

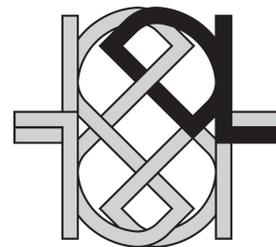
# Proceedings of the 20th International Workshop on Description Logics DL'07

June 8–10, 2007  
Free University of Bozen – Bolzano  
Brixen/Bressanone, Italy

EDITED BY  
DIEGO CALVANESE, ENRICO FRANCONI, VOLKER HAARSLEV, DOMENICO LEMBO,  
BORIS MOTIK, SERGIO TESSARIS, AND ANNI-YASMIN TURHAN



BOZEN · BOLZANO UNIVERSITY PRESS



Eds.: Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Sergio Tessaris, and Anni-Yasmin Turhan

Editing: Faculty of Computer Science, Free University of Bozen/Bolzano

Graphic design cover: Gruppe Gut Graphic Design, Bozen/Bolzano

Printing: DigiPrint, Bozen/Bolzano

Distribution:  
University Library of Bozen/Bolzano  
[www.unibz.it/library](http://www.unibz.it/library)

© 2007 by Bozen-Bolzano University Press  
Bozen/Bolzano  
All rights reserved

ISBN 978-88-6046-008-5

## Preface

Welcome to the 2007 International Workshop on Description Logics, DL07, in Brixen-Bressanone, Italy, 8-10 June 2007. The workshop, arriving this year to its 20th edition, continues the long-standing tradition of international workshops devoted to discussing developments and applications of knowledge representation formalisms and systems based on Description Logics. The list of the International Workshops on Description Logics can be found at <http://dl.kr.org>.

There were 86 papers submitted to DL07, more than double of the submissions in each of the past few editions. Each paper was reviewed by at least three members of the program committee or additional reviewers recruited by the PC members.

Papers were accepted in three categories: long papers, regular papers (with and without associated presentation), and posters. In addition to the presentation of the accepted papers, posters, and demos, the following speakers gave invited talks at the workshop:

- Alex Borgida: *Knowledge Representation Meets Databases — a view of the symbiosis*
- Hector Levesque: *Some Further Thoughts on Expressiveness and Tractability*
- Renee J. Miller: *Retrospective on Clio: Schema Mapping and Data Exchange in Practice*

The chairs the DL07 workshop gratefully acknowledge the support of the Free University of Bozen-Bolzano ([www.unibz.it](http://www.unibz.it)) in both the financial and logistical support, in particular with the registration process. Additionally, we would like to thank the organisers of DL’06, Bijan Parsia, Ulrike Sattler, and David Toman, for donating the surplus they made in 2006 in the form of Student Travel Awards.

Our thanks go to all the authors for submitting to DL’07, and to the invited speakers, PC members, and all additional reviewers who made the technical programme possible. In particular, we like to emphasise the effort of the four area chairs Volker Haarslev, Domenico Lembo, Boris Motik, and Anni-Yasmin Turhan in the co-ordination and organisation of both the review process and the workshop programme.

The organisation of the workshop also greatly benefited from the help of many people at the Free University of Bozen-Bolzano. Finally, we would like to acknowledge that the work of the PC was greatly simplified by using the EasyChair conference management system ([www.easychair.org](http://www.easychair.org)) developed by Andrei Voronkov.

Diego Calvanese, Enrico Franconi, and Sergio Tessaris  
Description Logics 2007 workshop chairs

## Conference Organisation

### Workshop Chairs

Diego Calvanese

Enrico Franconi

Sergio Tessaris

### Programme Chairs

Volker Haarslev

Domenico Lembo

Boris Motik

Anni-Yasmin Turhan

### Programme Committee

Carlos Areces

Alessandro Artale

Franz Baader

Alex Borgida

Andrea Cali

Bernardo Cuenca Grau

Francesco M Donini

Thomas Eiter

Birte Glimm

Ian Horrocks

Aditya Kalyanpur

Yevgeny Kazakov

Ralf Küsters

Maurizio Lenzerini

Thorsten Liebig

Carsten Lutz

Maarten Marx

Ralf Möller

Luigi Palopoli

Bijan Parsia

Peter Patel-Schneider

Riccardo Rosati

Ulrike Sattler

Stefan Schlobach

Luciano Serafini

Rob Shearer

Evren Sirin

Umberto Straccia

David Toman

Grant Weddell

Frank Wolter

Michael Zakharyashev

## Additional Reviewers

Sebastian Brandt  
Giuseppe De Giacomo  
Sofia Espinosa  
Valeria Fionda  
Achille Fokoue  
Alissa Kaplunova  
Atila Kaya  
Roman Kontchakov  
Tadeusz Litak  
Rainer Marrone  
Francisco Martin-Recuerda  
Davide Martinenghi  
Maja Milicic  
Magdalena Ortiz  
Charles Penwill  
Francesco Ricca  
Mariano Rodriguez  
Massimo Ruffolo  
Vladislav Ryzhikov  
Francesco Scarcello  
Baris Sertkaya  
Mantas Simkus  
Michael Smith  
Gaston Tagni  
Andrei Tamilin  
Dmitry Tsarkov  
Domenico Ursino  
Sebastian Wandelt  
Taowei Wang  
Evgeny Zolin

## Table of Contents

### Invited Talks.

Knowledge Representation Meets Databases — a view of the symbiosis ( <i>invited talk</i> ) . . . . .	1
<i>Alex Borgida</i>	
Some Further Thoughts on Expressiveness and Tractability ( <i>invited talk</i> ) . . .	12
<i>Hector Levesque</i>	
Retrospective on Clio: Schema Mapping and Data Exchange in Practice ( <i>invited talk</i> ) . . . . .	13
<i>Renee J. Miller</i>	

### Long Papers.

Blocking Automata for PSPACE DLs . . . . .	17
<i>Franz Baader, Jan Hladik, Rafael Penaloza</i>	
Actions and Programs over Description Logic Ontologies . . . . .	29
<i>Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Riccardo Rosati</i>	
Ontology Reuse: Better Safe than Sorry . . . . .	41
<i>Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, Ulrike Sattler</i>	
A New Mapping from $\mathcal{ALCI}$ to $\mathcal{ALC}$ . . . . .	53
<i>Yu Ding, Volker Haarslev, Jiewen Wu</i>	
Conjunctive Query Entailment for $\mathcal{SHOQ}$ . . . . .	65
<i>Birte Glimm, Ian Horrocks, Ulrike Sattler</i>	
Modularity in DL-Lite . . . . .	76
<i>Roman Kontchakov, Frank Wolter, Michael Zakharyashev</i>	
Data Complexity in the $\mathcal{EL}$ family of DLs . . . . .	88
<i>Adila Krisnadhi, Carsten Lutz</i>	
Inverse Roles Make Conjunctive Queries Hard . . . . .	100
<i>Carsten Lutz</i>	
Planning in Action Formalisms based on DLs: First Results . . . . .	112
<i>Maja Milicic</i>	
On Ordering Descriptions in a Description Logic . . . . .	123
<i>Jeffrey Pound, Lubomir Stanchev, David Toman, Grant Weddell</i>	

Deciding $\mathcal{ALBO}$ With Tableau .....	135
<i>Renate Schmidt, Dmitry Tishkovsky</i>	
<b>Regular Papers.</b>	
Description Logic vs. Order-Sorted Feature Logic .....	147
<i>Hassan Ait-Kaci</i>	
SEMilarity: Towards a Model-Driven Approach to Similarity .....	155
<i>Rudi Araújo, HSofia Pinto</i>	
Complexity of Reasoning over Entity Relationship Models .....	163
<i>Alessandro Artale, Diego Calvanese, Roman Kontchakov, Vladislav Ryzhikov, Michael Zakharyashev</i>	
Pinpointing in the Description Logic $\mathcal{EL}$ .....	171
<i>Franz Baader, Rafael Penaloza, Boontawe Sontisrivaraporn</i>	
Modal vs. Propositional Reasoning for Model Checking with Description Logics .....	179
<i>Shoham Ben-David, Richard Trefler, Grant Weddell</i>	
A General Framework for Covering Concepts Using Terminologies .....	187
<i>Boualem Benatallah, Mohand-Said Hacid, Alain Leger, Christophe Rey, Farouk Toumani</i>	
Expressing DL-Lite Ontologies with Controlled English .....	195
<i>Raffaella Bernardi, Diego Calvanese, Camilo Thorne</i>	
Consequence Finding in $\mathcal{ALC}$ .....	203
<i>Meghyn Bienvenu</i>	
On Importing Knowledge from DL Ontologies: some intuitions and problems	211
<i>Alex Borgida</i>	
A Constructive Semantics for $\mathcal{ALC}$ .....	219
<i>Loris Bozzato, Mauro Ferrari, Camillo Fiorentini, Guido Fiorino</i>	
MASTRO-I: Efficient Integration of Relational Data Through DL Ontologies	227
<i>Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Domenico Lembo, Antonella Poggi, Riccardo Rosati</i>	
A Proof Theory for DL-Lite .....	235
<i>Diego Calvanese, Evgeny Kharlamov, Werner Nutt</i>	
Dynamic Description Logic: Embracing Actions into Description Logic .....	243
<i>Liang Chang, Zhongzhi Shi, Lirong Qiu, Fen Lin</i>	
Partitioning ABoxes Based on Converting DL to Plain Datalog .....	251
<i>Jianfeng Du, Yi-Dong Shen</i>	

Exploiting Conjunctive Queries in Description Logic Programs . . . . .	259
<i>Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, Roman Schindlauer</i>	
OntoVQL: A Graphical Query Language for OWL Ontologies . . . . .	267
<i>Amineh Fadhil, Volker Haarslev</i>	
Induction of Optimal Semi-distances for Individuals based on Feature Sets . . . . .	275
<i>Nicola Fanizzi, Claudia d’Amato, Floriana Esposito</i>	
On Relating Heterogeneous Elements from Different Ontologies . . . . .	283
<i>Chiara Ghidini, Luciano Serafini, Sergio Tessaris</i>	
Contextualization of a DL Knowledge Base . . . . .	291
<i>Krzysztof Goczyła, Wojciech Waloszek, Aleksander Waloszek</i>	
EXPTIME Tableaux for $\mathcal{ALC}$ Using Sound Global Caching . . . . .	299
<i>Rajeev Gore, Linh Anh Nguyen</i>	
Optimizing Tableau Reasoning in $\mathcal{ALC}$ Extended with Uncertainty . . . . .	307
<i>Volker Haarslev, Hsueh-Ieng Pai, Nematollaah Shiri</i>	
Distributed Description Logics Revisited . . . . .	315
<i>Martin Homola</i>	
Multimedia Interpretation as Abduction . . . . .	323
<i>Atila Kaya, Sylvia Melzer, Ralf Möller, Sofia Espinosa, Michael Wessel</i>	
Prospects for and Issues With Mapping the Object-Role Modeling Language into $\mathcal{DLR}_{ifd}$ . . . . .	331
<i>C. Maria Keet</i>	
Adding ABoxes to a Description Logic with Uniqueness Constraints via Path Agreements . . . . .	339
<i>Vitaliy Khizder, David Toman, Grant Weddell</i>	
A Well-founded Semantics for Hybrid MKNF Knowledge Bases . . . . .	347
<i>Matthias Knorr, Jose Julio Alferes, Pascal Hitzler</i>	
Conjunctive Queries for $\mathcal{EL}$ with Role Composition . . . . .	355
<i>Markus Krötzsch, Sebastian Rudolph</i>	
Semantic difference in $\mathcal{ALN}$ . . . . .	363
<i>Alain Leger, Christophe Rey, Toumani Farouk</i>	
Consistent Query Answering over Description Logic Ontologies . . . . .	371
<i>Domenico Lembo, Marco Ruzzi</i>	
Exploiting Description Logic Reasoners in Inductive Logic Programming Systems: An Experience within the Semantic Web Area . . . . .	379
<i>Francesca Alessandra Lisi</i>	

Extracting Ontologies from Relational Databases . . . . .	387
<i>Lina Lubyte, Sergio Tessaris</i>	
Paraconsistent Resolution for Four-valued Description Logics . . . . .	395
<i>Yue Ma, Pascal Hitzler, Zuoquan Lin</i>	
Measuring Inconsistency for Description Logics Based on Paraconsistent Semantics . . . . .	403
<i>Yue Ma, Guilin Qi, Pascal Hitzler, Zuoquan Lin</i>	
Action Based ABox Update: an Example from the Chemical Compound Formulation . . . . .	411
<i>Alessandro Mosca, Matteo Palmonari</i>	
A Hypertableau Calculus for $\mathcal{SHIQ}$ . . . . .	419
<i>Boris Motik, Rob Shearer, Ian Horrocks</i>	
Expressive Querying over Fuzzy DL-Lite Ontologies . . . . .	427
<i>Jeff Z. Pan, Giorgos Stamou, Giorgos Stoilos, Edward Thomas</i>	
A Possibilistic Extension of Description Logics . . . . .	435
<i>Guilin Qi, Jeff Z. Pan, Qiu Ji</i>	
DL-based Alternating-Offers Protocol for Automated Multi-Issue Bilateral Negotiation . . . . .	443
<i>Azzurra Ragone, Tommaso Di Noia, Eugenio Di Sciascio, Francesco Donini</i>	
On Conjunctive Query Answering in $\mathcal{EL}$ . . . . .	451
<i>Riccardo Rosati</i>	
Combining Two Formalism for Reasoning about Concepts . . . . .	459
<i>Nikolay Shilov, Igor Anureev, Natalia Garanina</i>	
DLMedia: an Ontology Mediated Multimedia Information Retrieval System . . . . .	467
<i>Umberto Straccia, Giulio Visco</i>	
Approximate Subsumption for Complex Description Logics . . . . .	475
<i>Heiner Stuckenschmidt</i>	
Speeding up Approximation with Nicer Concepts . . . . .	483
<i>Anni-Yasmin Turhan, Yusri Bong</i>	
$\mathcal{DLclog}$ : A Hybrid System Integrating Rules and Description Logics with Circumscription . . . . .	491
<i>Fangkai Yang, Xiaoping Chen</i>	
A Boolean Lattice Based Fuzzy Description Logic in Web Computing . . . . .	499
<i>changli zhang, Jian Wu, Zhengguo Hu</i>	
Integrated Distributed Description Logics . . . . .	507
<i>Antoine Zimmermann</i>	

Modal Logic Applied to Query Answering and the Case for Variable Modalities .....	515
<i>Evgeny Zolin</i>	
<b>Statements of Interest.</b>	
The Minimal Finite Model Visualization as an Ontology Debugging Tool ...	523
<i>Guntis Barzdins, Martins Barinskis</i>	
Structural Theory of Science as a Systematic Framework for the Design of DL's and CD's for E-science .....	525
<i>Hansje Braam</i>	
Integrating Semantic Annotations in Bayesian Causal Models .....	527
<i>Hector Ceballos, Francisco Cantú Ortíz</i>	
Ontological Modelling for Neurovascular Disease Study: Issues in the Adoption of Description Logic .....	529
<i>Gianluca Colombo, Daniele Merico, Marco Antoniotti, Flavio De Paoli, Giancarlo Mauri</i>	
An ExpTime Tableau Decision Procedure for $\mathcal{ALCQI}$ .....	531
<i>Yu Ding, Volker Haarslev</i>	
Tools for the QuOnto System - Conversion between OWL and DL-Lite with Protégé-OWL Plug-in .....	533
<i>Manuel Dioturni, Maurizio Iacovella</i>	
Using Off-the-Shelf Reasoners for Reasoning over Distributed ABoxes .....	535
<i>Florian Fuchs, Michael Berger</i>	
Explaining Subsumption and Patching Non-Subsumption with Tableau Methods .....	537
<i>Thorsten Liebig, Stephan Scheele, Julian Lambertz</i>	
Model Checking of Restricted CTL* Formulas Using $\mathcal{ALCK}_{\mathcal{R}+}$ .....	539
<i>Taufiq Rochaeli, Claudia Eckert</i>	
Description Logics in the Calculus of Structures .....	541
<i>Jean-David Roubach, Pascal Yim, Joaquín Rodríguez</i>	
Efficient Query Answering Through Approximation .....	543
<i>Edward Thomas, Jeff Z. Pan</i>	
Practical Conforming Datatype Groups .....	545
<i>Dave Turner, Jeremy Carroll</i>	

# Knowledge Representation meets Databases — a view of the symbiosis —

Alex Borgida

Dept. of Computer Science  
Rutgers University  
New Brunswick, NJ 08904, USA  
[borgida@cs.rutgers.edu](mailto:borgida@cs.rutgers.edu)

**Abstract.** This rather informal paper surveys a personal selection of research projects which addressed new problems related to Databases, and whose solution was both inspired by ideas from the field of Knowledge Representation and Reasoning, and at the same time ended up contributing new ideas to that field. The problems include natural language access to databases, Information System (environment) design, permitting exceptions to integrity constraints, configuration databases, and goal-oriented schema design.

## 1 Introduction

It is possible to view KR&DB research from multiple viewpoints. First, it can be considered as applying database notions to knowledge representation systems. The typical concerns of database research are taken to be persistence and scale, which require special data storage techniques and possibly optimization of queries. Less frequently considered outside the DB field, but just as important, are notions such as concurrency control and recovery in case of failures. Thus for any KR&R system equipped with a Tell/Ask interface (e.g., a rule-based system, description logic reasoner), there have been investigations on how to add (some of) the above “database properties”. For example, Chaudri et al. [7] considered the issue of concurrency control for knowledge bases.

In the opposite direction, one can view KR&DB research as applying KR ideas to traditional database problems. For example, normally the standard first step in database schema design is drawing an Entity-Relationship diagram. It turns out that one can check the self-consistency of ER diagrams, or of single entity sets in them, by translating them into a description logic, and then using a standard DL reasoner, as done by the i.Com tool [9].

I will focus instead on the mutually supportive research at the meet of these two areas. Invoking the privilege of an invited speaker, I will concentrate exclusively on work that I have personally witnessed or been involved in<sup>1</sup>.

---

<sup>1</sup> I have had the great fortune to collaborate on much of this work with John Mylopoulos and Ron Brachman, who are not just outstanding scientists but, even more importantly to me, the nicest, most easy-going and supportive people one could hope to meet. It gives me great pleasure to acknowledge their invaluable help.

To give some shape to my presentation I will use a list of topics that have been the focus of considerable research in the KR&R field:

1. semantic networks
2. objects/classes with *instanceOf* and *IsA*
3. First Order Logic
4. Description Logics
5. goals

and discuss how attempting to solve new database-related problems led to solutions inspired by these notions, and frequently new results of interest to KR&R itself. Given the nature of the audience, I will also try to point occasionally to interesting connections to topics such as Description Logics and the Semantic Web.

## 2 Semantic Networks

In 1974/75 John Mylopoulos was flooded by a large group of new students interested in AI research (including James Allen, Phil Cohen, Hector Levesque, John Tsotsos and yours truly). He proposed to work on the problem of **natural language access to databases** - the Torus project. (Other significant AI research in NLP was being driven by the same problem, including William Woods’ LUNAR project, and William A. Martin’s EQS/OWL projects.) To solve the NLP problem, one needs, of course, a representation of the semantics of sentences, and this being the mid 1970’s, the answer was semantic networks: labeled directed graphs, which in our case had an open-ended label set for nodes, but only employ a fixed pre-determined set of possible edge labels.

Fig.1 is based on [14], and illustrates part of a semantic network graph describing the process of writing, sending and receiving recommendation letters.

What makes this interesting from a KR&DB point of view is that in order to answer questions such as “Did we receive any letters for Jimbo?”, the semantic representation needs to be connected to the database. Supposing that there is a database table

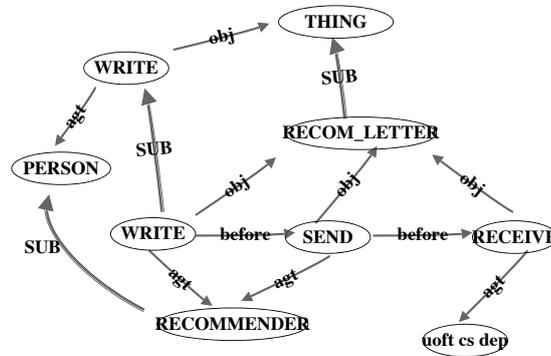
`RecLetterTable(Name,Source,Address,Text,DateReceived)`

this is accomplished by connecting every column to a node in the graph (e.g., `Name` to `Applicant`, `Source` to `Recommender`, ...). The result is that *the semantic network acts as a semantic data model for the database* — one providing, in fact, considerably richer semantics than standard ER diagrams. What was missing of course, as elsewhere in AI, is precise semantics for semantic networks.

### KR&R Resonances

The following quotes from the IJCAI’75 papers on Torus resonate interestingly with later topics in the field of DL:

- “Due to the properties of the sub/superset hierarchy, there is a unique position in the semantic net for each semantic graph we wish to integrate”. For



2007 © A. Borgida

7D1.07

Fig. 1. Part of a Torus Semantic Network

example, as illustrated in Fig.1, “recommender writing a recommendation letter” is a specialization of “writing”. Thus Torus’ (graph) classification resembles concept classification in DLs, but without a clear idea of what is a definition vs primitive, or what is the semantics of subsumption.

- The term “*definitional axis*” is mentioned in [15], but with no further explanation.
- Torus represents properties, such as `hasColor`, using so-called “characteristics”:

```

PHYS_OBJECT <--ch-- COLOR --val--> COLOR_VALUE
    /
    /
  eltOf
    /
PapaSmurf <--ch-- COLOR --val--> Blue
    /
    /
  
```

which resembles the representation suggested in the recent Dolce ontology [10], based on the notion of *qualia*.

- Some concepts, like `ADDRESS_VALUE`, have an associated “recognition function” to recognize instantiations, such as ‘65 st george street, toronto’. This prefigured the *test-defined concepts* of the Classic description logic [5] (and of Taxis [16]): these are concepts that have associated procedures for recognizing instances, so that they support instance classification, but are treated as primitive “blackbox” concepts as far as concept classification.

### 3 Inheritance and Meta-classes Everywhere

In 1977, Mylopoulos and Wong, with the assistance of Phil Bernstein, embarked on the Taxis project [16], whose goal was to develop a **language for designing/implementing Information Systems** at a conceptual, rather than “logical”/“physical” level. Taxis provided a conceptual modeling language not just for objects but also for procedures, exceptions, exception handlers, and eventually workflows. So, for example, after specifying Students, Courses, and how to Enroll students into courses, one could describe Part\_Time\_Students, Graduate\_Courses, and additional details concerning the procedure to enroll for these specialized arguments. My participation in the project concerned the formal semantics of Taxis, especially procedures, workflows (“scripts”), and the overall methodology of programming by specialization.

#### KR&R Resonances

- IsA hierarchies of procedures were also a new idea in KR&R — see [2].
- Taxis scripts [1] which prefigured workflows, were also organized in inheritance hierarchies. This echoes somewhat the application of DLs to workflow fragment management [11].

While Taxis pushed to the limits the use of inheritance, *meta-classes* become essential when moving to the creation of an **IS software development environments**. The result was the language Telos [17], which allowed one to define (conceptual) models, and to state meta-data constraints. For example, consider the use of a *property category* like “initial condition” in

```
PERSON
  initial condition
  age : {0}
```

This allowed what were essentially assertions in a complex temporal logic to be abbreviated so the time aspects did not appear in formulas. By thinking of properties, such as **age**, as classes (instantiated potentially for each individual of their domain class), property categories, like **initial condition**, become meta-classes, like **INITIAL\_PROP\_CLASS**: classes with classes as instances. So, given that property classes *p* have associated *p.Class*, *p.Name*, and *p.Range*, in our example we would have

```
p0.Class=PERSON, p0.Name = age , p0.Range = {0}
```

and *p0* would be made an instance of property meta-class **INITIAL\_PROP\_CLASS**, which would be defined to have an (obscure) invariant constraint

$$\forall p \in \text{INITIAL\_PROP\_CLASS} . \forall t \in \text{TIME} . \forall x . \\ (x \in p.\text{Class}@t \wedge x \notin p.\text{Class}@t-1) \Rightarrow x.(p.\text{Name})@t \in p.\text{Range}$$

describing the temporal semantics.

Incidentally, according to its formalization, a Telos KB consists of a set of “units”, each with 4 associated fields: [[source, identifier, destination, time-interval]]. For example,

```
p1 with [[Jimbo,instanceOf,PERSON,2/feb/1992 - 31/dec/2067]]
p2 with [[PERSON,age,{0},all_time]]
p3 with [[p2,instanceOf,INITIAL_PROP_CLASS,all_time]]
Jimbo with [[Jimbo,Jimbo,Jimbo, 2/feb/1992 - 31/dec/2067]]
```

As the last line above indicates, even objects were units with 4 fields.

### KR&R Resonances

- RDF(S) anyone? Telos’ treatment of properties as objects that link a source to a destination via a name echo RDF’s <subject, predicate, object> triples, especially the fact that properties can have properties themselves. The main difference is Telos’ addition of a temporal interval.
- While Telos was a theoretical language, it was made into a practical system, ConceptBase [12], by adding deductive rules to it. The considerable success of ConceptBase (downloaded at over 500 sites) lies at least in part in its uniform treatment of everything as an object that can have properties, which allows meta-statements to be easily recorded.

## 4 First Order Logic

Integrity Constraints are intended to detect inconsistencies after database updates. But, when dealing with the natural world, these constraints are almost always over-generalizations. Hence, one desires to **allow the coexistence of general constraints** such as

$$IC : \forall e. e \in EMP \Rightarrow (e.salary > 1000) \wedge (e.salary < e.manager.salary)$$

and occasional exceptions such as

```
calvin.salary=20000
calvin.manager = hobbes
hobbes.salary=15000
```

where the problem might be blamed on the fact that **hobbes** is only temporarily assigned to be **calvin**’s manager. But this leads to a difficulty: once even *one* exception is allowed to persist, this IC will always evaluate to false, and can no longer detect errors in future updates (e.g., when **judy**’s salary is changed to 900), which is “a new reason for it to be false”. Therefore we want to modify the constraint to restore its error detection role. In [3], I propose that one first rewrite IC in FOL without function symbols:

$$\forall e.(e \in EMP \wedge sal(e, se)) \Rightarrow (se > 1000) \wedge \forall me, sm.(mgr(e, me) \wedge sal(me, sm)) \Rightarrow se < me)$$

Then, rather than going for the obvious

$$\forall e.(e \neq hobbes \wedge e \in EMP \wedge sal(e, se)) \Rightarrow (se > 1000) \wedge \forall me, sm.(mgr(e, me) \wedge sal(me, sm)) \Rightarrow se < me)$$

(which is not good enough because in this IC there are two conditions that are being checked at the same time), we propose to use the the more subtle

$$\forall e.(e \in EMP \wedge sal(e, se)) \Rightarrow (se > 1000) \wedge ([mgr(e, me) \wedge \neg(e = calvin \wedge me = hobbes)] \wedge sal(me, sm)) \Rightarrow se < me$$

Interestingly, this corresponds to a *model theoretic* specification: minimally mutilate all models of the original IC, so that the exceptional fact (*manager(calvin, hobbes)*) is counter-factual. The nice property of this is that (i) there is a syntactic transformation corresponding to it, and (ii) the actual syntactic form of the ICs does not matter. (I owe Ray Reiter a debt for helping me see the above.)

### KR&R Resonances

My student, Mukesh Dalal used this idea of minimal mutilation of models to obtain a propositional knowledge-base revision operator  $update(kb, u)$  [8] — (almost) the first one that was insensitive to the syntactic form of the theory  $kb$  or the update  $u$ . (To get the models of  $update(kb, u)$ , it iteratively mutilated single atoms of  $kb$  models till at least one model consistent with  $u$  was found.) Interestingly, through the advice of David Israel, this seems to also have been the first AI paper to mention the Alchourrón - Gärdenfors - Makinson belief revision axioms.

## 5 The Classic Description Logic

Classic [5] was probably the most widely used second-generation description logic system: one with precise semantics, polynomial time subsumption checking, and complete algorithm (when **one-of** and **fills** were used with individuals that themselves had no properties, such as numbers, enumerations, etc.).

More significantly, Classic was used in real, industrial applications dealing with configuration management, as well as (attempted) support for data exploration/mining carried out by humans.

Not only was Classic first published in the SIGMOD database conference (rather than an AI conference), but **special strengths of description logics became apparent when viewing DLs as languages used for interacting with a database-like system**. Specifically, one can adopt a Levesque- and SQL-inspired view of a knowledge base management system as a black box with

- *create* operator to declare new identifiers, with possible definitions
- *constrain* operator to express integrity constraints on valid states of the KB
- *update* operator to manage facts about a specific world (A-box)
- *inquire* operator to ask about the state of the world (A-box)

Now, each of the first three operators  $x$  above involves a language  $L_x$ , while *inquire* involves two languages: one for stating questions,  $L_{query}$ , and one for expressing answers,  $L_{answer}$ . By considering what happens when each of these languages is a DL such as Classic, one gets the following insights:

- *L<sub>create</sub>*: DLs provide the (well-known) opportunity to add not just primitive but also defined concepts (“views”), and to automatically organize these in subsumption hierarchies.
- *L<sub>constrain</sub>*: DLs allow necessary conditions to be stated on primitive concepts, which are like Integrity Constraints. (These are the first key ingredient in configuration management applications.)
- *L<sub>update</sub>*: By asserting a description such as  $\forall \text{friends} . (\forall \text{gender} . \text{FEMALE})$  about *DonJuan*, one is able to say things about an indefinite number of objects – the current and any yet to be specified friends of Don Juan. This is the source of considerable expressive power for DL-based knowledge/data management, in contrast to null values in relational databases, and the second key to the success of Classic in configuration management. Note that this benefit of dealing with incomplete information accrues even when the language does not support disjunction, and has polynomial time reasoning.
- *L<sub>query</sub>*: Of course, DL concepts are well suited to retrieve sets of values — their instances; a benefit here is that queries can themselves be organized in subsumption hierarchies, which facilitates re-use and query refinement. (On the down side, DL concepts cannot express even some very simple conjunctive queries [4] — the bread and butter of database research.)
- *L<sub>answer</sub>*: When using DL concepts as part of answers, one gets the benefit of *descriptive*, rather than just enumerated, answers. For example, in response to the query “Who is female?” one might get not just Eve, but also “the friends of Don Juan”.

### KR&R Resonances

It may be worth recording that essential to the practical success of Classic was the ability to extend its expressive power as needed (through test functions). I believe that OWL has not yet met the hard-nosed challenge of real applications, where customers may walk away if their needs are not served. I do not see any reasons why OWL cannot be made extensible at least to the point of allowing arbitrary “concrete domains”, and then maintaining libraries of such extensions, just like we will maintain libraries of concepts (ontologies).

A more interesting reverberation was our application of Classic-like ideas to the specification of Corba services [6], which is exactly like the use of DLs for specifying services on the Semantic Web.

## 6 Goals

In the past decades, research in Requirements Engineering for software has undergone a revolution, whereby the standard functional specification stage is now preceded by a new phase of *early requirements*, dealing with the intentions of the agents and organizations in the environment where the software is to be used. Because of its focus on goals, and how they are to be achieved, this is

known as Goal-Oriented Requirements Engineering (GORE). One of the important aspects of GORE is dealing with so-called non-functional (soft) goals, such as efficiency, accuracy, etc. — goals that do not have clear criteria of success.

The field of database specification, on the other hand, has remained pretty-well unchanged since the mid-70's: one still starts by constructing a conceptual schema as an ER or maybe UML diagram. In recent joint work with Jiang, Topaloglou and Mylopoulos [13], we have started to look at a **goal-oriented approach to database design**. As in GORE, we start with various stakeholders, and their general hard and soft goals; decompose these into subgoals using AND/OR graphs; then apply means/ends analysis to find tasks that achieve them. See Fig. 2 for a small example, which uses the  $i^*$  notation [19]. In diagrams, we use contribution edges labeled with + or - (or

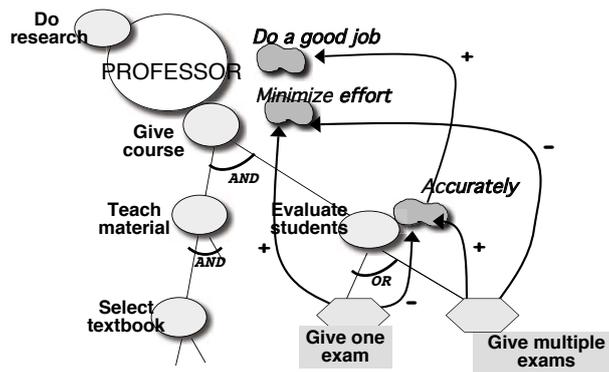


Fig. 2. A Goal Model in  $i^*$

even ++ and --) to indicate how goals influence each other. Thus, in Fig.2, the softgoal (peanut-shape) of evaluating students accurately contributes positively towards the “Do a good job” softgoal, but performing the task of giving a single exam (hexagon) contributes negatively towards accurate evaluation. One important advantage of goal-oriented approaches is that they provide consideration of design alternatives, and traceability for decisions based on such contribution dependencies. So, for example, we might choose between schemas `StudentTable1(studentId,examGrade)` and `StudentTable1(studentId,exam1,exame2,averageGrade)` based on which goals are more important.

Continuing with our goal-based schema design, our methodology suggests analyzing the textual description of goals, and the participants in the tasks, to obtain a list of “relevant concepts”, which is then organized into a *domain model* (expressed in some conceptual modeling language); its purpose is to provide a shared understanding of the domain for database designers and end-users. The *conceptual schema* of the database is then derived from this by addressing a list of questions concerning issues such as persistence, time, data quality, etc. For each such category, we have a number of alternative schema manipulation operators which can be applied to derive the final conceptual schema from the domain model.

### KR&R Resonances

Of course, one of the earliest KR&R proposals in Artificial Intelligence was Simon and Newell’s GPS, which was also concerned with means-ends analysis to achieve goals. And, in a separate context, it was Simon who introduced the important notion of “*satisficing*”, which is applicable for such goals.

Although it is clear how to formalize AND/OR goal decomposition even in Horn propositional logic, notions like softgoals, their satisficing, and contribution edges would seem to be inherently “soft” — hard to reason with. Sebastiani et al [18] however show how to formalize even this aspect: Since there could be conflicting evidence concerning any goal  $g$ , the secret is to replace proposition  $g$  by propositions *fully\_satisfied\_g*, *partially\_satisfied\_g*, *partially\_denied\_g*, and *fully\_denied\_g*. An edge  $g \xrightarrow{-} h$  then introduces axiom

$$partially\_satisfied\_g \Rightarrow partially\_denied\_h$$

while edge  $g \xrightarrow{+} h$  also adds axiom

$$fully\_satisfied\_g \Rightarrow fully\_denied\_h$$

By using a suitable extension of this set of axioms, and a min-sat solver, it is then possible to find minimal sets of “input/bottom” goals that guarantee desired top-level goals.

## 7 Conclusions

I have briefly reviewed a sample of database-inspired projects which had connections to KR&R topics: natural language access to databases  $\leftrightarrow$  semantic networks; information system design  $\leftrightarrow$  inheritance & metaclasses; exceptions to database integrity constraints  $\leftrightarrow$  First Order Logic and minimal mutilations; querying and verifying consistency of incomplete databases  $\leftrightarrow$  description logics; goal-based database design  $\leftrightarrow$  goal analysis and satisfaction/satisficing. In each case, I tried to give an impression of the benefits each field, DB and KR&R, derived from the other, as a result of the research carried out. Of course, the above survey was highly skewed towards my own experiences — there are many more such examples, both extant and to come.

## Acknowledgments

I am most grateful to all my collaborators over the years for making this such an enjoyable adventure. I single out Enrico Franconi for particular thanks for suggesting “KR&DB” as the topic of this paper, and for his comments on the write up. All remaining errors are of course my own.

This work was supported in part by the U.S. DHS under ONR grant N00014-07-1-0150.

## References

1. J.L. Barron, "Dialogue and Process Design for Interactive Information Systems using Taxis", Proceedings ACM SIGOA, pp. 12-20, Philadelphia, June 1982
2. Alexander Borgida: On the Definition of Specialization Hierarchies for Procedures. IJCAI 1981: 254-256
3. Alexander Borgida: Language Features for Flexible Handling of Exceptions in Information Systems. ACM Trans. Database Syst. 10(4): 565-603 (1985)
4. Alexander Borgida: On the Relative Expressiveness of Description Logics and Predicate Logics. Artif. Intell. 82(1-2): 353-367 (1996)
5. Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, Lori Alperin Resnick: CLASSIC: A Structural Data Model for Objects. SIGMOD Conference 1989: 58-67
6. Alexander Borgida, Premkumar T. Devanbu: Adding more "DL" to IDL: Towards More Knowledgeable Component Inter-Operability. ICSE 1999: 378-387
7. Chaudhri, V., Hadzilacos, V. and Mylopoulos, J., "Concurrency Control for Knowledge Bases", Third International Conference on Knowledge Representation and Reasoning, Boston, October 1992.
8. Mukesh Dalal: Investigations into a Theory of Knowledge Base Revision. AAAI 1988: 475-479
9. Enrico Franconi, Gary Ng: The i.com tool for Intelligent Conceptual Modeling. KRDB 2000: 45-53
10. Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, Luc Schneider: Sweetening Ontologies with DOLCE. EKAW 2002: 166-181
11. Antoon Goderis, Ulrike Sattler, Carole A. Goble: Applying Description Logics for Workflow Reuse and Repurposing. Description Logics 2005
12. Matthias Jarke, Rainer Gellersdrfer, Manfred A. Jeusfeld, Martin Staudt: ConceptBase - A Deductive Object Base for Meta Data Management. J. Intell. Inf. Syst. 4(2): 167-192 (1995)
13. Lei Jiang, Thodoros Topaloglou, Alexander Borgida, John Mylopoulos: Incorporating Goal Analysis in Database Design: A Case Study from Biological Data Management. RE 2006: 196-204
14. John Mylopoulos, Alexander Borgida, P. Cohen, Nick Roussopoulos, John K. Tsotsos, Harry K. T. Wong: TORUS - A Natural Language Understanding System For Data Management. IJCAI 1975: 414-421
15. John Mylopoulos, P. Cohen, Alexander Borgida, L. Sugar: Semantic Networks and the Generation of Context. IJCAI 1975: 134-142
16. John Mylopoulos, Philip A. Bernstein, Harry K. T. Wong: A Language Facility for Designing Database-Intensive Applications. SIGMOD 1978 (Abstract). ACM Trans. Database Syst. 5(2): 185-207 (1980)

17. John Mylopoulos, Alexander Borgida, Matthias Jarke, Manolis Koubarakis: Telos: Representing Knowledge About Information Systems. *ACM Trans. Inf. Syst.* 8(4): 325-362 (1990)
18. Roberto Sebastiani, Paolo Giorgini, John Mylopoulos: Simple and Minimum-Cost Satisfiability for Goal Models. *CAiSE 2004*: 20-35
19. Eric S. K. Yu, John Mylopoulos: From E-R to A-R — Modelling Strategic Actor Relationships for Business Process Reengineering. *Int. J. Cooperative Inf. Syst.* 4(2-3): 125-144 (1995)

## Some further thoughts on expressiveness and tractability

Hector Levesque

University of Toronto, Toronto, CANADA

**Abstract.** Since the mid 1980s, there has been considerable research studying the expressiveness and the tractability of various description logics. In this talk, we consider this issue as it was then and now. We argue that an overly strict interpretation may have caused the work on description logic to be seen as less relevant than it used to be to the general AI enterprise.

## Retrospective on Clio: Schema Mapping and Data Exchange in Practice

Renée J. Miller

Bell University Labs Chair of Information Systems  
Department of Computer Science  
University of Toronto  
miller@cs.toronto.edu

**Abstract.** Clio is a joint research project between the University of Toronto and IBM Almaden Research Center started in 1999 to address both foundational and systems issues related to the management of heterogeneous data. In this talk, I will take a look back over the last eight years of this project to review its achievements, the lessons learned, and the challenges that remain.

**Key words:** Schema mapping, mapping generation, mapping compilation and execution, data exchange, data integration

The Clio project was founded to tackle the challenging issues raised by the proliferation of independently developed data sources that are heterogeneous in their design and content. Heterogeneous data sets contain data that have been represented using different data models, different structuring primitives, or different modeling assumptions. Such data sets often have been developed and modeled with different requirements in mind. As a consequence, different schemas may have been used to represent the same or related data. To manage heterogeneous data, we must be able to manage these schemas and mappings between the schemas. Clio is a management system for heterogeneous data that couples traditional data management solutions with additional tools for creating, using and maintaining mappings between schemas. Our first results on schema mapping generation [MHH00] were first demonstrated at SIGMOD 2001 [HMH01]. The main contributions of the Clio project include the following. .

*Schema mapping generation.* In many integration applications, data that conforms to a (*source*) schema (also called a local schema), may be queried or viewed through another (*target*) schema (also called a global schema). The relationship between the source and the target schema is modeled through a set of artifacts called *schema mappings*. Prior to Clio, most work on automating schema mapping generation focused on finding simple attribute-attribute correspondences or matches (using text similarity or natural language techniques). Clio introduced a novel interactive mapping creation paradigm [MHH00] for creating mappings that represent structural transformations of data. These mappings are created

using the semantics encoded in the schemas, and represent the semantic relationship between schemas. We have investigated how to automate the mapping creation process, and how to effectively solicit user input when the semantics of the schemas are ambiguous or incomplete [MHH00,PVM<sup>+</sup>02]. This work was demonstrated at ICDE 2002 [HPV<sup>+</sup>02]. In addition, we have developed a data-driven visualization tool to help users understand and refine mappings as they are generated [YMHF01]. We use carefully chosen data examples to explain mappings and alternative mappings to help a user arrive at a correct and complete mapping. Most recently, we have considered the generation of mappings that may be correlated with other mappings [FHH<sup>+</sup>06], work that was demonstrated at ICDE 2007 [HHP<sup>+</sup>07]

*Schema mapping specification.* Mappings are specified using an expressive declarative language with a formal semantics. The mapping language is suitable for relational schemas and for nested structures including XML schema, DTDs, and concept hierarchies. The expressiveness of this mapping language enables a wide range of transformations and makes the language suitable for use in a variety of mapping applications [PVM<sup>+</sup>02,FHH<sup>+</sup>06].

In creating a mapping from a source schema to a target schema, we do not assume that the schemas represent the same data. Certainly, there may be source data that are not represented in the target. Additionally, there may be target data that are not represented in the source. An accurate mapping must be able to represent missing (unmapped) source and target data [PVM<sup>+</sup>02].

*Using Mappings.* The mappings produced by Clio can be used for both *data integration* (where the target data is virtual) [YP04] and for *data exchange* (where the target data is materialized) [PVM<sup>+</sup>02]. For data exchange, we faced the challenge of being able to generate new values for unspecified (unmapped) target data that are essential for ensuring the consistency of the target database. Clio presented the first algorithm for data exchange in 2002 using a solution that is guaranteed to generate a valid target database [PVM<sup>+</sup>02].

We have gone on to investigate some of the foundations of data exchange [FKMP03,FKMP05] with a formal study of how data exchange differs from data integration. Importantly, this formal study uses a mapping language (tuple-generating dependencies) that are inspired by the Clio system. We have considered how to characterize the best data exchange solution [FKP05], how to do data exchange in networks of peer schemas [FKMT06], how to compose mappings [FKPT05], and a number of other issues surrounding the use of mappings [Fag06].

*Mapping compilation and execution.* Mappings can be compiled into executable queries or programs which perform data exchange [PVM<sup>+</sup>02]. Depending on the application environment, Clio can generate SQL queries, Xquery, XSLT, among other formats, for execution. Alternatively, Clio also provides its own optimized execution engine for efficient evaluation of data exchange programs [JHPH07].

Our declarative mapping formalism is sufficiently expressive to capture the semantics of a wide variety of integration and exchange applications, and as a result, the ability to compile mappings into different runtime environments for efficient execution is critical.

*Managing and maintaining mappings.* Our declarative mappings can be adapted and reused more easily than low-level procedural mapping scripts. We have presented solutions for adapting mapping as schemas evolve [VMP03,VMP04,YP05], demonstrated at ICDE 2004 [VMPM04].

*Industrial impact.* Clio technology has been transferred into IBM’s product lines, and forms a core component of IBM’s Rational Data Architect [HHH<sup>+</sup>05]. The use of declarative mappings (over hand-coded procedural scripts) is now considered “best-practice” in commercial products and Clio has lead the way in changing the culture. In addition, Clio has been a leader in demonstrating that both schemas and mappings are dynamic artifacts that must be managed effectively.

## References

- [Fag06] Ronald Fagin. Inverting Schema Mappings. In *Proc. of the ACM Symp. on Principles of Database Systems (PODS)*, pages 50–59, 2006.
- [FHH<sup>+</sup>06] Ariel Fuxman, Mauricio A. Hernández, Howard Ho, Renée J. Miller, Paolo Papotti, and Lucian Popa. Nested Mappings: Schema Mapping Reloaded. In *Proc. of the Int’l Conf. on Very Large Data Bases (VLDB)*, pages 67–78, 2006.
- [FKMP03] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data Exchange: Semantics and Query Answering. In *Proc. of the Int’l Conf. on Database Theory (ICDT)*, pages 207–224, 2003.
- [FKMP05] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science*, 336(1):89–124, May 2005.
- [FKMT06] Ariel Fuxman, Phokion G. Kolaitis, Renée J. Miller, and Wang-Chiew Tan. Peer Data Exchange. *ACM Trans. on Database Systems (TODS)*, 31(4):1454–1498, 2006.
- [FKP05] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. *ACM Trans. on Database Systems (TODS)*, 30(1):174–210, 2005.
- [FKPT05] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang-Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. on Database Systems (TODS)*, 30(4):994–1055, 2005.
- [HHH<sup>+</sup>05] Laura M. Haas, Mauricio A. Hernández, Howard Ho, Lucian Popa, and Mary Roth. Clio grows up: from research prototype to industrial tool. In *ACM SIGMOD Int’l Conf. on the Management of Data*, pages 805–810, 2005.
- [HHP<sup>+</sup>07] Mauricio A. Hernández, Howard Ho, Lucian Popa, T. Fukuda, Ariel Fuxman, Renée J. Miller, and Paolo Papotti. Creating Nested Mappings with Clio. In *IEEE Proc. of the Int’l Conf. on Data Engineering (ICDE)*, 2007. System Demonstration.

- [HMH01] Mauricio A. Hernández, Renée J. Miller, and Laura Haas. Clio: A Semi-Automatic Tool for Schema Mapping. *ACM SIGMOD Int'l Conf. on the Management of Data*, 30(2):607, 2001. System Demonstration.
- [HPV<sup>+</sup>02] Mauricio A. Hernández, Lucian Popa, Yannis Velegarakis, Renée J. Miller, F. Naumann, and C.-T. Ho. Mapping XML and Relational Schemas with Clio. In *IEEE Proc. of the Int'l Conf. on Data Engineering (ICDE)*, pages 498–499, 2002. System Demonstration.
- [JHPH07] Haifeng Jiang, Howard Ho, Lucian Popa, and Wook-Shin Han. Mapping-Driven XML Transformation. In *International World Wide Web Conference*, 2007.
- [MHH00] Renée J. Miller, L. M. Haas, and Mauricio A. Hernández. Schema Mapping as Query Discovery. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 77–88, 2000.
- [MHH<sup>+</sup>01] Renée J. Miller, Mauricio A. Hernández, L. M. Haas, Ling-Ling Yan, Howard Ho, Ronald Fagin, and Lucian Popa. The Clio Project: Managing Heterogeneity. *SIGMOD Record*, 30(1):78–83, March 2001.
- [PVM<sup>+</sup>02] Lucian Popa, Yannis Velegarakis, Renée J. Miller, Mauricio A. Hernández, and Ronald Fagin. Translating Web Data. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 598–609, 2002.
- [VMP03] Yannis Velegarakis, Renée J. Miller, and Lucian Popa. Mapping Adaptation under Evolving Schemas. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 584–595, 2003.
- [VMP04] Yannis Velegarakis, Renée J. Miller, and Lucian Popa. On Preserving Mapping Consistency under Schema Changes. *The Int'l Journal on Very Large Data Bases*, 13(3):274–293, 2004.
- [VMPM04] Yannis Velegarakis, Renée J. Miller, Lucian Popa, and John Mylopoulos. ToMAS: A System for Adaptin Mappings as Schemas Evolve. In *IEEE Proc. of the Int'l Conf. on Data Engineering (ICDE)*, page 882, 2004. System Demonstration.
- [YMHF01] Ling-Ling Yan, Renée J. Miller, Laura Haas, and Ronald Fagin. Data-Driven Understanding and Refinement of Schema Mappings. *ACM SIGMOD Int'l Conf. on the Management of Data*, 30(2):485–496, May 2001.
- [YP04] Cong Yu and Lucian Popa. Constraint-Based XML Query Rewriting For Data Integration. In *ACM SIGMOD Int'l Conf. on the Management of Data*, pages 371–382, 2004.
- [YP05] Cong Yu and Lucian Popa. Semantic Adaptation of Schema Mappings when Schemas Evolve. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 1006–1017, 2005.

## Blocking Automata for PSpace DLs

Franz Baader<sup>1</sup>, Jan Hladik<sup>1</sup>, and Rafael Peñaloza<sup>2\*</sup>

<sup>1</sup> Theoretical Computer Science, TU Dresden, Germany  
[franz.baader@tu-dresden.de](mailto:franz.baader@tu-dresden.de), [jan.hladik@tu-dresden.de](mailto:jan.hladik@tu-dresden.de)

<sup>2</sup> Intelligent Systems, Uni Leipzig, Germany  
[penaloza@informatik.uni-leipzig.de](mailto:penaloza@informatik.uni-leipzig.de)

**Abstract.** In Description Logics (DLs), both tableau-based and automata-based algorithms are frequently used to show decidability and complexity results for basic inference problems such as concept satisfiability. Whereas tableau-based algorithms usually yield worst-case optimal algorithms in the case of PSPACE-complete logics, it is often very hard to design optimal tableau-based algorithms for EXPTIME-complete DLs. In contrast, the automata-based approach is usually well-suited to prove EXPTIME upper-bounds, but its direct application will usually also yield an EXPTIME-algorithm for a PSPACE-complete logic since the (tree) automaton constructed for a given concept is usually exponentially large. In the present paper, we formulate conditions under which an on-the-fly construction of such an exponentially large automaton can be used to obtain a PSPACE-algorithm. We illustrate the usefulness of this approach by proving a new PSPACE upper-bound for satisfiability of concepts w.r.t. acyclic terminologies in the DL  $\mathcal{SI}$ .

### 1 Introduction

Two of the most prominent methods for showing decidability and complexity results for DLs are the tableau-based [4] and the automata-based [7] approach. Both approaches basically depend on the tree-model property of the DL under consideration: if a concept is satisfiable, then it is also satisfiable in a tree-shaped model. They differ in how they test for the existence of a tree-shaped model. Tableau-based algorithms try to generate such a model in a top-down non-deterministic manner, starting with the root of the tree. Automata-based algorithms construct a tree automaton that accepts exactly the tree-shaped models of the concept, and then test the language accepted by this automaton for emptiness. The usual emptiness test for tree automata is deterministic and works in a bottom-up manner. This difference between the approaches also leads to different behaviour regarding elegance, complexity, and practicability.

If the logic has the *finite* tree model property, then termination of tableau-based algorithms is usually easy to achieve. If, in addition, the tree models these algorithms are trying to construct are of polynomial depth (as is the case for the PSPACE-complete problem of satisfiability in  $\mathcal{ALC}$ ), then one can usually modify

\* Funded by the DFG, Graduiertenkolleg Wissensrepräsentation, Uni Leipzig.

tableau-based algorithms such that they need only polynomial space: basically, they must only keep one path of the tree in memory [17]. However, the automaton constructed in the automata-based approach is usually exponential, and thus constructing it explicitly before applying the emptiness test requires exponential time and space. In [8], we formulate conditions on the constructed automaton that ensure—in the case of finite tree models of polynomially bounded depth—that an on-the-fly construction of the automaton during a non-deterministic top-down emptiness test yields a PSPACE algorithm.

If the logic does *not* have the *finite* tree model property, then applying the tableau-based approach in a straightforward manner leads to a non-terminating procedure. To ensure termination of tableau-based algorithms in this case, one must apply an appropriate cycle-checking technique, called “blocking” in the DL literature [4]. This is, for example, the case for satisfiability in  $\mathcal{ALC}$  w.r.t. general concept inclusions (GCIs) [1], which has both the finite model property and the tree model property, but not the finite tree model property. Since blocking usually occurs only after an exponential number of steps and since tableau-based algorithms are non-deterministic, the best complexity upper-bound that can be obtained this way is NEXPTIME. This is not optimal since satisfiability in  $\mathcal{ALC}$  w.r.t. GCIs is “only” EXPTIME-complete. The EXPTIME upper-bound can easily be shown with the automata-based approach: the constructed automaton is of exponential size, and the (bottom-up) emptiness test for tree automata runs in time polynomial in the size of the automaton.

Although the automata-based approach yields a worst-case optimal algorithm in this case, the obtained algorithm is not practical since it is also exponential in the best case: before applying the emptiness test, the exponentially large automaton must be constructed. In contrast, optimised implementations of tableau-based algorithms usually behave quite well in practice [9], in spite of the fact that they are not worst-case optimal.

There have been some attempts to overcome this mismatch between practical and worst-case optimal algorithms for EXPTIME-complete DLs. In [5] we show that the so-called inverse tableau method [19] can be seen as an on-the-fly implementation of the emptiness test in the automata-based approach, which avoids the a priori construction of the exponentially large automaton. Conversely, we show in [2] that the existence of a sound and complete so-called EXPTIME-admissible tableau-based algorithm for a logic always implies the existence of an EXPTIME automata-based algorithm. This allows us to construct only the (practical, but not worst-case optimal) tableau-based algorithm, and get the optimal EXPTIME upper-bound for free.

In the present paper, we extend the approach from [8] mentioned above such that it can also deal with PSPACE-complete logics that do not have the finite tree model property. A well-known example of such a logic is  $\mathcal{ALC}$  extended with transitive roles [14]. To illustrate the power of our approach, we use the more expressive DL  $\mathcal{ST}$  as an example, which extends  $\mathcal{ALC}$  with transitive and inverse roles. In addition, we also allow for acyclic concept definitions. To the

best of our knowledge, the result that satisfiability in  $\mathcal{SI}$  w.r.t. acyclic concept definitions is in PSPACE is new.

For lack of space we must omit most of the proofs of our results. Detailed proofs can be found in [3].

## 2 Hintikka trees for $\mathcal{SI}$ with general TBoxes

The DL  $\mathcal{SI}$  extends  $\mathcal{ALC}$  with transitive and inverse roles, i.e. if  $N_C$  is the set of concept names and  $N_R$  is the set of role names, then there is a set  $N_T \subseteq N_R$  of *transitive* role names and, if  $r \in N_R$ , we can use  $r^-$  in concept expressions like a role name. The semantics of  $\mathcal{SI}$  is defined as usual (see e.g. [11]). For convenience, we introduce the following notation: for an  $\mathcal{SI}$  role  $s$ , the *inverse of  $s$*  (denoted by  $\bar{s}$ ) is  $s^-$  if  $s$  is a role name, and  $r$  if  $s = r^-$  for a role name  $r$ . Since a role is interpreted as transitive iff its inverse is interpreted as transitive, we will use the predicate  $\text{trans}(r)$  on  $\mathcal{SI}$  roles to express that  $r$  or  $\bar{r}$  belongs to  $N_T$ .

An *acyclic TBox* is a set of concept definitions  $A \doteq C$ , where  $A$  is a concept name and  $C$  is an  $\mathcal{SI}$  concept term, with the additional condition that definitions are unique and acyclic in the sense that a concept name does not directly or indirectly appear in its own definition. A *general TBox* is an acyclic TBox which can additionally contain GCIs (general concept inclusion axioms) of the form  $C \sqsubseteq D$ , where both  $C$  and  $D$  are  $\mathcal{SI}$  concept terms. Please note that this definition is slightly non-standard because we do not allow cyclic definitions in general TBoxes. Obviously, an arbitrary set of definitions can be transformed into a general TBox by replacing cyclic definitions with GCIs. The semantics of TBoxes and the satisfiability problem are defined as usual (see e.g. [4]).

The definition of acyclic TBoxes ensures that the concept definitions simply introduce abbreviations (macro definitions), which could in principle be completely expanded by repeatedly replacing defined names by their definitions. Thus, acyclic TBoxes do not increase the expressive power, but they increase succinctness because expansion can lead to an exponential blow-up [13].

For the DL  $\mathcal{ALC}$ , it is known that the satisfiability problem is PSPACE-complete w.r.t. acyclic TBoxes [12] and EXPTIME-complete w.r.t. general TBoxes [16]. We will show in this paper that the same is true for  $\mathcal{SI}$ .

Tree models of satisfiable  $\mathcal{SI}$  concepts can be obtained by applying the well-known technique of unravelling [6]. For example, the  $\mathcal{SI}$  concept  $A$  is satisfiable w.r.t. the general TBox  $\{A \sqsubseteq \exists r.A\}$  in a one-element model whose single element belongs to  $A$  and is related to itself via  $r$ . The corresponding unravelled model consists of a sequence  $d_0, d_1, d_2, \dots$  of elements, all belonging to  $A$ , where  $d_i$  is related to  $d_{i+1}$  via  $r$ . Intuitively, Hintikka trees are tree models where every node is labelled with the concepts to which the element represented by the node belongs. These concepts are taken from the set of subconcepts of the concept to be tested for satisfiability and of the concepts occurring in the TBox. In our example, the nodes  $d_i$  would be labelled by  $A$  and  $\exists r.A$  since each  $d_i$  belongs to these concepts.

To simplify the formal definitions, we assume in the following that *all* concepts are in *negation normal form (NNF)*, i.e. negation appears only directly in front of concept names. Any  $\mathcal{SI}$  concept can be transformed into NNF in linear time using de Morgan’s laws, duality of quantifiers, and elimination of double negation. We denote the NNF of a concept  $C$  by  $\text{nnf}(C)$  and  $\text{nnf}(\neg C)$  by  $\neg C$ .

**Definition 1 (Subconcepts, Hintikka sets).** The set of *subconcepts* of an  $\mathcal{SI}$  concept  $C$  ( $\text{sub}(C)$ ) is the least set  $S$  that contains  $C$  and has the following properties: if  $S$  contains  $\neg A$  for a concept name  $A$ , then  $A \in S$ ; if  $S$  contains  $D \sqcup E$  or  $D \sqcap E$ , then  $\{D, E\} \subseteq S$ ; if  $S$  contains  $\exists r.D$  or  $\forall r.D$ , then  $D \in S$ . For a TBox  $\mathcal{T}$ ,  $\text{sub}(C, \mathcal{T})$  is defined as follows:

$$\text{sub}(C) \cup \bigcup_{A \doteq D \in \mathcal{T}} (\{A, \neg A\} \cup \text{sub}(D) \cup \text{sub}(\neg D)) \cup \bigcup_{D \sqsubseteq E \in \mathcal{T}} \text{sub}(\neg D \sqcup E)$$

A set  $H \subseteq \text{sub}(C, \mathcal{T})$  is called a *Hintikka set for  $C$*  if the following three conditions are satisfied: if  $D \sqcap E \in H$ , then  $\{D, E\} \subseteq H$ ; if  $D \sqcup E \in H$ , then  $\{D, E\} \cap H \neq \emptyset$ ; and there is no concept name  $A$  with  $\{A, \neg A\} \subseteq H$ .

For a TBox  $\mathcal{T}$ , a Hintikka set  $H$  is called  $\mathcal{T}$ -*expanded* if for every GCI  $D \sqsubseteq E \in \mathcal{T}$ , it holds that  $\neg D \sqcup E \in H$ , and for every concept definition  $A \doteq D \in \mathcal{T}$ , it holds that, if  $A \in H$  then  $D \in H$ , and if  $\neg A \in H$  then  $\neg D \in H$ .<sup>3</sup>

Hintikka trees for  $C$  and  $\mathcal{T}$  are infinite trees of a fixed arity  $k$ , which is determined by the number of existential restrictions, i.e. concepts of the form  $\exists r.D$ , in  $\text{sub}(C, \mathcal{T})$ . For a positive integer  $k$ , we denote the set  $\{1, \dots, k\}$  by  $K$ . The nodes of a  $k$ -ary tree can be denoted by the elements of  $K^*$ , with the empty word  $\varepsilon$  denoting the root, and  $ui$  the  $i$ th successor of  $u$ . In the case of labelled trees, we will refer to the label of the node  $u$  in the tree  $t$  by  $t(u)$ .

In the definition of Hintikka trees, we need to know which successor in the tree corresponds to which existential restriction. For this purpose, we fix a linear order on the existential restrictions in  $\text{sub}(C, \mathcal{T})$ . Let  $\varphi : \{\exists r.D \in \text{sub}(C, \mathcal{T})\} \rightarrow K$  be the corresponding ordering function, i.e.  $\varphi(\exists r.D)$  determines the successor node corresponding to  $\exists r.D$ . In general, such a successor node need not exist in a tree model. To obtain a full  $k$ -ary tree, Hintikka trees contain appropriate dummy nodes.

For technical reasons, which will become clear later on, the nodes of the Hintikka trees defined below are not simply labelled by Hintikka sets, but by quadruples  $(\Gamma, \Pi, \Omega, \rho)$ , where  $\rho$  is the role which connects the node with the father node,  $\Omega$  is the complete Hintikka set for the node,  $\Gamma \subseteq \Omega$  consists of the unique concept  $D$  contained in  $\Omega$  because of an existential restriction  $\exists \rho.D$  in the father node, and  $\Pi$  contains only those concepts that are contained in  $\Omega$  because of universal restrictions  $\forall \rho.E$  in the father node. We will use a special new role name  $\lambda$  for nodes that are not connected to the father node by a role,

<sup>3</sup> We will refer to this technique of handling concept definitions as *lazy unfolding*. Note that, in contrast to GCIs, concept definitions are only applied if  $A$  or  $\neg A$  is explicitly present in  $H$ .

i.e. the root node and those (dummy) nodes which are labelled with an empty set of concepts.

**Definition 2 (Hintikka trees).** The tuple  $((\Gamma_0, \Pi_0, \Omega_0, \varrho_0), (\Gamma_1, \Pi_1, \Omega_1, \varrho_1), \dots, (\Gamma_k, \Pi_k, \Omega_k, \varrho_k))$  is called  $C, \mathcal{T}$ -compatible if, for all  $i, 0 \leq i \leq k$ ,  $\Gamma_i \cup \Pi_i \subseteq \Omega_i$ ,  $\Omega_i$  is a  $\mathcal{T}$ -expanded Hintikka set, and the following holds for every existential concept  $\exists r.D \in \text{sub}(C, \mathcal{T})$ :

- if  $\exists r.D \in \Omega_0$ , then
  1.  $\Gamma_{\varphi(\exists r.D)}$  consists of  $D$ ;
  2.  $\Pi_{\varphi(\exists r.D)}$  consists of all concepts  $E$  for which there is a universal restriction  $\forall r.E \in \Omega_0$ , and it additionally contains  $\forall r.E$  if  $\text{trans}(r)$ ;
  3. for every concept  $\forall \bar{r}.F \in \Omega_{\varphi(\exists r.D)}$ ,  $\Omega_0$  contains  $F$ , and additionally  $\forall \bar{r}.F$  if  $\text{trans}(r)$ ;
  4.  $\varrho_{\varphi(\exists r.D)} = r$ ;
- if  $\exists r.D \notin \Omega_0$ , then  $\Gamma_{\varphi(\exists r.D)} = \Pi_{\varphi(\exists r.D)} = \Omega_{\varphi(\exists r.D)} = \emptyset$  and  $\varrho_{\varphi(\exists r.D)} = \lambda$ .

A  $k$ -ary tree  $t$  is called a *Hintikka tree for  $C$  and  $\mathcal{T}$*  if, for every node  $v \in K^*$ , the tuple  $(t(v), t(v1), \dots, t(vk))$  is  $C, \mathcal{T}$ -compatible, and  $t(\varepsilon)$  has empty  $\Gamma$ - and  $\Pi$ -components, an  $\Omega$ -component containing  $C$ , and  $\lambda$  as its  $\varrho$ -component.

Our definition of a Hintikka tree ensures that the existence of such a tree characterises satisfiability of  $\mathcal{SI}$  concepts. It basically combines the technique for handling transitive and inverse roles introduced in [10]<sup>4</sup> with the technique for dealing with acyclic TBoxes employed in [8].

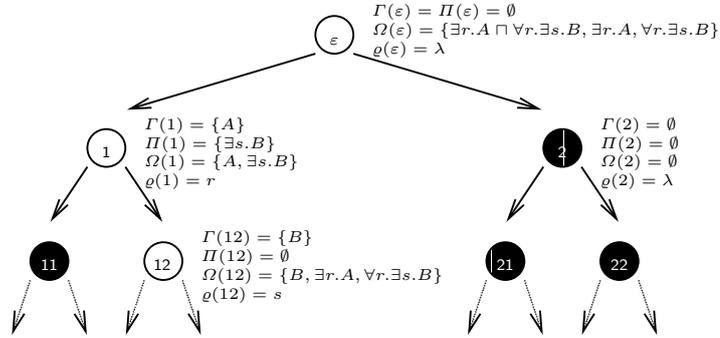


Fig. 1. A Hintikka tree for  $\exists r.A \sqcap \forall r.\exists s.B$ .

**Example 3.** Figure 1 shows a Hintikka tree for the concept  $\exists r.A \sqcap \forall r.\exists s.B$ , where  $r$  is a transitive role. Since there are two existential subconcepts, the model is a binary tree, and we assume that the first son stands for the concept

<sup>4</sup> there used in the context of tableau-based algorithms.

$\exists r.A$  and the second one for  $\exists s.B$ . Node number 1 is labelled with  $A$ ,  $\exists s.B$  and, since  $r$  is transitive, also with  $\forall r.\exists s.B$ . Node number 2 is a dummy node (all dummy nodes are shown as black). The label of node 12 has to contain  $B$ , but the  $\Omega$  set can contain any other concept that does not violate the conditions for Hintikka sets, e.g. those shown in the label. Thus, in this case, node 121 cannot be a dummy node, but it has to contain  $A$  and  $\exists s.B$ .

**Theorem 4.** The  $\mathcal{ST}$  concept  $C$  is satisfiable w.r.t. the general TBox  $\mathcal{T}$  iff there exists a Hintikka tree for  $C$  and  $\mathcal{T}$ .

### 3 Tree automata

The existence of a Hintikka tree can be decided with the help of so-called looping automata, i.e. automata on infinite trees without a special acceptance condition. After introducing these automata, we will first show how they can be used to decide satisfiability in  $\mathcal{ST}$  w.r.t. general TBoxes in exponential time. Then we will introduce a restricted class of looping automata, and use it to show that satisfiability in  $\mathcal{ST}$  w.r.t. acyclic TBoxes can be decided in polynomial space.

#### 3.1 Looping automata

The following definition of looping tree automata does not include an alphabet for labelling the nodes of the trees. In fact, when deciding the emptiness problem for such automata, only the *existence* of a tree accepted by the automaton is relevant, and not the labels of its nodes. Since all information relevant for the existence is contained in the states of the automaton, the node labels are redundant. For our reduction this implies that the automaton we construct for a given input  $C, \mathcal{T}$  does not actually accept the Hintikka trees for  $C, \mathcal{T}$ . Instead, it attempts to label the unlabelled tree as a Hintikka tree for the input. If  $C$  is satisfiable w.r.t.  $\mathcal{T}$ , then the automaton accepts the unlabelled tree and its *successful runs* are the Hintikka trees for  $C$  and  $\mathcal{T}$ .

**Definition 5 (Automaton, run).** A *looping tree automaton* over  $k$ -ary trees is a tuple  $(Q, \Delta, I)$ , where  $Q$  is a finite set of states,  $\Delta \subseteq Q^{k+1}$  is the transition relation, and  $I \subseteq Q$  is the set of initial states.

A *run* of this automaton on the (unique) unlabelled  $k$ -ary tree  $t$  is a labelled  $k$ -ary tree  $r : K^* \rightarrow Q$  such that  $(r(v), r(v1), \dots, r(vk)) \in \Delta$  for all  $v \in K^*$ . The run is *successful* if  $r(\varepsilon) \in I$ . The *emptiness problem for looping tree automata* is the problem of deciding whether a given looping tree automaton has a successful run or not.

In order to *decide the emptiness problem* in time polynomial in the size of the automaton, one computes the set of all bad states, i.e. states that do not occur in any run, in a *bottom-up* manner [18, 5]: states that do not occur as first component in the transition relation are bad, and if all transitions that have the state  $q$  as first component contain a state already known to be bad, then  $q$  is

also bad. The automaton has a successful run iff there is an initial state that is not bad.

For an  $\mathcal{SI}$  concept  $C$  and a general TBox  $\mathcal{T}$ , we can construct a looping tree automaton whose successful runs are exactly the Hintikka trees for  $C$  and  $\mathcal{T}$  as follows.

**Definition 6 (Automaton  $\mathcal{A}_{C,\mathcal{T}}$ ).** For an  $\mathcal{SI}$  concept  $C$  and a TBox  $\mathcal{T}$ , let  $k$  be the number of existential restrictions in  $\text{sub}(C, \mathcal{T})$ . Then the looping automaton  $\mathcal{A}_{C,\mathcal{T}} = (Q, \Delta, I)$  is defined as follows:

- $Q$  consists of all 4-tuples  $(\Gamma, \Pi, \Omega, \varrho)$  such that  $\Gamma \cup \Pi \subseteq \Omega \subseteq \text{sub}(C, \mathcal{T})$ ,  $\Gamma$  is a singleton set,  $\Omega$  is a  $\mathcal{T}$ -expanded Hintikka set for  $C$ , and  $\varrho$  occurs in  $C$  or  $\mathcal{T}$  or is equal to  $\lambda$ ;
- $\Delta$  consists of all  $C, \mathcal{T}$ -compatible tuples  $((\Gamma_0, \Pi_0, \Omega_0, \varrho_0), (\Gamma_1, \Pi_1, \Omega_1, \varrho_1), \dots, (\Gamma_k, \Pi_k, \Omega_k, \varrho_k))$ ;
- $I := \{(\emptyset, \emptyset, \Omega, \lambda) \in Q \mid C \in \Omega\}$ .

**Lemma 7.**  $\mathcal{A}_{C,\mathcal{T}}$  has a successful run iff  $C$  is satisfiable w.r.t.  $\mathcal{T}$ .

Since the cardinality of  $\text{sub}(C, \mathcal{T})$  and the size of each of its elements is linear in the size of  $C, \mathcal{T}$ , the size of the automaton  $\mathcal{A}_{C,\mathcal{T}}$  is exponential in the size of  $C, \mathcal{T}$ . Together with the fact that the emptiness problem for looping tree automata can be decided in polynomial time, this yields:

**Theorem 8.** Satisfiability in  $\mathcal{SI}$  w.r.t. general TBoxes is in EXPTIME.

This complexity upper-bound is optimal since EXPTIME-hardness follows from the known hardness result for  $\mathcal{ALC}$  with general TBoxes [16].

One could also try to solve the emptiness problem by constructing a successful run in a *top-down manner*: label the root with an element  $q_0$  of  $I$ , then apply a transition with first component  $q_0$  to label the successor nodes, etc. There are, however, two problems with this approach. Firstly, it yields a *non-deterministic* algorithm since  $I$  may contain more than one element, and in each step more than one transition may be applicable. Secondly, one must employ an appropriate cycle-checking technique (similar to blocking in tableau-based algorithms) to obtain a terminating algorithm. Applied to the automaton  $\mathcal{A}_{C,\mathcal{T}}$ , this approach would at best yield a (non-optimal) NEXPTIME satisfiability test.

### 3.2 Blocking-invariant automata

In order to obtain a PSPACE result for satisfiability w.r.t. *acyclic* TBoxes, we use the top-down emptiness test sketched above. In fact, in this case non-determinism is unproblematic since NPSpace is equal to PSPACE by Savitch’s theorem [15]. The advantage of the top-down over the bottom-up emptiness test is that it is not necessary to construct the whole automaton before applying the emptiness test. Instead, the automaton can be constructed on-the-fly. However, we still need to deal with the termination problem. For this purpose, we adapt the blocking technique known from the tableau-based approach. In the following, when we speak about a *path* in a  $k$ -ary tree, we mean a sequence of nodes  $v_1, \dots, v_m$  such that  $v_1$  is the root  $\varepsilon$  and  $v_{i+1}$  is a direct successor of  $v_i$ .

**Definition 9** ( $\leftarrow$ -invariant,  $m$ -blocking). Let  $\mathcal{A} = (Q, \Delta, I)$  be a looping tree automaton and  $\leftarrow$  be a binary relation over  $Q$ , called the *blocking relation*. If  $q \leftarrow p$ , then we say that  $q$  is *blocked* by  $p$ . The automaton  $\mathcal{A}$  is called  $\leftarrow$ -invariant if, for every  $q \leftarrow p$ , and  $(q_0, q_1, \dots, q_{i-1}, q, q_{i+1}, \dots, q_k) \in \Delta$ , it holds that  $(q_0, q_1, \dots, q_{i-1}, p, q_{i+1}, \dots, q_k) \in \Delta$ .

A  $\leftarrow$ -invariant automaton  $\mathcal{A}$  is called  $m$ -blocking if, for every successful run  $r$  of  $\mathcal{A}$  and every path  $v_1, \dots, v_m$  of length  $m$  in  $r$ , there are  $1 \leq i < j \leq m$  such that  $r(v_j) \leftarrow r(v_i)$ .

Obviously, any looping automaton  $\mathcal{A} = (Q, \Delta, I)$  is  $=$ -invariant (i.e. the blocking relation is equality) and  $m$ -blocking for every  $m > \#Q$  (where “ $\#Q$ ” denotes the cardinality of  $Q$ ). However, we are interested in automata and blocking relations where blocking occurs earlier than after a linear number of transitions.

To test an  $m$ -blocking automaton for emptiness, it is sufficient to construct partial runs of depth  $m$ . More formally, we define  $K^{\leq n} := \bigcup_{i=0}^n K^i$ . A *partial run of depth  $m$*  is a mapping  $r : K^{\leq m-1} \rightarrow Q$  such that  $(r(v), r(v_1), \dots, r(v_k)) \in \Delta$  for all  $v \in K^{\leq m-2}$ . It is *successful* if  $r(\varepsilon) \in I$ .

**Lemma 10.** An  $m$ -blocking automaton  $\mathcal{A} = (Q, \Delta, I)$  has a successful run iff it has a successful partial run of depth  $m$ .

```

1: if  $I \neq \emptyset$  then
2:   guess an initial state  $q \in I$ 
3: else
4:   return “empty”
5: if there is a transition from  $q$  then
6:   guess such a transition  $(q, q_1, \dots, q_k) \in \Delta$ 
7:   push(SQ,  $(q_1, \dots, q_k)$ ), push(SN, 0)
8: else
9:   return “empty”
10: while SN is not empty do
11:    $(q_1, \dots, q_k) := \text{pop}(\text{SQ}), n := \text{pop}(\text{SN}) + 1$ 
12:   if  $n \leq k$  then
13:     push(SQ,  $(q_1, \dots, q_k)$ ), push(SN,  $n$ )
14:     if length(SN) <  $m - 1$  then
15:       if there is a transition from  $q_n$  then
16:         guess a transition  $(q_n, q'_1, \dots, q'_k) \in \Delta$ 
17:         push(SQ,  $(q'_1, \dots, q'_k)$ ), push(SN, 0)
18:       else
19:         return “empty”
20: return “not empty”

```

**Fig. 2.** The non-deterministic top-down emptiness test for  $m$ -blocking automata.

For  $k > 1$ , the size of a successful partial run of depth  $m$  is still exponential in  $m$ . However, when checking for the existence of such a run, one can perform

a depth-first traversal of the run while constructing it. To do this, it is basically enough to have at most one path of length up to  $m$  in memory.<sup>5</sup> The algorithm that realizes this idea is shown in Figure 2. It uses two stacks: the stack  $SQ$  stores, for every node on the current path, the right-hand side of the transition which led to this node, and the stack  $SN$  stores, for every node on the current path, on which component of this right-hand side we are currently working. The entries of  $SQ$  and  $SN$  are elements of  $Q^k$  and  $K \cup \{0\}$ , respectively, and the number of entries is bounded by  $m$  for each stack.

Note that the algorithm does not require the automaton  $\mathcal{A}$  to be explicitly given. It can be constructed on-the-fly during the run of the algorithm.

**Definition 11.** Assume that we have a set of inputs  $\mathcal{I}$  and a construction that yields, for every  $i \in \mathcal{I}$ , an  $m_i$ -blocking automaton  $\mathcal{A}_i = (Q_i, \Delta_i, I_i)$  working on  $k_i$ -ary trees. We say that this construction is a PSPACE *on-the-fly construction* if there is a polynomial  $P$  such that, for every input  $i$  of size  $n$  we have

- $m_i \leq P(n)$  and  $k_i \leq P(n)$ ;
- every element of  $Q_i$  is of a size bounded by  $P(n)$ ;
- one can non-deterministically guess in time bounded by  $P(n)$  an element of  $I_i$  and, for a state  $q \in Q_i$ , a transition from  $\Delta_i$  with first component  $q$ .

The algorithms guessing an initial state (a transition starting with  $q$ ) are assumed to yield the answer “no” if there is no initial state (no such transition).

The following theorem is an easy consequence of the correctness of the top-down emptiness test described in Figure 2 and Savitch’s theorem [15].

**Theorem 12.** If the automata  $\mathcal{A}_i$  are obtained from the inputs  $i \in \mathcal{I}$  by a PSPACE on-the-fly construction, then the emptiness problem for  $\mathcal{A}_i$  can be decided by a deterministic algorithm in space polynomial in the size of  $i$ .

#### 4 Satisfiability in $\mathcal{ST}$ w.r.t. acyclic TBoxes

It is easy to see that the construction of the automaton  $\mathcal{A}_{C,\mathcal{T}}$  from a given  $\mathcal{ST}$  concept  $C$  and a general TBox  $\mathcal{T}$  satisfies all but one of the conditions of a PSPACE on-the-fly construction. The condition that is violated is the one requiring that blocking must occur after a polynomial number of steps. In the case of general TBoxes, this is not surprising since we know that the satisfiability problem is EXPTIME-hard. Unfortunately, this condition is also violated if  $\mathcal{T}$  is an acyclic TBox. The reason is that successor states may contain new concepts that are not really required by the definition of  $C, \mathcal{T}$ -compatible tuples, but are also not prevented by this definition, like  $\exists r.A$  and  $\forall r.\exists s.B$  in the label of node 12 in Example 3. In the case of acyclic TBoxes, we can construct a subautomaton that avoids such unnecessary concepts. It has fewer runs than  $\mathcal{A}_{C,\mathcal{T}}$ , but it does have a successful run whenever  $\mathcal{A}_{C,\mathcal{T}}$  has one. The construction of this subautomaton follows the following general pattern.

<sup>5</sup> This is similar to the so-called trace technique for tableau-based algorithms [17].

**Definition 13 (Faithful).** Let  $\mathcal{A} = (Q, \Delta, I)$  be a looping tree automaton on  $k$ -ary trees. The family of functions  $f_q : Q \rightarrow Q^S$  for  $q \in Q^S$  is *faithful* w.r.t.  $\mathcal{A}$  if  $I \subseteq Q^S \subseteq Q$ , and the following two conditions are satisfied for every  $q \in Q^S$ :

1. if  $(q, q_1, \dots, q_k) \in \Delta$ , then  $(q, f_q(q_1), \dots, f_q(q_k)) \in \Delta$ ;
2. if  $(q_0, q_1, \dots, q_k) \in \Delta$ , then  $(f_q(q_0), f_q(q_1), \dots, f_q(q_k)) \in \Delta$ .<sup>6</sup>

The *subautomaton*  $\mathcal{A}^S = (Q^S, \Delta^S, I)$  of  $\mathcal{A}$  induced by this family has the transition relation  $\Delta^S := \{(q, f_q(q_1), \dots, f_q(q_k)) \mid (q, q_1, \dots, q_k) \in \Delta \text{ and } q \in Q^S\}$ .

Instead of testing  $\mathcal{A}$  for emptiness, we can equivalently test  $\mathcal{A}^S$ .

**Lemma 14.** Let  $\mathcal{A}$  be a looping tree automaton and  $\mathcal{A}^S$  its subautomaton induced by the faithful family of functions  $f_q : Q \rightarrow Q^S$  for  $q \in Q^S$ . Then  $\mathcal{A}$  has a successful run iff  $\mathcal{A}^S$  has a successful run.

Intuitively, the range of  $f_q$  contains the states that are allowed after state  $q$  has been reached. Before we can define an appropriate family of functions for  $\mathcal{A}_{C, \mathcal{T}}$ , we must introduce some notation. For an  $\mathcal{ST}$  concept  $C$  and an acyclic TBox  $\mathcal{T}$ , the *role depth*  $\text{rd}_{\mathcal{T}}(C)$  of  $C$  w.r.t.  $\mathcal{T}$  is the maximal nesting of (universal and existential) role restrictions in the concept obtained by expanding  $C$  w.r.t.  $\mathcal{T}$ . Obviously,  $\text{rd}_{\mathcal{T}}(C)$  is polynomially bounded by the size of  $C, \mathcal{T}$ . For a set of  $\mathcal{ST}$  concepts  $S$ , its role depth  $\text{rd}_{\mathcal{T}}(S)$  w.r.t.  $\mathcal{T}$  is the maximal role depth w.r.t.  $\mathcal{T}$  of the elements of  $S$ . We define  $\text{sub}_{\leq n}(C, \mathcal{T}) := \{D \mid D \in \text{sub}(C, \mathcal{T}) \text{ and } \text{rd}_{\mathcal{T}}(D) \leq n\}$ , and  $S/r := \{D \in S \mid \text{there is an } E \text{ such that } D = \forall r.E\}$ .

The main idea underlying the next definition is the following. If  $\mathcal{T}$  is acyclic, then, since we use lazy unfolding of concept definitions, the definition of  $C, \mathcal{T}$ -compatibility requires, for a transition  $(q, q_1, \dots, q_k)$  of  $\mathcal{A}_{C, \mathcal{T}}$ , only the existence of concepts in  $q_i = (I_i, II_i, \Omega_i, \varrho_i)$  that are of a smaller depth than the maximal depth  $n$  of concepts in  $q$  if  $\varrho_i$  is not transitive. If  $\varrho_i$  is transitive, then  $II_i$  may also contain universal restrictions of depth  $n$ .

We can therefore remove from the states  $q_i$  all concepts with a higher depth and still maintain  $C, \mathcal{T}$ -compatibility.

**Definition 15 (Functions  $f_q$ ).** For two states  $q = (I, II, \Omega, \varrho)$  and  $q' = (I', II', \Omega', \varrho')$  of  $\mathcal{A}_{C, \mathcal{T}}$  with  $\text{rd}_{\mathcal{T}}(\Omega) = n$ , we define the function  $f_q(q')$  as follows:

- if  $\text{rd}_{\mathcal{T}}(I') \geq \text{rd}_{\mathcal{T}}(\Omega)$ , then  $f_q(q') := (\emptyset, \emptyset, \emptyset, \lambda)$ ;
- otherwise,  $f_q(q') := (I', II'', \Omega'', \varrho')$ , where
  - $P = \text{sub}_{\leq n}(C, \mathcal{T})/\varrho'$ , if  $\text{trans}(\varrho')$ ; otherwise  $P = \emptyset$ ;
  - $II'' = II' \cap (\text{sub}_{\leq n-1}(C, \mathcal{T}) \cup P)$ ;
  - $\Omega'' = \Omega' \cap (\text{sub}_{\leq n-1}(C, \mathcal{T}) \cup II'')$ .

The definition of  $II''$  implies that we remove from  $II'$  all concepts which have a higher depth than the maximum depth in  $\Omega$ , and we allow for a concept of the same depth as in  $\Omega$  only if it has the shape  $\forall \varrho'.E$  and  $\varrho'$  is transitive. If  $\mathcal{T}$

<sup>6</sup> Note that this condition does neither imply nor follow from condition 1, since  $q_0$  need not be equal to  $q$ , and it is not required that  $f_q(q)$  equals  $q$ .

is acyclic, then the set  $\Omega''$  defined above is still a  $\mathcal{T}$ -expanded Hintikka set after removing these concepts. Looking back at Example 3, this means that node 12 could *not* contain any existential or value restrictions because node 1 has role depth 1 and  $s$  is not transitive. Consequently, all sons of node 12 have to be dummy nodes.

**Lemma 16.** The family of mappings  $f_q$  (for states  $q$  of  $\mathcal{A}_{C,\mathcal{T}}$ ) introduced in Definition 15 is faithful w.r.t.  $\mathcal{A}_{C,\mathcal{T}}$ .

Consequently,  $\mathcal{A}_{C,\mathcal{T}}$  has a successful run iff the induced subautomaton  $\mathcal{A}_{C,\mathcal{T}}^S$  has a successful run.

**Lemma 17.** The construction of  $\mathcal{A}_{C,\mathcal{T}}^S$  from an input consisting of an  $\mathcal{SI}$  concept  $C$  and an acyclic TBox  $\mathcal{T}$  is a PSPACE on-the-fly construction.

The main thing to show <sup>7</sup> in the proof is that blocking always occurs after a polynomial number of steps. To show this, we use the following blocking relation:  $(\Gamma_1, \Pi_1, \Omega_1, \varrho_1) \leftarrow_{\mathcal{SI}} (\Gamma_2, \Pi_2, \Omega_2, \varrho_2)$  if  $\Gamma_1 = \Gamma_2$ ,  $\Pi_1 = \Pi_2$ ,  $\Omega_1/\bar{\varrho}_1 = \Omega_2/\bar{\varrho}_2$ , and  $\varrho_1 = \varrho_2$ . If  $m := \#\text{sub}(C, \mathcal{T})$ , then  $\mathcal{A}_{C,\mathcal{T}}^S$  is  $m^4$ -blocking w.r.t.  $\leftarrow_{\mathcal{SI}}$ . The main reasons for this to hold are the following: (i) if a successor node is reached w.r.t. a non-transitive role, then the role depth of the  $\Omega$ -component decreases, and the same is true if within two steps two different transitive roles are used; (ii) if a successor node is reached w.r.t. a transitive role, then there is an inclusion relationship between the  $\Pi$ -components of the successor node and its father; the same is true (though in the other direction) for the  $\Omega/\bar{\varrho}$ -components.

Since we know that  $C$  is satisfiable w.r.t.  $\mathcal{T}$  iff  $\mathcal{A}_{C,\mathcal{T}}$  has a successful run iff  $\mathcal{A}_{C,\mathcal{T}}^S$  has a successful run, Theorem 12 yields the desired PSPACE upper-bound. PSPACE-hardness for this problem follows directly from the known PSPACE-hardness of satisfiability w.r.t. the empty TBox in  $\mathcal{ALC}$  [17].

**Theorem 18.** Satisfiability in  $\mathcal{SI}$  w.r.t. acyclic TBoxes is PSPACE-complete.

## 5 Conclusion

We have developed a framework for automata that adapts the notion of *blocking* from tableau algorithms and makes it possible to show tight complexity bounds for PSPACE logics using the automata approach. In order to achieve this result, we replace the deterministic bottom-up emptiness test with a nondeterministic top-down test that can be interleaved with the construction of the automaton and aborted after a “blocked” state is reached. If the number of transitions before this happens is polynomial in the size of the input, emptiness of the automaton can be tested using space polynomial in the size of the input rather than time exponential in the size of the input. This illustrates the close relationship between tableau and automata algorithms.

As an example for an application of this method, we have shown how blocking automata can be used to decide satisfiability of  $\mathcal{SI}$  concepts w.r.t. acyclic TBoxes in PSPACE.

<sup>7</sup> A detailed proof can be found in [3].

## References

- [1] F. Baader, H.-J. Bürckert, B. Hollunder, W. Nutt, and J. H. Siekmann. Concept logics. In J. W. Lloyd, editor, *Computational Logics, Symposium Proceedings*, pages 177–201. Springer-Verlag, 1990.
- [2] F. Baader, J. Hladik, C. Lutz, and F. Wolter. From tableaux to automata for description logics. *Fundamenta Informaticae*, 57(2–4):247–279, 2003.
- [3] F. Baader, J. Hladik, and R. Peñaloza. PSPACE automata with blocking for description logics. LTCS-Report 06-04, Chair for Automata Theory, Dresden University of Technology, 2006. See <http://lat.inf.tu-dresden.de/research/reports.html>. A short version of this report is to appear in the proceedings of LATA’07.
- [4] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.
- [5] F. Baader and S. Tobies. The inverse method implements the automata approach for modal satisfiability. In *Proceedings of IJCAR-01*, volume 2083 of *LNAI*, pages 92–106. Springer-Verlag, 2001.
- [6] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.
- [7] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proceedings of IJCAI-99*, pages 84–89, 1999.
- [8] J. Hladik and R. Peñaloza. PSPACE automata for description logics. In B. Parsia, U. Sattler, and D. Toman, editors, *Proc. of DL’06*, volume 189 of *CEUR-WS*, 2006.
- [9] I. Horrocks and P. F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.
- [10] I. Horrocks, U. Sattler, and S. Tobies. A PSpace-algorithm for deciding  $ACCNT_{R^+}$ -satisfiability. LTCS-Report 98-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1998.
- [11] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of LPAR ’99*, number 1705 in *LNAI*. Springer-Verlag, 1999.
- [12] C. Lutz. Complexity of terminological reasoning revisited. In *Proceedings of LPAR ’99*, volume 1705 of *LNAI*, pages 181–200. Springer-Verlag, 1999.
- [13] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
- [14] U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *Proceedings of KI’96*, volume 1137 of *LNAI*, pages 333–345. Springer-Verlag, 1996.
- [15] W. J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. of Computer and System Science*, 4:177–192, 1970.
- [16] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of IJCAI-91*, pages 466–471, 1991.
- [17] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
- [18] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Science*, 32:183–221, 1986.
- [19] A. Voronkov. How to optimize proof-search in modal logics: new methods of proving redundancy criteria for sequent calculi. *ACM transactions on computational logic*, 2(2), 2001.

## Actions and Programs over Description Logic Ontologies

Diego Calvanese<sup>1</sup>, Giuseppe De Giacomo<sup>2</sup>, Maurizio Lenzerini<sup>2</sup>, Riccardo Rosati<sup>2</sup>

<sup>1</sup> Faculty of Computer Science  
Free University of Bozen-Bolzano  
Piazza Domenicani 3,  
I-39100 Bolzano, Italy  
calvanese@inf.unibz.it

<sup>2</sup> Dip. di Informatica e Sistemistica  
Università di Roma “La Sapienza”  
Via Salaria 113,  
I-00198 Roma, Italy  
lastname@dis.uniroma1.it

**Abstract.** We aim at representing and reasoning about actions and (high level) programs over ontologies expressed in Description Logics. This is a critical issue that has resisted good solutions for a long time. In particular, while well-developed theories of actions and high-level programs exist in AI, e.g., the ones based on SitCalc, these theories do not apply smoothly to Description Logic ontologies, due to the profoundly non-definitorial nature of such ontologies (cf. cyclic TBoxes). Here we propose a radical solution: we assume a functional view of ontologies and see them as systems that allow for two kinds of operations: ASK, which returns the (certain) answer to a query, and TELL, which produces a new ontology as a result of the application of an atomic action. We base atomic actions on instance level update and instance level erasure on the ontology. Building on this functional view, we introduce Golog/ConGolog-like high-level programs on ontologies. This paper demonstrates the effectiveness of the approach in general, and presents the following specific results: we characterize the notion of single-step executability of such programs, devise methods for reasoning about sequences of actions, and present (nice) complexity results in the case where the ontology is expressed in DL-Lite.

### 1 Introduction

Description Logics (DLs) [1] are generally advocated as the right tool to express ontologies, and this belief is one of the cornerstones of the Semantic Web [31, 15]. Notably, semantic web services [22] constitute another cornerstone of the Semantic Web. These are essentially high-level descriptions of computations that abstract from the technological issues of the actual programs that realize them. An obvious concern is to combine in some way the static descriptions of the information provided by ontologies with the dynamic descriptions of the computations provided by semantic web services. Interestingly, such a critical issue has resisted good solutions for a long time. Indeed even big efforts such as OWL-S [22] have not really succeeded.

In AI, the importance of combining static and dynamic knowledge has been recognized early [23, 24]. By now, well developed theories of actions and high level programs exist in AI, e.g., the ones based on Reiter’s variant of SitCalc [28]. Note that high-level programs [19, 10] share with semantic web services the emphasis on abstracting from

the technological issues of actual programs, and are indeed abstract descriptions of computations over a domain of interest.

Unfortunately, these theories do not apply smoothly to DL ontologies, due to the profoundly non-definitorial nature of such ontologies. Indeed, concepts and roles expressions in a DL do not provide *definitions* of concepts and roles in general, but usually only describe interrelations between them (cf. cyclic TBox interpreted according to the usual descriptive semantics [1]). Such non-definitorial nature of DL ontologies makes them one of the most difficult kinds of domain descriptions for reasoning about actions [2, 20].

Here we propose a radical solution: we assume a functional view [17] of ontologies and see them as systems that allow for two kinds of operations: ASK, which returns the (certain) answer to a query, and TELL, which produces a new ontology as a result of the application of an atomic action. Observe that this approach, whose origins come from [7, 13, 25, 32], has some subtle limitations, due to the fact that we lose the possibility of distinguishing between “knowledge” and “truth” as pointed out in [30]. On the other hand, it has a major advantage: it decouples reasoning on the static knowledge from the one on the dynamics of the computations over such knowledge. As a result, we gain the ability of lifting to DLs many of the results developed in reasoning about actions in the years.

We demonstrate such an approach in this paper. Specifically, we base atomic actions used by the TELL operation on instance level update and instance level erasure on the ontology [8, 9]. Building on this functional view, we introduce Golog/ConGolog-like high level programs over ontologies. We characterize the notion of single-step executability of such programs, devise methods for reasoning about sequences of actions, and present (nice) complexity results in the case where the ontology is expressed in DL-Lite. We stress that this paper is really an illustration of what a functional view on ontologies can bring about in combining static and dynamic aspects in the context of DL ontologies, and that many extensions of this work can be investigated (we will mention some of them in the conclusions).

## 2 Preliminaries

**DL ontologies.** Description Logics (DLs) [1] are knowledge representation formalisms that are tailored for representing the domain of interest in terms of *concepts* (or classes), which denote sets of objects, and *roles* (or relations), which denote binary relations between objects. DLs *ontologies* (aka knowledge bases) are formed by two distinct parts: the so-called *TBox*, which represents the *intensional level* of the ontology, and contains an intensional description of the domain of interest; and the so-called *ABox*, which represents the *instance level* of the ontology, and contains extensional information.

We give the semantics of a DL ontology in terms of interpretations over a fixed infinite domain  $\Delta$  of objects. We assume to have a constant for each object in  $\Delta$  denoting exactly that object. In this way we blur the distinction between constants and objects, so that we can use them interchangeably (with a little abuse of notation) without causing confusion (cf. standard names [18]).

An interpretation  $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$  consists of a first order structure over  $\Delta$ , where  $\cdot^{\mathcal{I}}$  is the interpretation function, i.e., a function mapping each concept to a subset of  $\Delta$  and each role to a subset of  $\Delta \times \Delta$ . We say that  $\mathcal{I}$  is a *model of a (TBox or ABox) assertion*  $\alpha$ , or also that  $\mathcal{I}$  *satisfies*  $\alpha$ , if  $\alpha$  is true in  $\mathcal{I}$ . We say that  $\mathcal{I}$  is a *model of the ontology*  $s = \langle \mathcal{T}, \mathcal{A} \rangle$ , or also that  $\mathcal{I}$  *satisfies the ontology*  $s$ , if  $\mathcal{I}$  is a model of all the assertions in  $\mathcal{T}$  and  $\mathcal{A}$ . Given a set  $\mathcal{S}$  of (TBox or ABox) assertions, we denote as  $Mod(\mathcal{S})$  the set of interpretations that are models of all assertions in  $\mathcal{S}$ . In particular, the set of *models of an ontology*  $s$ , denoted as  $Mod(s)$ , is the set of models of all assertions in  $\mathcal{T}$  and  $\mathcal{A}$ , i.e.,  $Mod(s) = Mod(\langle \mathcal{T}, \mathcal{A} \rangle) = Mod(\mathcal{T} \cup \mathcal{A})$ . An ontology  $s$  is *consistent* if  $Mod(s) \neq \emptyset$ , i.e., it has at least one model. We say that an ontology  $s$  *logically implies* an expression  $\alpha$  (e.g., an assertion, an instantiated union of conjunctive queries, etc.), written  $s \models \alpha$ , if for every interpretation  $\mathcal{I} \in Mod(s)$ , we have  $\mathcal{I} \in Mod(\alpha)$ , i.e., all the models of  $s$  are also models of  $\alpha$ . When dealing with queries, we are interested in *query answering* (for CQs and UCQs): given an ontology  $s$  and a query  $q(\mathbf{x})$  over  $s$ , return the *certain answers* to  $q(\mathbf{x})$  over  $s$ , i.e., all tuples  $\mathbf{t}$  of elements of  $\Delta^{\mathcal{I}}$  such that, when substituted to  $\mathbf{x}$  in  $q(\mathbf{x})$ , we have that  $s \models q(\mathbf{t})$ .

**DL-Lite<sub>F</sub>.** In this paper, we focus on a particular DL, namely *DL-Lite<sub>F</sub>*, belonging to the *DL-Lite* family [4, 5] of DLs, which are tailored towards capturing conceptual modeling constructs (such as those typical of UML Class Diagrams or Entity-Relationship Diagrams), while keeping reasoning, including conjunctive query answering, tractable and first-order reducible (i.e., LOGSPACE in data complexity). In *DL-Lite<sub>F</sub>*, which is the logic originating the whole *DL-Lite* family, concepts are defined as follows:

$$B ::= A \mid \exists R \qquad C ::= B \mid \neg B \qquad R ::= P \mid P^-$$

where  $A$  denotes an atomic concept,  $P$  an atomic role,  $B$  a *basic concept*, and  $C$  a general concept. A basic concept can be either an atomic concept, a concept of the form  $\exists P$ , i.e. the standard DL construct of unqualified existential quantification on roles, or a concept of the form  $\exists P^-$ , which involves *inverse roles*. A *DL-Lite<sub>F</sub>* TBox is a finite set of universal assertions of the form

$$\begin{aligned} B \sqsubseteq C & \quad \textit{inclusion assertion} \\ (\textit{funct } R) & \quad \textit{functionality assertion} \end{aligned}$$

Inclusion assertions are interpreted as usual in DLs, while functionality assertions express the (global) functionality of atomic roles or of inverses of atomic roles.

A *DL-Lite<sub>F</sub>* ABox is a finite set of membership assertions of the form,  $B(a)$  or  $R(a, b)$ , which state, respectively, that the object  $a$  is an instance of the basic concept  $B$ , and that the pair of objects  $(a, b)$  is an instance of the role  $R$ .

**Query answering of EQL-Lite(UCQ) queries over DL-Lite<sub>F</sub> ontologies.** As query language, here we consider *EQL-Lite(UCQ)* [6]. This is essentially formed by full (domain-independent) FOL query expressions built on top of atoms that have the form  $\mathbf{K}\alpha$ , where  $\alpha$  is a union of conjunctive queries<sup>1</sup>. The operator  $\mathbf{K}$  is a minimal knowl-

<sup>1</sup> For queries consisting of only one atom  $\mathbf{K}\alpha$ , the  $\mathbf{K}$  operator is omitted.

edge operator [17, 27, 18], which is used to formalize the epistemic state of the ontology. Informally, the formula  $\mathbf{K}\alpha$  is read as “ $\alpha$  is known to hold” or “ $\alpha$  is logically implied by the ontology”. Answering *EQL-Lite*(UCQ) queries over *DL-Lite<sub>F</sub>* ontologies is LOGSPACE, and, notably, can be reduced to evaluating (pure) FOL queries over the ABox, when considered as a database. We refer to [6] for more details.

**DL instance-level updates and erasure.** Following the work in [8, 9], we adopt Winslett’s notion of *update* [33, 34] and its counterpart, defined in [16], as the notion of *erasure*. However, we refine such notions to take into account that we are interested in studying changes at the instance level, while we insist that the intensional level of the ontology is considered stable and hence remains invariant. Intuitively, the result of updating (resp., erasing) an ontology  $s$  with a finite set  $\mathcal{F}$  of membership assertions is a new ontology that logically implies (resp., does not logically imply) all assertions in  $\mathcal{F}$ , and whose set of models minimally differs from the set of models of  $s$ . Unfortunately, as shown in [21, 8, 9], in general the result of update and erasure cannot be expressed in the same language as the original ontology.<sup>2</sup> Hence, we focus on maximally approximated update and erasure. The *maximally approximated update (erasure)* is an ontology in the same language as the original one and whose models are the models of the update (erasure) which minimally differ from the models of the original ontology.

Below, when we talk about update and erasure, we always consider their approximated versions. More precisely, let  $s = \langle \mathcal{T}, \mathcal{A} \rangle$  be an ontology and  $\mathcal{F}$  a finite set of membership assertions such that  $\text{Mod}(\mathcal{T} \cup \mathcal{F}) \neq \emptyset$ : we denote by  $s \circ_{\mathcal{T}} \mathcal{F}$  the (maximally approximated) *update of  $s$  with  $\mathcal{F}$* . Similarly, assuming  $\text{Mod}(\mathcal{T} \cup \neg\mathcal{F}) \neq \emptyset$ , where  $\neg\mathcal{F}$  denotes the set of membership assertions  $\{\neg F_i \mid F_i \in \mathcal{F}\}$ <sup>3</sup>: we denote by  $s \bullet_{\mathcal{T}} \mathcal{F}$  the (maximally approximated) *erasure of  $s$  with  $\mathcal{F}$* . Computing both (maximally approximated) update and erasure of a *DL-Lite<sub>F</sub>* ontology  $s$  with a set  $\mathcal{F}$  of membership assertions is polynomial in the sizes of  $s$  and  $\mathcal{F}$  [9].

### 3 Atomic actions

Under a functional view [17], ontologies are seen as systems that are able to perform two basic kinds of operations, namely ASK and TELL operations (cf. [17, 18]):

- ASK: given an ontology and a *query* (in the query language recognized by the ontology), returns a *finite* set of tuples of objects (constituting the answers to the query over the ontology).
- TELL: given an ontology and an *atomic action*, returns a new ontology resulting from executing the action, if the action is executable wrt the given ontology.

<sup>2</sup> The form of the *DL-Lite<sub>F</sub>* ABox considered above is that of the original proposal in [4], and is a restriction w.r.t. the one studied in [8], where instance-level updates in DLs of the *DL-Lite* family were first introduced. Specifically, here we do not allow for negation and “variables” in the membership assertions, cf. [8]. With this restriction *DL-Lite<sub>F</sub>* becomes akin to the vast majority of DLs, see [21], in that the result of updates and erasure is not expressible as a new *DL-Lite<sub>F</sub>* ABox, thus requiring approximation [9].

<sup>3</sup> Observe that  $\neg F_i$  might not be in the language of ABoxes, see [9].

In this paper, we focus on  $DL-Lite_F$  [4,5] as ontology language, and  $EQL-Lite(UCQ)$  [6] as query language. Hence, we base ASK on certain answers to such queries. Specifically, we denote by  $q(\mathbf{x})$  an ( $EQL-Lite(UCQ)$ ) query with distinguished variables  $\mathbf{x}$ . We define  $ASK(q(\mathbf{x}), s) = \{\mathbf{t} \mid s \models q(\mathbf{t})\}$ , where  $s$  is an ontology and  $\mathbf{t}$  is a tuple of constants of the same arity as  $\mathbf{x}$ . We denote by  $\phi$  queries with no distinguished variables. Such queries are called *boolean* queries and return either *true* (i.e., the empty tuple) or *false* (i.e., no tuples at all).

As for TELL, we base atomic actions on instance level update and erasure [8,9]. Specifically, we allow for *atomic actions* of the following form:

**update**  $L(\mathbf{x})$  **where**  $q(\mathbf{x})$   
**erase**  $L(\mathbf{x})$  **where**  $q(\mathbf{x})$

where  $q(\mathbf{x})$  stands for a query with  $\mathbf{x}$  as distinguished variables and  $L(\mathbf{x})$  stands for a set of membership assertions on constants and variables in  $\mathbf{x}$ . We define

$$\begin{aligned} \text{TELL}([\mathbf{update} \ L(\mathbf{x}) \ \mathbf{where} \ q(\mathbf{x})], s) &= s \circ_{\mathcal{T}} \bigcup_{\mathbf{t} \in \text{ASK}(q(\mathbf{x}), s)} L(\mathbf{t}) \\ &\text{if } \text{Mod}(\mathcal{T} \cup \bigcup_{\mathbf{t} \in \text{ASK}(q(\mathbf{x}), s)} L(\mathbf{t})) \neq \emptyset \\ \text{TELL}([\mathbf{erase} \ L(\mathbf{x}) \ \mathbf{where} \ q(\mathbf{x})], s) &= s \bullet_{\mathcal{T}} \bigcup_{\mathbf{t} \in \text{ASK}(q(\mathbf{x}), s)} L(\mathbf{t}) \\ &\text{if } \text{Mod}(\mathcal{T} \cup \neg \bigcup_{\mathbf{t} \in \text{ASK}(q(\mathbf{x}), s)} L(\mathbf{t})) \neq \emptyset \end{aligned}$$

If the conditions in the equivalences above are not satisfied, we say that the atomic action  $a$  is *not executable* in  $s$ . We extend ASK to expressions of the form  $ASK([\text{executable}(a)], s)$ , so as to be able to check executability of actions. Observe that the executability of actions as defined above can indeed be checked on the ontology.

Notice that both ASK and TELL for  $DL-Lite_F$  defined above can be computed in polynomial time, considering the size of the query fixed [6,9].

## 4 Programs

We now consider how atomic actions can be organized within a program. In particular, we focus on a variant of Golog [19, 10, 29] tailored to work on ontologies. Instead of situations, we consider ontologies, or, to be more precise, ontology states. We recall that when considering ontologies we assume the TBox to be invariant, so the only part of the ontology that can change as a result of an action (or a program) is the ABox.

While all constructs of the original Golog/ConGolog have a counterpart in our variant, here for brevity we concentrate on a core fragment only, namely:

$a$	atomic actions
$\epsilon$	the empty sequence of actions
$\delta_1; \delta_2$	sequential composition
<b>if</b> $\phi$ <b>then</b> $\delta_1$ <b>else</b> $\delta_2$	if-then-else
<b>while</b> $\phi$ <b>do</b> $\delta$	while
<b>pick</b> $q(x). \delta[x]$	pick

where  $a$  is an atomic instruction which corresponds to the execution of the atomic action  $a$ ;  $\epsilon$  is an empty sequence of instructions (needed for technical reasons) **if**  $\phi$  **then**  $\delta_1$  **else**  $\delta_2$  and **while**  $\phi$  **do**  $\delta$  are the standard constructs for conditional choice and iteration, where the test condition is a boolean query (or an executability check) to be asked to the current ontology; finally **pick**  $q(x). \delta[x]$  picks a tuple  $t$  in the answer to  $q(x)$ , instantiates the rest of the program  $\delta$  by substituting  $x$  with  $t$  and executes  $\delta$ . The latter construct is a variant of the pick construct in Golog: the main difference being that  $t$  is bounded by a query to the ontology. Also, while in Golog such a choice is nondeterministic, here we think of it as possibly made *interactively*, see below.

The general approach we follow is the *structural operational semantics* approach based on defining a single step of program execution [26, 10]. This single-step semantics is often called *transition semantics* or *computation semantics*. Namely, to formally define the semantics of our programs we make use of a *transition relation*, named *Trans*, and denoted by “ $\longrightarrow$ ”:

$$(\delta, s) \xrightarrow{a} (\delta', s')$$

where  $\delta$  is a program,  $s$  is an ontology in which the program is executed,  $a$  is the executed atomic action,  $s'$  is the ontology obtained by executing  $a$  in  $\delta$  and  $\delta'$  is what remains to be executed of  $\delta$  after having executed  $a$ .

We also make use of a *final predicate*, named *Final*, and denoted by “ $\surd$ ”:

$$(\delta, s) \surd$$

where  $\delta$  is a program that can be considered (successfully) terminated with the ontology  $s$ .

Such a relation and predicate can be defined inductively in a standard way, using the so called *transition (structural) rules*. The structural rules for defining the transition relation and the final predicate are given in Figure 1 and Figure 2 respectively. All structural rules have the following schema:

$$\frac{\text{CONSEQUENT}}{\text{ANTECEDENT}} \text{ if SIDE-CONDITION}$$

$$\begin{aligned}
 act : & \frac{(a, s) \xrightarrow{a} (\epsilon, \text{TELL}(a, s))}{true} \quad \text{if } a \text{ is executable in } s \\
 seq : & \frac{(\delta_1; \delta_2, s) \xrightarrow{a} (\delta'_1; \delta_2, s') \quad (\delta_1; \delta_2, s) \xrightarrow{a} (\delta'_2, s')}{(\delta_1, s) \xrightarrow{a} (\delta'_1; s') \quad (\delta_2, s) \xrightarrow{a} (\delta'_2; s')} \quad \text{if } (\delta_1, s)^\vee \\
 if : & \frac{(\text{if } \phi \text{ then } \delta_1 \text{ else } \delta_2, s) \xrightarrow{a} (\delta'_1, s')}{(\delta_1, s) \xrightarrow{a} (\delta'_1, s')} \quad \text{if } \text{ASK}(\phi, s) = true \\
 & \frac{(\text{if } \phi \text{ then } \delta_1 \text{ else } \delta_2, s) \xrightarrow{a} (\delta'_2, s')}{(\delta_2, s) \xrightarrow{a} (\delta'_2, s')} \quad \text{if } \text{ASK}(\phi, s) = false \\
 while : & \frac{(\text{while } \phi \text{ do } \delta, s) \xrightarrow{a} (\delta', \text{while } \phi \text{ do } \delta, s')}{(\delta, s) \xrightarrow{a} (\delta', s')} \quad \text{if } \text{ASK}(\phi, s) = true \\
 pick : & \frac{(\text{pick } q(x). \delta[x], s) \xrightarrow{a} (\delta'[t], s')}{(\delta[t], s) \xrightarrow{a} (\delta'[t], s')} \quad (\text{for } t = \text{CHOICE}[\text{ASK}(q(x), s)])
 \end{aligned}$$

**Fig. 1.** Transition rules

$$\begin{aligned}
 \epsilon : & \frac{(\epsilon, s)^\vee}{true} & seq : & \frac{(\delta_1; \delta_2, s)^\vee}{(\delta_1, s)^\vee \wedge (\delta_2; s)^\vee} \\
 if : & \frac{(\text{if } \phi \text{ then } \delta_1 \text{ else } \delta_2, s)^\vee}{(\delta_1, s)^\vee} \quad \text{if } \text{ASK}(\phi, s) = true \\
 & \frac{(\text{if } \phi \text{ then } \delta_1 \text{ else } \delta_2, s)^\vee}{(\delta_2, s)^\vee} \quad \text{if } \text{ASK}(\phi, s) = false \\
 while : & \frac{(\text{while } \phi \text{ do } \delta, s)^\vee}{true} \quad \text{if } \text{ASK}(\phi, s) = false \\
 & \frac{(\text{while } \phi \text{ do } \delta, s)^\vee}{(\delta, s)^\vee} \quad \text{if } \text{ASK}(\phi, s) = true \\
 pick : & \frac{(\text{pick } q(x). \delta[x], s)^\vee}{(\delta[t], s)^\vee} \quad (\text{for } t = \text{CHOICE}[\text{ASK}(q(x), s)])
 \end{aligned}$$

**Fig. 2.** Final rules

which is to be interpreted logically as:

$$\forall(\text{ANTECEDENT} \wedge \text{SIDE-CONDITION} \rightarrow \text{CONSEQUENT})$$

where  $\forall Q$  stands for the universal closure of all free variables occurring in  $Q$ , and, typically, ANTECEDENT, SIDE-CONDITION and CONSEQUENT share free variables. The structural rules define inductively a relation, namely: *the smallest relation satisfying the rules*.

Observe the use of the parameter CHOICE, which denotes a choice function, to determine the tuple to be picked in executing the pick constructs of programs. More precisely, CHOICE stands for any function, depending on an arbitrary number of parameters, returning a tuple from the set  $ASK(q(x), s)$ . In the original Golog/ConGolog proposal [19, 10] such a choice function (there also extended to other nondeterministic constructs) is implicit, the idea there being that Golog executions use a choice function that would lead to the termination of the program (angelic nondeterminism). In [29], a choice function is also implicit, but based on the idea that choices are done randomly (devilish nondeterminism). Here, we make use of choice functions explicitly, so as to have control on nondeterministic choices. Indeed, one interesting use of CHOICE is to model the delegation of choices to the client of the program, with the idea that the pick construct is interactive: it presents the result of the query to the client, who chooses the tuple  $s$ /he is interested in. For example, if the query is about hotels that are available in Rome, the client sees the list of available hotels resulting from the query and chooses the one  $s$ /he likes most. We say that a program is *deterministic* when no pick instructions are present or a fixed choice function for CHOICE is considered.

*Examples* Let us look at a couple of simple examples of programs. Consider the following ontology on companies and grants.

$$\begin{aligned} \exists \text{owns} &\sqsubseteq \text{Company} \\ \exists \text{owns}^- &\sqsubseteq \text{Company} \\ \text{PublicCompany} &\sqsubseteq \text{Company} \\ \text{PrivateCompany} &\sqsubseteq \text{Company} \\ \exists \text{grantAsked} &\sqsubseteq \text{exResearchGroup} \\ \exists \text{grantAsked}^- &\sqsubseteq \text{Company} \\ \text{IllegalOwner} &\sqsubseteq \text{Company} \end{aligned}$$

The first program we write aims at populating the concept *IllegalOwner* with those companies that own themselves, either directly or indirectly. We assume *temp* to be an additional role in the alphabet of the TBox. Then, the following deterministic program `ComputeIllegalOwners` can be used to populate *IllegalOwner*:

```

ComputeIllegalOwners =
erase temp(x1,x2) where q(x1,x2) <- temp(x1,x2);
erase IllegalOwner(x) where q(x) <- IllegalOwner(x);
update temp(x1,x2) where q(x1,x2) <- owns(x1,x2);
while (q() <- K(temp(y1,z), owns(z,y2)), not K(temp(y1,y2))) do (
  update temp(x1,x2) where
    q(x1,x2) <- K(temp(x1,z), owns(z,x2)), not K(temp(x1,x2))
);
update IllegalOwner(x) where q(x) <- temp(x,x)

```

The second program we look at is a program that, given a research group  $r$  and a company  $c$ , interactively –through a suitable choice function for CHOICE– selects a

public company owned by  $c$  to ask a grant to; if  $c$  does not own public companies, then it selects the company  $c$  itself:

```
askNewGrant(r,c) =
  if (q() <- owns(c,y), PublicCompany(y)) then (
    pick (q(x) <- owns(c,x), PublicCompany(x)). (
      update grantAsked(r,x) where true
    )
  )
  else update grantAsked(r,c) where true
```

## 5 Results

In this section, we assume that ontologies are expressed in  $DL-Lite_F$  and that the ASK and TELL operations are those defined for  $DL-Lite_F$  in Section 3.

Given an ontology  $s$  and a program  $\delta$ , we define the set *next step*, denoted by  $Next$ , as:

$$Next(\delta, s) = \{ \langle a, \delta', s' \rangle \mid (\delta, s) \xrightarrow{a} (\delta', s') \}$$

The following two theorems tell us that programs are indeed computable.

**Theorem 1.** *Let  $s$  be an ontology and  $\delta$  a program. Then, the set  $Next(\delta, s)$  has a finite cardinality, and can be computed in polynomial time in  $s$  and  $\delta$  (considering the size of the queries in  $\delta$  fixed). Moreover, if  $\delta$  is deterministic then, for each action  $a$ , the number of tuples  $\langle a, \delta', s' \rangle \in Next(\delta, s)$  is at most one (one if  $a$  is executable, zero otherwise).*

**Theorem 2.** *Let  $s$  be an ontology and  $\delta$  a program. Then, checking  $(\delta, s)^\vee$  can be done in polynomial time in  $s$  and  $\delta$  (considering the size of the queries in  $\delta$  fixed).*

Given an ontology  $s_0$  and a sequence  $\rho = a_1 \cdots a_n$  of actions, we say that  $\rho$  is a run of a program  $\delta_0$  over the ontology  $s_0$  if there are  $(\delta_i, s_i)$ , for  $i = 1, \dots, n$ , such that

$$(\delta_0, s_0) \xrightarrow{a_1} (\delta_1, s_1) \xrightarrow{a_2} \cdots \xrightarrow{a_n} (\delta_n, s_n)$$

We call  $\delta_n$  and  $s_n$  above respectively the program and the ontology resulting from the run  $\rho$ . If  $(\delta_n, s_n)$  is final (i.e.,  $(\delta_n, s_n)^\vee$ ), then we say that  $\rho$  is a *terminating run*. Note that, if the program  $\delta_0$  is deterministic, then  $(\delta_n, s_n)$  is functionally determined by  $(\delta_0, s_0)$  and  $\rho$ .

**Theorem 3.** *Let  $s_0$  be an ontology,  $\delta_0$  a deterministic program, and  $\rho = a_1 \cdots a_n$  a sequence of actions. Then checking whether  $\rho$  is a run of  $\delta_0$  starting from  $s_0$  can be done in polynomial time in the size of  $s_0$ ,  $\rho$ , and  $\delta_0$  (considering the size of the queries in  $\delta_0$  fixed)*

**Theorem 4.** *Let  $s_0$  be an ontology,  $\delta_0$  a deterministic program, and  $\rho$  a run of  $\delta_0$  starting from  $s_0$ . Then, computing the resulting program  $\delta_n$  and the resulting ontology  $s_n$ , as well as checking  $(\delta_n, s_n)^\vee$  and computing a query  $q(\mathbf{x})$  over  $s_n$ , can be done in polynomial time in the size of  $s_0$ ,  $\rho$ , and  $\delta_0$  (considering the size of the queries in  $\delta_0$  fixed).*

For nondeterministic programs, i.e., when we do not fix a choice function for CHOICE, Theorems 3 and 4 do not hold anymore. Indeed, it can be shown the problems in the theorems become NP-complete.

We conclude this section by turning to the two classical problem in reasoning about actions, namely the executability problem and the projection problem [28]. In our setting such problems are phrased as follows:

- *executability problem*: check whether a sequence of actions is executable in an ontology;
- *projection problem*: compute the result of a query in the ontology obtained by executing a sequence of actions in an initial ontology.

Now, considering that a sequence of actions can be seen as a simple deterministic program, from the theorems above we get the following result:

**Theorem 5.** *Let  $s_0$  be an ontology and  $\rho$  a sequence of actions. Then, checking the executability of  $\rho$  in  $s_0$ , and computing the result of a query  $q(x)$  over the ontology obtained by executing  $\rho$  in  $s_0$ , can both be done in polynomial time in the size of  $s_0$  and  $\rho$ .*

In fact, all the above results can be immediately extended (with different complexity bounds) to virtually every DL and associated ASK and TELL operations, as long as ASK and TELL are both decidable.

## 6 Conclusion

In this paper we have laid the foundations for an effective approach to reasoning about actions and programs over ontologies, based on a functional view of the ontology. Namely, the ontology is seen as a system that can perform two kinds of operations: ASK and TELL. We have focused on DL-Lite, but the approach applies to more expressive DLs. It suffices to have a decidable ASK, i.e., decidable query answering on the chosen query and ontology languages, and a decidable TELL, i.e., define atomic actions so that, through their effects, they produce one successor ontology (or, in fact, a finite number of successor ontologies) and such that their executability can be decided. Works such as those reported in [2, 20, 14] are certainly relevant.

Our approach (and the results for *DL-Lite<sub>F</sub>*) can be extended to all other programming constructs studied within Golog (i.e., non determinism, procedures) [19], ConGolog (i.e., concurrency, prioritized interrupts) [10] and, with some care –see the discussion on analysis and synthesis below– even to those in IndiGolog (search) [29].

Also, the works on forms of execution developed within Golog/ConGolog/IndiGolog can be lifted to DL ontologies by applying the proposed approach. Specifically, notions like online execution [29], offline execution [19, 10], monitored execution [11], can all be lifted to the setting studied here.

Golog/ConGolog-like programs do not have a store to keep memory of previous results of queries to the ontology. An interesting extension would be to introduce such a store, i.e., variables for storing results of queries or partial computations. Notice that this

would make also the program infinite state in general (the ontology is already infinite state). Also, this would make such programs much more alike programs in standard procedural languages such as *C* or *Java*, which manipulate global data structures –in our case the ontology– and local data structures –in our case the information stored in the variables of the program.

Finally, we can adopt the functional view of ontologies also to specify interactive and nonterminating processes acting on them, similarly to what is currently done when specifying web services on relational databases [3, 12].

We close the paper by noticing that, since the ontology is not finite state, tasks related to automated analysis and automated synthesis of programs (e.g., verifying executability on every ontology, verifying termination, synthesizing a plan that achieves a goal, or synthesizing a service that fulfills a certain specification) are difficult in general. This difficulty is shared with SitCalc-based and Golog/ConGolog-like high-level programs. One of the most promising techniques to effectively tackle such tasks is to rely on a suitable finite state *abstraction* (cf. [35]) of the ontology, and use such an abstraction in the analysis and in the synthesis.

**Acknowledgements.** This research has been partially supported by the FET project TONES (Thinking ONtologiES), funded within the EU 6th Framework Programme under contract FP6-7603.

## References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proc. of AAAI 2005*, pages 572–577, 2005.
3. D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based Semantic Web services with messaging. In *Proc. of VLDB 2005*, pages 613–624, 2005.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, pages 602–607, 2005.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. EQL-Lite: Effective first-order query processing in description logics. In *Proc. of IJCAI 2007*, pages 274–279, 2007.
7. G. De Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Moving a robot: the KR&R approach at work. In *Proc. of KR’96*, pages 198–209, 1996.
8. G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the update of description logic ontologies at the instance level. In *Proc. of AAAI 2006*, 2006.
9. G. De Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the approximation of instance level update and erasure in description logics. In *Proc. of AAAI 2007*, 2007.
10. G. De Giacomo, Y. Lespérance, and H. J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.
11. G. De Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *Proc. of KR’98*, pages 453–465, 1998.

12. A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven web services. In *Proc. of PODS 2006*, pages 90–99, 2006.
13. D. G. Giuseppe and R. Rosati. Minimal knowledge approach to reasoning about actions and sensing. *ETAI*, 3, Section C, 1999.
14. Y. Gu and M. Soutchanski. Decidable reasoning in a modified situation calculus. In *Proc. of IJCAI 2007*, pages 1891–1897, 2007.
15. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.
16. H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. of KR’91*, pages 387–394, 1991.
17. H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.
18. H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.
19. H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *J. of Logic Programming*, 31:59–84, 1997.
20. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Reasoning about actions using description logics with general TBoxes. In *Proc. of JELIA 2006*, 2006.
21. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In *Proc. of KR 2006*, pages 46–56, 2006.
22. D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, Solanki, N. Srinivasan, and K. Sycara. Bringing semantics to web services: The OWL-S approach. In *Proc. of the 1st Int. Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, 2004.
23. J. McCarthy. Towards a mathematical science of computation. In *Proc. of the IFIP Congress*, pages 21–28, 1962.
24. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.
25. R. P. A. Petrick and F. Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proc. of KR 2004*, pages 613–622, 2004.
26. G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.
27. R. Reiter. What should a database know? *J. of Logic Programming*, 14:127–153, 1990.
28. R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.
29. S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the semantics of deliberation in IndiGolog - from theory to implementation. *Ann. of Mathematics and Artificial Intelligence*, 41(2–4):259–299, 2004.
30. S. Sardiña, G. De Giacomo, Y. Lespérance, and H. J. Levesque. On the limits of planning over belief states under strict uncertainty. In *Proc. of KR 2006*, pages 463–471, 2006.
31. M. K. Smith, C. Welty, and D. L. McGuinness. OWL Web Ontology Language guide. W3C Recommendation, Feb. 2004. Available at <http://www.w3.org/TR/owl-guide/>.
32. M. B. van Riemsdijk, F. S. de Boer, M. Dastani, and J.-J. C. Meyer. Prototyping 3APL in the Maude term rewriting language. In *Proc. of 5th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 1279–1281, 2006.
33. M. Winslett. Reasoning about action using a possible models approach. In *Proc. of AAAI’98*, 1988.
34. M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.
35. L. D. Zuck and A. Pnueli. Model checking and abstraction to the aid of parameterized systems (a survey). *Computer Languages, Systems & Structures*, 30(3–4):139–169, 2004.

## Ontology Reuse: Better Safe than Sorry

Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov and Ulrike Sattler  
 The University of Manchester  
 School of Computer Science  
 Manchester, M13 9PL, UK  
 {bcg, horrocks, ykazakov, sattler}@cs.man.ac.uk

### 1 Motivation

The design, maintenance, reuse, and integration of ontologies are complex tasks. Like software engineers, ontology engineers need to be supported by tools and methodologies that help them to minimize the introduction of errors, i.e., to ensure that ontologies are consistent and do not have unexpected consequences. In order to develop this support, important notions from software engineering, such as *module*, *black-box behavior*, and *controlled interaction*, must be adapted.

Recently, there has been growing interest in the topic of modularity in ontology engineering [10, 9, 8, 5, 3], motivated by the above mentioned application needs. This paper extends our previous results[3]. We focus on the problem of “safe” reuse of ontologies and consider the scenario in which we are developing an ontology  $\mathcal{P}$  and want to reuse a set  $S$  of symbols—that is, concept names, role names and individuals—from a “foreign” ontology  $\mathcal{Q}$  without changing their meaning.

Suppose that an ontology engineer is building an ontology about research projects, which specifies different types of projects according to the research topics they focus on. For example, the concepts `Genetic_Disorder_Project` and `Cystic_Fibrosis_EUProject` describe projects about genetic disorders and European projects about cystic fibrosis respectively, as given by the axioms P1 and P2 in Figure 1. The ontology engineer is an expert on research projects; he knows, for example, that every instance of `EU_Project` must be an instance of `Project` (axiom P3) and that the role `has_Focus` can be applied only to instances of `Project` (axiom P4). He may be unfamiliar, however, with most of the topics the projects cover and, in particular, with the terms `Cystic_Fibrosis` and `Genetic_Disorder` mentioned in P1 and P2. In order to complete the projects ontology with suitable definitions of these medical terms, he decides to reuse the knowledge about these subjects from a well-established medical ontology.

Suppose that `Cystic_Fibrosis` and `Genetic_Disorder` are described in an ontology  $\mathcal{Q}$  containing axioms M1-M4 in Figure 1. The most straightforward way to reuse these concepts is to import in  $\mathcal{P}$  the ontology  $\mathcal{Q}$ —that is, to add the axioms from  $\mathcal{Q}$  to the axioms of  $\mathcal{P}$  and work with the extended ontology  $\mathcal{P} \cup \mathcal{Q}$ . Importing additional axioms into an ontology may result into new logical consequences. For example, axioms M1–M4 in  $\mathcal{Q}$  imply that every instance of `Cystic_Fibrosis` is an instance of `Genetic_Disorder`:

$$\mathcal{Q} \models \alpha := (\text{Cystic\_Fibrosis} \sqsubseteq \text{Genetic\_Disorder}) \quad (1)$$

Indeed,  $\alpha_1 = (\text{Cystic\_Fibrosis} \sqsubseteq \text{Genetic\_Disorder})$  follows from axioms M1 and M2 as well as from M1 and M3;  $\alpha$  follows from  $\alpha_1$  and M4. Using inclusion  $\alpha$  from

<b>Ontology of medical research projects <math>\mathcal{P}</math>:</b>	
P1	$\text{Genetic\_Disorder\_Project} \equiv \text{Project} \sqcap \exists \text{has\_Focus\_Genetic\_Disorder}$
P2	$\text{Cystic\_Fibrosis\_EUProject} \equiv \text{EUProject} \sqcap \exists \text{has\_Focus\_Cystic\_Fibrosis}$
P3	$\text{EUProject} \sqsubseteq \text{Project}$
P4	$\exists \text{has\_Focus\_T} \sqsubseteq \text{Project}$
E1	$\text{Project} \sqcap (\text{Genetic\_Disorder} \sqcup \text{Cystic\_Fibrosis}) \sqsubseteq \perp$
E2	$\forall \text{has\_Focus\_Cystic\_Fibrosis} \sqsubseteq \exists \text{has\_Focus\_Genetic\_Disorder}$
<b>Ontology of medical terms <math>\mathcal{Q}</math>:</b>	
M1	$\text{Cystic\_Fibrosis} \equiv \text{Fibrosis} \sqcap \exists \text{located\_In\_Pancreas} \sqcap \exists \text{has\_Origin\_Genetic\_Origin}$
M2	$\text{Genetic\_Fibrosis} \equiv \text{Fibrosis} \sqcap \exists \text{has\_Origin\_Genetic\_Origin}$
M3	$\text{Fibrosis} \sqcap \exists \text{located\_In\_Pancreas} \sqsubseteq \text{Genetic\_Fibrosis}$
M4	$\text{Genetic\_Fibrosis} \sqsubseteq \text{Genetic\_Disorder}$

Fig. 1: Reusing medical terminology in an ontology on research projects

(1) and axioms P1–P3 from ontology  $\mathcal{P}$  we can now prove that every instance of  $\text{Cystic\_Fibrosis\_EUProject}$  must also be an instance of  $\text{Genetic\_Disorder\_Project}$ :

$$\mathcal{P} \cup \mathcal{Q} \models \beta := (\text{Cystic\_Fibrosis\_EUProject} \sqsubseteq \text{Genetic\_Disorder\_Project}) \quad (2)$$

Note that, on the one hand,  $\mathcal{P} \not\models \beta$  and, on the other hand, the ontology engineer might be not aware of (2), even though it concerns the terms of primary scope in  $\mathcal{P}$ .

It is to be expected that axioms like  $\alpha$  in (1) from an imported ontology  $\mathcal{Q}$  cause new entailments like  $\beta$  in (2) over the terms defined in the main ontology  $\mathcal{P}$ . One would not expect, however, that the meaning of the terms defined in  $\mathcal{Q}$  changes as a consequence of the import since these terms are supposed to be completely specified within  $\mathcal{Q}$ . Such a side effect is highly undesirable for the modeling of ontology  $\mathcal{P}$  since the ontology engineer of  $\mathcal{P}$  might not be an expert on the subject of  $\mathcal{Q}$  and is not supposed to alter the meaning of the terms defined in  $\mathcal{Q}$ , not even implicitly. The meaning of the reused terms might change after the import due, for example, to modeling errors. In particular, suppose the ontology engineer has learned about the concepts  $\text{Genetic\_Disorder}$  and  $\text{Cystic\_Fibrosis}$  from the ontology  $\mathcal{Q}$  (including the dependency (1)) and has decided to introduce additional axioms formalizing the following statements:

“Every instance of  $\text{Project}$  is different from every instance of  $\text{Genetic\_Disorder}$  and every instance of  $\text{Cystic\_Fibrosis}$ .”<sup>(3)</sup>

“Every project that has  $\text{Focus}$  on  $\text{Cystic\_Fibrosis}$ , also has  $\text{Focus}$  on  $\text{Genetic\_Disorder}$ .”<sup>(4)</sup>

Note that the statements (3) and (4) add new information about projects and, intuitively, they should not change or constrain the meaning of the medical terms.

Suppose the ontology engineer has formalized statements (3) and (4) in  $\mathcal{P}$  using axioms E1 and E2 respectively. At this point, he has introduced modeling errors by translating the words *and* and *every* as conjunction  $\sqcap$  and value restriction  $\forall$  respectively. As

a consequence, axioms E1 and E2 do not correspond to (3) and (4): E1 actually formalizes the following statement: “*Every instance of Project is different from every common instance of Genetic\_Disorder and Cystic\_Fibrosis*”, and E2 expresses that “*Every object that has\_Focus only on Cystic\_Fibrosis if at all, also has\_Focus on Genetic\_Disorder*”. This kind of modeling errors are difficult to detect, especially when they do not lead to inconsistencies in the original ontology.

Note that, although axiom E1 does not correspond to fact (3), it is still a consequence of (3) and hence it should not constrain the meaning of the medical terms. In contrast, E2 is not a consequence of (4) and, in fact, it does constrain the meaning of these medical terms. Indeed, axioms E1 and E2 together with axioms P1-P4 from  $\mathcal{P}$  imply new axioms about the concepts Cystic\_Fibrosis and Genetic\_Disorder, namely their disjointness:

$$\mathcal{P} \models \gamma := (\text{Genetic\_Disorder} \sqcap \text{Cystic\_Fibrosis} \sqsubseteq \perp) \quad (5)$$

The entailment (5) can be proved using axiom E2 which is equivalent to:

$$\top \sqsubseteq \exists \text{has\_Focus.}(\text{Genetic\_Disorder} \sqcup \neg \text{Cystic\_Fibrosis}) \quad (6)$$

The inclusion (6) and P4 imply that every element in the domain must be a project—that is,  $\mathcal{P} \models (\top \sqsubseteq \text{Project})$ . Now, together with axiom E1, this implies (5). The axioms E1 and E2 not only imply new statements about the medical terms, but also cause inconsistencies when used together with the imported axioms from  $\mathcal{Q}$ . Indeed, from (1) and (5) we obtain  $\mathcal{P} \cup \mathcal{Q} \models \delta := (\text{Cystic\_Fibrosis} \sqsubseteq \perp)$  which expresses the inconsistency of the concept Cystic\_Fibrosis.

To summarize, we have seen that importing an external ontology can lead to undesirable side effects in our knowledge reuse scenario, like the entailment of new axioms or even inconsistencies over the reused vocabulary.

The contributions of this paper are as follows. First, we formalize some reasoning services that are relevant for ontology reuse. In particular, we propose the notion of safe reuse of a signature in an ontology. Second, we show that the problem of checking safety is undecidable in  $\mathcal{ALCO}$ . This result leaves us with two alternatives: we can either focus on simple DLs for which this problem is decidable, or we may look for sufficient conditions for safety—that is, an incomplete solution. We define in general terms the notion of a sufficient condition for safety—a *safety class*—and define a family of safety classes—called *locality*—with some compelling properties. We have implemented a safety checking algorithm and obtained empirical evidence of its usefulness in practice.

This paper comes with an extended version available online [4]; we refer the reader to the extended version for further technical details.

## 2 Conservative Extensions and Safety

As argued in the previous section, an important requirement for the reuse of an ontology  $\mathcal{Q}$  within an ontology  $\mathcal{P}$  should be that  $\mathcal{P} \cup \mathcal{Q}$  produces exactly the same logical consequences over the vocabulary of  $\mathcal{Q}$  as  $\mathcal{Q}$  alone does. This requirement can be naturally formulated using the well-known notion of a conservative extension, which has recently been investigated in the context of ontologies [7, 8].

**Definition 1 (Conservative Extension).** Let  $\mathcal{L}$  be a description logic and let  $\mathcal{O}_1 \subseteq \mathcal{O}$  be two ontologies, and  $\mathbf{S}$  a signature over  $\mathcal{L}$ . We say that  $\mathcal{O}$  is an  $\mathbf{S}$ -conservative extension of  $\mathcal{O}_1$  w.r.t.  $\mathcal{L}$ , if for every axiom  $\alpha$  over  $\mathcal{L}$  with  $\text{Sig}(\alpha) \subseteq \mathbf{S}$ , we have  $\mathcal{O} \models \alpha$  iff  $\mathcal{O}_1 \models \alpha$ . We say that  $\mathcal{O}$  is a conservative extension of  $\mathcal{O}_1$  w.r.t.  $\mathcal{L}$  if  $\mathcal{O}$  is an  $\mathbf{S}$ -conservative extension of  $\mathcal{O}_1$  w.r.t.  $\mathcal{L}$  for  $\mathbf{S} = \text{Sig}(\mathcal{O}_1)$ .

Definition 1 implies that, in order to show that  $\mathcal{P} \cup \mathcal{Q}$  is not a  $\mathbf{S}$ -conservative extension of  $\mathcal{Q}$  it suffices to find an axiom  $\alpha$  over  $\mathbf{S}$  that is implied by  $\mathcal{P} \cup \mathcal{Q}$  but not by  $\mathcal{Q}$  alone. In our example, the ontology  $\mathcal{P} \cup \mathcal{Q}$  is *not* a conservative extension of  $\mathcal{Q}$  w.r.t.  $\mathbf{S} = \{\text{Cystic\_Fibrosis}, \text{Genetic\_Disorder}\}$  since  $\mathcal{P} \cup \mathcal{Q}$  implies  $\alpha_1 = (\text{Cystic\_Fibrosis} \sqsubseteq \perp)$  and  $\alpha_2 = (\text{Genetic\_Disorder} \sqsubseteq \perp)$  over  $\mathbf{S}$ , but  $\mathcal{Q}$  does not.

Definition 1 applies to fixed  $\mathcal{P}, \mathcal{Q}$ . In realistic scenarios, however, the reused ontology  $\mathcal{Q}$  may *evolve* beyond the control of the designers of  $\mathcal{P}$ , which may not be authorized to modify  $\mathcal{Q}$ , or may decide at a later time to reuse the symbols `Cystic_Fibrosis` and `Genetic_Disorder` from a medical ontology other than  $\mathcal{Q}$ . Therefore, for application scenarios in which the external ontology  $\mathcal{Q}$  may change, it is reasonable to “abstract” from the particular  $\mathcal{Q}$  under consideration. In other words, the fact that the axioms in  $\mathcal{P}$  do not change the meaning of the external symbols in  $\mathbf{S}$  should be *independent* from the particular meaning of these symbols. This idea can be made precise as follows:

**Definition 2 (Safety for a Signature).** Let  $\mathcal{L}$  be an ontology language, and let  $\mathcal{O}$  be an ontology and  $\mathbf{S}$  a signature over  $\mathcal{L}$ . We say that  $\mathcal{O}$  is safe for  $\mathbf{S}$  w.r.t.  $\mathcal{L}$ , if for every ontology  $\mathcal{O}'$  over  $\mathcal{L}$  with  $\text{Sig}(\mathcal{O}) \cap \text{Sig}(\mathcal{O}') \subseteq \mathbf{S}$ , we have that  $\mathcal{O} \cup \mathcal{O}'$  is a conservative extension of  $\mathcal{O}'$  w.r.t.  $\mathcal{L}$ .

Definition 2 captures the intuition in our example: the axioms in  $\mathcal{P}$  should not yield new consequences over the signature  $\mathbf{S}$  and the signature  $\text{Sig}(\mathcal{Q})$  of the reused ontology  $\mathcal{Q}$ , independently of the particular  $\mathcal{Q}$  under consideration. In our example, the ontology  $\mathcal{O} = \{\text{E2}\}$  is not safe w.r.t.  $\mathbf{S} = \{\text{Cystic\_Fibrosis}, \text{Genetic\_Disorder}\}$  and  $\mathcal{L} = \mathcal{ALC}$ . Indeed, take  $\mathcal{Q}_1 = \{\top \sqsubseteq \text{Cystic\_Fibrosis}; \text{Genetic\_Disorder} \sqsubseteq \perp\}$ . Then,  $\mathcal{Q}_1 \cup \mathcal{O}$  is inconsistent whereas  $\mathcal{Q}_1$  is consistent. Consequently,  $\mathcal{Q}_1 \cup \mathcal{O}$  is not a  $\mathbf{S}$ -conservative extension of  $\mathcal{Q}_1$  w.r.t.  $\mathcal{L} = \mathcal{ALC}$ , and therefore  $\mathcal{O} = \{\text{E2}\}$  is not safe for  $\mathbf{S}$  and  $\mathcal{L}$ .

Proving that an ontology is safe is more involved than proving that it is not. One way to prove that  $\mathcal{O}$  is  $\mathbf{S}$ -safe is the following: if we can take an arbitrary interpretation for the symbols in  $\mathbf{S}$  and extend it to a model of  $\mathcal{O}$  by interpreting the additional symbols in  $\text{Sig}(\mathcal{O})$ , then  $\mathcal{O}$  must be  $\mathbf{S}$ -safe. This property can be formalized as follows:

**Definition 3.** Two interpretations  $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$  and  $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2})$  coincide on a signature  $\mathbf{S}$  (notation:  $\mathcal{I}_1|_{\mathbf{S}} = \mathcal{I}_2|_{\mathbf{S}}$ ) if  $\Delta^{\mathcal{I}_1} = \Delta^{\mathcal{I}_2}$  and  $X^{\mathcal{I}_1} = X^{\mathcal{I}_2}$  for every  $X \in \mathbf{S}$ .

**Lemma 1.** Let  $\mathcal{O}$  be a  $\text{SHOIQ}$  ontology and  $\mathbf{S}$  a signature such that for every interpretation  $\mathcal{I}$  there exists a model  $\mathcal{J}$  of  $\mathcal{O}$  such that  $\mathcal{J}|_{\mathbf{S}} = \mathcal{I}|_{\mathbf{S}}$ . Then  $\mathcal{O}$  is safe for  $\mathbf{S}$  w.r.t.  $\mathcal{L} = \text{SHOIQ}$ .

We can now prove that the ontology  $\mathcal{P}_1$  consisting of axioms P1-P4 is safe for  $\mathbf{S} = \{\text{Cystic\_Fibrosis}, \text{Genetic\_Disorder}\}$ . Take an arbitrary interpretation  $\mathcal{I}$  of  $\mathbf{S}$  and construct an interpretation  $\mathcal{J}$  to be identical to  $\mathcal{I}$  except for the interpretations of the atomic concepts `Genetic_Disorder_Project`, `Cystic_Fibrosis_EUProject`, `Project`, `EUProject` and

the atomic role has\_Focus, all of which we interpret in  $\mathcal{J}$  as the empty set. All the axioms P1–P4, E2 are satisfied in  $\mathcal{J}$  and hence  $\mathcal{J} \models \mathcal{P}_1$ .

Using Lemma 1, we are now ready to show the main result in this section:

**Theorem 1 (Undecidability for Safety of Ontologies).** *Given an  $\mathcal{ALC}$ -ontology  $\mathcal{O}$  and a signature  $\mathbf{S}$  it is undecidable whether  $\mathcal{O}$  is  $\mathbf{S}$ -safe w.r.t.  $\mathcal{L} = \mathcal{ALCCO}$ .*

*Proof.* The proof is based on a reduction to a domino tiling problem. A domino system is a triple  $D = (T, H, V)$  where  $T = \{1, \dots, k\}$  is a finite set of tiles and  $H, V \subseteq T \times T$  are horizontal and vertical matching relations. A solution for a domino system  $D$  is a mapping  $t_{i,j}$  that assigns to every pair of integers  $i, j \geq 1$  an element of  $T$ , such that  $\langle t_{i,j}, t_{i,j+1} \rangle \in V$  and  $\langle t_{i,j}, t_{i+1,j} \rangle \in H$ . A periodic solution for a domino system  $D$  is a solution  $t_{i,j}$  for which there exist integers  $m \geq 1, n \geq 1$  called periods such that  $t_{i+m,j} = t_{i,j}$  and  $t_{i,j+n} = t_{i,j}$  for every  $i, j \geq 1$ .

Let  $\mathcal{D}$  be the set of all domino systems,  $\mathcal{D}_s$  be the subset of  $\mathcal{D}$  that admit a solution and  $\mathcal{D}_{ps}$  be the subset of  $\mathcal{D}_s$  that admit a periodic solution. It is well-known [1, Theorem 3.1.7] that the sets  $\mathcal{D} \setminus \mathcal{D}_s$  and  $\mathcal{D}_{ps}$  are recursively inseparable, that is, there is no recursive (i.e. decidable) subset  $\mathcal{D}' \subseteq \mathcal{D}$  of domino systems such that  $\mathcal{D}_{ps} \subseteq \mathcal{D}' \subseteq \mathcal{D}_s$ . For every domino system  $D$ , we construct a signature  $\mathbf{S} = \mathbf{S}(D)$ , an ontology  $\mathcal{O} = \mathcal{O}(D)$  which consists of a single  $\mathcal{ALC}$ -axiom such that: **(a)** if  $D$  does not have a solution then  $\mathcal{O} = \mathcal{O}(D)$  is safe for  $\mathbf{S} = \mathbf{S}(D)$  w.r.t.  $\mathcal{L} = \mathcal{ALCCO}$ , and **(b)** if  $D$  has a periodic solution then  $\mathcal{O} = \mathcal{O}(D)$  is not safe for  $\mathbf{S} = \mathbf{S}(D)$  w.r.t.  $\mathcal{L} = \mathcal{ALCCO}$ .

In other words, for the set  $\mathcal{D}'$  consisting of the domino systems  $D$  such that  $\mathcal{O} = \mathcal{O}(D)$  is not safe for  $\mathbf{S} = \mathbf{S}(D)$  w.r.t.  $\mathcal{L} = \mathcal{ALCCO}$ , we have  $\mathcal{D}_{ps} \subseteq \mathcal{D}' \subseteq \mathcal{D}_s$ . Since  $\mathcal{D} \setminus \mathcal{D}_s$  and  $\mathcal{D}_{ps}$  are recursively inseparable, this implies undecidability for  $\mathcal{D}'$  and hence for the problem of checking if  $\mathcal{O}$  is an  $\mathbf{S}$ -safe w.r.t.  $\mathcal{L} = \mathcal{ALCCO}$ , because otherwise one can use this problem for deciding membership in  $\mathcal{D}'$ .

Given  $D = (T, H, V)$ , let  $\mathbf{S}$  consist of fresh atomic concepts  $A_i$  for every  $i \in T$  and atomic roles  $r_H$  and  $r_V$ . Consider an ontology  $\mathcal{O}_{\text{tile}}$  in Figure 2 constructed for  $D$ . Note that  $\text{Sig}(\mathcal{O}_{\text{tile}}) = \mathbf{S}$ . The axioms of  $\mathcal{O}_{\text{tile}}$  express the tiling conditions for a domino

$$\begin{aligned}
 (q_1) \quad & \top \sqsubseteq A_1 \sqcup \dots \sqcup A_k && \text{where } T = \{1, \dots, k\} \\
 (q_2) \quad & A_i \sqcap A_j \sqsubseteq \perp && 1 \leq i < j \leq k \\
 (q_3) \quad & A_i \sqsubseteq \exists r_H. (\bigsqcup_{(i,j) \in H} A_j) && 1 \leq i \leq k \\
 (q_4) \quad & A_i \sqsubseteq \exists r_V. (\bigsqcup_{(i,j) \in V} A_j) && 1 \leq i \leq k
 \end{aligned}$$

Fig. 2: An ontology  $\mathcal{O}_{\text{tile}} = \mathcal{O}_{\text{tile}}(D)$  expressing tiling conditions for a domino system  $D$

system  $D$ , namely  $(q_1)$  and  $(q_2)$  express that every domain element is assigned with a unique tile  $t \in T$ ;  $(q_3)$  and  $(q_4)$  express that every domain element has horizontal and vertical matching successors. Now let  $s, B \notin \mathbf{S}$ . Let  $\mathcal{O} := \{\beta\}$  where:

$$\beta := \top \sqsubseteq \exists s. \left[ \bigsqcup_{(C_i \sqsubseteq D_i) \in \mathcal{O}_{\text{tile}}} (C_i \sqcap \neg D_i) \sqcup (\exists r_H. \exists r_V. \mathbf{B} \sqcap \exists r_V. \exists r_H. \neg \mathbf{B}) \right]$$

We say that  $r_H$  and  $r_V$  commute in an interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  if for every domain elements  $a, b, c, d_1$  and  $d_2$  from  $\Delta^{\mathcal{I}}$  with  $\langle a, b \rangle \in r_H^{\mathcal{I}}, \langle b, d_1 \rangle \in r_V^{\mathcal{I}}, \langle a, c \rangle \in r_V^{\mathcal{I}}$ , and  $\langle c, d_2 \rangle \in r_H^{\mathcal{I}}$ , we have  $d_1 = d_2$ . The following claims can be easily proved:

*Claim 1.* If  $\mathcal{O}_{\text{tile}}(D)$  has a model  $\mathcal{I}$  in which  $r_H$  and  $r_V$  commute, then  $D$  has a solution.

*Claim 2.* If  $\mathcal{I}$  is a model of  $\mathcal{O} = \{\beta\}$ , then either  $\mathcal{I} \not\models \mathcal{O}_{\text{tile}}$  or  $r_H$  and  $r_V$  do not commute in  $\mathcal{I}$ .

To prove Property (a), we use Lemma 1 and demonstrate that if  $D$  has no solution then for every interpretation  $\mathcal{I}$  there exists a model  $\mathcal{J}$  of  $\mathcal{O}$  such that  $\mathcal{J}|_{\mathbf{S}} = \mathcal{I}|_{\mathbf{S}}$ , which implies that  $\mathcal{O}$  is safe for  $\mathbf{S}$  w.r.t.  $\mathcal{L}$ . Let  $\mathcal{I}$  be an arbitrary interpretation. Since  $D$  has no solution, then by the contra-position of Claim 1 either (1)  $\mathcal{I}$  is not a model of  $\mathcal{O}_{\text{tile}}$ , or (2)  $r_H$  and  $r_V$  do not commute in  $\mathcal{I}$ . We demonstrate for both of these cases how to construct the required model  $\mathcal{J}$  of  $\mathcal{O}$  such that  $\mathcal{J}|_{\mathbf{S}} = \mathcal{I}|_{\mathbf{S}}$ .

Case (1). If  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  is not a model of  $\mathcal{O}_{\text{tile}}$  then there exists an axiom  $(C_i \sqsubseteq D_i) \in \mathcal{O}_{\text{tile}}$  such that  $\mathcal{I} \not\models (C_i \sqsubseteq D_i)$ . That is, there exists a domain element  $a \in \Delta^{\mathcal{I}}$  such that  $a \in C_i^{\mathcal{I}}$  but  $a \notin D_i^{\mathcal{I}}$ . Let us define  $\mathcal{J}$  to be identical to  $\mathcal{I}$  except for the interpretation of the atomic role  $s$  which we define in  $\mathcal{J}$  as  $s^{\mathcal{J}} = \{\langle x, a \rangle \mid x \in \Delta\}$ . Since the interpretations of the symbols in  $\mathbf{S}$  has remained unchanged, we have  $a \in C_i^{\mathcal{J}}, a \in \neg D_i^{\mathcal{J}}$ , and so  $\mathcal{J} \models (\top \sqsubseteq \exists s.[C_i \sqcap \neg D_i])$ . This implies that  $\mathcal{J} \models \beta$ , and so, we have constructed a model  $\mathcal{J}$  of  $\mathcal{O}$  such that  $\mathcal{J}|_{\mathbf{S}} = \mathcal{I}|_{\mathbf{S}}$ .

Case (2). Suppose that  $r_H$  and  $r_V$  do not commute in  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ . This means that there exist domain elements  $a, b, c, d_1$  and  $d_2$  from  $\Delta^{\mathcal{I}}$  with  $\langle a, b \rangle \in r_H^{\mathcal{I}}, \langle b, d_1 \rangle \in r_V^{\mathcal{I}}, \langle a, c \rangle \in r_V^{\mathcal{I}}$ , and  $\langle c, d_2 \rangle \in r_H^{\mathcal{I}}$ , such that  $d_1 \neq d_2$ . Let us define  $\mathcal{J}$  to be identical to  $\mathcal{I}$  except for the interpretation of the atomic role  $s$  and the atomic concept  $B$ . We interpret  $s$  in  $\mathcal{J}$  as  $s^{\mathcal{J}} = \{\langle x, a \rangle \mid x \in \Delta\}$ . We interpret  $B$  in  $\mathcal{J}$  as  $B^{\mathcal{J}} = \{d_1\}$ . Note that  $a \in (\exists r_H. \exists r_V. B)^{\mathcal{J}}$  and  $a \in (\exists r_V. \exists r_H. \neg B)^{\mathcal{J}}$  since  $d_1 \neq d_2$ . So, we have  $\mathcal{J} \models (\top \sqsubseteq \exists s. [\exists r_H. \exists r_V. B \sqcap \exists r_V. \exists r_H. \neg B])$  which implies that  $\mathcal{J} \models \beta$ , and thus, we have constructed a model  $\mathcal{J}$  of  $\mathcal{O}$  such that  $\mathcal{J}|_{\mathbf{S}} = \mathcal{I}|_{\mathbf{S}}$ .

To prove Property (b), assume that  $D$  has a periodic solution  $t_{i,j}$  with the periods  $m, n \geq 1$ . We show that  $\mathcal{O}$  is not  $\mathbf{S}$ -safe w.r.t.  $\mathcal{L}$ . We build an  $\mathcal{ALCO}$ -ontology  $\mathcal{O}'$  with  $\text{Sig}(\mathcal{O}) \cap \text{Sig}(\mathcal{O}') \subseteq \mathbf{S}$  such that  $\mathcal{O} \cup \mathcal{O}' \models (\top \sqsubseteq \perp)$ , but  $\mathcal{O}' \not\models (\top \sqsubseteq \perp)$ . This will imply that  $\mathcal{O}$  is not safe for  $\mathcal{O}'$  w.r.t.  $\mathcal{L} = \mathcal{ALCO}$ , and hence, is not safe for  $\mathbf{S}$  w.r.t.  $\mathcal{L} = \mathcal{ALCO}$ . We define  $\mathcal{O}'$  such that every model of  $\mathcal{O}'$  is a finite encoding of the periodic solution  $t_{i,j}$ . For every pair  $(i, j)$  with  $1 \leq i \leq m$  and  $1 \leq j \leq n$ , introduce a fresh individual  $a_{i,j}$  and take  $\mathcal{O}'$  the extension of  $\mathcal{O}_{\text{tile}}$  with the following axioms:

$$\begin{aligned} (p_1) \{a_{i_1, j}\} \sqsubseteq \exists r_V. \{a_{i_2, j}\} & \quad (p_2) \{a_{i_1, j}\} \sqsubseteq \forall r_V. \{a_{i_2, j}\}, & \quad i_2 = i_1 + 1 \pmod{m} \\ (p_3) \{a_{i, j_1}\} \sqsubseteq \exists r_H. \{a_{i, j_2}\} & \quad (p_4) \{a_{i, j_1}\} \sqsubseteq \forall r_H. \{a_{i, j_2}\}, & \quad j_2 = j_1 + 1 \pmod{n} \\ (p_5) \top \sqsubseteq \bigsqcup_{1 \leq i \leq m, 1 \leq j \leq n} \{a_{i, j}\} & \end{aligned}$$

Axioms  $(p_1)$ – $(p_5)$  ensure that  $r_H$  and  $r_V$  commute in every model of  $\mathcal{O}'$ . Indeed  $\mathcal{O}'$  has a model corresponding to every periodic solution for  $D$  with periods  $m$  and  $n$ . Hence  $\mathcal{O}' \not\models (\top \sqsubseteq \perp)$ . Also, since every model of  $\mathcal{O}'$  is a model of  $\mathcal{O}_{\text{tile}}$  in which  $r_H$  and  $r_V$  commute, by Claim 2,  $\mathcal{O}' \cup \mathcal{O}$  is unsatisfiable, so  $\mathcal{O}' \cup \mathcal{O} \models (\top \sqsubseteq \perp)$ .  $\square$

### 3 Safety Classes

Theorem 1 leaves us with two alternatives: first, we can focus simple DLs for which this problem is decidable; second, we may look for sufficient conditions for the notion of safety—that is, if an ontology satisfies our conditions then we can guarantee that it is safe, but not necessarily vice versa. In this paper, we will explore the latter approach.

In general, any sufficient condition for safety can be represented by defining, for every signature  $\mathbf{S}$ , the set of ontologies over a language that satisfy the condition for that signature. These ontologies should be guaranteed to be safe.

**Definition 4 (Class of Ontologies, Safety Class).** A class of ontologies for a DL  $\mathcal{L}$  and a signature  $\mathbf{S}$  is a function  $\mathbf{O}(\cdot)$  that assigns to every subset  $\mathbf{S}'$  of  $\mathbf{S}$  a set  $\mathbf{O}(\mathbf{S}')$  of ontologies in  $\mathcal{L}$ ; it is anti-monotonic if for every  $\mathbf{S}_1 \subseteq \mathbf{S}_2$ , we have  $\mathbf{O}(\mathbf{S}_2) \subseteq \mathbf{O}(\mathbf{S}_1)$ ; it is subset-closed if for every  $\mathbf{S}$  and  $\mathcal{O}_1 \subseteq \mathcal{O}$  we have that  $\mathcal{O} \in \mathbf{O}(\mathbf{S})$  implies  $\mathcal{O}_1 \in \mathbf{O}(\mathbf{S})$ ; it is union-closed if  $\mathcal{O}_1 \in \mathbf{O}(\mathbf{S})$  and  $\mathcal{O}_2 \in \mathbf{O}(\mathbf{S})$  implies  $(\mathcal{O}_1 \cup \mathcal{O}_2) \in \mathbf{O}(\mathbf{S})$  for every  $\mathbf{S}$ . A safety class for  $\mathcal{L}$  is a class of ontologies  $\mathbf{O}(\cdot)$  for  $\mathcal{L}$  such that, for every  $\mathbf{S}$ , every ontology in  $\mathbf{O}(\mathbf{S})$  is safe for  $\mathbf{S}$ .

Safety classes may admit many natural properties, as given in Definition 4. *Anti-monotonicity* intuitively means that if an ontology  $\mathcal{O}$  can be proved to be safe w.r.t.  $\mathbf{S}$  using the sufficient condition, then  $\mathcal{O}$  can be proved to be safe w.r.t. every subset of  $\mathbf{S}$ . Similarly, *subset-closure* means that under the same assumption, every subset of  $\mathcal{O}$  can also be proved to be safe using the same sufficient condition. If a safety class is *union-closed* and two ontologies  $\mathcal{O}_1$  and  $\mathcal{O}_2$  can be proved safe using that sufficient test, then their union  $\mathcal{O}_1 \cup \mathcal{O}_2$  can also be proved safe using the same test.

#### 3.1 Locality

In this section we introduce a particular family of safety classes for  $\mathcal{L} = \mathit{SHOIQ}$ , that we call locality classes. In Section 2, we have seen that, according to Lemma 1, one way to prove that  $\mathcal{O}$  is  $\mathbf{S}$ -safe is to show that every  $\mathbf{S}$ -interpretation can be extended to a model of  $\mathcal{O}$ . Local ontologies are those for which safety can be used using Lemma 1.

**Definition 5 (Locality).** Given a  $\mathit{SHOIQ}$  signature  $\mathbf{S}$ , we say that a set of interpretations  $\mathbf{I}$  is local w.r.t.  $\mathbf{S}$  if for every  $\mathit{SHOIQ}$ -interpretation  $\mathcal{I}$  there exists an interpretation  $\mathcal{J} \in \mathbf{I}$  such that  $\mathcal{I}|_{\mathbf{S}} = \mathcal{J}|_{\mathbf{S}}$ . A class of interpretations is a function  $\mathbf{I}(\cdot)$  that given a  $\mathit{SHOIQ}$  signature  $\mathbf{S}$  returns a set of interpretations  $\mathbf{I}(\mathbf{S})$ ; it is local if  $\mathbf{I}(\mathbf{S})$  is local w.r.t.  $\mathbf{S}$  for every  $\mathbf{S}$ ; it is monotonic if  $\mathbf{S}_1 \subseteq \mathbf{S}_2$  implies  $\mathbf{I}(\mathbf{S}_1) \subseteq \mathbf{I}(\mathbf{S}_2)$ .

An axiom  $\alpha$  (an ontology  $\mathcal{O}$ ) is valid in  $\mathbf{I}$  if every interpretation  $\mathcal{I} \in \mathbf{I}$  is a model of  $\alpha$  (respectively  $\mathcal{O}$ ). Given a class of interpretations  $\mathbf{I}(\cdot)$ ,  $\mathbf{O}(\cdot)$  is the class of ontologies  $\mathbf{O}(\cdot)$  based on  $\mathbf{I}(\cdot)$  if for every  $\mathbf{S}$ ,  $\mathbf{O}(\mathbf{S})$  is the set of ontologies that are valid in  $\mathbf{I}(\mathbf{S})$ ; if  $\mathbf{I}(\cdot)$  is local then we say that  $\mathbf{O}(\cdot)$  is a class of local ontologies, and for every  $\mathbf{S}$  and  $\mathcal{O} \in \mathbf{O}(\mathbf{S})$  and every  $\alpha \in \mathcal{O}$ , we say that  $\mathcal{O}$ , respectively  $\alpha$  is local (based on  $\mathbf{I}(\cdot)$ ).

*Example 1.* Let  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\cdot)$  be a class of  $\mathit{SHOIQ}$  interpretations defined as follows. Given a signature  $\mathbf{S}$ , the set  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$  consist of interpretations  $\mathcal{J}$  such that  $r^{\mathcal{J}} = \emptyset$  for every atomic role  $r \notin \mathbf{S}$  and  $A^{\mathcal{J}} = \emptyset$  for every atomic concept  $A \notin \mathbf{S}$ . It is easy to show that

$\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$  is local for every  $\mathbf{S}$ , since for every interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  and the interpretation  $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$  defined by  $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}}$ ,  $r^{\mathcal{J}} = \emptyset$  for  $r \notin \mathbf{S}$ ,  $A^{\mathcal{J}} = \emptyset$  for  $A \notin \mathbf{S}$ , and  $X^{\mathcal{J}} := X^{\mathcal{I}}$  for the remaining symbols  $X$ , we have  $\mathcal{J} \in \mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$  and  $\mathcal{I}|_{\mathbf{S}} = \mathcal{J}|_{\mathbf{S}}$ . Since  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S}_1) \subseteq \mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S}_2)$  for every  $\mathbf{S}_1 \subseteq \mathbf{S}_2$ , we have that  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\cdot)$  is monotonic;  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\cdot)$  is also compact, since for every  $\mathbf{S}_1$  and  $\mathbf{S}_2$  the sets of interpretations  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S}_1)$  and  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S}_2)$  are defined differently only for elements in  $\mathbf{S}_1 \Delta \mathbf{S}_2$ .

Given a signature  $\mathbf{S}$ , the set  $\mathbf{Ax}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$  of axioms that are local w.r.t.  $\mathbf{S}$  based on  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$  consists of all axioms  $\alpha$  such for every  $\mathcal{J} \in \mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$ , we have that  $\mathcal{J} \models \alpha$ . Then the class of local ontologies based on  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\cdot)$  could be defined by  $\mathcal{O} \in \mathbf{O}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$  iff  $\mathcal{O} \subseteq \mathbf{Ax}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$ .

**Proposition 1 (Locality Implies Safety).** *Let  $\mathbf{O}(\cdot)$  be a class of ontologies based on a local class of interpretations  $\mathbf{I}(\cdot)$ . Then  $\mathbf{O}(\cdot)$  is a subset-closed and union-closed safety class for  $\mathcal{L} = \text{SHOIQ}$ . If additionally  $\mathbf{I}(\cdot)$  is monotonic, then  $\mathbf{O}(\cdot)$  is anti-monotonic.*

Proposition 1 and Example 1 suggest a particular way for proving safety of ontologies. Given an SHOIQ ontology  $\mathcal{O}$  and a signature  $\mathbf{S}$  it is sufficient to check if every axiom  $\alpha$  in  $\mathcal{O}$  is satisfied by every interpretation from  $\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$ ; that is, given  $\alpha$  and  $\mathbf{S}$ , it suffices to interpret every atomic concept and atomic role not in  $\mathbf{S}$  as the empty set and then check if  $\alpha$  is satisfied in all interpretations of the remaining symbols. Note that for defining  $\mathbf{O}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$ , we do not fix the interpretation of the individuals outside  $\mathbf{S}$ , but in principle, we could do that. The reason is that there is no elegant way how to describe such interpretations. Namely, every individual needs to be interpreted as an element of the domain, and there is no “canonical” element of every domain to choose, as opposed to the “canonical” subsets of (pairs of) the domain elements, which can be taken, say as the empty set or the set of all (pairs of) the domain elements. These observations suggest the following test for locality:

**Proposition 2 (Testing Locality).** *Given a SHOIQ-signature  $\mathbf{S}$ , concept  $C$ , axiom  $\alpha$  and ontology  $\mathcal{O}$  let  $\tau(C, \mathbf{S})$ ,  $\tau(\alpha, \mathbf{S})$  and  $\tau(\mathcal{O}, \mathbf{S})$  be defined recursively as follows:*

$$\begin{aligned}
 \tau(C, \mathbf{S}) ::= & \tau(A, \mathbf{S}) &= \perp \text{ if } A \notin \mathbf{S} \text{ and otherwise } = A; & (a) \\
 & | \tau(C_1 \sqcap C_2, \mathbf{S}) &= \tau(C_1, \mathbf{S}) \sqcap \tau(C_2, \mathbf{S}); & (b) \\
 & | \tau(\neg C_1, \mathbf{S}) &= \neg \tau(C_1, \mathbf{S}); & (c) \\
 & | \tau(\exists R.C_1, \mathbf{S}) &= \perp \text{ if } \text{Sig}(R) \not\subseteq \mathbf{S} \text{ and otherwise } = \exists R.\tau(C_1, \mathbf{S}); & (d) \\
 & | \tau(\geq n R.C_1, \mathbf{S}) &= \perp \text{ if } \text{Sig}(R) \not\subseteq \mathbf{S} \text{ and otherwise } = (\geq n R.\tau(C_1, \mathbf{S})). & (e) \\
 \tau(\alpha, \mathbf{S}) ::= & \tau(C_1 \sqsubseteq C_2, \mathbf{S}) &= (\tau(C_1, \mathbf{S}) \sqsubseteq \tau(C_2, \mathbf{S})); & (g) \\
 & | \tau(R_1 \sqsubseteq R_2, \mathbf{S}) &= (\perp \sqsubseteq \perp) \text{ if } \text{Sig}(R_1) \not\subseteq \mathbf{S}, \text{ otherwise} \\
 & &= \exists R_1.\top \sqsubseteq \perp \text{ if } \text{Sig}(R_2) \not\subseteq \mathbf{S}, \text{ otherwise } = (R_1 \sqsubseteq R_2); & (h) \\
 & | \tau(a : C, \mathbf{S}) &= a : \tau(C, \mathbf{S}); & (i) \\
 & | \tau(r(a, b), \mathbf{S}) &= \top \sqsubseteq \perp \text{ if } r \notin \mathbf{S} \text{ and otherwise } = r(a, b); & (j) \\
 & | \tau(\text{Trans}(r), \mathbf{S}) &= \perp \sqsubseteq \perp \text{ if } r \notin \mathbf{S} \text{ and otherwise } = \text{Trans}(r); & (k) \\
 & | \tau(\text{Funct}(R), \mathbf{S}) &= \perp \sqsubseteq \perp \text{ if } \text{Sig}(R) \not\subseteq \mathbf{S} \text{ and otherwise } = \text{Funct}(R). & (l) \\
 \tau(\mathcal{O}, \mathbf{S}) ::= & \bigcup_{\alpha \in \mathcal{O}} \tau(\alpha, \mathbf{S}) & & (m)
 \end{aligned}$$

Then,  $\mathcal{O} \in \mathbf{O}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\mathbf{S})$  iff every axiom in  $\tau(\mathcal{O}, \mathbf{S})$  is a tautology.

*Example 2.* Let  $\mathcal{O} = \{\alpha\}$  consists of axiom  $\alpha = \text{M2}$  from Figure 1. We demonstrate using Proposition 2 that  $\mathcal{O}$  is local w.r.t.  $\mathbf{S} = \{\text{Fibrosis}, \text{Genetic\_Origin}\}$ . According to Proposition 2, in order to check if  $\mathcal{O}$  is local w.r.t.  $\mathbf{S}_1$  it is sufficient to perform the following replacements in  $\alpha$  (the symbols from  $\mathbf{S}$  are underlined):

$$\text{M2} \quad \overbrace{\perp \text{ [by (a)]}}^{\perp \text{ [by (a)]}} \equiv \text{Fibrosis} \sqcap \overbrace{\exists \text{has\_Origin. Genetic\_Origin}}^{\perp \text{ [by (d)]}} \quad (7)$$

We obtain  $\tau(\text{M2}, \mathbf{S}) = (\perp \equiv \text{Fibrosis} \sqcap \perp)$  which is a *SHOIQ*-tautology. Hence  $\mathcal{O}$  is local w.r.t.  $\mathbf{S}$  and hence by Lemma 1 is  $\mathbf{S}$ -safe w.r.t. *SHOIQ*.

By Proposition 2, one can use available DL-reasoners for testing locality. If this is too costly, one can still formulate a tractable approximation of locality:

**Definition 6 (Syntactic Locality for *SHOIQ*).** Let  $\mathbf{S}$  be a signature. The following grammar recursively defines two sets of concepts  $\mathbf{Con}^0(\mathbf{S})$  and  $\mathbf{Con}^\Delta(\mathbf{S})$  for  $\mathbf{S}$ :

$$\begin{aligned} \mathbf{Con}^0(\mathbf{S}) &::= A^\emptyset \mid \neg C^\Delta \mid C \sqcap C^\emptyset \mid \exists R^\emptyset.C \mid \exists R.C^\emptyset \mid (\geq n R^\emptyset.C) \mid (\geq n R.C^\emptyset). \\ \mathbf{Con}^\Delta(\mathbf{S}) &::= \neg C^\emptyset \mid C_1^\Delta \sqcap C_2^\Delta. \end{aligned}$$

where  $A^\emptyset \notin \mathbf{S}$  is an atomic concept,  $R$  is a role, and  $C$  is a concept,  $C^\emptyset \in \mathbf{Con}^0(\mathbf{S})$ ,  $C_i^\Delta \in \mathbf{Con}^\Delta(\mathbf{S})$ ,  $i = 1, 2$ , and  $R^\emptyset$  is (possibly inverse of) an atomic role  $r^\emptyset \notin \mathbf{S}$ . An axiom  $\alpha$  is syntactically local w.r.t.  $\mathbf{S}$  if it is of one of the following forms: (1)  $R^\emptyset \sqsubseteq R$ , or (2)  $\text{Trans}(R^\emptyset)$ , or (3)  $\text{Funct}(R^\emptyset)$ , or (4)  $C^\emptyset \sqsubseteq C$ , or (5)  $C \sqsubseteq C^\Delta$ , or (6)  $a : C^\Delta$ . A *SHOIQ*-ontology  $\mathcal{O}$  is syntactically local w.r.t.  $\mathbf{S}$  if every  $\alpha \in \mathcal{O}$  is syntactically local.

It is easy to see from the inductive definitions of  $\mathbf{Con}^0(\mathbf{S})$  and  $\mathbf{Con}^\Delta(\mathbf{S})$  in Definition 6 that for every interpretation  $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$  from  $\mathbf{I}_{A \leftarrow \emptyset}^{\mathcal{I}}(\mathbf{S})$  we have that  $(R^\emptyset)^\mathcal{I} = \emptyset$ ,  $(C^\emptyset)^\mathcal{I} = \emptyset$  and  $(C^\Delta)^\mathcal{I} = \Delta^\mathcal{I}$ ,  $C^\emptyset \in \mathbf{Con}^0(\mathbf{S})$  and  $C^\Delta \in \mathbf{Con}^\Delta(\mathbf{S})$ . Hence, every syntactically local axiom is satisfied in every interpretation  $\mathcal{I}$  from  $\mathbf{I}_{A \leftarrow \emptyset}^{\mathcal{I}}(\mathbf{S})$ , and so is also semantically local. Furthermore, it can even be shown that the safety class for *SHOIQ* based on syntactic locality enjoys all of the properties from Definition 4—that is, it is anti-monotone, subset-closed and union-closed.

*Example 3 (Example 2 continued).* Axiom M2 from Figure 1 is syntactically local w.r.t.  $\mathbf{S}_1 = \{\text{Fibrosis}, \text{Genetic\_Origin}\}$ :

$$\text{M2} \quad \overbrace{\perp \text{ [matches } A^\emptyset]}^{\in \mathbf{Con}^0(\mathbf{S}_1) \text{ [matches } A^\emptyset]} \equiv \text{Fibrosis} \sqcap \overbrace{\exists \text{has\_Origin. Genetic\_Origin}}^{\in \mathbf{Con}^0(\mathbf{S}_1) \text{ [matches } \exists R^\emptyset.C]} \quad (8)$$

$\in \mathbf{Con}^0(\mathbf{S}_1) \text{ [matches } C \sqcap C^\emptyset]$

It is easy to show that syntactic locality can be checked in polynomial time with respect to the size of the input ontology and input signature.

Note that semantic locality does not imply syntactic locality. For example, the axiom  $\alpha = (A \sqsubseteq A \sqcup B)$  is local w.r.t. every  $\mathbf{S}$  since it is a tautology, but it is not syntactically local w.r.t.  $\mathbf{S} = \{A, B\}$  since it involves symbols in  $\mathbf{S}$  only.

$\mathbf{I}_{A \leftarrow *}(S)$	$r, A \notin S : r^{\mathcal{J}} A^{\mathcal{J}}$	$\mathbf{I}_{A \leftarrow *}(S)$	$r, A \notin S : r^{\mathcal{J}} A^{\mathcal{J}}$
$\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(S)$	$\emptyset \quad \emptyset$	$\mathbf{I}_{A \leftarrow \Delta}^{r \leftarrow \emptyset}(S)$	$\emptyset \quad \Delta^{\mathcal{J}}$
$\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \Delta \times \Delta}(S)$	$\Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \quad \emptyset$	$\mathbf{I}_{A \leftarrow \Delta}^{r \leftarrow \Delta \times \Delta}(S)$	$\Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \quad \Delta^{\mathcal{J}}$
$\mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow \text{id}}(S)$	$\{\langle x, x \rangle \mid x \in \Delta^{\mathcal{J}}\} \quad \emptyset$	$\mathbf{I}_{A \leftarrow \Delta}^{r \leftarrow \text{id}}(S)$	$\{\langle x, x \rangle \mid x \in \Delta^{\mathcal{J}}\} \quad \Delta^{\mathcal{J}}$

$\alpha$ Axiom	$? \alpha \in \mathbf{Ax}$	$\begin{matrix} r \leftarrow \emptyset \\ A \leftarrow \emptyset \end{matrix}$	$\begin{matrix} r \leftarrow \Delta \times \Delta \\ A \leftarrow \emptyset \end{matrix}$	$\begin{matrix} r \leftarrow \text{id} \\ A \leftarrow \emptyset \end{matrix}$	$\begin{matrix} r \leftarrow \emptyset \\ A \leftarrow \Delta \end{matrix}$	$\begin{matrix} r \leftarrow \Delta \times \Delta \\ A \leftarrow \Delta \end{matrix}$	$\begin{matrix} r \leftarrow \text{id} \\ A \leftarrow \Delta \end{matrix}$
P4 $\exists \text{has\_Focus}.\top \sqsubseteq \text{Project}$		✓	✗	✗	✓	✓	✓
P5 $\text{BioMedical\_Project} \equiv \text{Project} \sqcap \text{Bio\_Medicine} \sqcap \exists \text{has\_Focus}.\text{Bio\_Medicine}$		✓	✓	✓	✗	✗	✗
P6 $\text{Project} \sqcap \text{Bio\_Medicine} \sqsubseteq \perp$		✓	✓	✓	✗	✗	✗
P7 $\text{Funct}(\text{has\_Focus})$		✓	✗	✓	✓	✗	✓
P8 $\text{Human\_Genome} : \text{Project}$		✗	✗	✗	✓	✓	✓
P9 $\text{has\_Focus}(\text{Human\_Genome}, \text{Gene})$		✗	✓	✗	✗	✓	✗
E2 $\forall \text{has\_focus}.\text{Cystic\_Fibrosis} \sqsubseteq \sqsubseteq \exists \text{has\_Focus}.\text{Cystic\_Fibrosis}$		✗	✗	✗	✗	✗	✗

Table 1: Examples for and Comparison Between Different Local Classes of Interpretations

The locality condition in Example 1 is just a particular example of a locality class. Other classes of local interpretations can be constructed in a similar way, by fixing the interpretations of the symbols not in  $S$  to different values. In Table 1 we provide several such classes of local interpretations by fixing the interpretation of atomic roles outside  $S$  to either the empty set  $\emptyset$ , the universal relation  $\Delta \times \Delta$ , or the identity relation  $\text{id}$  on  $\Delta$ , and the interpretation of atomic concepts outside  $S$  to either the empty set  $\emptyset$  or the set  $\Delta$  of all domain elements. Each class of local interpretations in Table 1 defines a corresponding class of local ontologies. In Table 1 we have listed all of these classes together with examples of typical types of axioms used in ontologies. Table 1 shows that different types of locality conditions are appropriate for different types of axioms. Note that E2 is not local for any of our locality conditions, since E2 is not safe for  $S$ .

One could design algorithms for testing locality for the classes of interpretations in Table 1 similar to the one presented in Proposition 2. E.g., locality for the class  $\mathbf{I}_{A \leftarrow \Delta}^{r \leftarrow \emptyset}(S)$  can be tested as in Proposition 2, where the case (a) of the definition for  $\tau(C, S)$  is replaced with: “ $\tau(A, S) = \top$  if  $A \notin S$  and otherwise  $= A$ ”. For the remaining classes of interpretations, that is for  $\mathbf{I}_{A \leftarrow *}^{r \leftarrow \Delta \times \Delta}(S)$  and  $\mathbf{I}_{A \leftarrow *}^{r \leftarrow \text{id}}(S)$ , checking locality is not straightforward, since it is not clear how to eliminate the universal roles and identity roles from the axioms and preserve validity in the respective classes of interpretations. Still, it is easy to design tractable syntactic approximations for all these locality conditions by modifying Definition 6 accordingly. In Figure 3 we give recursive definitions for syntactically local axioms  $\tilde{\mathbf{Ax}}_{A \leftarrow *}(S)$  that correspond to the classes of interpretations  $\mathbf{I}_{A \leftarrow *}(S)$  from Table 1, where some cases in the recursive definitions are present only for the indicated classes of interpretations.

In order to check safety in practice, one may try to apply different sufficient tests and check if any of them succeeds. For such a purpose, one could combine two dif-

$$\begin{array}{ll}
 \mathbf{Con}^{\emptyset}(\mathbf{S}) ::= (\neg C^{\Delta}) \mid (C \sqcap C^{\emptyset}) & \mathbf{Con}^{\Delta}(\mathbf{S}) ::= (\neg C^{\emptyset}) \mid (C_1^{\Delta} \sqcap C_2^{\Delta}) \\
 \mid (\exists R.C^{\emptyset}) \mid (\geq n R.C^{\emptyset}) & \mathbf{I}_{A \leftarrow \Delta}^{r \leftarrow * }(\cdot) : \mid A^{\Delta} \\
 \mathbf{I}_{A \leftarrow \emptyset}^{r \leftarrow * }(\cdot) : \mid A^{\emptyset} & \mathbf{I}_{A \leftarrow * }^{r \leftarrow \Delta \times \Delta}(\cdot) : \mid (\exists R^{\Delta \times \Delta}.C^{\Delta}) \mid (\geq n R^{\Delta \times \Delta}.C^{\Delta}) \\
 \mathbf{I}_{A \leftarrow * }^{r \leftarrow \emptyset}(\cdot) : \mid (\exists R^{\emptyset}.C) \mid (\geq n R^{\emptyset}.C) & \mathbf{I}_{A \leftarrow * }^{r \leftarrow \text{id}}(\cdot) : \mid (\exists R^{\text{id}}.C^{\Delta}) \mid (\geq 1 R^{\text{id}}.C^{\Delta}). \\
 \mathbf{I}_{A \leftarrow * }^{r \leftarrow \text{id}}(\cdot) : \mid (\geq m R^{\text{id}}.C), m \geq 2. & \\
 \\
 \mathbf{Ax}_{A \leftarrow * }^{r \leftarrow * }(\mathbf{S}) ::= C^{\emptyset} \sqsubseteq C \mid C \sqsubseteq C^{\Delta} \mid a : C^{\Delta} & \text{Where:} \\
 \mathbf{I}_{A \leftarrow * }^{r \leftarrow \emptyset}(\cdot) : \mid R^{\emptyset} \sqsubseteq R \mid \text{Trans}(r^{\emptyset}) \mid \text{Func}t(R^{\emptyset}) & A^{\emptyset}, A^{\Delta}, r^{\emptyset}, r^{\Delta \times \Delta}, r^{\text{id}} \notin \mathbf{S}; \\
 \mathbf{I}_{A \leftarrow * }^{r \leftarrow \Delta \times \Delta}(\cdot) : \mid R \sqsubseteq R^{\Delta \times \Delta} \mid \text{Trans}(r^{\Delta \times \Delta}) \mid r^{\Delta \times \Delta}(a, b) & R^{\emptyset}, R^{\Delta \times \Delta}, R^{\text{id}} \notin \mathbf{RolS}; \\
 \mathbf{I}_{A \leftarrow * }^{r \leftarrow \text{id}}(\cdot) : \mid \text{Trans}(r^{\text{id}}) \mid \text{Func}t(R^{\text{id}}) & C^{\emptyset} \in \mathbf{Con}^{\emptyset}(\mathbf{S}), C_i^{\Delta} \in \mathbf{Con}^{\Delta}(\mathbf{S}); \\
 & C \text{ is any concept, } R \text{ is any role}
 \end{array}$$

Fig. 3: Syntactic Approximations to the Locality Classes

ferent safety classes and obtain a more powerful one by checking whether an ontology satisfies either the first or the second condition. The combination can be achieved by forming a union of safety classes: given two safety classes  $\mathbf{O}_1(\cdot)$  and  $\mathbf{O}_2(\cdot)$ , their union  $(\mathbf{O}_1 \cup \mathbf{O}_2)(\cdot)$  defined by  $(\mathbf{O}_1 \cup \mathbf{O}_2)(\mathbf{S}) = \mathbf{O}_1(\mathbf{S}) \cup \mathbf{O}_2(\mathbf{S})$ , also gives a safety class. It is easy to demonstrate that if both safety classes  $\mathbf{O}_1(\cdot)$  and  $\mathbf{O}_2(\cdot)$  are anti-monotonic or subset-closed then their union is also anti-monotonic or subset-closed. Unfortunately the union-closure property for safety classes is not preserved under union of safety classes. For example, the union  $(\mathbf{O}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset} \cup \mathbf{O}_{A \leftarrow \Delta}^{r \leftarrow \Delta \times \Delta})(\cdot)$  of the classes  $\mathbf{O}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\cdot)$  and  $\mathbf{O}_{A \leftarrow \Delta}^{r \leftarrow \Delta \times \Delta}(\cdot)$  is not union-closed since it captures, for example, the ontology  $\mathcal{O}_1$  consisting of axioms P4–P7 from Table 1, which satisfies the first locality condition, the ontology  $\mathcal{O}_2$  consisting of axioms P8–P9 satisfies the second locality condition, but their union  $\mathcal{O}_1 \cup \mathcal{O}_2$  is not even safe for  $\mathbf{S}$ .

It can be shown that the classes  $\mathbf{O}_{A \leftarrow \emptyset}^{r \leftarrow \emptyset}(\cdot)$  and  $\mathbf{O}_{A \leftarrow \Delta}^{r \leftarrow \Delta \times \Delta}(\cdot)$  of local ontologies are maximal union-closed safety classes for  $\mathcal{SHIQ}$ —that is, there is no union-closed class that strictly extends them.

We have verified empirically that syntactic locality provides a powerful sufficient test for safety which works for many real-world ontologies. We have implemented a (syntactic) locality checker and run it over ontologies from a library of 300 ontologies of various sizes and complexity some of which import each other [6].<sup>1</sup> For all ontologies  $\mathcal{P}$  that import an ontology  $\mathcal{Q}$ , we check syntactic locality of  $\mathcal{P}$  for  $\mathbf{S} = \text{Sig}(\mathcal{P}) \cap \text{Sig}(\mathcal{Q})$ .

It turned out that from 96 ontologies that import other ontologies, all but 11 are syntactically local w.r.t. the given interface signature. From the 11 non local ontologies, 7 are written in the OWL-Full species of OWL to which our framework does not yet apply. The remaining 4 non-localities are due to the presence of so-called *mapping axioms* of the form  $A \equiv B'$ , where  $A \notin \mathbf{S}$  and  $B' \in \mathbf{S}$ . Note that these axioms simply indicate that the concept names  $A, B'$  in the two ontologies under consideration are synonyms. Indeed, we were able to easily fix these non-localities as follows: we replace every occurrence of  $A$  in  $\mathcal{P}$  with  $B'$  and then remove this axiom from the ontology. After this transformation, all 4 non-local ontologies turned out to be local.

<sup>1</sup> The library is available at <http://www.cs.man.ac.uk/~horrocks/testing/>

## 4 Outlook

This paper extends the framework for modular reuse of ontologies presented in [3]. We have formalized the notion of safe reuse of ontologies. We have shown that checking safety of an ontology w.r.t. a signature is undecidable for  $\mathcal{ALCO}$ . We have provided a general notion of a sufficient condition for checking safety—a safety class—and examples of safety classes based on semantic and syntactic restrictions. The former can be checked using a reasoner and the latter can be checked syntactically in polynomial time. It turns out that these sufficient conditions for safety work surprisingly well for many real-world ontologies. In a recent paper [2], we have also demonstrated how to use safety classes for extracting modules from ontologies.

## References

1. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives of Mathematical Logic. Springer-Verlag, 1997. Second printing (Universitext) 2001.
2. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: Extracting modules from ontologies. In *Proc. of the 16th International World Wide Web Conference (WWW-2007)*. ACM, 2007.
3. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. A logical framework for modularity of ontologies. In *Proc of IJCAI 2007, Hyderabad, India, January 2007*, pages 298–304. AAAI, 2007.
4. B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Ontology reuse: Better safe than sorry, 2007. Available at <http://www.cs.man.ac.uk/~bcg/extended.pdf>.
5. B. Cuenca Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and Web Ontologies. In *Proceedings of KR-2006, June 2-5, 2006*, pages 198–209. AAAI Press, 2006.
6. T. Gardiner, D. Tsarkov, and I. Horrocks. Framework for an automated comparison of description logic reasoners. In *Proc. of ISWC 2006, Athens, GA, USA*, pages 654–667, 2006.
7. S. Ghilardi, C. Lutz, and F. Wolter. Did I Damage my Ontology? A Case for Conservative Extensions in Description Logics. In *Proceedings, KR-2006, Lake District, UK, June 2-5, 2006*, pages 187–197. AAAI Press, 2006.
8. C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. In *Proc. of IJCAI-2007, India, January 2007*, pages 453–459. AAAI, 2007.
9. N. Noy. Semantic integration: A survey on ontology-based approaches. *SIGMOD Record*, 33:65–70, 2004.
10. J. Seidenberg and A. Rector. Web ontology segmentation: Analysis, classification and use. In *Proceedings of WWW 2006, Edinburgh, UK, May 23-26, 2006*, pages 13–22. ACM, 2006.

## A New Mapping from $\mathcal{ALCT}$ to $\mathcal{ALC}$

Yu Ding and Volker Haarslev and Jiewen Wu

Concordia University, Montreal, Quebec, Canada  
 {ding\_yu|haarslev|w\_jiewen}@cse.concordia.ca

### 1 Introduction

It is well known that a *correspondence theory* for description logics (DLs), propositional modal logics (MLs) and propositional dynamic logics (PDLs) was given in [Sch91] and that an *axiomatic system* for a description logic with inverse roles was presented in the same paper. Schild’s paper covers what is to be discussed in this paper although his paper did not mention *nominals*<sup>1</sup> as well as expressive *roles* explicitly. On the other hand, reductions to eliminate *converse programs* (*inverse roles* in DLs) are known even for full PDL ( $\mathcal{ALCT}_{reg}$ ); the original reduction from converse PDL to PDL was in [Gia96]. Moreover, a direct tableaux method for converse PDL and further discussions on the elimination of *converse programs* were given in [GM00]. Besides *graded modalities*, it was pointed out in [Gia96] that the previous reduction technique is also applicable to *nominals*.

This paper shares some viewpoints well made in [Gia96]. The proposed mapping process here is based on the simple idea to capture possible back-propagation caused by the use of inverse roles<sup>2</sup>. This process consists of three steps, *tagging*, *recording*, *polarisation*, which are introduced below. Concept expressions/formulae are assumed to be in *negation norm form* (NNF). For simplicity, *existential restrictions* and *universal restrictions* are called *modal constraints* somewhere. We refer to [BCM<sup>+</sup>03] for usual background knowledge on *description logics* (DLs).

### 2 Concept Satisfiability with General Concept Inclusions

Concept expressions/formulae are in NNF. For simplicity, we consider a GCI (*general concept inclusion*) of the form  $\top \sqsubseteq C$ , where  $C$  is in NNF. The following shows how to use three simple steps (*tagging*, *recording* and *polarisation*) to convert a concept formula (in NNF) of a source logic with *inverse roles* to a target logic without *inverse roles*. Terminological knowledge bases with general concept inclusions (GCIs) are also considered.

<sup>1</sup> Such viewpoint can be found in the early literature on *hybrid logics*, which are logics extending the *propositional modal logic* with *nominals* (a.k.a. named states).

<sup>2</sup> The proposed mapping technique also relies on the model properties of the description logics concerned in the paper.

**Definition 1. (Tagging-1)** *The tagging technique introduces new concept names for modal constraints of concept expressions and axioms. The function  $tag(\cdot)$  on a concept formula  $x$  is defined as:*

- (1) if  $x$  is  $C \sqcap D$ , then  $tag(x) = tag(C) \sqcap tag(D)$ ;
- (2) if  $x$  is  $C \sqcup D$ , then  $tag(x) = tag(C) \sqcup tag(D)$ ;
- (3) if  $x$  is  $\exists R.C$ , then  $tag(x) = \exists R.(tag(C))$ ;
- (4) if  $x$  is  $\forall R.C$ , then  $tag(x) = Q(x) \oplus \forall R.(tag(C))$ ;
- (5) if  $x$  is  $\top \sqsubseteq C$ , then  $tag(x) = \top \sqsubseteq tag(C)$ ;
- (6) otherwise  $tag(x) = x$ .

where  $Q(x)$  is a fresh name unique for each  $x$ ;  $C, D$  are subformulae; the symbol  $\oplus$  represents the conjunction operator in exactly the same way as  $\sqcap$ .

$\mathcal{U}(R)$  denotes tagged *universal constraints* of the form  $Q(x) \oplus \forall R.(tag(C))$ , where  $R$  is a role and  $\mathcal{U}$  are sets indexed by roles.  $E_0/E_1$  denotes the formulae before and after tagging;  $\mathcal{K}_0/\mathcal{K}_1$  denotes the GCI before and after tagging. Let  $\forall R.C$  be a subformula before tagging, we have after tagging:

- ( $\star$ ) for  $x = \forall R.C$ , there is  $Q(x) \oplus \forall R.tag(C) \in \mathcal{U}(R)$ ;

**Definition 2. (Recording-1)** *Initialize  $\mathcal{K}_a = \emptyset$ . For each set  $\mathcal{U}(R)$  indexed by each role  $R$ , and for each element  $(Q(x) \oplus \forall R.(tag(C))) \in \mathcal{U}(R)$ , perform the operation:  $\mathcal{K}_a = \mathcal{K}_a \cup \{\top \sqsubseteq tag(C) \sqcup \forall R^-. \neg Q(x)\}$ .*

**Definition 3. (Polarisation-1)**  *$Pol(x)$  is performed on the tagged input formula  $E_1$  to get a polarized  $E_2$ , and on  $\mathcal{K}_1 \cup \mathcal{K}_a$  to get the polarized  $\mathcal{K}_2$ :*

- (1) if  $x$  is  $C \sqcap D$ , then  $Pol(x) = Pol(C) \sqcap Pol(D)$ ;
- (2) if  $x$  is  $C \sqcup D$ , then  $Pol(x) = Pol(C) \sqcup Pol(D)$ ;
- (3) if  $x$  is  $\exists R.C$ , then  $Pol(x) = \exists R^a.Pol(C)$ ;
- (4) if  $x$  is  $\forall R.C$ , then  $Pol(x) = \forall R^a.Pol(C)$ ;
- (5) if  $x$  is  $\exists R^-.C$ , then  $Pol(x) = \exists R^b.Pol(C)$ ;
- (6) if  $x$  is  $\forall R^-.C$ , then  $Pol(x) = \forall R^b.Pol(C)$ ;
- (7) if  $x$  is  $\top \sqsubseteq C$ , then  $Pol(x) = \top \sqsubseteq Pol(C)$ ;
- (8) otherwise,  $Pol(x) = x$ .

where  $R^a$  ( $R^b$ ) is a fresh role name unique for  $R$  ( $R^-$ ).

### 3 Concept Satisfiability/Abox Consistency with Tbox

The Tbox (a.k.a. terminological box) is a set of unfoldable axioms. The notion of Tbox is related some fundamental notions such as *name unfolding* and GCI *absorption* [BCM<sup>+</sup>03]. By *descriptive semantics, equality axioms* like  $A \equiv C$  are expressed in two *inclusion axioms*  $A \sqsubseteq C$  and  $\neg A \sqsubseteq \neg C$ . The right-hand-sides of the axioms are in MNF. An acyclic Tbox of only such inclusion axioms is called *simplified* [Lut99]. In the following, we show how to use the three simple steps (i.e., *tagging, recording, polarisation*) for Tboxes.

**Definition 4. (Acyclic Ordering)** *The ordering relation<sup>3</sup> is as following:*

<sup>3</sup> Due to acyclicity,  $ord(A) \succ ord(A)$  is not induced.

- (1) for each axiom  $A \sqsubseteq C$ , there is  $\text{ord}(A) \succ \text{ord}(C)$ ;
- (2)  $\text{ord}(C \sqcap D) \succ \text{ord}(C)$  and  $\text{ord}(C \sqcap D) \succ \text{ord}(D)$ ;
- (3)  $\text{ord}(C \sqcup D) \succ \text{ord}(C)$  and  $\text{ord}(C \sqcup D) \succ \text{ord}(D)$ ;
- (4)  $\text{ord}(\exists R.C) \succ \text{ord}(C)$ ;
- (5)  $\text{ord}(\forall R.C) \succ \text{ord}(C)$ ;

An Abox consists of *concept assertions* and *role assertions*. If an Abox has several unconnected components, each of them can be treated alike separately. We assume one individual has at most one label because  $d : C$  and  $d : D$  can be replaced by  $d : C \sqcap D$ . W.l.o.g. we consider a single-component Abox  $\mathcal{A}_0$  and each individual has at most one label. We denote the label for  $d_i$  as  $\mathcal{L}(d_i)$ .

**Definition 5. (Tagging-2)** The function  $\text{tag}(x)$  is:

- (1) if  $x$  is  $A \sqsubseteq C$ , then  $\text{tag}(x) = A \sqsubseteq \text{tag}(C)$ ;
- (2) if  $x$  is  $\exists R.C$ , then  $\text{tag}(x) = P(x) \oplus \exists R.(\text{tag}(C))$ ;
- (3) if  $x$  is individual  $d_i$  with  $\mathcal{L}(d_i)$ , then  $\text{tag}(x) = d_i : P(x) \oplus \text{tag}(\mathcal{L}(d_i))$ ;
- (4) if  $x$  is an individual  $d_i$  of no label, then  $\text{tag}(x) = d_i : P(x)$ ;
- (5) otherwise, call *Tagging-1* for  $x$ .

where  $P(x)$  is a unique name for each  $x$ , and the sign  $\oplus$  stands for  $\sqcap$ .

The original Abox/Tbox are denoted as  $\mathcal{A}_0/\mathcal{T}_0$ , their tagged counterparts are denoted as  $\mathcal{A}_1/\mathcal{T}_1$ . Notice we do not tag any *role assertions*. We also write  $P(d_i)$  instead of  $P(x)$  if the tag is for an individual  $d_i$ . The set of tags for all individuals of the Abox is  $\mathcal{D} = \{P(d_i) | d_i \in \mathcal{A}_0\}$ . Let  $C$  denote any (sub)formula:

- ( $\star$ ) for  $x = \exists R.C$ , there is  $P(x) \oplus \exists R.\text{tag}(C) \in \mathcal{E}(R)$ ;
- ( $\star$ ) for  $y = \forall R.C$ , there is  $P(y) \oplus \forall R.\text{tag}(C) \in \mathcal{U}(R)$ ;
- ( $\star$ ) for  $z = d_i$ , there is  $P(d_i) \in \mathcal{D}$ ;

We additionally stipulates  $\text{ord}(P(d_i)) \succ \text{ord}(\text{tag}(\mathcal{L}(d_j)))$  for any individual  $d_i$  and  $d_j$ . This forces  $P(d_i)$  to get a higher order than  $\text{tag}(\mathcal{L}(d_j))$  (and higher than subformulae of  $\text{tag}(\mathcal{L}(d_j))$ ) but does not introduce cycles<sup>4</sup>.

**Definition 6. (Recording-2)** For two tuples  $\beta \in \mathcal{U}(\ast)$  and  $\alpha \in (\mathcal{E}(\ast) \cup \mathcal{U}(\ast) \cup \mathcal{D})$  where  $\ast$  denotes any role name, if the following conditions are met:

- (1)  $\text{ord}(\alpha) \succ \text{ord}(\beta)$ ; and
- (2)  $\alpha = P(x) \oplus \forall R_1.\text{tag}(C)$  or  
 $\alpha = P(x) \oplus \exists R_1.\text{tag}(C)$  or  
 $\alpha = P(x)$  and  $x$  is some Abox individual  $d_i$ ; and
- (3)  $\beta = P(y) \oplus \forall R_2.\text{tag}(D)$ ;

then perform the operation:  $\mathcal{T}_a = \mathcal{T}_a \cup \{P(x) \sqsubseteq \forall R_2^- . \neg P(y) \sqcup \text{tag}(D)\}$ .

**Definition 7. (Polarisation-2)**  $\text{Pol}(x)$  is performed on the tagged Abox  $\mathcal{A}_1$  to get  $\mathcal{A}_2$ , and on the augmented Tbox  $\mathcal{T}_1 \cup \mathcal{T}_a$  to get  $\mathcal{T}_2$ :

- (1) if  $x$  is  $A \sqsubseteq C$ , then  $\text{Pol}(x) = A \sqsubseteq \text{Pol}(C)$ ;

<sup>4</sup> Please notice  $\succ$  is transitive. The extra requirement forces  $\text{ord}(P(d_i) \oplus \text{tag}(\mathcal{L}(d_i))) \succ \text{ord}(P(d_i)) \succ \text{ord}(\text{tag}(\mathcal{L}(d_i)))$ . For  $i \neq j$ , we have: (1)  $\text{ord}(\text{tag}(\mathcal{L}(d_i)))$  and  $\text{ord}(\text{tag}(\mathcal{L}(d_j)))$  are incomparable; (2)  $\text{ord}(P(d_i))$  and  $\text{ord}(P(d_j))$  are incomparable; (3)  $\text{ord}(P(d_i)) \succ \text{ord}(\text{tag}(\mathcal{L}(d_j)))$ .

- (2) if  $x$  is  $(c, d) : R \in \mathcal{A}_1$ , then  $\mathcal{A}_2 = \mathcal{A}_1 \cup \{(c, d) : R^a, (d, c) : R^b\}$ ;
  - (3) if  $x$  is  $(c, d) : R^- \in \mathcal{A}_1$ , then  $\mathcal{A}_2 = \mathcal{A}_1 \cup \{(d, c) : R^a, (c, d) : R^b\}$ ;
  - (4) if  $x$  is  $d_i : P(d_i) \oplus \mathcal{L}(d_i) \in \mathcal{A}_1$ , then  $\mathcal{A}_2 = \mathcal{A}_1 \cup \{d_i : P(d_i) \oplus Pol(\mathcal{L}(d_i))\}$ ;
  - (5) otherwise, call *Polarisation-1*.
- where  $R^a$  ( $R^b$ ) is a fresh role name unique for  $R$  ( $R^-$ ).

Though in the above only acyclic Tboxes are emphasized, the exact mapping method also applies to cyclic Tboxes by simply dropping the acyclic ordering condition prescribed at the *recording* step.

## 4 Experiments

We have implemented the mapping as presented in Section 1 to evaluate its practicality. All satisfiability tests were performed with RacerPro 1.9.0 on a Pentium PC with 3.5 GB memory. The tested ontologies were also converted on the same machine. Note that the expressivity of the original ontologies is  $\mathcal{ALCI}$ .

KB Name	Coherence Check (original Tbox)	Conversion	Coherence Check (converted Tbox)
galen-ir1-alc1-new1	9.141	50.657	84.625
galen-ir2-alc1-new1	9.549	52.547	76.156
uml-no-max-min-new4	timeout after 1 hour	1.156	0.110
revised-9-alc1 (partial)	timeout after 20 mins	17.016	3.297

**Table 1.** Experimental results (all times are given in seconds)

KB Name	Num. of Axioms (original/converted)	Classes/Properties (original Tbox)	Classes/Properties (converted Tbox)
galen-ir1-alc1-new1	4645/5495	3107/234	3597/228
galen-ir2-alc1-new1	4666/5508	3107/234	3597/228
uml-no-max-min-new4	524/739	233/213	448/213
revised-9-alc1 (partial)	3077/3099	2427/56	2449/37

**Table 2.** KBs before and after conversion (number of axioms, classes and properties)

Table 1 shows some empirical results (coherence check only), where the time indicated is the average of 5 independent runs of the conversion system. It can be seen that although more time is spent for testing Tbox coherence for the converted versions of the first two KBs, the performance is still acceptable since the KB sizes after conversion are nearly five times of the original ones. Evidently, for the UML ontology the runtime after conversion is quite impressive. Besides, we have also divided the UML ontology into two sub-ontologies, both of which, if converted, require less time to compute the satisfiability of all the concepts. Dramatic increase of performance is shown in the last case, where the ontology contains one major class extracted from ontology “revised-9-alc1”.

## 5 Discussions

The question about whether a *decision procedure* that has to work both “forward” and “backward” could be implemented to run efficiently was raised long ago in the literature, among them we mention [Gia96]. Nonetheless, the highly optimized tableau-based reasoning systems (the 3rd generation [BCM<sup>+</sup>03]) are convincingly found “to behave quite well” in practice for many realistic problems. As *description logics* are widely used in diverse application domains, different “application patterns” produce a lot of realistic problems that might not be quite “tractable” as previous ones in terms of “problem size” and “problem structure”. Several application domains are known to easily give “practically intractable problems”, e.g., the *model checking* field, probably due to their indistinct narrative styles, i.e., extensive use of tightly constrained constraints. Recently, it was even found that some small-size ontologies are “hard” enough to kill some best tableau-based DL systems currently available. The existence of latest “intractable realistic problems” is more baffling than any indistinct narrative style that people have seen before.

Schild has provided an axiomatic system [Sch91] for  $\mathcal{ALCI}$ , the DL extending the basic description logic  $\mathcal{ALC}$  with *inverse roles*. A *correspondence theory* for description logics, propositional modal logics and propositional dynamic logics was also given in [Sch91]. In [Gia95] and [CGR98], the “converse elimination technique” was presented for the CPDL and  $\mathcal{ALCI}_{reg}$ . Their technique is more general than what is presented in this paper and was extended in various aspects. Important literature on “converse elimination” and a direct tableaux approach include [Gia96] and [GM00]. Their transformation leads to target problems in the ExpTime class. Their technique could possibly lead to good implementations in practice. However, it is not very clear if there was any empirical result about their elimination of converse for the so-called “realistic problems”.

A worst-case optimal tableau procedure for testing concept satisfiability w.r.t. general Tbox was given in [DM00] for  $\mathcal{ALC}$  in details. Their technique of *caching* intermediate results and nogoods has deep influence on tableau-based DL systems. The belief that description logics without inverse roles lend themselves better to optimisations (e.g. the caching technique) originates from [DM00] and is well supported from practice. Lutz discussed the complexity cliff phenomenon for the problem of concept satisfiability test w.r.t. an acyclic Tbox in several logics in [Lut99]. One of the results showed is a tableaux procedure that takes a polynomial space for concept satisfiability test w.r.t. an acyclic Tbox in  $\mathcal{ALC}$ . The *pre-completion technique* was proposed in [DLNS94] [Hol94] for reducing the Abox consistency problem to a number<sup>5</sup> of concept satisfiability tests to be carried out independently. For a pre-completion technique for  $\mathcal{SH}$  Abox (which strictly contains the logic  $\mathcal{ALC}$ ) w.r.t. Tbox, see [TG99].

The proposed mapping in this paper allows tableau-based decision procedures to safely use the *global sub-tableaux caching technique*. This gives a hope that the run-time performance should not be much worse than using the *dynamic block-*

<sup>5</sup> It is the exact number of Abox individuals.

ing and the *pseudo-model merging* techniques, two best optimisations currently available [BCM<sup>+</sup>03] to tableau-based decision procedures for DLs with inverse roles. This conjecture is supported by our first-hand experiments. Further, the *pseudo-model merging* technique coexists with the *global sub-tableaux caching* technique. Also, the new axioms introduced in the recording step can be “selectively” used to simulate the well-known *tableau expansion rules* in such a way that if the well-known tableau algorithm (that allows bi-directional propagation of constraints) constructs a pre-model for the source problem, then this construction process can be repeated to construct a pre-model for the target problem at an equal cost. The only possible disadvantage of the proposed mapping is that it introduces extra concept names and extra axioms (with one disjunction per axiom). However it should be noted that these extra names and axioms are for “simulating” the well-known tableau expansion rules that rely on the *dynamic blocking technique*. Moreover, the newly recorded axioms are not necessarily GCIs but can always be unfoldable axioms (as shown in Section 3).

We require each role has a unique inverse role. For a role  $R$ , for example, we consider  $R^-$  as the only *inverse role*. This takes a linear cost. The presented transformation is *equisatisfiability preserving*, and is fine-grained in the sense that the target problems stay in the same complexity class as the source problems. The *recording* operation descends from the C-rule (the Ramsey-Rule) which states an equivalence of  $\exists R.C \sqsubseteq D$  and  $C \sqsubseteq \forall R^-.D$  [Ram31]. This equivalence was rediscovered in DLs and was lately used for new *absorption techniques*, for example [HW06] and [SGP06].

This paper largely follows and extends our previous work in [DH05]. In [DH05], we proposed three different ways to deal with  $\mathcal{ALCI}$ . The first was a *dynamic caching* technique that extends the *dynamic blocking* technique to work on different traces and thus allows *anywhere blocking*. The second was a *reachability analysis* to guarantee the soundness of the *global sub-tableaux caching* technique through a pre-compilation of a Tbox. The third was about the equivalence mentioned above. In this paper, rather than to enrich the *absorption technique* as previously perceived, we used the equivalence in a different and novel way. Here we have presented two versions of a mapping technique to deal with GCIs and (acyclic) Tboxes<sup>6</sup>. It also works for nominals and expressive roles. Moreover, the mapping without *polarisation* is quite interesting in itself for it brings a *back-propagation don't-care property* for the target problems (now in DLs with inverse roles). Here is a summary conclusion of the proposed mapping technique:

- every knowledge base in fragments of  $\mathcal{SHOI}$  that contain  $\mathcal{SHI}$  (or  $\mathcal{ALCOI}$ ) can be converted to a unfoldable Tbox in the corresponding fragments containing  $\mathcal{SH}$  (or  $\mathcal{ALCO}$ );
- every (acyclic) unfoldable Tbox in  $\mathcal{ALCHI}$  (or its fragments) can be converted to an (acyclic) unfoldable Tbox in  $\mathcal{ALCH}$  (or corres. fragments);
- the mapping is fine-grained in the sense that the target problems stay in the same complexity class as the source problems.

<sup>6</sup> The transformation is presented for the Abox consistency check problem w.r.t. (acyclic) Tboxes.

It is observed in our current experiments that some (not all) very hard ontologies the coherence of which could not even be tested are able to be classified in reasonable time. For optimisations of the classification, they are beyond the satisfiability test based (tableau-based) decision procedures. It is well known that classification could even be done without resorting to any satisfiability test at all, for example [BHN<sup>+</sup>92] [TH05]. Right now, we are preparing an optimized and extended implementation. An in-depth empirical analysis is under way. For a parallel work on a worst-case ExpTime (binary coding of numbers) tableau-based decision procedure for  $\mathcal{ALCQI}$ , a description logic containing both *qualified number restrictions* and *inverse roles*, see [DH07].

## 6 Acknowledgments

The authors thank referees for giving many insightful comments so that an improvement on the paper is possible; and thank Xi Deng for proof-reading the latest version. Also sincere thanks from the first author goes to Angel for helping a draft version before the Christmas season.

## References

- [BCM<sup>+</sup>03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [BHN<sup>+</sup>92] Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profittlich, and Enrico Franconi. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems, or Making KRIS Get a Move On. In *KR*, pages 270–281, 1992.
- [CGR98] Diego Calvanese, Giuseppe De Giacomo, and Riccardo Rosati. A Note on Encoding Inverse Roles and Functional Restrictions in  $\mathcal{ALC}$  Knowledge Bases. *CEUR Workshop DL-98*, 1998.
- [DH05] Yu Ding and Volker Haarslev. Towards Efficient Reasoning for Description Logics with Inverse Roles. *DL-Workshop’05*, 2005.
- [DH07] Yu Ding and Volker Haarslev. An ExpTime Tableau-based Decision Procedure for  $\mathcal{ALCQI}$ . *DL-Workshop’07*, 2007.
- [DLNS94] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Deduction in Concept Languages: From Subsumption to Instance Checking. *J. Log. Comput.*, 4(4):423–452, 1994.
- [DM00] Francesco M. Donini and Fabio Massacci. EXPTIME Tableaux for  $\mathcal{ALC}$ . *Artificial Intelligence*, 124(1):87–138, 2000.
- [FL79] Michael J. Fischer and Richard E. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.
- [Gia95] Giuseppe De Giacomo. *Decidability of Class-based Knowledge Representation Formalisms*. PhD thesis, Università di Roma “La Sapienza”, Roma, Italy, 1995.

- [Gia96] Giuseppe De Giacomo. Eliminating “Converse” from Converse PDL. *Journal of Logic, Language and Information*, 5(2):193–208, 1996.
- [GM00] Giuseppe De Giacomo and Fabio Massacci. Combining Deduction and Model Checking into Tableaux and Algorithms for Converse-PDL. *Inf. Comput.*, 162(1-2):117–137, 2000.
- [Hol94] B. Hollunder. *Algorithmic Foundations of Terminological Knowledge Representation Systems*. PhD thesis, Universit, at des Saarlandes, Germany, 1994.
- [HW06] Alexander K. Hudek and Grant Weddell. Binary Absorption in Tableaux-Based Reasoning for Description Logics. *DL-Workshop’06*, 2006.
- [Lut99] Carsten Lutz. Complexity of Terminological Reasoning Revisited. *LPAR-1999*, 99:181–200, 1999.
- [Ram31] F. Plank Ramsey. *Truth and Probability*. In Foundations of Mathematics and Other Logical Essays, R.B. Braithwaite, ed., New York., 1931.
- [Sav70] Walter J. Savitch. Relationships between Nondeterministic and Deterministic Tape Complexities. *J. of Computer and System Sci.*, 4(2):177–192, 1970.
- [Sch91] Klaus Schild. A Correspondence Theory for Terminological Logics: Preliminary Report. *IJCAI*, pages 466–471, 1991.
- [SGP06] Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From wine to water: Optimizing description logic reasoning for nominals. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *KR*, pages 90–99. AAAI Press, 2006.
- [Tes01] Sergio Tessaris. *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, Universit of Manchester, Manchester, UK, 2001.
- [TG99] Sergio Tessaris and Graham Gough. Abox Reasoning with Transitive Roles and Axioms. *DL WorkShop’99*, 1999.
- [TH05] Dmitry Tsarkov and Ian Horrocks. Optimised Classification for Taxonomic Knowledge Bases. In Ian Horrocks, Ulrike Sattler, and Frank Wolter, editors, *Description Logics*, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

## A One Working Example

Give a concept  $E_0 = \exists s^-.C_2$ , and a Tbox  $\mathcal{K}_0$  of the following nine general axioms in logic  $\mathcal{ALCI}$ :

$$\begin{array}{ll}
 (a_1) \top \sqsubseteq \neg C_1 \sqcup \exists q.C_2 & (a_2) \top \sqsubseteq \neg C_2 \sqcup \forall s.C_4 \\
 (a_3) \top \sqsubseteq \neg C_4 \sqcup \exists p.C_3 & (a_4) \top \sqsubseteq \neg C_3 \sqcup \exists s^-.C_2 \\
 (a_5) \top \sqsubseteq \neg C_2 \sqcup \forall s.C_5 & (a_6) \top \sqsubseteq \neg C_5 \sqcup \forall p^-.C_6 \\
 (a_7) \top \sqsubseteq \neg C_6 \sqcup \forall p^-.C_7 & (a_8) \top \sqsubseteq \neg C_7 \sqcup \forall s^-.C_8 \\
 (a_9) \top \sqsubseteq \neg C_8 \sqcup \forall q^-.C_9
 \end{array}$$

**Source Problem:** the satisfiability of  $E_0$  w.r.t.  $\mathcal{K}_0$  in  $\mathcal{ALCI}$ .

**Step-1:** perform tagging. For each  $a_i$ , there is  $a'_i = \text{tag}(a_i)$ .

$$\begin{array}{ll}
 (a'_1) \top \sqsubseteq \neg C_1 \sqcup \exists q.C_2 & \text{(no change from } a_1) \\
 (a'_2) \top \sqsubseteq \neg C_2 \sqcup (A_1 \oplus \forall s.C_4) & \\
 (a'_3) \top \sqsubseteq \neg C_4 \sqcup \exists p.C_3 & \text{(no change from } a_3) \\
 (a'_4) \top \sqsubseteq \neg C_3 \sqcup \exists s^-.C_2 & \text{(no change from } a_4)
 \end{array}$$

$$\begin{aligned}
 (a'_5) \top &\sqsubseteq \neg C_2 \sqcup (A_2 \oplus \forall s. C_5) \\
 (a'_6) \top &\sqsubseteq \neg C_5 \sqcup (A_3 \oplus \forall p^-. C_6) \\
 (a'_7) \top &\sqsubseteq \neg C_6 \sqcup (A_4 \oplus \forall p^-. C_7) \\
 (a'_8) \top &\sqsubseteq \neg C_7 \sqcup (A_5 \oplus \forall s^-. C_8) \\
 (a'_9) \top &\sqsubseteq \neg C_8 \sqcup (A_6 \oplus \forall q^-. C_9)
 \end{aligned}$$

The tagging operation changes nothing for axioms  $a_1, a_3$  and  $a_4$ .  $A_j$  are newly introduced tags for occurrences of universal constraints. Now,  $\mathcal{K}_1 = \{a'_i\}$ ; the tagged concept is  $E_1 = \text{tag}(E_0) = \exists s^-. C_2$ .

**Step-2:** perform recording. Each  $A_j$  has an axiom  $r_j$ .

$$\begin{aligned}
 (r_1) \top &\sqsubseteq \forall s^-. \neg A_1 \sqcup C_4 & (r_2) \top &\sqsubseteq \forall s^-. \neg A_2 \sqcup C_5 \\
 (r_3) \top &\sqsubseteq \forall p. \neg A_3 \sqcup C_6 & (r_4) \top &\sqsubseteq \forall p. \neg A_4 \sqcup C_7 \\
 (r_5) \top &\sqsubseteq \forall s. \neg A_5 \sqcup C_8 & (r_6) \top &\sqsubseteq \forall q. \neg A_6 \sqcup C_9
 \end{aligned}$$

Now,  $\mathcal{K}_a = \{r_j\}$ .

**Step-3:** perform polarisation. Notice  $\mathcal{K}_2 = \text{Poly}(\mathcal{K}_1 \cup \mathcal{K}_a)$ . Accordingly, we have  $c_i = \text{Poly}(a'_i)$  and  $d_j = \text{Poly}(r_j)$  as following.

$$\begin{aligned}
 (c_1) \top &\sqsubseteq \neg C_1 \sqcup \exists q^a. C_2 \\
 (c_2) \top &\sqsubseteq \neg C_2 \sqcup (A_1 \oplus \forall s^a. C_4) \\
 (c_3) \top &\sqsubseteq \neg C_4 \sqcup \exists p^a. C_3 \\
 (c_4) \top &\sqsubseteq \neg C_3 \sqcup \exists s^b. C_2 \\
 (c_5) \top &\sqsubseteq \neg C_2 \sqcup (A_2 \oplus \forall s^a. C_5) \\
 (c_6) \top &\sqsubseteq \neg C_5 \sqcup (A_3 \oplus \forall p^b. C_6) \\
 (c_7) \top &\sqsubseteq \neg C_6 \sqcup (A_4 \oplus \exists p^b. C_7) \\
 (c_8) \top &\sqsubseteq \neg C_7 \sqcup (A_5 \oplus \forall s^b. C_8) \\
 (c_9) \top &\sqsubseteq \neg C_8 \sqcup (A_6 \oplus \forall q^a. C_9) \\
 (d_1) \top &\sqsubseteq \forall s^b. \neg A_1 \sqcup C_4 & (d_2) \top &\sqsubseteq \forall s^b. \neg A_2 \sqcup C_5 \\
 (d_3) \top &\sqsubseteq \forall p^a. \neg A_3 \sqcup C_6 & (d_4) \top &\sqsubseteq \forall p^a. \neg A_4 \sqcup C_7 \\
 (d_5) \top &\sqsubseteq \forall s^a. \neg A_5 \sqcup C_8 & (d_6) \top &\sqsubseteq \forall q^a. \neg A_6 \sqcup C_9
 \end{aligned}$$

We get  $E_2 = \text{Poly}(E_1) = \text{Poly}(\exists s^-. C_2) = \exists s^b. C_2$ ; and  $\mathcal{K}_2 = \{c_i\} \cup \{d_j\}$  as listed above. The newly introduced roles  $\{s^a, p^a, q^a, s^b, p^b, q^b\}$  replace the old roles  $\{s, p, q, s^-, p^-, q^-\}$ . Replace  $\oplus$  with  $\sqcap$  and we get the problem in  $\mathcal{ALC}$ .

**Target problem:** the satisfiability of  $E_2$  w.r.t.  $\mathcal{K}_2$  in  $\mathcal{ALC}$ .

## B Proofs

**Definition 8. (Fischer-Ladner Closure)** *The Fischer-Ladner closure [FL79]  $FL(H)$  of a set of formulae  $H$  is the least set of formulae which is inductively generated as follows:*

- (1)  $H$  is a subset of  $FL(H)$ ;
- (2) if  $C \in FL(H)$ , then so is  $\neg C$ ;
- (3) if  $C \sqcap D \in FL(H)$ , then so are  $C$  and  $D$ ;
- (4) if  $C \sqcup D \in FL(H)$ , then so are  $C$  and  $D$ ;
- (5) if  $\exists R. C \in FL(H)$ , then so is  $C$ ;
- (6) if  $\forall R. C \in FL(H)$ , then so is  $C$ .

To denote a *modal constraint*, we use  $\exists R. \text{tag}(C)$ . The  $\text{tag}(\cdot)$  operation (recursively) converts one *modal constraint* to a conjunction having two conjuncts.

In each tagged constraint we stipulate that the tag is on the left and the *modal constraint* is on the right. This excludes cases like  $\exists R.tag(C) \sqcap Q(x)$ .

**Definition 9.** Let  $Q(x) \sqcap \exists R.tag(C)$  be a tagged constraint in the Tbox  $\mathcal{KT}$  and the formula  $E$ . Consider the Fischer-Ladner closure  $FL(\mathcal{KT} \cup \{E\})$ , a generating formula for  $\exists R.tag(C)$  is a formulae  $\alpha$  such that  $FL(\{\alpha\}) \supset FL(\{\exists R.tag(C)\})$ , and  $\alpha \notin \{\exists R.tag(C), \neg(\exists R.tag(C))\}$ .

**Definition 10.** The generating formula  $\alpha$  for  $\exists R.tag(C)$  is least if there exists no other generating formula  $\beta$  s.t.  $FL(\{\beta\}) \subset FL(\{\alpha\})$ .

A few comments are necessary: (1) Each tag  $Q(x)$  is unique and occurs only as a conjunct; its negation  $\neg Q(x)$  occurs only as a disjunct; (2)  $Q(x) \sqcap \exists R.tag(C)$  and  $\neg(Q(x) \sqcap \exists R.tag(C))$  are two least generating formulae for  $\exists R.tag(C)$ . We call the former *positive* and the latter *negative*; (3) The formulae have already been converted into NNF before performing the mapping; (4) The tagging operation assigns a set of unique tags for one modal constraint and that a least generating formula for the tag  $Q_i(x)$  will also be a least generating formula for the tagged modal constraint; (5) Only *positive generating formulae* need to be considered [BCM<sup>+</sup>03] when building a model<sup>7</sup>. These lead to the notion<sup>8</sup> of *p-model*.

**Definition 11.** Given  $E$  and  $\mathcal{KT}$  that are tagged formula and Tbox. Let  $Q_1(x) \sqcap \exists R.tag(C), \dots, Q_m(x) \sqcap \exists R.tag(C)$  be tagged constraints. A *p-model* for  $E$  and  $\mathcal{KT}$  is a model  $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  such that for any  $n \in \Delta^{\mathcal{I}}$  there are  
 (1) if  $n \in (Q_i(x))^{\mathcal{I}}$  then  $n \in (\exists R.tag(C))^{\mathcal{I}}$ ; and  
 (2) if  $n \in (\exists R.tag(C))^{\mathcal{I}}$ , then there is  $Q_i(x)$  for  $\exists R.tag(C)$  s.t.  $n \in (Q_i(x))^{\mathcal{I}}$ .

**Lemma 1.** Given  $E$  and  $\mathcal{KT}$  as the tagged formula and the tagged Tbox. Let  $Q_i(x) \sqcap \exists R.tag(C)$  be tagged constraints in  $E$  and  $\mathcal{KT}$ , where  $i = 1, \dots, m$ .  $E$  and  $\mathcal{KT}$  is satisfiable only if it is satisfiable in a *p-model*.

*Proof.* Only proof outline. Let  $M_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$  be any model for  $E$  and  $\mathcal{KT}$ .

We use the well-known *sub-model generating technique*<sup>9</sup> to get a *p-model* from  $M_1$ . To guide the extraction process, it is only necessary to consider the *positive generating formulae*. To be precise, for any  $n \in \Delta^{\mathcal{I}_1}$ , the extraction process ignores assertions like  $n \in (\neg(Q_i \sqcap \exists R.tag(C)))^{\mathcal{I}_1}$ , where  $\exists R.tag(C)$  is a tagged constraints with a unique set of tags  $\{Q_1, Q_2, \dots, Q_m\}$ .

Starting from a (root) node with  $\mathcal{L}(n) \supseteq \{E\} \cup \mathcal{KT}$ , a guided extraction process (focusing on *positive least generating formulae*) generates a sub-model

<sup>7</sup> This is true for  $\mathcal{ALCI}$  and expressive logics without the qualified number restrictions.

<sup>8</sup> Consider a multi-valued function from modal constraints to tags  $f(\exists R.tag(C)) = \{Q_1, \dots, Q_m\}$  s.t.  $f(x) \cap f(y) = \emptyset$  for  $x \neq y$ . The model of interest is such a model  $(\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$  that  $\forall n \in \Delta^{\mathcal{J}}$ : (i) for all  $1 \leq i \leq m$ ,  $(Q_i)^{\mathcal{J}} \subseteq (\exists R.tag(C))^{\mathcal{J}}$ ; and (ii) if  $n \in (\exists R.tag(C))^{\mathcal{J}}$ , then  $n \in (Q_i)^{\mathcal{J}}$  for some  $1 \leq i \leq m$ .

<sup>9</sup> It has also been extensively used in the literature in the completeness proof of certain tableau-calculus for DLs with inverse roles.

from  $M_1$ . It is verifiably a model by routinely showing/checking it is saturated and clash-free, and it meets  $p$ -model definition. So, there is a  $p$ -model  $(\Delta^{\mathcal{I}_2}, \mathcal{I}_2)$  provided that  $(\Delta^{\mathcal{I}_1}, \mathcal{I}_1)$  be a model (which is saturated and has no clash).  $\square$

**Lemma 2.** *Given a concept formula  $E'$  and a Tbox  $\mathcal{KT}'$ . Let the tagged formula and Tbox be  $E$  and  $\mathcal{KT}$ .  $E'$  is satisfiable w.r.t.  $\mathcal{KT}'$  iff  $E$  is satisfiable w.r.t.  $\mathcal{KT}$ .*

*Proof.* The proof uses the same *sub-model generating technique* as above and a mapping from the  $\exists\forall R.tag(C)$  to  $Q_i \sqcap \exists\forall R.tag(C)$  and vice versa.

From the  $p$ -model (out of the above sub-model generating) for  $E$  and  $\mathcal{KT}$ , taking away (negated) tags leads to a model for  $E'$  and  $\mathcal{KT}'$  (having no tags).

In the other direction, for any model of  $E'$  and  $\mathcal{KT}'$ , a mapping reflecting the tag operation (from the  $\exists\forall R.tag(C)$  to  $Q_i \sqcap \exists\forall R.tag(C)$  for some  $Q_i$ ) will lead to a model (interpreting tags) for  $E$  and  $\mathcal{KT}$ . Since a  $p$ -model is a model, this concludes that the tagging operation preserves satisfiability.  $\square$

### B.1 Concept Satisfiability Test with General Concept Inclusions

**Lemma 3.** *if  $E_1$  and  $\mathcal{K}_1$  has a  $p$ -model, then  $\mathcal{K}_a$  is satisfiable in that model.*

*Proof.* This follows from the definition of  $p$ -model  $Q(x)^{\mathcal{I}} \subseteq (\forall R.tag(C))^{\mathcal{I}}$ .  $\square$

**Lemma 4.**  *$E_1$  is satisfiable w.r.t.  $\mathcal{K}_1 \cup \mathcal{K}_a$  iff  $E_1$  is satisfiable w.r.t.  $\mathcal{K}_1$ .*

*Proof.* (Only If Direction) It is trivial.

(If Direction) Let  $\mathcal{M}_2$  a  $p$ -model for  $E_1$  and  $\mathcal{K}_1$ . According to the lemma above,  $\mathcal{K}_a$  is always satisfied in the  $p$ -model for both  $E_1$  and  $\mathcal{K}_1$ . It follows that  $\mathcal{M}_2$  is a model for  $E_1$  and  $\mathcal{K}_1 \cup \mathcal{K}_a$ .  $\square$

**Lemma 5.**  *$E_1$  is satisfiable w.r.t.  $\mathcal{K}_1 \cup \mathcal{K}_a$  iff  $E_2$  is satisfiable w.r.t.  $\mathcal{K}_2$ .*

*Proof.* Note  $E_2 = Pol(E_1)$  and  $\mathcal{K}_2 = Pol(\mathcal{K}_1 \cup \mathcal{K}_a)$ .

(If Direction) Let  $\mathcal{M}_2 = (\Delta^{\mathcal{I}_2}, \mathcal{I}_2)$  be a  $p$ -model (possibly non-tree) for  $E_2$  and  $\mathcal{K}_2$ . For  $m', n' \in \Delta^{\mathcal{I}_2}$ , consider a mapping to  $m, n \in \Delta^{\mathcal{I}_1}$  such that

- (1) if  $(m', n') \in (R^a)^{\mathcal{I}_2}$ , then  $(m, n) \in R^{\mathcal{I}_1}$ ;
- (2) if  $(m', n') \in (S^b)^{\mathcal{I}_2}$ , then  $(m, n) \in (S^-)^{\mathcal{I}_1}$ ;
- (3) if  $m', n' \in Poly(C)^{\mathcal{I}_2}$ , then  $m, n \in C^{\mathcal{I}_1}$ .

(Only If Direction) Let  $\mathcal{M}_1 = (\Delta^{\mathcal{I}_1}, \mathcal{I}_1)$  be a  $p$ -model (possibly non-tree) for  $E_1$  and  $\mathcal{K}_1$ . For  $m, n \in \Delta^{\mathcal{I}_1}$ , consider a mapping that maps them to  $m', n' \in \Delta^{\mathcal{I}_2}$

- (1) if  $(m, n) \in R^{\mathcal{I}_1}$ , then  $(m', n') \in (R^a)^{\mathcal{I}_2}$  and  $(n', m') \in (R^b)^{\mathcal{I}_2}$ ;
- (2) if  $(m, n) \in (S^-)^{\mathcal{I}_1}$ , then  $(m', n') \in (S^b)^{\mathcal{I}_2}$  and  $(n', m') \in (S^a)^{\mathcal{I}_2}$ ;
- (3) if  $m, n \in C^{\mathcal{I}_1}$  then  $m', n' \in Poly(C)^{\mathcal{I}_2}$ .

$R, S^-$  are roles in  $\mathcal{ALCC}$ ;  $R^a, R^b, S^a, S^b$  are roles in  $\mathcal{ALC}$ .

Note the definition of  $p$ -models and the special axioms acquired by the recording operation. In both directions, at each element of the target interpretation, all constraints are satisfied both locally and w.r.t. its neighbor elements provided the given  $(\Delta^{\mathcal{I}_k}, \mathcal{I}_k)$  be a  $p$ -model. This concludes that the polarisation operation preserves equisatisfiability (for a tagged and recorded problem).  $\square$

**Lemma 6.**  $\|E_2\| + \|\mathcal{K}_2\|$  is of  $O(n^2)$  where  $n = \|E_0\| + \|\mathcal{K}_0\|$ .

**Theorem 1.** (1)  $E_0$  is satisfiable w.r.t.  $\mathcal{K}_0$  iff  $E_2$  is satisfiable w.r.t.  $\mathcal{K}_2$ . (2) The satisfiability of  $E_0$  w.r.t.  $\mathcal{K}_0$  in  $\mathcal{ALCI}$  can be decided by a test of  $E_2$  w.r.t.  $\mathcal{K}_2$  in  $\mathcal{ALC}$ . (3) The concept satisfiability w.r.t. GCIs in  $\mathcal{ALCI}$  can be decided in exponential time by the tableaux procedure in  $\mathcal{ALC}$  [DM00].

### B.2 Concept Satisfiability/Abox Consistency with (Acyclic) Tbox

Regarding to acyclic Tboxes in DLs without inverse roles, we refer to [Lut99] and [Tes01] for their results. We list supporting theorems and lemmas. The proofs are similar to the ones previously done for general axioms.

**Theorem 2. (Acyclic ALC Tbox[Lut99])** The concept satisfiability w.r.t. an acyclic Tbox in  $\mathcal{ALC}$  is decidable in PSPACE by a tableau-based procedure.

**Lemma 7.** For an input concept  $E_0$  and an acyclic Tbox  $\mathcal{T}_0$  in  $\mathcal{ALCI}$ , by tagging-recording-polarisation the concept  $E_2$  and the acyclic Tbox  $\mathcal{T}_2$  in  $\mathcal{ALC}$  is of size  $O(n^3)$ , where  $n = \|E_0\| + \|\mathcal{T}_0\|$ .

*Proof.* (outline only) By a similar step-by-step proof<sup>10</sup> (as in the case of general axioms), it is able to show  $E_2$  and  $\mathcal{T}_2$  is equisatisfiable to  $E_0$  and  $\mathcal{T}_0$ . The acyclicity of  $\mathcal{T}_2$  is easy to verify. The number of combinations (formulae  $\alpha, \beta$  s.t.  $\text{ord}(\alpha) \succ \text{ord}(\beta)$ ) is at most  $n^2$  and each new axiom is of size at most  $n$ .  $\square$

**Lemma 8.** The concept satisfiability problem w.r.t. an acyclic Tbox in  $\mathcal{ALCI}$  can be decided in PSPACE by tableau procedures as given in [Lut99].

**Lemma 9.** For Abox Consistency,  $\mathcal{T}_2$  is acyclic. The conversion (a streamlined tagging, recording and polarisation operations) takes a polynomial space.

**Lemma 10.** The consistency of an Abox w.r.t. an acyclic Tbox in  $\mathcal{ALCI}$  can be decided by the consistency check of an Abox w.r.t. an acyclic Tbox in  $\mathcal{ALC}$ .

**Theorem 3. (Precompletion[Tes01])** A precompletion of an Abox w.r.t. a Tbox can be nondeterministically computed in a polynomial space; the size of each precompletion is polynomial bounded.

**Theorem 4.** The consistency of an Abox w.r.t. an acyclic Tbox in  $\mathcal{ALCI}$  can be decided in PSPACE by tableau-based decision procedures.

*Proof.* (1) Perform the conversion to get the target problem, which is of a polynomial size to the source problem. The conversion itself takes a polynomial space; (2) Nondeterministically compute one precompletion. Then individuals of the target Abox are subject to satisfiability tests by the PSPACE procedure in [Lut99] w.r.t. the target Tbox independently. If each individual of the target Abox is satisfiable, the source problem is consistent. The Abox consistency problem w.r.t. an acyclic Tbox is decidable in a nondeterministic polynomial space. Consider Savitch’s theorem[Sav70]. This ends the proof<sup>11</sup>.  $\square$

<sup>10</sup> Proofs of equisatisfiability do not use acyclicity or properties of Aboxes in our case. These properties are however needed in proofs of particular decision procedures that are PSPACE and they are taken into account by previous work [Lut99] [Tes01]. We cite those results as theorems.

<sup>11</sup> Similar arguments were extensively used for PSPACE tableaux in the literature.

## Conjunctive Query Entailment for *SHOQ*

Birte Glimm<sup>\*</sup>, Ian Horrocks, and Ulrike Sattler

[glimm,horrocks,sattler]@cs.man.ac.uk  
The University of Manchester, UK

**Abstract.** An important reasoning task, in addition to the standard DL reasoning services, is conjunctive query answering. In this paper, we present a decision procedure for conjunctive query entailment in the expressive Description Logic *SHOQ*. This is, to the best of our knowledge, the first decision procedure for conjunctive query entailment in a logic that allows for nominals. We achieve this by combining the techniques used in the conjunctive query entailment procedure for *SHIQ* with the techniques proposed for a restricted class of conjunctive queries in *SHOQ*.

### 1 Introduction

Existing Description Logic (DL) reasoners<sup>1</sup> provide automated reasoning support for checking concepts for satisfiability and subsumption, and also for answering queries that retrieve known instances of concepts and roles. There are, however, still many open questions regarding the development of algorithms that decide conjunctive query (CQ) entailment in expressive Description Logics. For example, proposed techniques for deciding CQ entailment in expressive DLs mostly require that all roles that occur in the query are simple, i.e., neither transitive nor have transitive subroles. Furthermore, none of the existing conjunctive query answering techniques [10, 8, 2, 7, 9] is able to handle nominals. In this paper, we address both these issues and present a decision procedure for entailment of arbitrary CQs in the very expressive DL *SHOQ*, i.e., we allow for both non-simple roles in the query and nominals. We also do not impose any restrictions on the structure of the queries as it is the case for a previously proposed query entailment technique for *SHOQ* [5].

Our algorithm combines ideas from the CQ entailment decision procedure for *SHIQ* [3, 4] with the technique for deciding entailment of a restricted class of CQs in *SHOQ* [5] (i.e., the algorithm does not accept arbitrary CQs as input, but only queries of a particular shape). We first rewrite a query into a set of queries that have a kind of forest shape. By applying the rolling-up or tuple-graph technique [2, 10], we build concepts that capture the rewritten queries. We then show that we can use the obtained concepts to reduce the task of deciding query entailment to the task of testing the consistency of extended knowledge bases.

<sup>\*</sup> This work was supported by an EPSRC studentship.

<sup>1</sup> For example, FaCT++ <http://owl.man.ac.uk/factplusplus>, KAON2 <http://kaon2.semanticweb.org>, Pellet <http://pellet.owldl.com>, or Racer Pro <http://www.racer-systems.com>

## 2 Preliminaries

We assume readers to be familiar with the syntax and semantics of the DL *SHOQ* (for details see [1]). Since, in the presence of nominals, the ABox can be internalised, we assume that a *SHOQ* knowledge base  $\mathcal{K}$  is a pair  $(\mathcal{T}, \mathcal{R})$  over a signature  $\mathcal{S} = (N_C, N_R)$ , where  $\mathcal{T}$  is a TBox,  $\mathcal{R}$  is a role hierarchy, and  $N_C$  and  $N_R$  are countable, infinite, and pairwise disjoint sets of *concept names* and *role names* respectively. We assume that the set  $N_C$  contains a subset  $N_I$  of *nominal names* and the set  $N_R$  contains a subset  $N_{tR}$  of *transitive role names*. We say that a role name  $r$  is simple if there is no  $s \in N_{tR}$  such that  $s \sqsubseteq_{\mathcal{R}}^* r$ , where  $\sqsubseteq_{\mathcal{R}}$  is the reflexive transitive closure of  $\sqsubseteq$  over  $\mathcal{R}$ .

**Definition 1.** Let  $\mathcal{S} = (N_C, N_R)$  be a signature and  $N_V$  a countably infinite set of variable names disjoint from  $N_C$  and  $N_R$ . Let  $C$  be a *SHOQ*-concept over  $\mathcal{S}$ ,  $r \in N_R$  a role name, and  $x, y \in N_V$ . An atom is an expression  $C(x)$  or  $r(x, y)$ , and we refer to these types of atoms as *concept atoms* and *role atoms* respectively. A Boolean conjunctive query  $q$  is a non-empty set of atoms. We use  $\text{Vars}(q)$  to denote the set of all variables occurring in  $q$  and  $\#(q)$  for the cardinality of  $q$ . A sub-query of  $q$  is simply a subset of  $q$  (including  $q$  itself).

Let  $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathcal{I})$  be an interpretation. For a total function  $\pi: \text{Vars}(q) \rightarrow \Delta^{\mathcal{I}}$ , we write

- $\mathcal{I} \models^{\pi} C(x)$  if  $\pi(x) \in C^{\mathcal{I}}$ ;
- $\mathcal{I} \models^{\pi} r(x, y)$  if  $(\pi(x), \pi(y)) \in r^{\mathcal{I}}$ .

If  $\mathcal{I} \models^{\pi} a$  for all atoms  $a \in q$ , we write  $\mathcal{I} \models^{\pi} q$ . We say that  $\mathcal{I}$  satisfies  $q$  and write  $\mathcal{I} \models q$  if there exists a mapping  $\pi$  such that  $\mathcal{I} \models^{\pi} q$ . We call such a  $\pi$  a match for  $q$  in  $\mathcal{I}$ . For a *SHOQ* knowledge base  $\mathcal{K}$ , we say that  $\mathcal{K}$  entails  $q$  and write  $\mathcal{K} \models q$  if  $\mathcal{I} \models \mathcal{K}$  implies  $\mathcal{I} \models q$ .

The *query entailment problem* is defined as follows: given a knowledge base  $\mathcal{K}$  and a query  $q$ , decide whether  $\mathcal{K} \models q$ . Please note that we do not allow for constants (individual names) in the query. In the presence of nominals this is clearly without loss of generality. Query entailment is the decision problem corresponding to query answering, which is a computation problem. For query answering, let the variables of a conjunctive query be typed: each variable can either be existentially quantified (also called *non-distinguished*) or free (also called *distinguished* or *answer variables*). Let  $q$  be a query in  $n$  variables (i.e.,  $\#(\text{Vars}(q)) = n$ ), of which  $v_1, \dots, v_m$  ( $m \leq n$ ) are answer variables,  $\mathcal{K}$  a *SHOQ* knowledge base, and  $\text{nom}(\mathcal{K})$  the set of nominals that occur in  $\mathcal{K}$ . The *answers* of  $\mathcal{K}$  to  $q$  are those  $m$ -tuples  $(o_1, \dots, o_m) \in \text{nom}(\mathcal{K})$  such that, for all models  $\mathcal{I}$  of  $\mathcal{K}$ ,  $\mathcal{I} \models^{\pi} q$  for some  $\pi$  that satisfies  $\pi(v_i) \in \{o_i^{\mathcal{I}}\}$  for all  $i$  with  $1 \leq i \leq m$ . It is not hard to see that the answers of  $\mathcal{K}$  to  $q$  can be computed by testing, for each  $(o_1, \dots, o_m) \in \text{nom}(\mathcal{K})^m$ , whether the query  $q'$  obtained from  $q$  by adding, for each  $v_i$  with  $1 \leq i \leq m$ , an atom  $(\{o_i\})(v_i)$ , is entailed by  $\mathcal{K}$ .<sup>2</sup> The answer

<sup>2</sup> Please note that, in the presence of constants, it is more common to replace the distinguished variables  $v_1, \dots, v_m$  with the constants  $o_1, \dots, o_m$  instead of adding constant atoms.

to  $q$  is then the set of all  $m$ -tuples  $(o_1, \dots, o_m)$  for which  $\mathcal{K} \models q_{[v_1, \dots, v_m / o_1, \dots, o_m]}$ . Let  $k = \sharp(\text{nom}(\mathcal{K}))$  be the number of nominals used in the  $\mathcal{K}$ . Since  $\mathcal{K}$  is finite, clearly  $k$  is finite. Hence, deciding which tuples belong to the set of answers can be checked with at most  $k^m$  entailment tests. This is clearly not very efficient, but optimisations can be used, e.g., to identify a (hopefully small) set of candidate tuples.

In the remainder of this paper, we concentrate on query entailment. In the following, we use  $\mathcal{K}$  for a  $\mathcal{SHOQ}$  knowledge base and  $q$  for a Boolean CQ over a common signature  $\mathcal{S}$ . We use  $\text{nom}(\mathcal{K})$  for the set of nominals that occur in  $\mathcal{K}$  and we assume that  $\text{nom}(\mathcal{K})$  is non-empty without further notice. This is w.l.o.g. since otherwise we can always add an axiom  $\{o\} \sqsubseteq \top$  to the TBox for a new nominal  $o \in N_I$ .

As for the CQ entailment algorithm for  $\mathcal{SHIQ}$ , we first show that we can restrict our attention to the canonical models of  $\mathcal{K}$ . Canonical models have a kind of forest shape, i.e., the elements in the model can be seen as a collection of trees such that each nominal builds the root of a tree. Additionally, there can be arbitrary relational structures between the nominals and relations between an element from within a tree back to some nominal. In order to emphasise the forest shape of the canonical models, we also define forest bases, where we omit the shortcuts induced by transitive roles. The role of canonical models in our decision procedure is roughly speaking the following: for deciding query entailment, we first rewrite a given query into a set of forest-shaped queries such that the rewritten queries can be expressed as concepts. We use the forest structure of the canonical models in order to show that the disjunction of the obtained concepts is indeed enough for deciding query entailment.

**Definition 2.** A tree  $T$  is a prefix-closed subset of  $\mathbf{N}^*$ . For  $w, w' \in T$ , we call  $w'$  a successor of  $w$  if  $w' = w \cdot c$  for some  $c \in \mathbf{N}$ , where “ $\cdot$ ” denotes concatenation. The empty word  $\varepsilon$  is the root of the tree. Given a set of roots  $R = \{r_1, \dots, r_n\}$ , a forest  $F$  w.r.t.  $R$  is a subset of  $R \times \mathbf{N}^*$  such that, for each  $r_i \in R$ ,  $f(r_i) = (r_i, \varepsilon)$  and the set  $\{w \mid (r_i, w) \in F\}$  is a tree.

A forest base for  $\mathcal{K}$  is an interpretation  $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$  that interprets transitive roles in an unrestricted (i.e., not necessarily transitive) way and, additionally, satisfies the following conditions:

- T1  $\Delta^{\mathcal{J}}$  is a forest w.r.t.  $\text{nom}(\mathcal{K})$ , and
- T2 if  $((o, w), (o', w')) \in r^{\mathcal{J}}$ , then either  $w' = \varepsilon$  or  $o = o'$  and  $w'$  is a successor of  $w$ .

An interpretation  $\mathcal{I}$  is canonical for  $\mathcal{K}$ , if there exists a forest base  $\mathcal{J}$  for  $\mathcal{K}$  such that  $\mathcal{I}$  is identical to  $\mathcal{J}$  except that, for all non-simple roles  $r$ , we have

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq_{\mathcal{R}} r, s \in N_{tR}} (s^{\mathcal{J}})^+$$

In this case, we say that  $\mathcal{J}$  is a forest base for  $\mathcal{I}$  and, if  $\mathcal{I} \models \mathcal{K}$ , we say that  $\mathcal{I}$  is a canonical model for  $\mathcal{K}$ .

**Lemma 1.**  $\mathcal{K} \not\models q$  iff there is some canonical model  $\mathcal{I}$  of  $\mathcal{K}$  such that  $\mathcal{I} \not\models q$ .

*Proof Sketch:* The if direction is trivial. For the only if direction, the proof is similar to the one for  $\mathcal{SHIQ}$ . Let  $\mathcal{I}$  be such that  $\mathcal{I} \models \mathcal{K}$  and  $\mathcal{I} \not\models q$ . Intuitively, we first unravel  $\mathcal{I}$  into a model  $\mathcal{I}'$  of  $\mathcal{K}$  and then construct a forest base from the unravelled model. Finally, we obtain a canonical model from the forest base by transitively closing all roles  $r \in N_{tR}$ . Since in the unravelling process we only “break” cycles, the query is still not satisfied in the constructed canonical model.

### 3 Reducing Query Entailment to Concept Unsatisfiability

In this section, we introduce the basic ideas that have been used in the development of algorithms for CQ entailment. In the following section, we show more formally how the techniques presented here can be combined in order to obtain a decision procedure for  $\mathcal{SHOQ}$ .

The initial ideas used in this paper were first introduced by Calvanese et al. [2] for deciding CQ containment and hence CQ entailment for  $\mathcal{DLR}_{reg}$ . The authors show how a query  $q$  can be expressed as a concept  $C_q$ , such that  $q$  is entailed by a knowledge base iff adding  $\top \sqsubseteq \neg C_q$  makes the KB inconsistent. In order to obtain the concept  $C_q$ , the query  $q$  is represented as a directed, labelled graph. This graph, called a tuple graph or a query graph, is traversed in a depth-first manner and, during the traversal, nodes and edges are replaced with appropriate concept expressions, leading to the concept  $C_q$  after completing the traversal.

The nodes in a query graph correspond to the variables in the query and are labelled with the concepts that occur in the corresponding concept atoms. The edges correspond to the role atoms in  $q$  and are labelled accordingly. E.g., let  $q_1 = \{C(x), s(x, y), D(y)\}$  and  $q_2 = \{C(x), r(x, y), r(x, y'), s(y, z), s(y', z), D(z)\}$ . The query graphs for  $q_1$  and  $q_2$  are depicted in Figure 1 and Figure 2 respectively. We call  $q_2$  a cyclic query since its underlying undirected query graph is cyclic. For acyclic queries such as  $q_1$ , we can build the concept that represents  $q_1$  as follows: start at  $x$  and traverse the graph to  $y$ . Since  $y$  is a leaf node, remove  $y$  and its incoming edge and conjoin  $\exists s.D$  to the label  $C$  of  $x$ , resulting in  $C \sqcap \exists s.D$  for  $C_{q_1}$ . A given KB  $\mathcal{K}$  entails  $q_1$  iff  $\mathcal{K} \cup \{\top \sqsubseteq \neg C_{q_1}\}$  is inconsistent. Please note that in the absence of inverse roles in the logic, this process requires that the query graph has the form of a directed tree.

This reduction is not directly extendable to cyclic queries since, due to the tree model property of most DLs, a concept cannot capture cyclic relationships. For simpler logics, only ABox assertions can enforce cyclic relational structures in every model. One could argue, therefore, that we can replace variables in a cycle with individual names from the ABox. By identifying variables with each other, however, some cyclic queries become acyclic. For example, identifying  $y$  and  $y'$  in  $q_2$  leads to an acyclic query that can be expressed as  $C \sqcap \exists r.(\exists s.D)$ . Hence,  $\mathcal{K} \models q_2$  if  $\mathcal{K} \cup \{\top \sqsubseteq \neg(C \sqcap \exists r.(\exists s.D))\}$  is inconsistent.

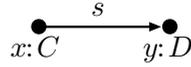


Fig. 1: The (acyclic) query graph for  $q_1$ .

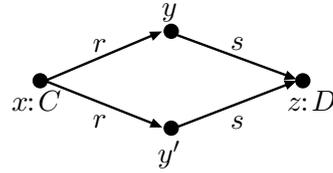


Fig. 2: A query graph for the cyclic query  $q_2$ .

Last year, we presented a decision procedure for entailment of a restricted class of CQs in  $\mathcal{SHOQ}$  [5]. Due to the absence of inverse roles in  $\mathcal{SHOQ}$ , the algorithm handles only queries where the tree parts of the query form a directed tree and all cyclic subgraphs build a directed cycle. For example, the query  $\{r(x, y), r(z, y)\}$  is not accepted as an input for the algorithm. The technique introduces, however, ideas for dealing with cycles that can arise when nominals and non-simple roles are present in the knowledge base. For example, Figure 3 represents a model for the knowledge base  $\mathcal{K}$  containing the axiom  $\{o\} \sqsubseteq \neg C \sqcap \neg D \sqcap \exists s.(C \sqcap \exists r.(D \sqcap \exists s.\{o\}))$  with  $s \in N_{tR}$ . The query  $q_3 = \{C(x), D(y), r(x, y), s(y, x)\}$  (see Figure 4) would clearly be satisfied in each model of  $\mathcal{K}$ , although in the relevant matches neither  $x$  nor  $y$  can be mapped to the nominal  $o^{\mathcal{I}}$  and without using the nominal  $o$  in the query concept, we cannot enforce the coreference for closing the cycle.

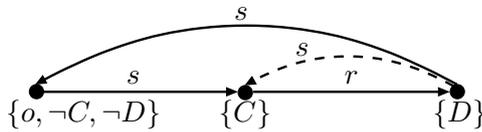


Fig. 3: The dashed line indicates the relationship added due to  $s$  being transitive. Therefore, there is a cycle not directly containing the nominal  $o$ .

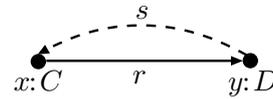


Fig. 4: The query graph for  $q_3$ .

In canonical models for a  $\mathcal{SHOQ}$  knowledge base, such a directed cycle among non-nominals can only occur due to a transitive role that provides a shortcut for “skipping” the nominal. Hence, a nominal is always at least indirectly involved, e.g.,  $o$  in the current example. The proposed algorithm allows, therefore, the replacement of the role atom  $s(y, x)$  with two role atoms  $s(y, v), s(v, x)$  for a new variable  $v$ . We can then guess that  $v$  corresponds to the nominal  $o$  and express the query as a concept in which we use  $\{o\}$  to close the cycle.

In the decision procedure for CQ entailment in  $\mathcal{SHIQ}$  [3, 4], the rewriting steps also allow for eliminating shortcuts induced by transitive roles that do not involve nominals (ABox individuals in the case of  $\mathcal{SHIQ}$ ). Let  $\mathcal{K}_4 = (\mathcal{T}, \mathcal{R}, \mathcal{A})$

be a  $\mathcal{SHIQ}$  knowledge base with  $\mathcal{T} = \emptyset$ ,  $\mathcal{R} = \{r \sqsubseteq t\}$ ,  $\mathcal{A} = \{(\exists s.\exists r.\exists r.\exists r.\top)(a)\}$ , and  $t \in N_{tR}$  (see Figure 5). The cyclic query  $q_4 = \{r(x_1, x_2), r(x_2, x_3), r(x_3, x_4), t(x_1, x_4)\}$  (see Figure 6) is clearly entailed by  $\mathcal{K}_4$ .

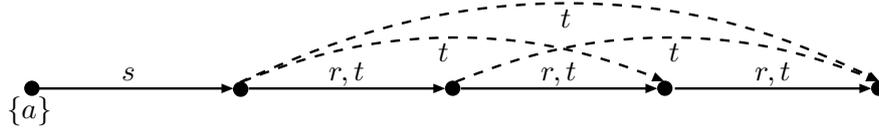


Fig. 5: A representation of a canonical model  $\mathcal{I}$  for  $\mathcal{K}_4$ . Transitive “shortcuts” are again indicated by dashed lines.

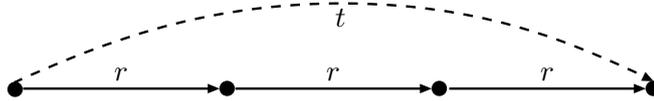


Fig. 6: The query graph for the query  $q_4$ .

In order to obtain a tree-shaped query that matches in such a canonical model, the rewriting steps for  $\mathcal{SHIQ}$  allow, for each role atom  $s(x, x')$  in a query such that there is a role  $s' \in N_{tR}$  with  $s' \sqsubseteq_{\mathcal{R}} s$ , to replace  $s(x, x')$  with up to  $\sharp(q)$  role atoms  $s'(x_1, x_2), \dots, s'(x_{\ell-1}, x_\ell)$  such that  $x_1 = x$  and  $x_\ell = x'$ . In the above example, we can replace  $t(x_1, x_4)$  with  $t(x_1, x_2), t(x_2, x_3), t(x_3, x_4)$  and obtain the query  $q'_4 = \{r(x_1, x_2), r(x_2, x_3), r(x_3, x_4), t(x_1, x_2), t(x_2, x_3), t(x_3, x_4)\}$ . Using role conjunctions, we can now build the query concept  $C_{q'_4} = \exists(r \sqcap t).(\exists(r \sqcap t).(\exists(r \sqcap t).\top))$ . We can now use the concept  $C_{q'_4}$  as described above and reduce the query entailment problem for a  $\mathcal{SHIQ}$  knowledge base to a knowledge base consistency problem for  $\mathcal{SHIQ}^\sqcap$ , i.e.,  $\mathcal{SHIQ}$  with role conjunctions.

Usually, there is not just one query concept for a given query. In the rewriting process, we build query concepts for *all* tree-shaped queries obtained by identifying variables and by replacing role atoms as described above. We then check whether  $\mathcal{K}$  entails the disjunction of the obtained concepts, which can be reduced to checking the consistency of  $\mathcal{K}$  extended with all axioms  $\top \sqsubseteq \neg C_q$  such that  $C_q$  is one of the obtained query concepts.

We now define the different rewriting steps more formally and show how the different rewriting steps can be combined into a decision procedure for general CQs in  $\mathcal{SHOQ}$ .

#### 4 Conjunctive Query Entailment for $\mathcal{SHOQ}$

For deciding whether a given Boolean CQ is entailed by a  $\mathcal{SHOQ}$  knowledge base, we transform the query in a four stage process into a set of  $\mathcal{SHOQ}^\sqcap$

concepts, i.e.,  $\mathcal{SHOQ}$  with role conjunctions. We can then reduce the task of deciding CQ entailment to the task of deciding  $\mathcal{SHOQ}^\square$  knowledge base consistency.

In the first step, called collapsing, we can identify variables. In the second step, we can replace role atoms of the form  $r(x, x')$  for which  $r$  is non-simple with up to  $\sharp(q)$  role atoms. This allows for explicating all shortcuts due to transitive roles in the query. In the third step, we “guess” which variables correspond to nominals and filter out those queries that can still not be expressed as a  $\mathcal{SHOQ}^\square$  concept. Those queries are trivially false since the structure specified by the query cannot be enforced by a  $\mathcal{SHOQ}$  concept and hence cannot be mapped to the canonical models of the knowledge base. Finally, we express the resulting queries as concepts and use these concepts for deciding entailment of the original query.

**Definition 3.** A collapsing of  $q$  is obtained as follows:

1. Build a partition  $\mathcal{P}$  of  $\text{Vars}(q)$ ,
2. choose, for each  $P \in \mathcal{P}$ , one variable name  $x \in P$ , and
3. replace each occurrence of  $x' \in P$  with  $x$ .

We use  $\text{co}(q)$  to denote the set of all queries that are a collapsing of  $q$ .

A transitivity rewriting of  $q$  is obtained by fixing a set  $V \subseteq N_V$  of variables not occurring in  $q$  such that  $\sharp(V) \leq \sharp(\text{Vars}(q))$  and by choosing, for each role atom  $r(x, x') \in q$  such that there is an  $s \in N_{tR}$  with  $s \sqsubseteq_{\mathcal{R}} r$ , to replace  $r(x, x')$  with  $1 \leq \ell \leq \sharp(q)$  role atoms  $s(x_1, x_2), \dots, s(x_{\ell-1}, x_\ell)$ , where  $x_1 = x$ ,  $x_\ell = x'$ , and  $x_2, \dots, x_\ell \in \text{Vars}(q) \cup V$ . We use  $\text{tr}_{\mathcal{K}}(q)$  to denote the set of all queries that are a transitivity rewriting of a query  $q_{co} \in \text{co}(q)$ .

We assume that  $\text{tr}_{\mathcal{K}}(q)$  contains no isomorphic queries, i.e., differences in (newly introduced) variable names only are neglected.

We now show how we can filter out those queries that are trivially false since they have a structure that cannot occur in canonical models. For this, we use forest structures that are similar to canonical models. We first guess which variables of the query correspond to nominals. Between those variables, the role atoms of the query can induce arbitrary relational structures. All other variables are mapped to trees such that for a role atom  $r(x, y)$  either the image of  $y$  is a successor of the image of  $x$  in the tree or  $y$  corresponds to a nominal and  $r(x, y)$  corresponds to an edge back to some nominal.

**Definition 4.** A tree mapping w.r.t.  $q$  is a total and bijective function  $f$  from  $\text{Vars}(q)$  to a tree  $T$  such that  $r(x, x') \in q$  implies that  $f(x')$  is a successor of  $f(x)$ . A query  $q$  is tree-shaped if there exists a tree mapping w.r.t.  $q$ .

A root choice is a subset of  $\text{Vars}(q)$ . A forest mapping w.r.t.  $q$  and a root choice  $R$  is a total function  $f$  from  $\text{Vars}(q)$  to a forest  $F$  w.r.t.  $R$  such that if  $r(x, x') \in q$ , then either  $x' \in R$  or there is some  $x_r \in R$  such that  $f(x) = (x_r, w)$  and  $f(x') = (x_r, w \cdot c)$ . We say that  $q$  is forest-shaped w.r.t.  $R$  if either  $R = \emptyset$  and  $q$  is tree-shaped or  $R \neq \emptyset$  and there exists a forest mapping w.r.t.  $q$  and  $R$ .

We use  $\text{fr}_{\mathcal{K}}(q)$  to denote the set of all pairs  $(q_{tr}, R)$  such that  $q_{tr} \in \text{tr}_{\mathcal{K}}(q)$ ,  $R$  is a root choice w.r.t.  $q_{tr}$ , and  $q_{tr}$  is forest-shaped w.r.t.  $R$ .

Similarly to forest-shaped queries, we define tree- and forest-shaped matches on canonical models.

**Definition 5.** Let  $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathcal{I})$  be a canonical model for  $\mathcal{K}$  such that  $\mathcal{I} \models^{\pi} q$ . A match  $\pi$  induces a root choice  $R = \{x \mid \pi(x) = (o, \varepsilon) \text{ for some } o \in \text{nom}(\mathcal{K})\}$ . We call  $\pi$  a tree match if  $R = \emptyset$  and there exists a bijective function  $f$  from  $\text{ran}(\pi)$  to a tree  $T$  such that  $r(x, x') \in q$  implies that  $f(\pi(x'))$  is a successor of  $f(\pi(x))$ . We call  $\pi$  a forest match if either  $\pi$  is a tree match or there is a total mapping  $f$  from  $\text{ran}(\pi)$  to a forest  $F$  w.r.t.  $R$  such that  $r(x, x') \in q$  implies that either  $f(\pi(x')) = (x', \varepsilon)$  or there is some  $x_r \in R$  such that  $f(\pi(x)) = (x_r, w)$ ,  $f(\pi(x')) = (x_r, w \cdot c)$ .

The following lemma shows that we can indeed omit queries that are not forest-shaped w.r.t. some subset of variables  $R$ .

**Lemma 2.** Let  $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathcal{I})$  be a model for  $\mathcal{K}$ .

- (1) If  $\mathcal{I}$  is canonical and  $\mathcal{I} \models q$ , then there is a pair  $(q_{tr}, R) \in \text{fr}_{\mathcal{K}}(q)$  and a forest match  $\pi$  such that  $\mathcal{I} \models^{\pi} q_{tr}$  and  $R$  is the root choice induced by  $\pi$ .
- (2) If  $(q_{tr}, R) \in \text{fr}_{\mathcal{K}}(q)$  and  $\mathcal{I} \models q_{tr}$ , then  $\mathcal{I} \models q$ .

The proof is very similar to the proofs for *SHIQ* [3, 4]. Intuitively, we use the canonical model  $\mathcal{I}$  to guide the rewriting process in the proof of part (1) of Lemma 2. Part (2) of Lemma 2 mainly follows from the fact that we only use non-simple roles in the transitivity rewritings.

We now build a query that consists of only concept atoms for each  $(q_{tr}, R) \in \text{fr}_{\mathcal{K}}(q)$  by replacing the variables from  $R$  with nominals from  $\text{nom}(\mathcal{K})$  and applying the rolling-up technique.

**Definition 6.** Let  $(q_{tr}, R) \in \text{fr}_{\mathcal{K}}(q)$ . A grounding for  $q_{tr}$  w.r.t.  $R$  is a total function  $\tau: R \rightarrow \text{nom}(\mathcal{K})$ . Let  $f$  be a forest mapping w.r.t.  $q$  and  $R$ . We build  $\text{con}(q_{tr}, R, \tau)$  as follows:

1. For each  $r(x, x_r) \in q_{tr}$  with  $x_r \in R$ , replace  $r(x, x_r)$  with  $(\exists r. \{\tau(x_r)\})(x)$ .
2. For each  $x_r \in R$  add a concept atom  $(\{\tau(x_r)\})(x_r)$  to  $q_{tr}$ .
3. We now inductively assign to each  $x \in \text{Vars}(q_{tr})$  a concept  $\text{con}(x)$  as follows:
  - if there is no role atom  $r(x, x') \in q_{tr}$ , then  $\text{con}(x) := \prod_{C(x) \in q_{tr}} C$ ,
  - if there are role atoms  $r(x, x_1), \dots, r(x, x_k) \in q_{tr}$ , then

$$\text{con}(x) := \prod_{C(x) \in q_{tr}} C \sqcap \prod_{1 \leq i \leq k} \exists (\prod_{r(x, x_i) \in q_{tr}} r) . \text{con}(x_i).$$

4. Finally,  $\text{con}(q_{tr}, R, \tau) = \{(\text{con}(x))(x) \mid x \in \text{Vars}(q_{tr}) \text{ and there is no role atom } r(x', x) \in q_{tr}\}$ .

We use  $\text{con}_{\mathcal{K}}(q)$  for the set  $\{\text{con}(q_{tr}, R, \tau) \mid (q_{tr}, R) \in \text{fr}_{\mathcal{K}}(q) \text{ and } \tau \text{ is a grounding w.r.t. } R\}$ .

Please note that after the first step, the resulting query consists of a set of unconnected components such that each component is a tree-shaped query with a distinguished root variable that has no incoming edges. In step 4, we collect all query concepts for these root variables in the set  $\text{con}(q_{tr}, R, \tau)$ . Hence  $\text{con}(q_{tr}, R, \tau)$  is a conjunctive query of the form  $\{C_1(x_1), \dots, C_n(x_n)\}$  with  $x_i \neq x_j$  for  $1 \leq i < j \leq n$  and each  $C_i$  is a  $\text{SHOQ}^\square$ -concept.

**Lemma 3.** *Let  $\mathcal{I}$  be a model of  $\mathcal{K}$ .*

- (1) *If  $\mathcal{I}$  is canonical and  $\mathcal{I} \models q$ , then there is some  $\text{con}(q_{tr}, R, \tau) \in \text{con}_{\mathcal{K}}(q)$  such that  $\mathcal{I} \models \text{con}(q_{tr}, R, \tau)$ .*
- (2) *If  $\mathcal{I} \models \text{con}(q_{tr}, R, \tau)$  for some  $\text{con}(q_{tr}, R, \tau) \in \text{con}_{\mathcal{K}}(q)$ , then  $\mathcal{I} \models q$ .*

Informally, the use of nominals in the constructed concepts still enforces the same structures that are required by the query.

We now show that the union of the queries in  $\text{con}_{\mathcal{K}}(q)$  can be used to decide entailment of  $q$ . Since we now use unions of conjunctive queries, we introduce their semantics more formally:

**Definition 7.** *A union of Boolean conjunctive queries is a formula  $q_1 \vee \dots \vee q_n$ , where each disjunct  $q_i$  is a Boolean conjunctive query. A knowledge base  $\mathcal{K}$  entails a union of Boolean conjunctive queries  $q_1 \vee \dots \vee q_n$ , written as  $\mathcal{K} \models q_1 \vee \dots \vee q_n$ , if, for each interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models \mathcal{K}$ , there is some  $i$  with  $1 \leq i \leq n$  such that  $\mathcal{I} \models q_i$ .*

W.l.o.g. we assume that the variable names in each disjunct are different from the variable names in the other disjuncts. This can always be achieved by naming variables apart.

Putting everything together, we get the following theorem, which shows that the queries in  $\text{con}_{\mathcal{K}}(q)$  are indeed enough to decide whether  $\mathcal{K} \models q$ .

**Theorem 1.** *Let  $\{q_1, \dots, q_\ell\} = \text{con}_{\mathcal{K}}(q)$ . Then  $\mathcal{K} \models q$  iff  $\mathcal{K} \models q_1 \vee \dots \vee q_\ell$ .*

Please note that each disjunct  $q_i$  is a set of concept atoms of the form  $\{C_1^i(x_1^i), \dots, C_{n_i}^i(x_{n_i}^i)\}$ , i.e., each  $q_i$  contains  $n_i$  unconnected components. In the following, we assume for convenience that conjunctive queries are written as a conjunction of atoms instead of in the set notation, e.g., we now write each of the disjuncts  $\{C_1^i(x_1^i), \dots, C_{n_i}^i(x_{n_i}^i)\}$  as  $C_1^i(x_1^i) \wedge \dots \wedge C_{n_i}^i(x_{n_i}^i)$ . By transforming the disjunction of queries in  $\text{con}_{\mathcal{K}}(q)$  into conjunctive normal form (cf. [10, 7.3.2]), we can reduce the problem of deciding whether  $\mathcal{K} \models q_1 \vee \dots \vee q_\ell$  to deciding whether  $\mathcal{K}$  entails each union of connected conjunctive queries  $\{at_1\} \vee \dots \vee \{at_\ell\}$  such that  $at_i$  a concept atom from  $q_i$ . Let  $\text{con}_{\mathcal{K}}(q) = \{q_1, \dots, q_\ell\}$ . We use  $\text{cnf}(\text{con}_{\mathcal{K}}(q))$  for the conjunctive normal form of  $q_1 \vee \dots \vee q_\ell$ . We now show how we can decide entailment of unions of conjunctive queries, where each conjunct consists of one concept atom only. This suffices to decide conjunctive query entailment for  $\text{SHOQ}$ .

**Definition 8.** Let  $\mathcal{K} = (\mathcal{T}, \mathcal{R})$  be a  $\mathcal{SHOQ}$  knowledge base,  $q$  a Boolean conjunctive query, and  $\{C_1(x_1) \vee \dots \vee C_\ell(x_\ell)\}$  a conjunct from  $\text{cnf}(\text{con}_{\mathcal{K}}(q))$ . An extended knowledge base w.r.t.  $\mathcal{K}$  and  $q$  is a pair  $(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R})$  such that  $\mathcal{T}_q$  contains an axiom  $\top \sqsubseteq \neg C_i$  for each  $i$  with  $1 \leq i \leq \ell$ .

We can now use the extended knowledge bases in order to decide conjunctive query entailment as follows:

**Theorem 2.**  $\mathcal{K} \models q$  iff each extended knowledge base  $\mathcal{K}_q$  w.r.t.  $\mathcal{K}$  and  $q$  is inconsistent.

Please note that the extended knowledge bases are in  $\mathcal{SHOQ}^\square$ . It is, however, not hard to see how the Tableaux algorithm for  $\mathcal{SHOQ}$  [6] can be extended to handle such extended KBs, which then provides a decision procedure for CQ entailment in  $\mathcal{SHOQ}$ .

## 5 Conclusions

In the previous section, we have presented a decision procedure for CQ entailment in  $\mathcal{SHOQ}$ . This is, to the best of our knowledge, the first CQ entailment decision procedure that can handle nominals. In addition, we allow for non-simple roles in the query as well, which is a feature that is known to be tricky. Since the set of rewritten queries can potentially be large, the algorithm is more suitable for showing decidability of the problem rather than building the foundation of implementable algorithms. By analysing the role hierarchy one can, however, avoid several rewritings. Going back to the example knowledge base  $\mathcal{K}_4$  and query  $q_4$  (see Figure 5 and 6) in Section 3, it is not hard to see that with the given role hierarchy the atom  $t(x_1, x_4)$  is redundant. Every match for the remaining role atoms implies the existence of a suitable  $t$ -edge. Since after removing this redundant role atom the query is acyclic, no rewriting is necessary in order to decide entailment. It will be part of our future work to investigate whether such an analysis of query and role hierarchy can be used instead of the transitivity rewriting step. Furthermore, we will try to find tight complexity bounds for the conjunctive query entailment problem in  $\mathcal{SHOQ}$  and we will try to show decidability of conjunctive query entailment in  $\mathcal{SHOIQ}$ .

## References

1. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS 1998*, 1998.
3. Birte Glimm, Ian Horrocks, Carsten Lutz, and Uli Sattler. Conjunctive query answering in the description logic *SHIQ*. In *Proc. of IJCAI 2007*, 2007.
4. Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic *SHIQ*. Technical report, The University of Manchester, 2007. <http://www.cs.man.ac.uk/~glimmbx/download/GHLS07b.pdf>.
5. Birte Glimm, Ian Horrocks, and Ulrike Sattler. Conjunctive query answering for description logics with transitive roles. In *Proc. DL 2006*, 2006.
6. Ian Horrocks and Ulrike Sattler. Ontology reasoning in the *SHOQ* description logic. In *Proc. of IJCAI 2001*, 2001.
7. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of LPAR 2004*, 2004.
8. Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *Proc. of ECAI 1996*, 1996.
9. Maria M. Ortiz, Diego Calvanese, and Thomas Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of AAAI 2006*, 2006.
10. Sergio Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001.

## Modularity in DL-Lite

Roman Kontchakov<sup>1</sup>, Frank Wolter<sup>2</sup>, and Michael Zakharyashev<sup>1</sup>

<sup>1</sup> School of Computer Science and Information Systems, Birkbeck College, London,  
`{roman,michael}@dcs.bbk.ac.uk`

<sup>2</sup> Department of Computer Science, University of Liverpool, U.K.,  
`frank@csc.liv.ac.uk`

**Abstract.** We develop a formal framework for modular ontologies by analysing four notions of conservative extensions and their applications in refining, re-using, merging, and segmenting ontologies. For two members of the *DL-Lite* family of description logics, we prove important meta-properties of these notions such as robustness under joins, vocabulary extensions, and iterated import of ontologies. The computational complexity of the corresponding reasoning tasks is investigated.

### 1 Introduction

In computer science and related areas, ontologies are used to define the meaning of vocabularies designed to speak about some domains of interest. In ontology languages based on description logics (DLs), such an ontology typically consists of a TBox stating which inclusions hold between complex concepts built over the vocabulary. An increasingly important application of ontologies is management of large amounts of data, where the ontology is used to provide flexible and efficient access to repositories consisting of data sets of instances of concepts and relations of the vocabulary. In DLs, such repositories are typically modelled as ABoxes.

Developing ontologies for this and other purposes is a difficult task. When dealing with DLs, the ontology designer is supported by efficient reasoning tools for classification, instance checking and some other reasoning problems. However, it is generally recognised that this support is not sufficient when ontologies are not developed as ‘monolithic entities’ but rather result from importing, merging, combining, re-using, refining and extending existing ontologies. In all those cases, reasoning support for analysing the impact of the respective operation on the ontology would be highly desirable. Typical reasoning tasks in this case may include the following:

- If we add some new concepts, relations and axioms to our ontology, can new assertions over the vocabulary of the original TBox be derived from the extended TBox?
- When importing an ontology, do we change the meaning of its vocabulary?
- When looking for a definition of some concepts, what part of the existing ontology defining them should be used?

Recently, the notion of conservative extension has been identified as fundamental for dealing with problems of this kind [1–5]. Parameterising this notion by a language  $\mathcal{L}$ , we say that a TBox  $\mathcal{T}$  is a *conservative extension* of a TBox  $\mathcal{T}'$  w.r.t.  $\mathcal{L}$  if  $\mathcal{T} \models \alpha$  implies  $\mathcal{T}' \models \alpha$ , for every  $\alpha$  from  $\mathcal{L}$  which only uses the vocabulary of  $\mathcal{T}'$ . In these papers, the main emphasis has been on languages  $\mathcal{L}$  consisting of TBox axioms over some description logic (such as  $\mathcal{ALC}$ ) and the much stronger notion of *model conservativity* which corresponds to the assumption that  $\alpha$  can be taken from any language with standard Tarski semantics (e.g., second-order logic). Considering TBox axioms is motivated by the fact that ontologies are developed and represented via such axioms. They are the syntactic objects an ontology designer is working with, and a possibility to derive some new axioms appears therefore to be a good indicator as to whether the meaning of symbols has changed in any relevant sense. The notion of model conservativity is motivated by its flexibility: whatever language  $\mathcal{L}$  is chosen, no new consequences in  $\mathcal{L}$  will be derivable [5, 4]. A third option (which lies between the two above as far as expressivity is concerned) is as follows: if the main application of the ontologies  $\mathcal{T}$  and  $\mathcal{T}'$  is to provide a vocabulary and its meaning for posing queries to ABoxes, then it appears to be of interest to regard  $\mathcal{T}$  as a conservative extension of  $\mathcal{T}'$  if, for every ABox  $\mathcal{A}$  and every (say, positive existential) query  $q$  in the vocabulary of  $\mathcal{T}'$ , any answer to  $q$  given by  $(\mathcal{T}, \mathcal{A})$  is given by  $(\mathcal{T}', \mathcal{A})$  as well. It can thus be seen that there is a variety of notions of conservativity which can be used to formally define modularity in ontologies. The choice of the appropriate one depends on what the ontologies are supposed to be used for, the computational complexity of the corresponding reasoning tasks, and the relevant meta-properties and ‘robustness’ of the notion of conservativity.

Here we investigate these and related notions of conservative extensions for the *DL-Lite* family of description logics [6–8]. *DL-Lite* and its variants are weak description logics that have been designed in order to facilitate efficient query-answering over large data sets. We introduce four different notions of conservativity for two languages within this family, motivate their relevance for modularity and re-use of ontologies, study their meta-properties, and determine the computational complexity of the corresponding reasoning tasks. All the proofs can be found in the Appendix available at <http://www.csc.liv.ac.uk/~frank>.

## 2 The *DL-Lite* Family

The *DL-Lite* family of DLs has been introduced and investigated in [6–8] with the aim of establishing maximal subsets of DL constructors for which the data complexity of query answering stays within LOGSPACE. The ‘covering’ DL of the *DL-Lite* family is known as *DL-Lite<sub>bool</sub>* [8]. As *DL-Lite<sub>bool</sub>* itself contains classical propositional logic, query answering in it is CONP-hard, but by taking the Horn-fragment *DL-Lite<sub>horn</sub>* of *DL-Lite<sub>bool</sub>*, one obtains a language for which query answering is within LOGSPACE [8] (precise formulations of these results are given below).

The language of  $DL\text{-Lite}_{bool}$  has *object names*  $a_1, a_2, \dots$ , *concept names*  $A_1, A_2, \dots$ , and *role names*  $P_1, P_2, \dots$ . Complex roles  $R$  and  $DL\text{-Lite}_{bool}$  concepts  $C$  are defined as follows:

$$\begin{aligned} R & ::= P_i \quad | \quad P_i^-, \\ B & ::= \perp \quad | \quad \top \quad | \quad A_i \quad | \quad \geq q R, \\ C & ::= B \quad | \quad \neg C \quad | \quad C_1 \sqcap C_2, \end{aligned}$$

where  $q \geq 1$ . The concepts of the form  $B$  above are called *basic*. A  $DL\text{-Lite}_{bool}$  concept inclusion is of the form  $C_1 \sqsubseteq C_2$ , where  $C_1$  and  $C_2$  are  $DL\text{-Lite}_{bool}$  concepts. A  $DL\text{-Lite}_{bool}$  TBox is a finite set of  $DL\text{-Lite}_{bool}$  concept inclusions. (Other concept constructs like  $\exists R$ ,  $\leq q R$  and  $C_1 \sqcup C_2$  will be used as standard abbreviations.)

As mentioned above, we also consider the *Horn* fragment  $DL\text{-Lite}_{horn}$  of  $DL\text{-Lite}_{bool}$ : a  $DL\text{-Lite}_{horn}$  concept inclusion is of the form

$$\bigsqcap_k B_k \sqsubseteq B,$$

where  $B$  and the  $B_k$  are basic concepts. In this context, basic concepts will also be called  $DL\text{-Lite}_{horn}$  concepts. Note that the axioms  $\bigsqcap_k B_k \sqsubseteq \perp$  and  $\top \sqsubseteq B$  are legal in  $DL\text{-Lite}_{horn}$ . A  $DL\text{-Lite}_{horn}$  TBox is a finite set of  $DL\text{-Lite}_{horn}$  concept inclusions. For other fragments of  $DL\text{-Lite}_{bool}$  we refer the reader to [6–8]. It is worth noting that in  $DL\text{-Lite}_{horn}$  we can express both *global functionality* of a role and *local functionality* (i.e., functionality restricted to a (basic) concept  $B$ ) by means of the axioms  $\geq 2 R \sqsubseteq \perp$  and  $B \sqcap \geq 2 R \sqsubseteq \perp$ , respectively.

Let  $\mathcal{L}$  be either  $DL\text{-Lite}_{bool}$  or  $DL\text{-Lite}_{horn}$ . An  $\mathcal{L}$ -ABox is a set of assertions of the form  $C(a_i)$ ,  $R(a_i, a_j)$ , where each  $C$  is an  $\mathcal{L}$ -concept,  $R$  a role, and  $a_i, a_j$  are object names. An  $\mathcal{L}$  knowledge base ( $\mathcal{L}$ -KB) is a pair  $(\mathcal{T}, \mathcal{A})$  consisting of an  $\mathcal{L}$ -TBox  $\mathcal{T}$  and an  $\mathcal{L}$ -ABox  $\mathcal{A}$ .

An *interpretation*  $\mathcal{I}$  is a structure of the form  $(\Delta^{\mathcal{I}}, A_1^{\mathcal{I}}, \dots, P_1^{\mathcal{I}}, \dots, a_1^{\mathcal{I}}, \dots)$ , where  $\Delta^{\mathcal{I}}$  is non-empty,  $A_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ ,  $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$  and  $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$  such that  $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$ , for  $a_i \neq a_j$  (i.e., we adopt the unique name assumption). The *extension*  $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$  of a concept  $C$  is defined as usual. A concept inclusion  $C_1 \sqsubseteq C_2$  is *satisfied* in  $\mathcal{I}$  if  $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ ; in this case we write  $\mathcal{I} \models C_1 \sqsubseteq C_2$ .  $\mathcal{I}$  is a *model* for a TBox  $\mathcal{T}$  if all concept inclusions from  $\mathcal{T}$  are satisfied in  $\mathcal{I}$ . A concept inclusions  $C_1 \sqsubseteq C_2$  *follows from*  $\mathcal{T}$ ,  $\mathcal{T} \models C_1 \sqsubseteq C_2$  in symbols, if every model for  $\mathcal{T}$  satisfies  $C_1 \sqsubseteq C_2$ . A concept  $C$  is  $\mathcal{T}$ -*satisfiable* if there exists a model  $\mathcal{I}$  for  $\mathcal{T}$  with  $C^{\mathcal{I}} \neq \emptyset$ . We say that  $\mathcal{I}$  is a *model* for an  $\mathcal{L}$ -KB  $(\mathcal{T}, \mathcal{A})$  if  $\mathcal{I}$  is a model for  $\mathcal{T}$  and every assertion of  $\mathcal{A}$  is satisfied in  $\mathcal{I}$ .

An (*essentially positive existential*)  $\mathcal{L}$ -query  $q(x_1, \dots, x_n)$  is a formula

$$\exists y_1 \cdots \exists y_m \varphi(x_1, \dots, x_n, y_1, \dots, y_m),$$

where  $\varphi$  is constructed, using only  $\wedge$  and  $\vee$ , from atoms of the form  $C(t)$  and  $P(t_1, t_2)$ , with  $C$  being an  $\mathcal{L}$ -concept,  $P$  a role, and  $t_i$  being either a variable from the list  $x_1, \dots, x_n, y_1, \dots, y_m$  or an object name. Given an  $\mathcal{L}$ -KB  $(\mathcal{T}, \mathcal{A})$  and an

$\mathcal{L}$ -query  $q(\mathbf{x})$ , with  $\mathbf{x} = x_1, \dots, x_n$ , we say that an  $n$ -tuple  $\mathbf{a}$  of object names is an *answer* to  $q(\mathbf{x})$  w.r.t.  $(\mathcal{T}, \mathcal{A})$  and write  $(\mathcal{T}, \mathcal{A}) \models q(\mathbf{a})$  if, for every model  $\mathcal{I}$  for  $(\mathcal{T}, \mathcal{A})$ , we have  $\mathcal{I} \models q(\mathbf{a})$ . The data complexity of the query answering problem for  $DL-Lite_{horn}$  knowledge bases is in LOGSPACE, while for  $DL-Lite_{bool}$  it is CONP-complete [8].

### 3 Types of Conservativity and Modularity

In this section, we introduce four different notions of conservative extension for  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$ , discuss their applications, investigate their meta-properties, and determine the computational complexity of the corresponding reasoning tasks. By a *signature* we understand here a finite set  $\Sigma$  of concept names and role names.<sup>3</sup> Given a concept, role, TBox, ABox, or query  $E$ , we denote by  $\text{sig}(E)$  the *signature* of  $E$ , that is, the set of concept and role names that occur in  $E$ . It is worth noting that the symbols  $\perp$  and  $\top$  are regarded as logical symbols. Thus,  $\text{sig}(\perp) = \text{sig}(\top) = \emptyset$ . A concept (role, TBox, ABox, query)  $E$  is called a  $\Sigma$ -*concept* (*role*, *TBox*, *ABox*, *query*, respectively) if  $\text{sig}(E) \subseteq \Sigma$ . Thus,  $P^-$  is a  $\Sigma$ -role iff  $P \in \Sigma$ . As before, we will use  $\mathcal{L}$  as a generic name for  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$ .

**Definition 1** (deductive conservative extension). Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be  $\mathcal{L}$ -TBoxes and  $\Sigma$  a signature. We call  $\mathcal{T}_1 \cup \mathcal{T}_2$  a (*deductive*) *conservative extension* of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t.  $\Sigma$  if, for every  $\mathcal{L}$ -concept inclusion  $C_1 \sqsubseteq C_2$  with  $\text{sig}(C_1 \sqsubseteq C_2) \subseteq \Sigma$ , we have  $\mathcal{T}_1 \models C_1 \sqsubseteq C_2$  whenever  $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$ .

This notion of deductive conservative extension is appropriate in the following situations; see also [2]. (i) Suppose that  $\mathcal{T}_1$  is a TBox which does not cover part of its domain in sufficient detail. An ontology engineer, say Eve, decides to expand it by axioms  $\mathcal{T}_2$ , but wants to be sure that by doing this she does not interfere with the derivable inclusions between  $\Sigma$ -concepts. Then she should check whether  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a conservative extension of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t. to  $\Sigma$ . (ii) If the designer of an ontology  $\mathcal{T}_2$  *imports* an ontology  $\mathcal{T}_1$  and wants to ensure that no extra inclusions between  $\text{sig}(\mathcal{T}_1)$ -concepts are derivable after importing the ontology, then again she should check whether  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a conservative extension of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t.  $\text{sig}(\mathcal{T}_1)$ . Observe that in  $DL-Lite_{bool}$ ,  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a deductive conservative extension of  $\mathcal{T}_1$  iff every  $\mathcal{T}_1$ -satisfiable  $DL-Lite_{bool}$  concept  $C$  with  $\text{sig}(C) \subseteq \Sigma$  is also  $\mathcal{T}_1 \cup \mathcal{T}_2$ -satisfiable.

**Theorem 1.** For any  $DL-Lite_{horn}$  TBoxes  $\mathcal{T}_1, \mathcal{T}_2$  and any signature  $\Sigma$ , the following two conditions are equivalent:

- $\mathcal{T}_1 \cup \mathcal{T}_2$  is a conservative extension of  $\mathcal{T}_1$  in  $DL-Lite_{bool}$  w.r.t.  $\Sigma$ ;
- $\mathcal{T}_1 \cup \mathcal{T}_2$  is a conservative extension of  $\mathcal{T}_1$  in  $DL-Lite_{horn}$  w.r.t.  $\Sigma$ .

<sup>3</sup> In the languages we consider, object names do not occur in TBoxes. Therefore, in this paper, we assume that signatures do not contain object names. When considering languages with nominals one would have to allow for object names in signatures.

For  $DL-Lite_{horn}$  TBoxes, the problem of deciding whether  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a conservative extension of  $\mathcal{T}_1$  in  $DL-Lite_{horn}$  w.r.t  $\Sigma$  is  $CONP$ -complete. For  $DL-Lite_{bool}$  TBoxes, this problem is  $\Pi_2^P$ -complete.

Observe that the complexity lower bounds follow immediately from the same lower bounds for the corresponding reasoning problems in classical propositional (Horn) logic. The upper bounds are proved in the Appendix. We remind the reader that conservativity is much harder for most DLs: it is  $EXPTIME$ -complete for  $\mathcal{EL}$  [9],  $2EXPTIME$ -complete for  $\mathcal{ALC}$  and  $\mathcal{ALCQI}$ , and undecidable for  $\mathcal{ALCQIO}$  [2, 4]. To explain, at a very high level, the reason for these results we consider the notion of a conservative extension in  $DL-Lite_{bool}$ : let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be TBoxes and  $\Sigma$  a signature with  $\Sigma \subseteq \text{sig}(\mathcal{T}_1)$ .  $\mathcal{T}_1 \cup \mathcal{T}_2$  is not a conservative extension of  $\mathcal{T}_1$  in  $DL-Lite_{bool}$  w.r.t.  $\Sigma$  if, and only if, there exists a concept  $C$  with  $\text{sig}(C) \subseteq \Sigma$  such that  $C$  is satisfiable relative to  $\mathcal{T}_1$  but not relative to  $\mathcal{T}_1 \cup \mathcal{T}_2$ . We call such a concept  $C$  a *witness-concept*. Thus, a decision procedure for conservativity can be regarded as a systematic search for such a witness-concept. In standard description logics such as  $DL-Lite_{bool}$ ,  $\mathcal{EL}$ ,  $\mathcal{ALC}$ , etc. the space of all possible witnesses is infinite. (This observation implies that from the decidability of the problem whether a concept is satisfiable w.r.t. a TBox it does not necessarily follow that conservativity is decidable.) Now, we prove in the Appendix that for  $DL-Lite_{bool}$  the existence of some witness concept implies the existence of a witness concept of size *polynomial* in the size of  $\mathcal{T}_1$  and  $\mathcal{T}_2$  and which uses only the numeral parameters which occur in number restrictions from  $\mathcal{T}_1 \cup \mathcal{T}_2$ . In contrast, in  $\mathcal{EL}$  one can construct examples in which minimal witnesses for non-conservativity are of double exponential size in the size of  $\mathcal{T}_1$  and  $\mathcal{T}_2$  [9]. In  $\mathcal{ALC}$ , one can even enforce minimal witness concepts of triple exponential size [2]. The reason for this difference is the availability of *qualified* quantification in those language, and its absence in  $DL-Lite_{bool}$ . The result on the size of witness concepts for  $DL-Lite_{bool}$  is easily converted into a decision procedure for non-conservativity which is in  $\Pi_2^P$ : just (non-deterministically) guess a concept  $C$  of polynomial size in the size of  $\mathcal{T}_1$  and  $\mathcal{T}_2$  and with  $\text{sig}(C) \subseteq \Sigma$  and check, by calling an NP-oracle, whether (i)  $C$  is satisfiable w.r.t.  $\mathcal{T}_1$  and (ii) not satisfiable w.r.t.  $\mathcal{T}_1 \cup \mathcal{T}_2$ . Because of the larger size of minimal witnesses, no such procedure exists for  $\mathcal{EL}$  or  $\mathcal{ALC}$ .

Consider now the situation when the ontology designer is not only interested in preserving derivable concept inclusions, but also in preserving answers to queries, for both  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$  TBoxes.

**Definition 2** (query conservative extension). Let  $\mathcal{T}_1, \mathcal{T}_2$  be  $\mathcal{L}$ -TBoxes and  $\Sigma$  a signature. We call  $\mathcal{T}_1 \cup \mathcal{T}_2$  a *query conservative extension* of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t.  $\Sigma$  if, for every  $\mathcal{L}$ -ABox  $\mathcal{A}$  with  $\text{sig}(\mathcal{A}) \subseteq \Sigma$ , every  $\mathcal{L}$ -query  $q$  with  $\text{sig}(q) \subseteq \Sigma$ , and every tuple  $\mathbf{a}$  of object names from  $\mathcal{A}$ , we have  $(\mathcal{T}_1, \mathcal{A}) \models q(\mathbf{a})$  whenever  $(\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}) \models q(\mathbf{a})$ .

It is easy to see that query conservativity implies deductive conservativity for both logics  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$ . Indeed, let  $\mathcal{L}$  be one of  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$ . Suppose that we have  $\mathcal{T}_1 \not\models C_1 \sqsubseteq C_2$  but  $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$ ,

for some  $\mathcal{L}$ -concept inclusion  $C_1 \sqsubseteq C_2$  with  $\text{sig}(C_1 \sqsubseteq C_2) \subseteq \Sigma$ . Consider the ABox  $\mathcal{A} = \{C_1(a)\}$  and the query  $q = C_2(a)$ . Then clearly  $(\mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{A}) \models q$ , while  $(\mathcal{T}_1, \mathcal{A}) \not\models q$ . Note that in  $DL\text{-Lite}_{horn}$ ,  $C_1 = B_1 \sqcap \dots \sqcap B_k$  and  $C_2 = B$ , where  $B, B_1, \dots, B_k$  are basic concepts.

The following example shows, in particular, that the converse implication does not hold.

*Example 1.* (1) To see that there are deductive conservative extensions which are not query conservative, take  $\mathcal{T}_1 = \emptyset$ ,  $\mathcal{T}_2 = \{A \sqsubseteq \exists P, \exists P^- \sqsubseteq B\}$  and  $\Sigma = \{A, B\}$ . Then  $\mathcal{T}_2$  is a deductive conservative extension of  $\mathcal{T}_1$  (in both  $DL\text{-Lite}_{bool}$  and  $DL\text{-Lite}_{horn}$ ) w.r.t.  $\Sigma$ . However, it is not a query conservative extension: let  $\mathcal{A} = \{A(a)\}$  and  $q = \exists y B(y)$ ; then  $(\mathcal{T}_1, \mathcal{A}) \not\models q$  but  $(\mathcal{T}_2, \mathcal{A}) \models q$ .

(2) Note also that query conservativity in  $DL\text{-Lite}_{horn}$  does not imply query conservativity in  $DL\text{-Lite}_{bool}$ . Indeed, let  $\mathcal{T}_1 = \emptyset$ ,  $\mathcal{T}_2 = \{A \sqsubseteq \exists P, A \sqcap \exists P^- \sqsubseteq \perp\}$  and  $\Sigma = \{A\}$ . Then  $\mathcal{T}_2$  is not a query conservative extension of  $\mathcal{T}_1$  in  $DL\text{-Lite}_{bool}$  w.r.t.  $\Sigma$ : just take  $\mathcal{A}$  as before and  $q = \exists y \neg A(y)$ . But it is a query conservative extension in  $DL\text{-Lite}_{horn}$ .

In the definition of essentially positive existential queries for  $DL\text{-Lite}_{bool}$  above, we have allowed negated concepts in queries and ABoxes. An alternative approach would be to allow only *positive* concepts. These two types of queries give rise to different notions of query conservativity: under the second definition, the TBox  $\mathcal{T}_2$  from Example 1 (2) is a query conservative extension of  $\mathcal{T}_1 = \emptyset$  w.r.t.  $\{A\}$ , even in  $DL\text{-Lite}_{bool}$ . We argue, however, that it is the *essentially positive* queries that should be considered in the context of this investigation. The reason is that, with positive queries, the addition of the *definition*  $B \equiv \neg A$  to  $\mathcal{T}_2$  and  $B$  to  $\Sigma$  would result in a TBox which is not a query conservative extension in  $DL\text{-Lite}_{bool}$  of  $\mathcal{T}_1$  any longer. This kind of non-robust behaviour of the notion of conservativity is clearly undesirable. Obviously, the definitions we gave are robust under the addition of such definitions. Moreover, two extra robustness conditions hold true.

**Definition 3 (robustness).** Let  $\Sigma$  be a signature and  $\text{cons}_\Sigma$  a ‘conservativity’ relation between TBoxes w.r.t.  $\Sigma$ . (For example,  $\text{cons}_\Sigma(\mathcal{T}_1, \mathcal{T}_1 \cup \mathcal{T}_2)$  can be defined as ‘ $\mathcal{T}_1 \cup \mathcal{T}_2$  is a deductive conservative extension of  $\mathcal{T}_1$  in  $DL\text{-Lite}_{bool}$  w.r.t.  $\Sigma$ ’).

- We say that  $\text{cons}_\Sigma$  is *robust under joins* if  $(\mathcal{T}_0, \mathcal{T}_0 \cup \mathcal{T}_1), (\mathcal{T}_0, \mathcal{T}_0 \cup \mathcal{T}_2) \in \text{cons}_\Sigma$  and  $\text{sig}(\mathcal{T}_1) \cap \text{sig}(\mathcal{T}_2) \subseteq \Sigma$  imply  $(\mathcal{T}_0, \mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{T}_2) \in \text{cons}_\Sigma$ ;
- $\text{cons}_\Sigma$  is *robust under vocabulary extensions* if  $(\mathcal{T}_1, \mathcal{T}_1 \cup \mathcal{T}_2) \in \text{cons}_\Sigma$  implies  $(\mathcal{T}_1, \mathcal{T}_1 \cup \mathcal{T}_2) \in \text{cons}_{\Sigma'}$ , for all signatures  $\Sigma'$  with  $\text{sig}(\mathcal{T}_1 \cup \mathcal{T}_2) \cap \Sigma' \subseteq \Sigma$ .

Roughly speaking, robustness under joins means that an ontology can be safely imported into joins of independent ontologies if each of them safely imports the ontology: if the shared symbols of  $\mathcal{T}_1$  and  $\mathcal{T}_2$  are from  $\Sigma$ , and both  $\mathcal{T}_1 \cup \mathcal{T}_0$  and  $\mathcal{T}_2 \cup \mathcal{T}_0$  are conservative extensions of  $\mathcal{T}_0$  w.r.t.  $\Sigma$ , then the join  $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_0$  is a conservative extension of  $\mathcal{T}_0$  w.r.t.  $\Sigma$ . In practice, this property supports collaborative ontology development in the following sense: it implies that if two

(or more) ontology developers extend a given ontology  $\mathcal{T}_0$  independently and do not use common symbols with the exception of those in a certain signature  $\Sigma$  then they can safely form the union of  $\mathcal{T}_0$  and all their additional axioms provided that their individual extensions are safe for  $\Sigma$  (in the sense of deductive or, respectively, query conservativity). This property is closely related to the well-known *Robinson consistency lemma* and *interpolation* (see e.g., [10]) and has been investigated in the context of modular software specification [11] as well. We refer the reader to the Appendix for a more detailed discussion.

Robustness under vocabulary extensions is even closer to interpolation: it states that once we know conservativity w.r.t.  $\Sigma$ , we also know conservativity with respect to any signature with extra fresh symbols. The practical relevance of this property is as follows: when specifying the signature  $\Sigma$  for which an ontology developer wants to check conservativity, the developer only has to decide which symbols from  $\mathcal{T}_1$  and  $\mathcal{T}_2$  she wants to include into  $\Sigma$ . The answer to the query does not depend on whether  $\Sigma$  contains fresh symbols or not.

**Theorem 2.** *Both deductive and query conservativity in both  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$  are robust under joins and vocabulary extensions.*

Actually, in  $DL-Lite_{horn}$  and  $DL-Lite_{bool}$  we even have a much stronger form of interpolation which is known as the *uniform interpolation property* [12, 13]. Let  $\mathcal{T}$  be a TBox and  $\Sigma$  a signature. A TBox  $\mathcal{T}'$  is called a *uniform interpolant* for  $\mathcal{T}$  w.r.t.  $\Sigma$  in  $\mathcal{L}$  if  $\mathcal{T}'$  is an  $\mathcal{L}$ -TBox,  $\text{sig}(\mathcal{T}') \subseteq \Sigma$ ,  $\mathcal{T} \models \mathcal{T}'$ , and for all  $\mathcal{L}$ -concept inclusions  $C_1 \sqsubseteq C_2$  with  $\mathcal{T} \models C_1 \sqsubseteq C_2$  and  $\text{sig}(C_1, C_2) \cap \text{sig}(\mathcal{T}) \subseteq \Sigma$ , we have  $\mathcal{T}' \models C_1 \sqsubseteq C_2$ .

Intuitively, a uniform interpolant for  $\mathcal{T}$  w.r.t.  $\Sigma$  contains exactly the same information about  $\Sigma$  in terms of concept inclusions as  $\mathcal{T}$  *without using additional symbols*. For most DLs, such as  $\mathcal{ALC}$ , uniform interpolants do not necessarily exist [2].

**Theorem 3.** *Let  $\mathcal{L}$  be  $DL-Lite_{horn}$  or  $DL-Lite_{bool}$ . Then for every  $\mathcal{L}$ -TBox  $\mathcal{T}$  and every signature  $\Sigma$  there exists a uniform interpolant for  $\mathcal{T}$  w.r.t.  $\Sigma$  in  $\mathcal{L}$ .*

We note that one has to be very careful when interpreting the meaning of uniform interpolants. Consider, for instance,  $\mathcal{T} = \{A \sqsubseteq \exists P, A \sqcap \exists P^- \sqsubseteq \perp\}$  and  $\Sigma = \{A\}$ . The TBox  $\mathcal{T}' = \emptyset$  is a uniform interpolant of  $\mathcal{T}$  w.r.t.  $\Sigma$  in  $DL-Lite_{bool}$ . However, as we saw in Example 1, the TBoxes  $\mathcal{T}$  and  $\mathcal{T}'$  behave differently with respect to queries in  $\Sigma$ :  $(\mathcal{T}, \{A(a)\}) \models \exists x \neg A(x)$  but  $(\mathcal{T}', \{A(a)\}) \not\models \exists x \neg A(x)$ .

As sketched above, one application of deductive and query conservativity is to ensure that, when importing an ontology  $\mathcal{T}$ , one does not change the meaning of its vocabulary (in terms of concept inclusions or answers to queries). We now consider the situation where the ontology  $\mathcal{T}$  to be imported is not known because, for example, it is still under development or because different ontologies can be chosen. In this case,  $\mathcal{T}$  should be regarded as a ‘black box’ which supplies information about a signature  $\Sigma$ . To ensure that the meaning of the symbols in  $\Sigma$  as defined by this black box is not changed by importing it into  $\mathcal{T}_1$ , one has to check the following condition:

**Definition 4 (safety).** Let  $\Sigma$  be a signature and  $\mathcal{T}_1$  an  $\mathcal{L}$ -TBox. We say that  $\mathcal{T}_1$  is *safe for  $\Sigma$  w.r.t. deductive (or query) conservativity in  $\mathcal{L}$*  if, for all  $\mathcal{L}$ -TBoxes  $\mathcal{T}$  with  $\text{sig}(\mathcal{T}) \cap \text{sig}(\mathcal{T}_1) \subseteq \Sigma$ ,  $\mathcal{T}_1 \cup \mathcal{T}$  is a deductive (respectively, query) conservative extension of  $\mathcal{T}$  in  $\mathcal{L}$  w.r.t.  $\Sigma$ .

This notion has been introduced in [3] where the reader can find further discussion. A natural generalisation of safety, considered in [5], is the following property:

**Definition 5 (strong deductive/query conservative extension).** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be  $\mathcal{L}$ -TBoxes and  $\Sigma$  a signature. We call  $\mathcal{T}_1 \cup \mathcal{T}_2$  a *strong deductive (query) conservative extension of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t.  $\Sigma$*  if  $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}$  is a deductive (respectively, query) conservative extension of  $\mathcal{T}_1 \cup \mathcal{T}$  in  $\mathcal{L}$  w.r.t.  $\Sigma$ , for every  $\mathcal{L}$ -TBox  $\mathcal{T}$  with  $\text{sig}(\mathcal{T}) \cap \text{sig}(\mathcal{T}_1 \cup \mathcal{T}_2) \subseteq \Sigma$ .

Observe that safety is indeed a special case of strong conservativity: it covers exactly the case where the TBox  $\mathcal{T}_1$  in the definition of strong conservativity is empty. A typical application of strong conservativity for ontology re-use is as follows (see [5]). Suppose that there is a large ontology  $\mathcal{O}$  and a subset  $\Sigma$  of its signature. Assume also that the ontology designer wants to use what  $\mathcal{O}$  says about  $\Sigma$  in her own ontology  $\mathcal{T}$  she is developing at the moment. Then instead of importing  $\mathcal{O}$  as a whole, it would be preferable to find a small subset  $\mathcal{T}_1$  of  $\mathcal{O}$ , which says precisely the same about  $\Sigma$  as  $\mathcal{O}$  does, and import only this small  $\mathcal{T}_1$  rather than the large  $\mathcal{O}$ . But then what are the conditions we should impose on  $\mathcal{T}_1$ ? An obvious minimal requirement is that by importing  $\mathcal{T}_1$  into  $\mathcal{T}$  we obtain the same consequences for subsumptions/queries over  $\Sigma$  as by importing  $\mathcal{O}$  into  $\mathcal{T}$ . Depending on whether concept inclusions or answers to queries are of interest, one therefore wants  $\mathcal{O} = \mathcal{T}_1 \cup \mathcal{T}_2$  to be a strong deductive or query conservative extension of  $\mathcal{T}_1$  w.r.t.  $\Sigma$ . We refer the reader to [5] for further discussion and algorithms for extracting such TBoxes from a given TBox.

*Example 2.* (1) Let us see first that strong deductive conservativity is indeed a stronger notion than deductive conservativity, for  $DL\text{-}Lite_{bool}$  and  $DL\text{-}Lite_{horn}$ . Consider again the TBoxes  $\mathcal{T}_1 = \emptyset$ ,  $\mathcal{T}_2 = \{A \sqsubseteq \exists R, A \sqcap \exists R^- \sqsubseteq \perp\}$ , and  $\Sigma = \{A\}$ . Then  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a deductive conservative extension of  $\mathcal{T}_1$  w.r.t.  $\Sigma$ . However,  $\mathcal{T}_1 \cup \mathcal{T}_2$  is *not* a strong deductive conservative extension of  $\mathcal{T}_1$  w.r.t.  $\Sigma$ . Let  $\mathcal{T} = \{\top \sqsubseteq A\}$ . Then we have  $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T} \models \top \sqsubseteq \perp$  but  $\mathcal{T}_1 \cup \mathcal{T} \not\models \top \sqsubseteq \perp$ .

(2) We show now that an analogue of Theorem 1 does not hold for strong deductive conservativity. Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be the following  $DL\text{-}Lite_{horn}$  TBoxes

$$\begin{aligned} \mathcal{T}_1 &= \{A \sqcap B \sqsubseteq \perp, \top \sqsubseteq \exists P_1, \top \sqsubseteq \exists P_2, \exists P_1^- \sqsubseteq A, \exists P_2^- \sqsubseteq B\}, \\ \mathcal{T}_2 &= \{\top \sqsubseteq \exists R, A \sqcap \exists R^- \sqsubseteq \perp, B \sqcap \exists R^- \sqsubseteq \perp\}, \end{aligned}$$

and let  $\Sigma = \{A, B, P_1, P_2\}$ .  $\mathcal{T}_2$  says that  $\top \not\sqsubseteq A \sqcup B$ . Now, in  $DL\text{-}Lite_{bool}$ ,  $\mathcal{T}_1 \cup \mathcal{T}_2$  is not a strong deductive conservative extension of  $\mathcal{T}_1$  w.r.t.  $\Sigma$ : just take  $\mathcal{T} = \{\top \sqsubseteq A \sqcup B\}$ . However,  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a strong deductive conservative extension of  $\mathcal{T}_1$  in  $DL\text{-}Lite_{horn}$ .

Obviously, the robustness conditions introduced above are of importance for the strong versions of conservativity as well.

**Theorem 4.** *Both strong deductive and strong query conservativity are robust under joins and vocabulary extensions for  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$ .*

In addition to these types of robustness, the following condition, which is dual to robustness under joins, is of crucial importance for iterated applications of the notion of safety for a signature. Suppose that  $\mathcal{T}$  is safe for  $\Sigma_1 \cup \Sigma_2$  under some notion of conservativity and  $\Sigma_1 \cap \Sigma_2 = \emptyset$ . Then, for any  $\mathcal{T}_1$  with  $\text{sig}(\mathcal{T}_1) \cap (\text{sig}(\mathcal{T}) \cup \Sigma_2) \subseteq \Sigma_1$ , the TBox  $\mathcal{T} \cup \mathcal{T}_1$  should be safe for  $\Sigma_2$  for the same notion of conservativity. Without this property, one might have the situation that a TBox is safe for a signature  $\Sigma_1 \cup \Sigma_2$ , but after importing a TBox for  $\Sigma_1$  the resulting TBox is not safe for  $\Sigma_2$  any longer, which is clearly undesirable.

**Theorem 5 (robustness under joins of signatures).** *Let  $\mathcal{L}$  be either  $DL-Lite_{bool}$  or  $DL-Lite_{horn}$ . If an  $\mathcal{L}$ -TBox  $\mathcal{T}$  is safe for a signature  $\Sigma_1 \cup \Sigma_2$  w.r.t. deductive/query conservativity in  $\mathcal{L}$ ,  $\Sigma_1 \cap \Sigma_2 = \emptyset$  and  $\mathcal{T}_1$  is a satisfiable  $\mathcal{L}$ -TBox with  $\text{sig}(\mathcal{T}_1) \cap (\text{sig}(\mathcal{T}) \cup \Sigma_2) \subseteq \Sigma_1$ , then  $\mathcal{T} \cup \mathcal{T}_1$  is safe for  $\Sigma_2$  w.r.t. deductive/query-conservativity in  $\mathcal{L}$ .*

This result follows immediately from the fact that any two satisfiable  $\mathcal{L}$ -TBoxes in disjoint signatures are strong query conservative extensions of each other. This property fails for a number of stronger notions of conservativity, for example, model conservativity.

The next theorem shows that in all those cases where we have not provided counterexamples the introduced notions of conservativity are equivalent:

$DL-Lite_{horn}$	deductive $\not\approx$ query $\not\approx$ strong deductive $\equiv$ strong query
$DL-Lite_{bool}$	deductive $\approx$ query $\equiv$ strong deductive $\equiv$ strong query

It also establishes the complexity of the corresponding decision problems.

**Theorem 6.** *Let  $\mathcal{L}$  be either  $DL-Lite_{bool}$  or  $DL-Lite_{horn}$ ,  $\mathcal{T}_1$  and  $\mathcal{T}_2$   $\mathcal{L}$ -TBoxes, and  $\Sigma$  a signature.*

*For  $\mathcal{L} = DL-Lite_{bool}$ , the following conditions are equivalent:*

- (1)  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a query conservative extension of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t.  $\Sigma$ ;
- (2)  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a strong deductive conservative extension of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t.  $\Sigma$ ;
- (3)  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a strong query conservative extension of  $\mathcal{T}_1$  in  $\mathcal{L}$  w.r.t.  $\Sigma$ .

*For  $\mathcal{L} = DL-Lite_{horn}$ , conditions (2) and (3) are equivalent, while (1) is strictly weaker than each of them.*

*For  $DL-Lite_{horn}$ , the decision problems corresponding to conditions (1)–(3) are all CONP-complete. For  $DL-Lite_{bool}$ , these problems are  $\Pi_2^P$ -complete.*

We believe that the equivalences stated in Theorem 6 are somewhat surprising. For example, it can be easily seen that for  $\mathcal{ALC}$  none of those equivalences holds true.

## 4 Model-Theoretic Characterisations of Conservativity

All the results discussed above are proved with the help of the model-theoretic characterisations of our notions of conservativity formulated below.

Let  $\Sigma$  be a signature and  $Q$  a set of positive natural numbers containing 1. By a  $\Sigma Q$ -concept we mean any concept of the form  $\perp$ ,  $\top$ ,  $A_i$ ,  $\geq q R$ , or its negation for some  $A_i \in \Sigma$ ,  $\Sigma$ -role  $R$  and  $q \in Q$ . A  $\Sigma Q$ -type is a set  $\mathbf{t}$  of  $\Sigma Q$ -concepts containing  $\top$  such that the following conditions hold:

- for every  $\Sigma Q$ -concept  $C$ , either  $C \in \mathbf{t}$  or  $\neg C \in \mathbf{t}$ ,
- if  $q < q'$  are both in  $Q$  and  $\geq q' R \in \mathbf{t}$  then  $\geq q R \in \mathbf{t}$ ,
- if  $q < q'$  are both in  $Q$  and  $\neg(\geq q R) \in \mathbf{t}$  then  $\neg(\geq q' R) \in \mathbf{t}$ .

It should be clear that, for each  $\Sigma Q$ -type  $\mathbf{t}$  with  $\perp \notin \mathbf{t}$ , there is an interpretation  $\mathcal{I}$  and a point  $x$  in it such that, for every  $C \in \mathbf{t}$ , we have  $x \in C^{\mathcal{I}}$ . In this case we say that  $\mathbf{t}$  is *realised at  $x$  in  $\mathcal{I}$* , or that  $\mathbf{t}$  is the  $\Sigma Q$ -type of  $x$  in  $\mathcal{I}$ .

**Definition 6.** A set  $\Xi$  of  $\Sigma Q$ -types is said to be  $\mathcal{T}$ -realisable if there is a model for  $\mathcal{T}$  realising *all* types from  $\Xi$ . We also say that  $\Xi$  is *precisely  $\mathcal{T}$ -realisable* if there is a model  $\mathcal{I}$  for  $\mathcal{T}$  such that  $\mathcal{I}$  realises all types in  $\Xi$  and every  $\Sigma Q$ -type realised in  $\mathcal{I}$  is in  $\Xi$ .

Given a signature  $\Sigma$ , we say that interpretations  $\mathcal{I}$  and  $\mathcal{J}$  are  $\Sigma$ -isomorphic and write  $\mathcal{I} \sim_{\Sigma} \mathcal{J}$  if there is a bijection  $f: \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{J}}$  such that  $f(a^{\mathcal{I}}) = a^{\mathcal{J}}$ , for every object name  $a$ ,  $x \in A^{\mathcal{I}}$  iff  $f(x) \in A^{\mathcal{J}}$ , for every concept name  $A \in \Sigma$ , and  $(x, y) \in P^{\mathcal{I}}$  iff  $(f(x), f(y)) \in P^{\mathcal{J}}$ , for every role name  $P \in \Sigma$ . Clearly,  $\Sigma$ -isomorphic interpretations cannot be distinguished by TBoxes, ABoxes, or queries *over  $\Sigma$* .

Given a set  $\mathcal{I}_i$ ,  $i \in I$ , of interpretations with  $1 \in I$ , define the interpretation (the *disjoint union* of the  $\mathcal{I}_i$ )  $\mathcal{J} = \bigoplus_{i \in I} \mathcal{I}_i$ , where  $\Delta^{\mathcal{J}} = \{(i, w) \mid i \in I, w \in \Delta_i\}$ ,  $a^{\mathcal{J}} = (1, a^{\mathcal{I}_1})$ , for every object name  $a$ ,  $A^{\mathcal{J}} = \{(i, w) \mid w \in A^{\mathcal{I}_i}\}$ , for every concept name  $A$ , and  $P^{\mathcal{J}} = \{((i, w_1), (i, w_2)) \mid (w_1, w_2) \in P^{\mathcal{I}_i}\}$ , for every role name  $P$ . Given an interpretation  $\mathcal{I}$ , we set  $\mathcal{I}^{\omega} = \bigoplus_{i \in \omega} \mathcal{I}_i$ , where  $\mathcal{I}_i = \mathcal{I}$  for  $i \in \omega$ . Again, it should be clear that TBoxes, ABoxes or queries (over any signature) cannot distinguish between  $\mathcal{I}$  and  $\mathcal{I}^{\omega}$ .

The following lemma provides an important model-theoretic property of *DL-Lite<sub>bool</sub>* which is used to establish model-theoretic characterisations of various notions of conservativity.

**Lemma 1.** *Let  $\mathcal{J}$  be an (at most countable) model for  $\mathcal{T}_1$  and  $\Sigma$  a signature with  $\Sigma \subseteq \text{sig}(\mathcal{T}_1)$ . Suppose that there is a model for  $\mathcal{T}_1 \cup \mathcal{T}_2$  realising precisely the same  $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$ -types as  $\mathcal{J}$ , where  $Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$  is the set of all numerical parameters occurring in  $\mathcal{T}_1 \cup \mathcal{T}_2$  together with 1. Then there is a model  $\mathcal{I}^*$  for  $\mathcal{T}_1 \cup \mathcal{T}_2$  such that  $\mathcal{I}^* \sim_{\Sigma} \mathcal{J}^{\omega}$ .*

*In particular,  $\mathcal{I}^* \models \mathcal{A}$  iff  $\mathcal{J} \models \mathcal{A}$ , for all ABoxes  $\mathcal{A}$  over  $\Sigma$ ,  $\mathcal{I}^* \models \mathcal{T}$  iff  $\mathcal{J} \models \mathcal{T}$ , for all TBoxes  $\mathcal{T}$  over  $\Sigma$ , and  $\mathcal{I}^* \models q(\mathbf{a})$  iff  $\mathcal{J} \models q(\mathbf{a})$ , for all queries  $q(\mathbf{a})$  over  $\Sigma$ .*

In the case of  $DL-Lite_{horn}$  we need some extra definitions. By a  $\Sigma Q^h$ -concept we mean any concept of the form  $\perp$ ,  $A_i$  or  $\geq q R$ , for some  $A_i \in \Sigma$ ,  $\Sigma$ -role  $R$  and  $q \in Q$ . Given a  $\Sigma Q$ -type  $\mathbf{t}$ , let  $\mathbf{t}^+ = \{B \in \mathbf{t} \mid B \text{ a } \Sigma Q^h\text{-concept}\}$ . Say that a  $\Sigma Q$ -type  $\mathbf{t}_1$  is *h-contained* in a  $\Sigma Q$ -type  $\mathbf{t}_2$  if  $\mathbf{t}_1^+ \subseteq \mathbf{t}_2^+$ . The following two notions characterise conservativity for  $DL-Lite_{horn}$ :

**Definition 7.** A set  $\Xi$  of  $\Sigma Q$ -types is said to be *sub-precisely  $\mathcal{T}$ -realisable* if there is a model  $\mathcal{I}$  for  $\mathcal{T}$  such that  $\mathcal{I}$  realises all types from  $\Xi$ , and every  $\Sigma Q$ -type realised in  $\mathcal{I}$  is h-contained in some type from  $\Xi$ . We also say that  $\Xi$  is *join-precisely  $\mathcal{T}$ -realisable* if there is a model  $\mathcal{I}$  for  $\mathcal{T}$  such that, for every  $\Sigma Q$ -type  $\mathbf{t}$  realised in  $\mathcal{I}$ ,  $\Xi_{\mathbf{t}} = \{\mathbf{t}_i \in \Xi \mid \mathbf{t}^+ \subseteq \mathbf{t}_i^+\} \neq \emptyset$  and  $\mathbf{t}^+ = \bigcap_{\mathbf{t}_i \in \Xi_{\mathbf{t}}} \mathbf{t}_i^+$ . (It follows that  $\mathbf{t}^+ \subseteq \mathbf{t}_i^+$ , for all  $\mathbf{t}_i \in \Xi_{\mathbf{t}}$ , and thus,  $\Xi$  is sub-precisely  $\mathcal{T}$ -realisable.)

The table below gives characterisations of our four notions of conservativity in the following form: let  $\Sigma$  be a signature and  $\mathcal{L}$  be either  $DL-Lite_{bool}$  or  $DL-Lite_{horn}$ ; then  $\mathcal{T}_1 \cup \mathcal{T}_2$  is an  $\alpha$  conservative extension of  $\mathcal{T}_1$  w.r.t.  $\Sigma$  in  $\mathcal{L}$  iff every precisely  $\mathcal{T}_1$ -realisable set  $\Xi$  of  $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$  types is ‘...  $\mathcal{T}_1 \cup \mathcal{T}_2$ -realisable’ ( $Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$  is the set of numerical parameters occurring in  $\mathcal{T}_1 \cup \mathcal{T}_2$  together with 1).

conservativity $\alpha$	language $\mathcal{L}$	
	$DL-Lite_{bool}$	$DL-Lite_{horn}$
deductive	$\mathcal{T}_1 \cup \mathcal{T}_2$ -realisable	$\mathcal{T}_1 \cup \mathcal{T}_2$ -realisable
query	precisely $\mathcal{T}_1 \cup \mathcal{T}_2$ -realisable	sub-precisely $\mathcal{T}_1 \cup \mathcal{T}_2$ -realisable
strong deductive		join-precisely $\mathcal{T}_1 \cup \mathcal{T}_2$ -realisable
strong query		

These characterisations are proved in the Appendix, where it is also shown that in each case it suffices to consider only those sets  $\Xi$  the size of which is bounded by a polynomial function in the size of the TBoxes. Then, for  $DL-Lite_{bool}$  TBoxes  $\mathcal{T}_1$  and  $\mathcal{T}_2$ , one can decide whether  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a *not* a strong deductive conservative extension by (non-deterministically) guessing a polynomial set  $\Xi$  of  $\Sigma Q_{\mathcal{T}_1 \cup \mathcal{T}_2}$ -types and checking that it is precisely  $\mathcal{T}_1$ -realisable and *not* precisely  $\mathcal{T}_1 \cup \mathcal{T}$ -realisable. The Appendix provides a polynomial *non-deterministic* algorithm deciding whether a given set of  $\Sigma Q$ -types is precisely  $\mathcal{T}$ -realisable, which yields a  $\Pi_2^p$  upper bound for strong deductive conservativity in  $DL-Lite_{bool}$ . Similarly, for  $DL-Lite_{horn}$  TBoxes  $\mathcal{T}_1$  and  $\mathcal{T}_2$ , one can decide whether  $\mathcal{T}_1 \cup \mathcal{T}_2$  is a query (or strong deductive) conservative extension of  $\mathcal{T}_1$  by guessing  $\Xi$  and checking that it is precisely  $\mathcal{T}_1$ -realisable and *not* sub-precisely (respectively, join-precisely)  $\mathcal{T}_1 \cup \mathcal{T}_2$ -realisable. The Appendix provides polynomial *deterministic* algorithms deciding whether a given set of  $\Sigma Q$ -types is precisely, sub-precisely and join-precisely  $\mathcal{T}$ -realisable, for a  $DL-Lite_{horn}$  TBox  $\mathcal{T}$ , which give CONP upper bounds for query and strong deductive conservativity. The lower bounds follow immediately from the corresponding lower bounds for propositional logic.

## 5 Conclusion

We have analysed the relation between different notions of conservative extension in description logics  $DL-Lite_{bool}$  and  $DL-Lite_{horn}$ , and proved that the corresponding reasoning problems are not harder than the same problems in propositional logic. Moreover, we have also shown that important meta-properties for modular ontology engineering, such as robustness under joins, vocabulary extensions, and iterated import of ontologies, hold true for these notions of conservativity.

**Acknowledgements.** This work was partially supported by U.K. EPSRC grant GR/S63175.

## References

1. Antoniou, G., Kehagias, A.: A note on the refinement of ontologies. *Int. J. of Intelligent Systems* **15**(7) (2000) 623–632
2. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logic. In: *Proc. of KR 2006*, AAAI Press (2006) 187–197
3. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modularity of ontologies. In: *Proc. of IJCAI 2007*. (2007) 298–303
4. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: *Proc. of IJCAI 2007*. (2007) 453–458
5. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: Extracting modules from ontologies. In: *Proc. of the 16th International World Wide Web Conference (WWW-2007)*. (2007)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: *Proc. of AAAI 2005*. (2005) 602–607
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: *Proc. of KR 2006*. (2006) 260–270
8. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: DL-Lite in the light of first-order logic. In: *Proc. of AAAI 2007*. (2007)
9. Lutz, C., Wolter, F.: Conservative extensions in the lightweight description logic EL. To appear in the *Proceedings of 21st Conference on Automated Deduction (CADE-21)* (2007)
10. Chang, C., Keisler, H.: *Model Theory*. Elsevier (1990)
11. Diaconescu, R., Goguen, J., Stefaneas, P.: Logical support for modularisation. In Huet, G., Plotkin, G., eds.: *Logical Environments*, Cambridge University Press, New York (1993) 83–130
12. Pitts, A.: On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. Symbolic Logic* **57**(1) (1992) 33–52
13. Visser, A.: Uniform interpolation and layered bisimulation. In Hájek, P., ed.: *Gödel '96 (Brno, 1996)*. Volume 6 of *Lecture Notes in Logic*. Springer Verlag (1996) 139–164
14. Papadimitriou, C.: *Computational Complexity*. Addison Wesley Publ. Co. (1994)

## Data Complexity in the $\mathcal{EL}$ family of DLs

Adila Krisnadhi<sup>1</sup> and Carsten Lutz<sup>2</sup>

<sup>1</sup> Faculty of Computer Science, University of Indonesia

<sup>2</sup> Institute for Theoretical Computer Science, TU Dresden, Germany

adila@cs.ui.ac.id, lutz@tcs.inf.tu-dresden.de

### 1 Introduction

In recent years, lightweight description logics (DLs) have experienced increased interest because they admit highly efficient reasoning on large-scale ontologies. Most prominently, this is witnessed by the ongoing research on the DL-Lite and  $\mathcal{EL}$  families of DLs, but see also [11, 14] for other examples. The main application of  $\mathcal{EL}$  and its relatives is as an ontology language [5, 1, 3]. In particular, the DL  $\mathcal{EL}^{++}$  proposed in [1] admits tractable reasoning while still providing sufficient expressive power to represent, for example, life-science ontologies. In contrast, the DL-Lite family of DLs is specifically tailored towards data intensive applications [8, 6, 7]. In such applications, instance checking and conjunctive query answering are the most relevant reasoning tasks and should thus be computationally cheap, preferably tractable. When determining the computational complexity of these task for a given DL, it is often realistic to consider *data complexity*, where the size of the input is measured only in terms of the ABox (which represents the data and tends to be large), but not in terms of the TBox and the query concept (which tend to be comparatively small). This is in contrast to *combined complexity*, where also the size of the TBox and query concept are taken into account.

The aim of this paper is to analyse the suitability of the  $\mathcal{EL}$  family of DLs for data intensive applications. In particular, we analyse the data complexity of instance checking and conjunctive query answering in extensions of  $\mathcal{EL}$ . For the DL-Lite family, such an investigation has been carried out in [7], with complexities ranging from LOGSPACE-complete to coNP-complete. It follows from the results in [7] that, at least w.r.t. general TBoxes, we cannot expect the data complexity to be below PTIME for members of the  $\mathcal{EL}$  family. The reason is that, in a crucial aspect, DL-Lite is even more lightweight than  $\mathcal{EL}$ : in contrast to the latter, the former does not allow for qualified existential (nor universal) restrictions and thus the interaction between different domain elements is very limited. When analyzing the data complexity of instance checking and conjunctive query answering in  $\mathcal{EL}$  and its extensions, we therefore concentrate on mapping out the boundary of tractability.

We consider a wide range of extensions of  $\mathcal{EL}$ , and analyze the data complexity of the mentioned tasks with acyclic TBoxes and with general TBoxes. When selecting extensions of  $\mathcal{EL}$ , we focus on DLs for which instance checking has been proved *intractable* regarding combined complexity in [1]. We show that, in most of these extensions, instance checking is also intractable regarding data complexity. The notable

exceptions are  $\mathcal{EL}$  extended with globally functional roles and  $\mathcal{EL}$  extended with inverse roles. It is shown in [2] that instance checking in these DLs is EXPTIME-complete regarding combined complexity. On the other hand, it follows from results in [11] that instance checking is tractable regarding data complexity in  $\mathcal{ELI}^f$ , the extension of  $\mathcal{EL}$  with both globally functional and inverse roles. In this paper, we extend this result to conjunctive query answering in  $\mathcal{ELI}^f$  is still tractable regarding data complexity.

We recommend to the reader the papers [15, 16], which also analyze the complexity of conjunctive query answering in extensions of  $\mathcal{EL}$ . The results in these papers have been obtained independently of and in parallel to the results in the current paper.

## 2 Preliminaries

We use standard notation for the syntax and semantics of  $\mathcal{EL}$  and its extensions, see [4]. The additional constructors we consider are atomic negation  $\neg A$ , disjunction  $C \sqcup D$ , sink restrictions  $\forall r.\perp$ , value restrictions  $\forall r.C$ , at-most restrictions ( $\leq nr$ ), at-least restrictions ( $\geq nr$ ), inverse roles  $\exists r^-.C$ , role negation  $\exists \neg r.C$ , role union  $\exists r \cup s.C$ , and transitive closure of roles  $\exists r^+.C$ . We denote extensions of  $\mathcal{EL}$  in a canonical way, writing e.g.  $\mathcal{EL}^{\forall r.\perp}$  for  $\mathcal{EL}$  extended with sink restrictions and  $\mathcal{EL}^{C \sqcup D}$  for  $\mathcal{EL}$  extended with disjunction. Since we perform a very fine grained analysis,  $\mathcal{EL}^{(\leq nr)}$  means the extension of  $\mathcal{EL}$  with ( $\leq nr$ ) for some *fixed*  $n$  (but not for some fixed  $r$ ). We will also consider  $\mathcal{EL}$  extended with *global* at-most restrictions:  $\mathcal{EL}^{kf}$  denotes the version of  $\mathcal{EL}$  obtained by reserving a set of *k-functional* roles that satisfy  $|\{e \mid (d, e) \in r^{\mathcal{I}}\}| \leq k$  for all interpretations  $\mathcal{I}$  and all  $d \in \Delta^{\mathcal{I}}$ . Instead of 1-functional roles, we will speak of functional roles as usual.

We will consider acyclic TBoxes which are defined in the usual way, and general TBoxes which are finite sets of concept inclusions  $C \sqsubseteq D$ . As usual when analyzing data complexity, we do not admit complex concepts in the ABox. Thus, ABoxes are sets of assertions  $A(a)$  and  $r(a, b)$ , where  $A$  is a concept name. Most of our results do not depend on the unique name assumption (UNA), which states that  $a^{\mathcal{I}} \neq b^{\mathcal{I}}$  for all distinct individual names  $a, b$ . Whenever they do, we will state explicitly whether the UNA is adopted or not. We write  $\mathcal{A}, \mathcal{T} \models C(a)$  to denote that  $a$  is an instance of  $C$  w.r.t.  $\mathcal{A}$  and  $\mathcal{T}$  (defined in the usual way). Also, we use  $\text{Ind}(\mathcal{A})$  to denote the set of individual names occurring in  $\mathcal{A}$ .

Since conjunctive query answering is not a decision problem, we will study *conjunctive query entailment* instead. For us, a *conjunctive query* is a set  $q$  of atoms  $A(v)$  and  $r(u, v)$ , where  $A$  is a concept name,  $r$  a role name or an inverse role, and  $u, v$  are variables. We use  $\text{Var}(q)$  to denote the variables used in  $q$ . If  $\mathcal{I}$  is an interpretation and  $\pi$  is a mapping from  $\text{Var}(q)$  to  $\Delta^{\mathcal{I}}$ , we write  $\mathcal{I} \models^{\pi} A(v)$  if  $\pi(v) \in A^{\mathcal{I}}$ ,  $\mathcal{I} \models^{\pi} r(u, v)$  if  $(\pi(u), \pi(v)) \in r^{\mathcal{I}}$ , and  $\mathcal{I} \models^{\pi} q$  if  $\mathcal{I} \models^{\pi} \alpha$  for all  $\alpha \in q$ . If  $\pi$  is not important, we simply write  $\mathcal{I} \models q$ . Finally,  $\mathcal{A}, \mathcal{T} \models q$  means that for all models  $\mathcal{I}$  of the ABox  $\mathcal{A}$  and the TBox  $\mathcal{T}$ , we have  $\mathcal{I} \models q$ . Now, *conjunctive query entailment* is to decide given  $\mathcal{A}$ ,  $\mathcal{T}$ , and  $q$ , whether  $\mathcal{A}, \mathcal{T} \models q$ . It is not hard to see that instance checking is a special case of conjunctive query entailment. Note that we do not allow individual names in conjunctive queries in place of variables. It is well-known that conjunctive query entail-

ment in which individual names are allowed in the query can be polynomially reduced to conjunctive query entailment as introduced here, see for example [9].

### 3 Lower Bounds

In [17], Schaerf proves that instance checking in  $\mathcal{EL}^{\neg A}$  w.r.t. empty TBoxes is co-NP-hard regarding data complexity. He uses a reduction from a variant of SAT that he calls 2+2-SAT. Our lower bounds for extensions of  $\mathcal{EL}$  are obtained by variations of Schaerf’s reduction. They all apply to the case of acyclic TBoxes.

Before we start, a note on TBoxes is in order. We will usually not consider the case where there is no TBox at all because, then, ABoxes that are restricted to concept *names* are extremely inexpressive. Actually, it is not hard to show that, without TBoxes, conjunctive query containment is tractable regarding data complexity for all extensions of  $\mathcal{EL}$  considered in this paper with the exception of  $\mathcal{EL}^{kf}$ , for which it is coNP-complete (which is proved below).

#### 3.1 Basic Cases

A 2+2 *clause* is of the form  $(p_1 \vee p_2 \vee \neg n_1 \vee \neg n_2)$ , where each of  $p_1, p_2, n_1, n_2$  is a propositional letter or a truth constant 1, 0. A 2+2 *formula* is a finite conjunction of 2+2 clauses. Now, 2+2-SAT is the problem of deciding whether a given 2+2 formula is satisfiable. It is shown in [17] that 2+2-SAT is NP-complete. To get started with our lower bound proofs, we repeat Schaerf’s proof showing that instance checking in  $\mathcal{EL}$  extended with primitive negation is co-NP-hard regarding data complexity.

Let  $\varphi = c_0 \wedge \dots \wedge c_{n-1}$  be a 2+2-formula in  $m$  propositional letters  $q_0, \dots, q_{m-1}$ . Let  $c_i = p_{i,1} \vee p_{i,2} \vee \neg n_{i,1} \vee \neg n_{i,2}$  for all  $i < n$ . We use  $f$ , the propositional letters  $q_0, \dots, q_{m-1}$ , the truth constants 1, 0, and the clauses  $c_0, \dots, c_{n-1}$  as individual names. Define the TBox  $\mathcal{T}$  as  $\{\bar{A} \doteq \neg A\}$  and the ABox  $\mathcal{A}_\varphi$  as follows, where  $c, p_1, p_2, n_1$ , and  $n_2$  are role names:

$$\begin{aligned} \mathcal{A}_\varphi := & \{A(1), \bar{A}(0), c(f, c_0), \dots, c(f, c_{n-1})\} \\ & \cup \bigcup_{i < n} \{p_1(c_i, p_{i,1}), p_2(c_i, p_{i,2}), n_1(c_i, n_{i,1}), n_2(c_i, n_{i,2})\} \end{aligned}$$

Models of  $\mathcal{A}_\varphi$  and  $\mathcal{T}$  represent truth assignments for  $\varphi$  by way of setting  $q_i$  to true iff  $q_i \in A^{\mathcal{I}}$ . Set  $C := \exists c. (\exists p_1. \bar{A} \sqcap \exists p_2. \bar{A} \sqcap \exists n_1. A \sqcap \exists n_2. A)$ . Intuitively,  $C$  expresses that  $\varphi$  is not satisfied, i.e., there is a clause in which the two positive literals and the two negative literals are all false. It is not hard to show that  $\mathcal{A}_\varphi, \mathcal{T} \not\models C(f)$  iff  $\varphi$  is satisfiable. Thus, instance checking in  $\mathcal{EL}^{\neg A}$  w.r.t. acyclic TBoxes is co-NP-hard regarding data complexity.

This reduction can easily be adapted to  $\mathcal{EL}^{\forall r. \perp}$ . In all interpretations  $\mathcal{I}$ ,  $\exists r. \top$  and  $\forall r. \perp$  partition the domain  $\Delta^{\mathcal{I}}$  and can thus be used to simulate the concept name  $A$  and its negation  $\neg A$  in the original reduction. We can thus simply replace the TBox  $\mathcal{T}$  with  $\mathcal{T}' := \{A \doteq \exists r. \top, \bar{A} \doteq \forall r. \perp\}$ .

In some extensions of  $\mathcal{EL}$ , we only find concepts that cover the domain, but not necessarily partition it. An example is  $\mathcal{EL}^{(\leq kr)}$ ,  $k \geq 1$ , in which  $\exists r. \top$  and  $(\leq kr)$

provide a covering (for  $k = 0$ , observe that  $(\leq k r)$  is equivalent to  $\forall r.\perp$ ). Interestingly, this does not pose a problem for the reduction. In the case of  $\mathcal{EL}^{(\leq kr)}$ , we use the TBox  $\mathcal{T} := \{A \doteq \exists r.\top, \bar{A} \doteq (\leq k r)\}$ , and the ABox  $\mathcal{A}_\varphi$  as well as the query concept  $C$  remain unchanged. Let us show that  $\mathcal{A}_\varphi, \mathcal{T} \not\models C(f)$  iff  $\varphi$  is satisfiable. For the “if” direction, it is straightforward to convert a truth assignment satisfying  $\varphi$  into a model  $\mathcal{I}$  of  $\mathcal{A}_\varphi$  and  $\mathcal{T}$  such that  $f \notin C^{\mathcal{I}}$ . For the “only if” direction, let  $\mathcal{I}$  be a model of  $\mathcal{A}_\varphi$  and  $\mathcal{T}$  such that  $f \notin C^{\mathcal{I}}$ . Define a truth assignment  $t$  by choosing for each propositional letter  $q_i$ , a truth value  $t(q_i)$  such that  $t(q_i) = 1$  implies  $q_i^{\mathcal{I}} \in A$  and  $t(q_i) = 0$  implies  $q_i^{\mathcal{I}} \in \bar{A}$ . Such a truth assignment exists since  $A$  and  $\bar{A}$  cover the domain. However, it is not necessarily unique since  $A$  and  $\bar{A}$  need not be disjoint. To show that  $t$  satisfies  $\varphi$ , assume that it does not. Then there is a clause  $c_i = (p_{i,1} \vee p_{i,2} \vee \neg n_{i,1} \vee \neg n_{i,2})$  that is not satisfied by  $t$ . By definition of  $t$  and  $\mathcal{A}_\varphi$ , it is not hard to show that  $c_i^{\mathcal{I}} \in (\exists p_1.\bar{A} \sqcap \exists p_2.\bar{A} \sqcap \exists n_1.A \sqcap \exists n_2.A)^{\mathcal{I}}$  and thus  $f \in C^{\mathcal{I}}$ , which is a contradiction.

The cases  $\mathcal{EL}^{\forall r.C}$  and  $\mathcal{EL}^{\exists \neg r.C}$  can be treated similarly because a covering of the domain can be achieved by choosing the concepts  $\exists r.\top$  and  $\forall r.X$  in the case of  $\mathcal{EL}^{\forall r.C}$ , and  $\exists r.\top$  and  $\exists \neg r.\top$  in the case of  $\mathcal{EL}^{\exists \neg r.C}$ . In the case,  $\mathcal{EL}^{C \sqcup D}$ , we use a TBox  $\mathcal{T}' := \{V \doteq X \sqcup Y\}$ . In all models of  $\mathcal{T}'$ , the extension of  $V$  is covered by the concepts  $X$  and  $Y$ . Thus, we can use the above ABox  $\mathcal{A}_\varphi$ , add  $V(q_i)$  for all  $i < m$ , and use the TBox  $\mathcal{T} := \mathcal{T}' \cup \{A \doteq X, \bar{A} \doteq Y\}$  and the same query concept  $C$  as above. The case  $\mathcal{EL}^{\exists r^+.C}$  is quite similar. In all models of the TBox  $\mathcal{T}' := \{V \doteq \exists r^+.C\}$ , the extension of  $V$  is covered by the concepts  $\exists r.C$  and  $\exists r.\exists r^+.C$ . Thus, we can use the same ABox and query concept as for  $\mathcal{EL}^{C \sqcup D}$ , together with the TBox  $\mathcal{T} := \mathcal{T}' \cup \{A \doteq \exists r.C, \bar{A} \doteq \exists r.\exists r^+.C\}$ .

**Theorem 1.** *For the following, instance checking w.r.t. acyclic TBoxes is co-NP-hard regarding data complexity:  $\mathcal{EL}^{\neg A}$ ,  $\mathcal{EL}^{\forall r.\perp}$ ,  $\mathcal{EL}^{\forall r.C}$ ,  $\mathcal{EL}^{\exists \neg r.C}$ ,  $\mathcal{EL}^{C \sqcup D}$ ,  $\mathcal{EL}^{\exists r^+.C}$ , and  $\mathcal{EL}^{(\leq kr)}$  for all  $k \geq 0$ .*

### 3.2 Cases that depend on the UNA

The results in the previous subsection are independent of whether or not the UNA is adopted. In the following, we consider some cases that depend on the (non-)UNA, starting with  $\mathcal{EL}^{(\geq kr)}$ .

In  $\mathcal{EL}^{(\geq kr)}$ ,  $k \geq 2$ , it does not seem possible to find two concepts that a priori cover the domain and can be used to represent truth values in truth assignments. However, if we add slightly more structure to the ABox, such concepts can be found. We treat only the case  $k = 3$  explicitly, but it easily generalizes to other values of  $k$ . Consider the ABox

$$\mathcal{A} = \{r(a, b_1), r(a, b_2), r(a, b_3), r(b_1, b_2), r(b_2, b_3), r(b_1, b_3)\}.$$

Without the UNA, there are two cases for models of  $\mathcal{A}$ : either two of  $b_1, b_2, b_3$  identify the same domain element or they do not. In the first case,  $a$  satisfies  $\exists r^4.\top$ , where  $\exists r^4$  denotes the four-fold nesting of  $\exists r$ . In the second case,  $a$  satisfies  $(\geq 3 r)$ . It follows that we can reduce satisfiability of 2+2 formulas using a reduction very similar to the one for  $\mathcal{EL}^{(\neg A)}$ . The main differences are that (i) a copy of  $\mathcal{A}$  is plugged in for each  $q_i$ , with  $a$  replaced by  $q_i$  and (ii) we use the TBox  $\mathcal{T} := \{A \doteq \exists r^4.\top, \bar{A} \doteq (\geq 3 r)\}$ .

Unlike the previous results, this lower bound clearly depends on the fact that the UNA is not adopted. We leave it as an open problem whether instance checking in  $\mathcal{EL}^{(\geq k r)}$  w.r.t. acyclic TBoxes is tractable if the UNA is adopted. In the following, we show that instance checking becomes coNP-hard under the UNA if we admit general TBoxes. Again, we only treat the case  $k = 3$  explicitly. Define a TBox

$$\mathcal{T} := \{ \begin{array}{l} V \sqsubseteq \exists r.X \sqcap \exists r.Y \sqcap \exists r.Z \\ (\geq 3r) \sqsubseteq A \\ \exists r.(X \sqcap Y) \sqsubseteq \bar{A} \quad \exists r.(X \sqcap Z) \sqsubseteq \bar{A} \quad \exists r.(Y \sqcap Z) \sqsubseteq \bar{A} \end{array} \}.$$

In models of  $\mathcal{T}$ , the extension of  $V$  is covered by  $A$  and  $\bar{A}$ . Therefore, we can adapt the reduction by using the reduction ABox defined for  $\mathcal{EL}^{C \sqcup D}$ .

**Theorem 2.** *For  $\mathcal{EL}^{(\geq k r)}$  with  $k \geq 2$ , instance checking is co-NP-hard in the following cases: (i) w.r.t. acyclic TBoxes and without UNA and (ii) w.r.t. general TBoxes and with (or without) UNA.*

Another case that depends on the (non-)UNA is  $\mathcal{EL}^{kf}$  with  $k \geq 2$ . We can prove coNP-hardness provided that the UNA is not adopted. For the case  $\mathcal{EL}^{1f}$ , we will prove in Section 4 that instance checking (and even conjunctive query entailment) is tractable regarding data complexity, with or without the UNA. For simplicity, we only treat the case  $\mathcal{EL}^{2f}$  explicitly. It is easy to generalize our argument to larger values of  $k$ . Like in  $\mathcal{EL}^{(\geq 3r)}$ , in  $\mathcal{EL}^{2f}$  it does not seem possible to find two concepts that cover the domain without providing additional structure via an ABox. Set

$$\mathcal{A}' = \{r(a, b_1), r(a, b_2), r(a, b_3), r(b_1, b_2), A(b_1), A(b_2), B(b_3)\}.$$

where  $r$  is 2-functional and thus at least two of  $b_1, b_2, b_3$  have to identify the same domain element. We can distinguish two cases: either  $b_3$  is identified with  $b_1$  or  $b_2$ , then  $a$  satisfies  $\exists r.(A \sqcap B)$ . Or  $b_1$  and  $b_2$  are identified, then  $a$  satisfies  $\exists r^3.\top$ , where  $\exists r^3$  denotes the three-fold nesting of  $\exists r$ . It follows that we can reduce satisfiability of 2+2 formulas using a reduction very similar to that for  $\mathcal{EL}^{(\geq 3r)}$  above. Interestingly, we do not need a TBox at all to make this work. We take the original ABox  $\mathcal{A}_\varphi$  defined for  $\mathcal{EL}^{-A}$ , add a copy of  $\mathcal{A}'$  for each  $q_i$  with  $a$  replaced by  $q_i$ , and replace  $A(1)$  with  $\{r(1, e), A(e), B(e)\}$  and  $\bar{A}(0)$  with  $\{r(\perp, e_0), r(e_0, e_1), r(e_1, e_2)\}$ . Thus, 1 satisfies  $\exists r.(A \sqcap B)$  (representing true) and 0 satisfies  $\exists r^3.\top$  (representing false). It remains to modify the query concept to  $C' := \exists c.(\exists p_1.\exists r^3.\top \sqcap \exists p_2.\exists r^3.\top \sqcap \exists n_1.\exists r.(A \sqcap B) \sqcap \exists n_2.\exists r.(A \sqcap B))$ .

With the UNA and without TBoxes, instance checking in  $\mathcal{EL}^{kf}$ ,  $k \geq 2$  is tractable regarding data complexity. The same holds for conjunctive query answering. In a nutshell, a polytime algorithm is obtained by considering the input ABox as a (complete) description of an interpretation and then checking all possible matches of the conjunctive query. A special case that has to be taken into account are inconsistent ABoxes such as those containing  $\{r(a, b_1), r(a, b_2), r(a, b_3)\}$  for a 2-functional role  $r$  and with the  $b_i$  mutually distinct. Such inconsistencies are easily detected. If found, the algorithm returns “yes” because an inconsistent ABox entails every consequence.

If we add TBoxes, instance checking in  $\mathcal{EL}^{kf}$ ,  $k \geq 2$  becomes co-NP-hard also with the UNA. We only treat the case  $k = 3$ , but our arguments generalize. As in the case of  $\mathcal{EL}^{2f}$  without UNA, we have to give additional structure to the ABox. Consider the TBox  $\mathcal{T}' = \{V \doteq \exists r.B\}$  and the ABox

$$\mathcal{A} = \{V(a), r(a, b_1), r(a, b_2), r(a, b_3), s(a, b_1), s'(a, b_2), s'(a, b_3)\}.$$

with  $r$  a 3-functional role. Then  $a$  satisfies  $\exists r.B$  in all models  $\mathcal{I}$  of  $\mathcal{A}$  and  $\mathcal{T}'$ . Because of the UNA, we can distinguish two cases: either  $b_1$  satisfies  $B$  or one of  $b_2, b_3$  does. In the first case,  $a$  satisfies  $\exists s.A$  and in the second case, it satisfies  $\exists s'.A$ . We can now continue the reduction as in the previous cases. Start with the ABox  $\mathcal{A}_\varnothing$  from the reduction for  $\mathcal{EL}^{-A}$  and add  $V(q_i)$  for all  $i < m$ . Then use the TBox  $\mathcal{T} = \mathcal{T}' \cup \{A \doteq \exists s.A, \bar{A} \doteq \exists s'.A\}$  and the original query concept  $C$ .

**Theorem 3.** For  $\mathcal{EL}^{kf}$  with  $k \geq 2$ , instance checking is

- tractable w.r.t. the empty TBox and with UNA;
- co-NP-hard in the following cases: (i) w.r.t. the empty TBox and without UNA, and (ii) w.r.t. acyclic TBoxes and with UNA.

## 4 Upper Bound

We consider the extension  $\mathcal{ELI}^f$  of  $\mathcal{EL}$  with inverse roles and globally functional roles. If any of these two is added to  $\mathcal{EL}$ , instance checking w.r.t. general TBoxes becomes EXPTIME-complete regarding combined complexity [1]. However, it follows from the results on Horn- $\mathcal{SHIQ}$  in [11] that instance checking in  $\mathcal{ELI}^f$  w.r.t. general TBoxes is tractable regarding data complexity. A direct proof can be found in [12]. Here, we show that even conjunctive query answering in  $\mathcal{ELI}^f$  is tractable regarding data complexity.

In  $\mathcal{ELI}^f$ , roles and also their inverses can be declared functional using statements  $\top \sqsubseteq (\leq 1 r)$  in the TBox. For conveniently dealing with inverse roles, we use the following convention: if  $r = s^-$  (with  $s$  a role name), then  $r^-$  denotes  $s$ .

As a preliminary, we assume that TBoxes and ABoxes are in a certain normal form, which we introduce next. For TBoxes, we assume that all concept inclusions are of one of the following forms, where  $A, A_1, A_2$ , and  $B$  are concept names or  $\top$  and  $r$  is a role name or an inverse role:

$$\begin{array}{lll} A \sqsubseteq B, & A \sqsubseteq \exists r.B, & \top \sqsubseteq (\leq 1 r) \\ A_1 \sqcap A_2 \sqsubseteq B, & \exists r.A \sqsubseteq B & \end{array}$$

The normal form for ABoxes simply requires that  $r(a, b) \in \mathcal{A}$  iff  $r^-(b, a) \in \mathcal{A}$ , for all role names  $r$  and individual names  $a, b$ .

Let  $\mathcal{A}$  be an ABox and  $\mathcal{T}$  a TBox.  $\mathcal{T}$  can be converted into normal form  $\mathcal{T}'$  in polytime, by introducing additional concept names. See [1] for more details. Converting  $\mathcal{A}$  into normal form  $\mathcal{A}'$  can obviously also be done in polytime. Moreover, it is not too difficult to see that for every conjunctive query  $q$  not using any of the concept names that occur in  $\mathcal{T}'$  but not in  $\mathcal{T}$ , we have  $\mathcal{A}, \mathcal{T} \models q$  iff  $\mathcal{A}', \mathcal{T}' \models q$ .

Another (standard) assumption that we make w.l.o.g. is that conjunctive queries are connected, i.e., for all  $u, v \in \text{Var}(q)$ , there are atoms  $r(u_0, u_1), \dots, r(u_{n-1}, u_n) \in q$ ,  $n \geq 0$ , such that  $u = u_0$  and  $v = u_n$ . Entailment of non-connected queries is easily (and polynomially) reduced to entailment of connected queries, see e.g. [9].

Our algorithm for conjunctive query answering in  $\mathcal{EL}^f$  is based on canonical models. To introduce canonical models, we need some preliminaries. Let  $\mathcal{T}$  be a TBox and  $\Gamma$  a finite set of concept names. We use

$$\text{sub}_{\mathcal{T}}(\Gamma) := \{A \in \mathbf{N}_{\mathcal{C}}^{\mathcal{T}} \mid \prod_{A' \in \Gamma} A' \sqsubseteq_{\mathcal{T}} A\}$$

to denote the *closure* of  $\Gamma$  under subsuming concept names w.r.t.  $\mathcal{T}$ . For the next definition, the reader should intuitively assume that we want to make all elements of  $\Gamma$  (jointly) true at a domain element in a model of  $\mathcal{T}$ . If  $A \in \Gamma$  and  $A \sqsubseteq \exists r.B \in \mathcal{T}$ , then we say that  $\Gamma$  has  $\exists r.B$ -obligation  $O$ , where

$$O = \text{sub}_{\mathcal{T}}(\{B\} \cup \{B' \in \mathbf{N}_{\mathcal{C}}^{\mathcal{T}} \mid \exists A' \in \Gamma : \exists r^{-}.A' \sqsubseteq B' \in \mathcal{T}\} \cup O'),$$

and  $O' = \emptyset$  if  $\top \sqsubseteq (\leq 1 r) \notin \mathcal{T}$  and  $O' = \{B' \in \mathbf{N}_{\mathcal{C}}^{\mathcal{T}} \mid \exists A' \in \Gamma : A' \sqsubseteq \exists r.B' \in \mathcal{T}\}$  otherwise.

Let  $\mathcal{T}$  be a TBox and  $\mathcal{A}$  an ABox, both in normal form, for which we want to decide conjunctive query entailment (for a yet unspecified query  $q$ ). To define a canonical model for  $\mathcal{A}$  and  $\mathcal{T}$ , we have to require that  $\mathcal{A}$  is *admissible* w.r.t.  $\mathcal{T}$ . What admissibility means depends on whether or not we make the UNA:  $\mathcal{A}$  is admissible w.r.t.  $\mathcal{T}$  if (i) the UNA is made and  $\mathcal{A}$  is consistent w.r.t.  $\mathcal{T}$  or (ii) the UNA is not made and  $(\top \sqsubseteq (\leq 1 r)) \in \mathcal{T}$  implies that there are no  $a, b, c \in \text{Ind}(\mathcal{A})$  with  $r(a, b), r(a, c) \in \mathcal{A}$  and  $b \neq c$ .

We define a sequence of interpretations  $\mathcal{I}_0, \mathcal{I}_1, \dots$ , and the canonical model for  $\mathcal{A}$  and  $\mathcal{T}$  will then be the limit of this sequence. To facilitate the construction, it is helpful to use domain elements that have an internal structure. An *existential* for  $\mathcal{T}$  is a concept  $\exists r.A$  that occurs on the right-hand side of some inclusion in  $\mathcal{T}$ . A *path*  $p$  for  $\mathcal{T}$  is a finite (possibly empty) sequence of existentials for  $\mathcal{T}$ . We use  $\text{ex}(\mathcal{T})$  to denote the set of all existentials for  $\mathcal{T}$ ,  $\text{ex}(\mathcal{T})^*$  to denote the set of all paths for  $\mathcal{T}$ , and  $\varepsilon$  to denote the empty path. All interpretations  $\mathcal{I}_i$  in the above sequence will satisfy

$$\Delta^{\mathcal{I}_i} := \{\langle a, p \rangle \mid a \in \text{Ind}(\mathcal{A}) \text{ and } p \in \text{ex}^*(\mathcal{T})\}$$

For convenience, we use a slightly non-standard representation of interpretations when defining the sequence  $\mathcal{I}_0, \mathcal{I}_1, \dots$  and canonical interpretations: the function  $\cdot^{\mathcal{I}}$  maps every element  $d \in \Delta^{\mathcal{I}}$  to a set of concept names  $d^{\mathcal{I}}$  instead of every concept name  $A$  to a set of elements  $A^{\mathcal{I}}$ . It is obvious how to translate back and forth between the standard representation and this one, and we will switch freely in what follows.

To start to construction of the sequence  $\mathcal{I}_0, \mathcal{I}_1, \dots$ , define  $\mathcal{I}_0$  as follows:

$$\begin{aligned} \Delta^{\mathcal{I}_0} &:= \{\langle a, \varepsilon \rangle \mid a \in \text{Ind}(\mathcal{A})\} \\ r^{\mathcal{I}_0} &:= \{\langle \langle a, \varepsilon \rangle, \langle b, \varepsilon \rangle \rangle \mid r(a, b) \in \mathcal{A}\} \\ \langle a, \varepsilon \rangle^{\mathcal{I}_0} &:= \{A \in \mathbf{N}_{\mathcal{C}} \mid \mathcal{A}, \mathcal{T} \models A(a)\} \\ a^{\mathcal{I}_0} &:= \langle a, \varepsilon \rangle \end{aligned}$$

Now assume that  $\mathcal{I}_i$  has already been defined. We want to construct  $\mathcal{I}_{i+1}$ . An element  $\langle a, p \rangle \in \Delta^{\mathcal{I}_i}$  is a *leaf* in  $\mathcal{I}_i$  if there is no  $\alpha \in \text{ex}(\mathcal{T})$  such that  $\langle a, p\alpha \rangle \in \Delta^{\mathcal{I}_i}$ . If existant, select a leaf  $\langle a, p \rangle$  and an  $\alpha = \exists r.A \in \text{ex}(\mathcal{T})$  such that  $\langle a, p \rangle^{\mathcal{I}_i}$  has  $\alpha$ -obligation  $O$  and (i)  $(\top \sqsubseteq (\leq 1 r)) \notin \mathcal{T}$  or (ii) there is no  $\langle b, q \rangle \in \Delta^{\mathcal{I}_i}$  with  $(\langle a, p \rangle, \langle b, q \rangle) \in r^{\mathcal{I}_i}$ . Then do the following:

- add  $\langle a, p\alpha \rangle$  to  $\Delta^{\mathcal{I}_i}$ ;
- if  $r$  is a role name, add  $(\langle a, p \rangle, \langle a, p\alpha \rangle)$  to  $r^{\mathcal{I}_i}$ ;
- if  $r = s^-$ , add  $(\langle a, p\alpha \rangle, \langle a, p \rangle)$  to  $s^{\mathcal{I}_i}$ ;
- set  $\langle a, p\alpha \rangle^{\mathcal{I}_i} := O$ .

The resulting interpretation is  $\mathcal{I}_{i+1}$  (and  $\mathcal{I}_{i+1} = \mathcal{I}_i$  if there are no  $\langle a, p \rangle$  and  $\alpha$  to be selected). We assume that the selected leaf  $\langle a, p \rangle$  is such that the length of  $p$  is minimal, and thus all obligations are eventually satisfied.

Finally, the canonical model  $\mathcal{I}$  for  $\mathcal{A}$  and  $\mathcal{T}$  is defined by setting  $\Delta^{\mathcal{I}} := \bigcup_i \Delta^{\mathcal{I}_i}$ ,  $A^{\mathcal{I}} := \bigcup_i A^{\mathcal{I}_i}$ ,  $r^{\mathcal{I}} := \bigcup_i r^{\mathcal{I}_i}$ , and  $a^{\mathcal{I}} := a^{\mathcal{I}_0}$ . A proof of the following result can be found in the full version [13].

**Lemma 1.** *The canonical model  $\mathcal{I}$  for  $\mathcal{T}$  and  $\mathcal{A}$  is a model of  $\mathcal{T}$  and of  $\mathcal{A}$ .*

Our aim is to prove that we can verify whether  $\mathcal{A}$  and  $\mathcal{T}$  entail a conjunctive query  $q$  by checking whether the canonical model  $\mathcal{I}$  for  $\mathcal{A}$  and  $\mathcal{T}$  matches  $q$ . Key to this result is the observation that the canonical model of  $\mathcal{A}$  and  $\mathcal{T}$  can be homomorphically embedded into any model of  $\mathcal{A}$  and  $\mathcal{T}$ . We first define homomorphisms and then state the relevant lemma.

Let  $\mathcal{I}$  and  $\mathcal{J}$  be interpretations. A function  $h : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{J}}$  is a *homomorphism* from  $\mathcal{I}$  to  $\mathcal{J}$  if the following holds:

1. for all individual names  $a$ ,  $h(a^{\mathcal{I}}) = a^{\mathcal{J}}$ ;
2. for all concept names  $A$  and all  $d \in \Delta^{\mathcal{I}}$ ,  $d \in A^{\mathcal{I}}$  implies  $h(d) \in A^{\mathcal{J}}$ ;
3. for all  $d, e \in \Delta^{\mathcal{I}}$  with  $(d, e) \in r^{\mathcal{I}}$ ,  $r$  a (possibly inverse) role,  $(h(d), h(e)) \in r^{\mathcal{J}}$ .

**Lemma 2.** *Let  $\mathcal{I}$  be the canonical model for  $\mathcal{A}$  and  $\mathcal{T}$ , and  $\mathcal{J}$  a model of  $\mathcal{A}$  and  $\mathcal{T}$ . Then there is a homomorphism  $h$  from  $\mathcal{I}$  to  $\mathcal{J}$ .*

**Proof.** Let  $\mathcal{I}$  and  $\mathcal{J}$  be as in the lemma. For each interpretation  $\mathcal{I}_i$  in the sequence  $\mathcal{I}_0, \mathcal{I}_1, \dots$  used to construct  $\mathcal{I}$ , we define a homomorphism  $h_i$  from  $\mathcal{I}_i$  to  $\mathcal{J}$ . The limit of the sequence  $h_0, h_1, \dots$  is then the desired homomorphism  $h$  from  $\mathcal{I}$  to  $\mathcal{J}$ . To start, define  $h_0$  by setting  $h_0(\langle a, \varepsilon \rangle) := a^{\mathcal{J}}$  for all individual names  $a$ . Clearly,  $h_0$  is a homomorphism:

- Condition 1 is satisfied by construction.
- For Condition 2, let  $\langle a, \varepsilon \rangle \in A^{\mathcal{I}_0}$ . Then  $\mathcal{A}, \mathcal{T} \models A(a)$ . Since  $\mathcal{J}$  is a model of  $\mathcal{A}$  and  $\mathcal{T}$ ,  $h_0(\langle a, \varepsilon \rangle) = a^{\mathcal{J}} \in A^{\mathcal{J}}$ .
- For Condition 3, let  $(\langle a, \varepsilon \rangle, \langle b, \varepsilon \rangle) \in r^{\mathcal{I}_0}$ . Then  $r(a, b) \in \mathcal{A}$  and since  $\mathcal{J}$  is a model of  $\mathcal{A}$  and by definition of  $h_0$ , we have  $(h_0(\langle a, \varepsilon \rangle), h_0(\langle b, \varepsilon \rangle)) \in r^{\mathcal{J}}$ .

Now assume that  $h_i$  has already been defined. If  $\mathcal{I}_{i+1} = \mathcal{I}_i$ , then  $h_{i+1} = h_i$ . Otherwise, there is a unique  $\langle a, p \rangle \in \Delta^{\mathcal{I}_{i+1}} \setminus \Delta^{\mathcal{I}_i}$ . Let  $p = q\alpha$ . Then  $\langle a, q \rangle \in \Delta^{\mathcal{I}_i}$ , and there is an  $\alpha = \exists r.B$ -obligation  $O$  of  $\langle a, q \rangle^{\mathcal{I}_i}$  such that  $\langle a, p \rangle^{\mathcal{I}_{i+1}} = \text{sub}_{\mathcal{T}}(O)$ . Let  $A \in \langle a, q \rangle^{\mathcal{I}_i}$  such that  $A \sqsubseteq \exists r.B \in \mathcal{T}$ . By Condition 2 of homomorphisms, we have  $d = h_i(\langle a, q \rangle) \in A^{\mathcal{J}}$ . Since  $A \sqsubseteq \exists r.B \in \mathcal{T}$ , there is an  $e \in B^{\mathcal{J}}$  with  $(d, e) \in r^{\mathcal{J}}$ . Define  $h_{i+1}$  as the extension of  $h_i$  with  $h_{i+1}(\langle a, p \rangle) := e$ . We prove that the three conditions of homomorphisms are preserved:

- Condition 1 is untouched by the extension.
- For Condition 2, let  $\langle a, p \rangle \in A^{\mathcal{I}_{i+1}}$ . By definition of obligations, we have that  $\exists r^-. \prod_{B' \in \langle a, q \rangle^{\mathcal{I}_i}} B' \sqsubseteq_{\mathcal{T}} \text{sub}_{\mathcal{T}}(O)$ . Since  $h_i(\langle a, q \rangle) = d$  and by Condition 2 of homomorphisms,  $d \in (\prod_{B' \in \langle a, q \rangle^{\mathcal{I}_i}} B')^{\mathcal{J}}$ . Since  $(d, e) \in r^{\mathcal{J}}$ , we thus have  $e \in (\prod_{B' \in \text{sub}_{\mathcal{T}}(O)} B')^{\mathcal{J}}$  and it remains to remind that  $A' \in \langle a, p \rangle^{\mathcal{I}_{i+1}} = \text{sub}_{\mathcal{T}}(O)$ .
- Condition 3 was satisfied by  $\mathcal{I}_i$  and is preserved by the extension to  $\mathcal{I}_{i+1}$ .  $\square$

**Lemma 3.** *Let  $\mathcal{I}$  be the canonical model for  $\mathcal{A}$  and  $\mathcal{T}$ , and  $q$  a conjunctive query. Then  $\mathcal{A}, \mathcal{T} \models q$  iff  $\mathcal{I} \models q$ .*

**Proof.** Let  $\mathcal{I}$  and  $q$  be as in the lemma. If  $\mathcal{I} \not\models q$ , then  $\mathcal{A}, \mathcal{T} \not\models q$  since, by Lemma 1,  $\mathcal{I}$  is a model of  $\mathcal{A}$  and  $\mathcal{T}$ . Now assume  $\mathcal{I} \models^{\pi} q$ , and let  $\mathcal{J}$  be a model of  $\mathcal{A}$  and  $\mathcal{T}$ . By Lemma 2, there is a homomorphism  $h$  from  $\mathcal{I}$  to  $\mathcal{J}$ . Define  $\pi' : \text{Var}(q) \rightarrow \Delta^{\mathcal{J}}$  by setting  $\pi'(v) := h(\pi(v))$ . It is easily seen that  $\mathcal{J} \models^{\pi'} q$ .  $\square$

Thus, we can decide query entailment by looking only at the canonical model. At this point, we are faced with the problem that we cannot simply construct the canonical model  $\mathcal{I}$  and check whether  $\mathcal{I} \models q$  since  $\mathcal{I}$  is infinite. However, we can show that if  $\mathcal{I} \models q$ , then  $\mathcal{I} \models^{\pi} q$  for some match  $\pi$  that maps all variables to elements that can be reached by travelling only a bounded number of role edges from some ABox individual. Thus, it suffices to construct a sufficiently large “initial part” of  $\mathcal{I}$  and check whether it matches  $q$ .

To make this formal, let  $n$  be the size of  $\mathcal{A}$ ,  $m$  the size of  $\mathcal{T}$ , and  $k$  the size of  $q$ . In the following, we use  $|p|$  to denote the length of a path  $p$ . The *initial canonical model*  $\mathcal{I}'$  for  $\mathcal{A}$  and  $\mathcal{T}$  is obtained from the canonical model  $\mathcal{I}$  for  $\mathcal{A}$  and  $\mathcal{T}$  by setting

$$\begin{aligned} \Delta^{\mathcal{I}'} &:= \{\langle a, p \rangle \mid |p| \leq 2^m + k\} \\ A^{\mathcal{I}'} &:= A^{\mathcal{I}} \cap \Delta^{\mathcal{I}'} \\ r^{\mathcal{I}'} &:= r^{\mathcal{I}} \cap (\Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'}) \\ a^{\mathcal{I}'} &:= a^{\mathcal{I}} \end{aligned}$$

**Lemma 4.** *Let  $\mathcal{I}$  be the canonical model for  $\mathcal{A}$  and  $\mathcal{T}$ ,  $\mathcal{I}'$  the initial canonical model, and  $q$  a conjunctive query. Then  $\mathcal{I} \models q$  iff  $\mathcal{I}' \models q$ .*

**Proof.** Let  $\mathcal{I}, \mathcal{I}'$ , and  $q$  be as in the lemma. It is obvious that  $\mathcal{I}' \models q$  implies  $\mathcal{I} \models q$ . For the converse direction, let  $\mathcal{I} \models^{\pi} q$ . If  $\pi(v) = \langle a, p \rangle$  with  $|p| \leq 2^m + k$  for all  $v \in \text{Var}(q)$ , then  $\mathcal{I}' \models^{\pi} q$ . Otherwise, since queries are connected, there is a  $v \in \text{Var}(q)$  with  $\pi(v) = \langle a, p_0 \rangle$ ,  $|p_0| > 2^m$ , and such that for all  $u \in \text{Var}(q)$  with  $\pi(u) = \langle b, q \rangle$ , we have  $a = b$  and  $p_0$  is a prefix of  $q$ .

Since  $|p_0| > 2^m$ , we can split  $p_0$  into  $p_1p_2p_3$  such that  $\langle a, p_1 \rangle^{\mathcal{I}} = \langle a, p_1p_2 \rangle^{\mathcal{I}}$ , and  $p_2 \neq \varepsilon$ . Now, let  $\pi' : \text{Var}(q) \rightarrow \Delta^{\mathcal{I}}$  be obtained by setting  $\pi'(v) := \langle a, p_1p_3q \rangle$  if  $\pi(v) = \langle a, p_1p_2p_3q \rangle$ . We show the following: for all  $v \in \text{Var}(q)$ ,

1.  $\pi(v)^{\mathcal{I}} = \pi'(v)^{\mathcal{I}}$ ;
2.  $\mathcal{I} \models^{\pi'} q$ .

For Point 1, let  $\pi(v) = \langle a, p_1p_2p_3q \rangle$ . Then  $\pi(v') = \langle a, p_1p_3q \rangle$ . We prove by induction on the length of  $p$  that for all prefixes  $p$  of  $p_3q$ ,  $\langle a, p_1p \rangle^{\mathcal{I}} = \langle a, p_1p_2p \rangle^{\mathcal{I}}$ . For  $p = \varepsilon$ , this is true by choice of  $p_1$  and  $p_2$ . Now assume that the claim has already been shown for  $p$ , and let  $\alpha \in \text{ex}(\mathcal{T})$  such that  $p\alpha$  is a prefix of  $p_3q$ . Since  $\langle ap_1p \rangle^{\mathcal{I}} = \langle a, p_1p_2p \rangle^{\mathcal{I}}$ ,  $\langle ap_1p\alpha \rangle^{\mathcal{I}}$  is the  $\alpha$ -obligation of  $\langle ap_1p \rangle^{\mathcal{I}}$ , and  $\langle ap_1p_2p\alpha \rangle^{\mathcal{I}}$  is the  $\alpha$ -obligation of  $\langle ap_1p_2p \rangle^{\mathcal{I}}$ , it is readily checked that  $\langle ap_1p\alpha \rangle^{\mathcal{I}} = \langle a, p_1p_2p\alpha \rangle^{\mathcal{I}}$ . This finishes the proof of Point 1

For Point 2, let  $A(v) \in q$ . By Point 1,  $\mathcal{I} \models^{\pi} A(v)$  implies  $\mathcal{I} \models^{\pi'} A(v)$ . Now let  $r(u, v) \in q$ . Then  $(\pi(u), \pi(v)) \in r^{\mathcal{I}}$ . By construction of  $\mathcal{I}$ , this implies that one of the following holds:

1.  $\pi(u) = \langle a, p_1p_2p_3q \rangle$  and  $\pi(v) = \langle a, p_1p_2p_3q\alpha \rangle$  for some  $\alpha = \exists r.B \in \text{ex}(\mathcal{T})$ ;
2.  $\pi(u) = \langle a, p_1p_2p_3q\alpha \rangle$  and  $\pi(v) = \langle a, p_1p_2p_3q \rangle$  for some  $\alpha = \exists r^-.B \in \text{ex}(\mathcal{T})$ .

In Case 1, we have  $\pi'(u) = \langle a, p_1p_3q \rangle$  and  $\pi(v) = \langle a, p_1p_3q\alpha \rangle$ . Again by construction of  $\mathcal{I}$ , this means  $(\pi'(u), \pi'(v)) \in r^{\mathcal{I}}$ . Case 2 is analogous.

When applying this construction exhaustively, we eventually obtain a  $\pi^*$  such that  $\pi^*(v) = \langle a, p \rangle$  with  $|p| \leq 2^m + k$  for all  $v \in \text{Var}(q)$   $\square$

The initial canonical model  $\mathcal{I}'$  for  $\mathcal{A}$  and  $\mathcal{T}$  can be constructed in time polynomial in the size of  $\mathcal{A}$ . In particular, (i)  $\mathcal{I}'_0$  can be constructed in polytime since, due to the results of [11, 12], instance checking in  $\mathcal{ELI}^f$  is tractable regarding data complexity; (ii) obligations can be computed in polytime since subsumption in  $\mathcal{ELI}^f$  w.r.t. general TBoxes is decidable and the required checks are independent of the size of  $\mathcal{A}$ ; (iii) the number of elements in the initial canonical model is bounded by  $\ell := n \cdot m^{2^m+k}$  and is thus independent of the size of  $\mathcal{A}$ .

Our algorithm for deciding entailment of a conjunctive query  $q$  by a TBox  $\mathcal{T}$  and ABox  $\mathcal{A}$  in normal form is as follows. If the UNA is made, we first check consistency of  $\mathcal{A}$  w.r.t.  $\mathcal{T}$  using one of the polytime algorithms from [11, 12]. If  $\mathcal{A}$  is inconsistent w.r.t.  $\mathcal{T}$ , we answer “yes”. If the UNA is not made, then we convert  $\mathcal{A}$  into an ABox  $\mathcal{A}'$  that is admissible w.r.t.  $\mathcal{T}$ , and continue working with  $\mathcal{A}'$ . Obviously, the conversion can be done in time polynomial in the size of  $\mathcal{A}$  simply by identifying ABox individuals. Both with and without UNA, at this point we have an ABox that is admissible w.r.t.  $\mathcal{T}$ . The next step is to construct the initial canonical structure  $\mathcal{I}'$  for  $\mathcal{T}$  and  $\mathcal{A}$ , and then check matches of  $q$  against this structure. The latter can be done in time polynomial in the size of  $\mathcal{A}$ : there are at most  $\ell^k$  (and thus polynomially many) mappings  $\tau : \text{Var}(q) \rightarrow \Delta^{\mathcal{I}'}$ , and each of them can be checked for being a match in polynomial time.

**Theorem 4.** *In  $\mathcal{ELI}^f$ , conjunctive query w.r.t. general TBoxes is in P regarding data complexity.*

A matching lower bound can be taken from [7] (which relies on the presence of general TBoxes and already applies to the instance problem), and thus we obtain P-completeness.

Extensions of $\mathcal{EL}$	w.r.t. acyclic TBoxes	w.r.t. general TBoxes
$\mathcal{EL}^{\neg A}$	coNP-complete [17]	coNP-complete [17]
$\mathcal{EL}^{C \sqcup D}$	coNP-complete	coNP-complete
$\mathcal{EL}^{\forall r, \perp}, \mathcal{EL}^{\forall r, C}$	coNP-complete	coNP-complete
$\mathcal{EL}^{(\leq kr)}, r \geq 0$	coNP-complete	coNP-complete
$\mathcal{EL}^{kf}, k \geq 2$ w/o UNA	coNP-complete (even w/o TBox)	coNP-complete
$\mathcal{EL}^{kf}, k \geq 2$ with UNA	coNP-complete (in P w/o TBox)	coNP-complete
$\mathcal{EL}^{(\geq kr)}, k \geq 2$ w/o UNA	coNP-complete	coNP-complete
$\mathcal{EL}^{(\geq kr)}, k \geq 2$ with UNA	in coNP	coNP-complete
$\mathcal{EL}^{\exists \neg r, C}$	coNP-hard	coNP-hard
$\mathcal{EL}^{\exists r \cup s, C}$	coNP-hard	coNP-hard
$\mathcal{EL}^{\exists r^+, C}$	coNP-hard	coNP-hard
$\mathcal{ELI}^f$	in P	P-complete

**Table 1.** Complexity of instance checking and conjunctive query entailment

## 5 Summary

The results of our investigation are summarized in Table 1, and in all cases they apply both to instance checking and conjunctive query entailment. The coNP upper bounds are a consequence of the results in [9]. When the UNA is not explicitly mentioned, the results hold both with and without UNA. We point out two interesting issues. First, for all of the considered extensions we were able to show tractability regarding data complexity if and only if the logic is *convex regarding instances*, i.e.,  $\mathcal{A}, \mathcal{T} \models C(a)$  with  $C = D_0 \sqcup \dots \sqcup D_{n-1}$  implies  $\mathcal{A}, \mathcal{T} \models D_i(a)$  for some  $i < n$ . It would be interesting to capture this phenomenon in a general result. And second, it is interesting to point out that subtle differences such as the UNA or local versus global functionality (for the latter, see  $\mathcal{EL}^{(\leq 1r)}$  vs.  $\mathcal{ELI}^f$ ) can have an impact on tractability.

As future work, it would be interesting extend our upper bound by including more operators from the tractable description logic  $\mathcal{EL}^{++}$  as proposed in [1]. For a start, it is not hard to show that conjunctive query entailment in full  $\mathcal{EL}^{++}$  is undecidable due to the presence of role inclusions  $r_1 \circ r_2 \sqsubseteq s$ . In the following, we briefly sketch the proof, which is by reduction of the problem of deciding whether the intersection of two languages defined by given context-free grammars  $G_i = (N_i, T, P_i, S_i)$ ,  $i \in \{1, 2\}$ , is empty. We assume w.l.o.g. that the set of non-terminals  $N_1$  and  $N_2$  are disjoint. Then define a TBox

$$\mathcal{T} := \{ \top \sqsubseteq \exists r_a. \top \mid a \in T \} \cup \{ r_{A_1} \circ \dots \circ r_{A_n} \sqsubseteq r_A \mid A \rightarrow A_1 \dots A_n \in P_1 \cup P_2 \}.$$

It is not too difficult to see that  $L(G_1) \cap L(G_2) \neq \emptyset$  iff the conjunctive query  $S_1(u, v) \wedge S_2(u, v)$  is matched by the ABox  $\{ \top(a) \}$  and TBox  $\mathcal{T}$ .

**Acknowledgement** We are grateful to Markus Krötzsch, Meng Suntsirivaraporn, and the anonymous reviewers for valuable comments on earlier versions of this paper.

## References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the  $\mathcal{EL}$  envelope. In *Proc. of the 19th Int. Joint Conf. on AI (IJCAI-05)*, pages 364–369. Morgan Kaufmann, 2005.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the  $\mathcal{EL}$  envelope. Submitted to a Journal. 2007
3. F. Baader, C. Lutz, and B. Suntsirivaraporn. Is tractable reasoning in extensions of the description logic  $\mathcal{EL}$  useful in practice? In *Proc. of the 4th Int. WS on Methods for Modalities (M4M'05)*, 2005.
4. F. Baader, D. L. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.
5. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. of the 16th European Conf. on AI (ECAI-2004)*, pages 298–302. IOS Press, 2004.
6. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-lite: Tractable description logics for ontologies. In *Proc. of the 20th National Conf. on AI (AAAI'05)*, pages 602–607. AAAI Press, 2005.
7. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on KR (KR'06)*. AAAI Press, 2006.
8. D. Calvanese, G. D. Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-lite: Practical reasoning for rich dls. In *Proc. of the 2004 Int. WS on DLs (DL2004)*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
9. B. Glimm and I. Horrocks and C. Lutz and U. Sattler. Conjunctive Query Answering for the Description Logic  $\mathcal{SHIQ}$ . In *Proc. of the 20th Int. Joint Conf. on AI (IJCAI-07)*. AAAI Press, 2007.
10. G. D. Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th National Conf. on AI (AAAI'94). Volume 1*, pages 205–212. AAAI Press, 1994.
11. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on AI (IJCAI'05)*, pages 466–471. Professional Book Center, 2005.
12. A. Krisnadhi. Data complexity of instance checking in the  $\mathcal{EL}$  family of description logics. Master thesis, TU Dresden, Germany, 2007.
13. A. Krisnadhi and C. Lutz. Data complexity of instance checking in the  $\mathcal{EL}$  family of description logics. Available from <http://lat.inf.tu-dresden.de/~clu/papers/>
14. M. Krötzsch, S. Rudolf, and P. Hitzler. On the complexity of horn description logics. In *Proc. of the 2nd WS on OWL: Experiences and Directions*, number 216 in CEUR-WS (<http://ceur-ws.org/>), 2006.
15. M. Krötzsch and S. Rudolf. Conjunctive Queries for  $\mathcal{EL}$  with Composition of Roles. In *Proc. of the 2007 Int. WS on DLs (DL2007)*. CEUR-WS.org, 2007.
16. R. Rosati. On conjunctive query answering in  $\mathcal{EL}$ . In *Proc. of the 2007 Int. WS on DLs (DL2007)*. CEUR-WS.org, 2007.
17. A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

## Inverse Roles Make Conjunctive Queries Hard

Carsten Lutz

Institute for Theoretical Computer Science, TU Dresden, Germany  
lutz@tcs.inf.tu-dresden.de

**Abstract.** Conjunctive query answering is an important DL reasoning task. Although this task is by now quite well-understood, tight complexity bounds for conjunctive query answering in expressive DLs have never been obtained: all known algorithms run in deterministic double exponential time, but the existing lower bound is only an EXPTIME one. In this paper, we prove that conjunctive query answering in  $\mathcal{ALCT}$  is 2-EXPTIME-hard (and thus complete), and that it becomes NEXPTIME-complete under some reasonable assumptions.

### 1 Introduction

When description logic (DL) knowledge bases are used in applications with a large amount of instance data, ABox querying is the most important reasoning problem. The most basic query mechanism for ABoxes is *instance retrieval*, i.e., returning all the individuals from an ABox that are known to be instances of a given query concept. Instance retrieval can be viewed as a well-behaved generalization of subsumption and satisfiability, which are the standard reasoning problems on TBoxes. In particular, algorithms for the latter can typically be adapted to instance retrieval in a straightforward way, and the computational complexity coincides in almost all cases (see [13] for an exception). In 1998, Calvanese et al. introduced *conjunctive query answering* as a more powerful query mechanism for DL ABoxes. Since then, conjunctive queries have received considerable interest in the DL community, see for example the papers [2, 3, 5–8, 12]. In a nutshell, conjunctive query answering generalizes instance retrieval by admitting also queries whose relational structure is not tree-shaped. This generalization is both natural and useful because the relational structure of ABoxes is usually not tree-shaped as well.

In contrast to the case of instance retrieval, developing algorithms for conjunctive query answering is not merely a matter of extending algorithms for satisfiability, but requires developing new techniques. In particular, all hitherto known algorithms for DLs that include  $\mathcal{ALC}$  as a fragment run in deterministic double exponential runtime, in contrast to algorithms for deciding subsumption and satisfiability which require only exponential time even for DLs much more expressive than  $\mathcal{ALC}$ . Since the introduction of conjunctive query answering as a reasoning problem for DLs, it has remained an open question whether or not this increase in runtime can be avoided. In other words, it has not been clear whether generalizing instance retrieval to the more powerful conjunctive query answering is penalized by higher computational complexity. In this paper, we answer this question by showing that conjunctive query answering is computationally more expensive than instance retrieval when inverse roles are present. More precisely,

we prove the following two results about  $\mathcal{ALCI}$ , the extension of  $\mathcal{ALC}$  with inverse roles:

(1) Rooted conjunctive query answering in  $\mathcal{ALCI}$  is co-NEXPTIME-complete, where *rooted* means that conjunctive queries are required to be connected and contain at least one answer variable. The phrase “rooted” derives from the fact that every match of such a query is rooted in at least one ABox individual. The lower bound even holds for ABoxes of the form  $\{C(a)\}$  and w.r.t. empty TBoxes.

(2) Conjunctive query answering in  $\mathcal{ALCI}$  is 2-EXPTIME-complete. The lower bound even holds for ABoxes of the form  $\{C(a)\}$  and when queries do not contain any answer variables (or when they contain answer variables, but are not connected).

In the conference version of this paper, we will complement these results by showing that the high computational complexity of conjunctive query answering is indeed due to inverse roles. We will show that conjunctive query answering in  $\mathcal{ALC}$  and  $\mathcal{SHQ}$ , the fragment of  $\mathcal{SHIQ}$  without inverse roles, is only EXPTIME-complete. In this abstract, we concentrate on the lower bounds due to space limitations.

## 2 Preliminaries

We assume standard notation for the syntax and semantics of  $\mathcal{ALCI}$  knowledge bases [1]. In particular, a *TBox* is a set of concept inclusions  $C \sqsubseteq D$  and a *knowledge base (KB)* is a pair  $(\mathcal{T}, \mathcal{A})$  consisting of a TBox  $\mathcal{T}$  and an ABox  $\mathcal{A}$ . Let  $N_V$  be a countably infinite set of *variables*. An *atom* is an expression  $C(v)$  or  $r(v, v')$ , where  $C$  is an  $\mathcal{ALCI}$  concept,  $r$  is a (possibly inverse) role, and  $v, v' \in N_V$ . A *conjunctive query*  $q$  is a finite set of atoms. We use  $\text{Var}(q)$  to denote the set of variables occurring in the query  $q$ . Let  $\mathcal{A}$  be an ABox,  $\mathcal{I}$  a model of  $\mathcal{A}$ ,  $q$  a conjunctive query, and  $\pi : \text{Var}(q) \rightarrow \Delta^{\mathcal{I}}$  a total function. We write  $\mathcal{I} \models^{\pi} C(v)$  if  $(\pi(v)) \in C^{\mathcal{I}}$  and  $\mathcal{I} \models^{\pi} r(v, v')$  if  $(\pi(v), \pi(v')) \in r^{\mathcal{I}}$ . If  $\mathcal{I} \models^{\pi} at$  for all  $at \in q$ , we write  $\mathcal{I} \models^{\pi} q$  and call  $\pi$  a *match* for  $\mathcal{I}$  and  $q$ . We say that  $\mathcal{I}$  *satisfies*  $q$  and write  $\mathcal{I} \models q$  if there is a match  $\pi$  for  $\mathcal{I}$  and  $q$ . If  $\mathcal{I} \models q$  for all models  $\mathcal{I}$  of a KB  $\mathcal{K}$ , we write  $\mathcal{K} \models q$  and say that  $\mathcal{K}$  *entails*  $q$ . The *query entailment problem* is, given a knowledge base  $\mathcal{K}$  and a query  $q$ , to decide whether  $\mathcal{K} \models q$ . This is the decision problem corresponding to query answering (which is a search problem), see e.g. [6] for details.

## 3 Rooted Query Entailment in $\mathcal{ALCI}$ is co-NEXPTIME-complete

Let  $\mathcal{ALC}^{\text{rs}}$  be the variation of  $\mathcal{ALC}$  in which all roles are interpreted as reflexive and symmetric relations. Our proof of the lower bound stated as (1) above proceeds by first polynomially reducing rooted query entailment in  $\mathcal{ALC}^{\text{rs}}$  w.r.t. the empty TBox to rooted query entailment in  $\mathcal{ALCI}$  w.r.t. the empty TBox. Then, we prove co-NEXPTIME-hardness of rooted query entailment in  $\mathcal{ALC}^{\text{rs}}$ .

Regarding the first step, we only sketch the basic idea, which is simply to replace each symmetric role  $r$  with the composition of  $r^-$  and  $r$ . Although  $r$  is not interpreted in a symmetric relation in  $\mathcal{ALCI}$ , the composition of  $r^-$  and  $r$  is clearly symmetric. To achieve reflexivity, we ensure that  $\exists r^- . \top$  is satisfied by all relevant individuals and

for all relevant roles  $r$ . Thus, every individual can reach itself by first travelling  $r^-$  and then  $r$ , which corresponds to a reflexive loop. Since we are working without TBoxes and thus cannot use statements such as  $\top \sqsubseteq \exists r^-. \top$ , a careful manipulation of the ABox and query is needed. Details are given in appendix A.

Before we prove co-NEXPTIME-hardness of rooted query entailment in  $\mathcal{ALC}^{\text{rs}}$ , we discuss a preliminary. An interpretation  $\mathcal{I}$  of  $\mathcal{ALC}^{\text{rs}}$  is *tree-shaped* if there is a bijection  $f$  from  $\Delta^{\mathcal{I}}$  into the set of nodes of a finite undirected tree  $(V, E)$  such that  $(d, e) \in s^{\mathcal{I}}$ , for some role name  $s$ , implies that  $d = e$  or  $\{f(d), f(e)\} \in E$ . The proof of the following result is standard, using unravelling of non-tree-shaped models.

**Lemma 1.** *If  $\mathcal{A}$  is an  $\mathcal{ALC}^{\text{rs}}$ -ABox and  $q$  a conjunctive query, then  $\mathcal{A} \not\models q$  implies that there is a tree-shaped model  $\mathcal{I}$  of  $\mathcal{A}$  such that  $\mathcal{I} \not\models q$ .*

Because  $\mathcal{A} \models q$  clearly implies that  $\mathcal{I} \models q$  for all tree-shaped models  $\mathcal{I}$  of  $\mathcal{A}$ , this lemma means that we can concentrate on tree-shaped interpretations when deciding conjunctive query entailment. We will exploit this fact to give an easier explanation of the reduction that is to follow.

We now give a reduction from a NEXPTIME-complete variant of the tiling problem to the complement of rooted query entailment in  $\mathcal{ALC}^{\text{rs}}$ .

**Definition 1 (Domino System).** *A domino system  $\mathcal{D}$  is a triple  $(T, H, V)$ , where  $T = \{0, 1, \dots, k-1\}$ ,  $k \geq 0$ , is a finite set of tile types and  $H, V \subseteq T \times T$  represent the horizontal and vertical matching conditions. Let  $\mathcal{D}$  be a domino system and  $c = c_0, \dots, c_{n-1}$  an initial condition, i.e. an  $n$ -tuple of tile types. A mapping  $\tau : \{0, \dots, 2^{n+1}-1\} \times \{0, \dots, 2^{n+1}-1\} \rightarrow T$  is a solution for  $\mathcal{D}$  and  $c$  iff for all  $x, y < 2^{n+1}$ , the following holds (where  $\oplus_i$  denotes addition modulo  $i$ ):*

- if  $\tau(x, y) = t$  and  $\tau(x \oplus_{2^{n+1}} 1, y) = t'$ , then  $(t, t') \in H$
- if  $\tau(x, y) = t$  and  $\tau(x, y \oplus_{2^{n+1}} 1) = t'$ , then  $(t, t') \in V$
- $\tau(i, 0) = c_i$  for  $i < n$ .

For a proof of NEXPTIME-hardness of this version of the domino problem, see e.g. Corollary 4.15 in [9].

We show how to translate a given domino system  $\mathcal{D}$  and initial condition  $c = c_0 \dots c_{n-1}$  into an ABox  $\mathcal{A}_{\mathcal{D},c}$  and query  $q_{\mathcal{D},c}$  such that each (tree-shaped) model  $\mathcal{I}$  of  $\mathcal{A}_{\mathcal{D},c}$  that satisfies  $\mathcal{I} \not\models q_{\mathcal{D},c}$  encodes a solution to  $\mathcal{D}$  and  $c$ , and conversely each solution to  $\mathcal{D}$  and  $c$  gives rise to a (tree-shaped) model of  $\mathcal{A}_{\mathcal{D},c}$  with  $\mathcal{I} \not\models q_{\mathcal{D},c}$ . The ABox  $\mathcal{A}_{\mathcal{D},c}$  contains only the assertion  $C_{\mathcal{D},c}(a)$ , with  $C_{\mathcal{D},c}$  a conjunction  $C_{\mathcal{D},c}^1 \sqcap \dots \sqcap C_{\mathcal{D},c}^7$  whose conjuncts we define in the following. For convenience, let  $m = 2n+2$ . The purpose of the first conjunct  $C_{\mathcal{D},c}^1$  is to enforce a binary tree of depth  $m$  whose leaves are labelled with the numbers  $0, \dots, 2^m-1$  of a binary counter implemented by the concept names  $A_0, \dots, A_{m-1}$ . We use concept names  $L_0, \dots, L_m$  to distinguish the different levels of the tree. This is necessary because we work with reflexive and symmetric roles. In the following  $\forall s^i.C$  denotes the  $i$ -fold nesting  $\forall s. \dots \forall s.C$ . In particular,  $\forall s^0.C$  is  $C$ .

$$C_{\mathcal{D},c}^1 := L_0 \sqcap \prod_{i < m} \forall s^i. (L_i \rightarrow (\exists s.(L_{i+1} \sqcap A_i) \sqcap \exists s.(L_{i+1} \sqcap \neg A_i))) \sqcap$$

$$\prod_{i < m} \forall s^i. \prod_{j < i} ((L_i \sqcap A_j) \rightarrow \forall s.(L_{i+1} \rightarrow A_j) \sqcap$$

$$(L_i \sqcap \neg A_j) \rightarrow \forall s.(L_{i+1} \rightarrow \neg A_j))$$

From now on, leafs in this tree are called  $L_m$ -nodes. Intuitively, each  $L_m$ -node corresponds to a position in the  $2^{n+1} \times 2^{n+1}$ -grid that we have to tile: the counter  $A_x$  realized by the concept names  $A_0, \dots, A_n$  binarily encodes the horizontal position, and the counter  $A_y$  realized by  $A_{n+1}, \dots, A_m$  encodes the vertical position. We now extend the tree with some additional nodes. Every  $L_m$ -node gets three successor nodes labelled with  $F$ , and each of these  $F$ -nodes has a successor node labelled  $G$ . To distinguish the three different  $G$ -nodes below each  $L_m$ -node, we additionally label them with the concept names  $G_1, G_2, G_3$ .

$$C_{\mathcal{D},c}^2 := \forall s^m. (L_m \rightarrow (\bigwedge_{1 \leq i \leq 3} \exists s. (F \sqcap \exists s. (G \sqcap G_i))))$$

We want that each  $G_1$ -node represents the grid position identified by its ancestor  $L_m$ -node, the sibling  $G_2$  node represents the horizontal neighbor position in the grid, and the sibling  $G_3$ -node represents the vertical neighbor.

$$C_{\mathcal{D},c}^3 := \forall s^m. (L_m \rightarrow (\bigwedge_{i \leq n} ((A_i \rightarrow \forall s^2. (G_1 \sqcup G_3 \rightarrow A_i)) \sqcap (\neg A_i \rightarrow \forall s^2. (G_1 \sqcup G_3 \rightarrow \neg A_i))) \sqcap \bigwedge_{n < i < m} ((A_i \rightarrow \forall s^2. (G_1 \sqcup G_2 \rightarrow A_i)) \sqcap (\neg A_i \rightarrow \forall s^2. (G_1 \sqcup G_2 \rightarrow \neg A_i))) \sqcap E_2 \sqcap E_3))$$

where  $E_2$  is an  $\mathcal{ALC}$ -concept ensuring that the  $A_x$  value at each  $G_2$ -node is obtained from the  $A_x$ -value of its  $G$ -node ancestor by incrementing modulo  $2^{n+1}$ ; similarly,  $E_3$  expresses that the  $A_y$  value at each  $G_3$ -node is obtained from the  $A_y$ -value of its  $G$ -node ancestor by incrementing modulo  $2^{n+1}$ . It is not hard to work out the details of these concepts, see e.g. [11] for more details. The *grid representation* that we have enforced is shown in Figure 1. To represent tiles, we introduce a concept name  $D_i$  for each  $i \in T$  and put

$$C_{\mathcal{D},c}^4 := \forall s^{m+2}. (G \rightarrow (\bigwedge_{i \in T} D_i \sqcap \bigwedge_{i,j \in T, i \neq j} \neg(D_i \sqcap D_j)))$$

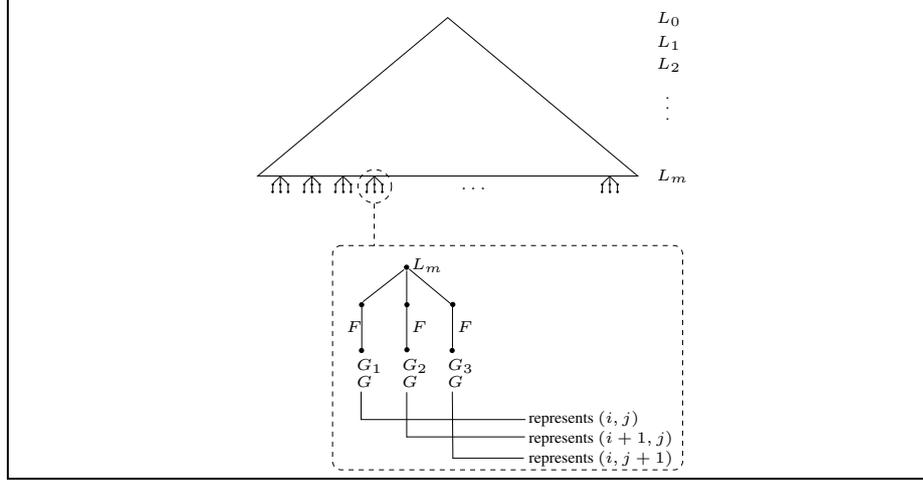
The initial condition is easily guaranteed by

$$C_{\mathcal{D},c}^5 := \bigwedge_{i < n} \forall s^{m+2}. ((\bigwedge_{j \leq n, \text{bit}_j(i)=0} \neg A_j \sqcap \bigwedge_{j \leq n, \text{bit}_j(i)=1} A_j \sqcap \bigwedge_{n < j < m} \neg A_j) \rightarrow T_{c_i}),$$

where  $\text{bit}_j(i)$  denotes the value of the  $j$ -th bit in the binary representation of  $i$ . To enforce the matching conditions, we proceed in two steps. First we ensure that they are satisfied locally, i.e., among the three  $G$ -nodes below each  $L_m$ -node:

$$C_{\mathcal{D},c}^6 := \forall s^{m+2}. (L_m \rightarrow (\bigwedge_{i \in T} (\exists s^2. (G_1 \sqcap D_i) \rightarrow \forall s^2. (G_2 \rightarrow \bigwedge_{(i,j) \in H} D_j)) \sqcap \bigwedge_{i \in T} (\exists s^2. (G_1 \sqcap D_i) \rightarrow \forall s^2. (G_3 \rightarrow \bigwedge_{(i,j) \in V} D_j))))$$

Second, we enforce the following condition, which together with local satisfaction of the matching conditions ensures their global satisfaction:



**Fig. 1.** The structure encoding the  $2^{n+1} \times 2^{n+1}$ -grid.

(\*) if the  $A_x$  and  $A_y$ -values of two  $G$ -nodes coincide, then their tile types coincide.

In (\*), a  $G$ -node can be any of a  $G_1$ -,  $G_2$ -, or  $G_3$ -node. To enforce (\*), we use the query. Before we give details, let us finish the definition of the concept  $C_{\mathcal{D},c}^7$ . The last conjunct  $C_{\mathcal{D},c}^7$  enforces two technical conditions that will be explained later: if  $d$  is an  $F$ -node and  $e$  its  $G$ -node successor, then

(T1)  $d$  and  $e$  are labelled dually regarding  $A_i, \neg A_i$  for all  $i < m$ , i.e.,  $d$  satisfies  $A_i$  iff  $e$  satisfies  $\neg A_i$ ;

(T2)  $d$  and  $e$  are labelled dually regarding  $D_0, \dots, D_{k-1}$ , i.e., for all  $j < k$ , if  $d$  satisfies  $D_j$ , then  $e$  satisfies  $D_0, \dots, D_{j-1}, \neg D_j, D_{j+1}, \dots, D_{k-1}$ .

We use the following concept:

$$C_{\mathcal{D},c}^7 := \forall s^{m+1}. (F \rightarrow (\bigwedge_{i < m} (A_i \rightarrow \forall s. (G \rightarrow \neg A_i)) \sqcap (\neg A_i \rightarrow \forall s. (G \rightarrow A_i)) \sqcap \bigwedge_{i \in T} \exists s. (G \sqcap D_i) \rightarrow (\neg D_i \sqcap \bigwedge_{j < k, j \neq i} D_j)))$$

We now construct the query  $q_{\mathcal{D},c}$  that does *not* match the grid representation iff (\*) is satisfied. In other words,  $q_{\mathcal{D},c}$  matches the grid representation if there are two  $G$ -nodes that agree on the value of the counters  $A_x$  and  $A_y$ , but are labelled with different tile types. Because of Lemma 1, we can concentrate on the grid representation as shown in Figure 1 while constructing  $q_{\mathcal{D},c}$ , and need not worry about models in which domain elements that are different in Figure 1 are identified.

The construction of  $q_{\mathcal{D},c}$  is in several steps, starting with the query  $q_{\mathcal{D},c}^i$  on the left-hand side of Figure 2, where  $i \in \{0, \dots, m - 1\}$ . In the queries  $q_{\mathcal{D},c}^i$ , all the edges



Together, this gives four options for removing the two mentioned length-three cycles. However, two of these options are ruled out because the resulting collapsings have no match in the grid representation. The first such case is when we identify  $v$  with  $v_0$  and  $v'$  with  $v_{2m+3}$ . Then  $v_0$  and  $v_{2m+3}$  have to satisfy  $G$ . To continue our argument, we make a case distinction on the two options that we have for eliminating the cycle  $\{v_{\text{ans}}, v_{m+1}, v_{m+2}\}$ .

Case (1). If we identify  $v_{\text{ans}}$  and  $v_{m+1}$ , the path from the  $G$ -variable  $v_0$  to  $v_{\text{ans}}$  is only of length  $m + 1$ . In our grid representation, all paths from a  $G$ -node to an ABox individual (i.e., the root) are of length  $m + 2$ , so there can be no match of this collapsing.

Case (2). If we identify  $v_{\text{ans}}$  and  $v_{m+2}$ , the path from  $v_{\text{ans}}$  to the  $G$ -variable  $v_{2m+3}$  is only of length  $m + 1$  and again there is no match.

We can argue analogously for the case where we identify  $v$  with  $v'_0$  and  $v'$  with  $v'_{2m+3}$ . Therefore, the two remaining collapsings for eliminating the cycles  $\{v, v_0, v'_0\}$  and  $\{v', v_{2m+3}, v'_{2m+3}\}$  are the following:

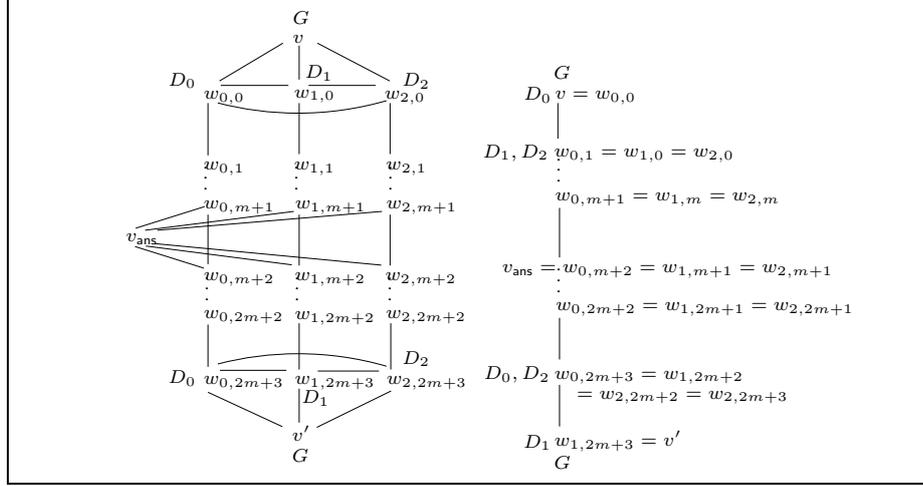
- (a) identify  $v$  with  $v_0$  and  $v'$  with  $v'_{2m+3}$ ;
- (b) identify  $v$  with  $v'_0$  and  $v'$  with  $v_{2m+3}$ .

In the first case, we further have to identify  $v_{\text{ans}}$  with  $v_{m+2}$  and  $v'_{m+1}$ , for otherwise we can argue as above that there is no match. In the second case, we have to identify  $v_{\text{ans}}$  with  $v_{m+1}$  and  $v'_{m+2}$ . After this has been done, there is only one way to eliminate the cycle  $v = v_0, \dots, v_{2m+3}, v' = v'_{2m+3}, \dots, v'_0$  such that the result is a chain of length  $2m + 4$  with the  $G$ -variables at both ends and the answer variable exactly in the middle (any other way to collapse means that there are no matches). The reflexive loops at the endpoints of the resulting chain and at  $v_{\text{ans}}$  can simply be dropped since we work with reflexive roles. The resulting cycle-free queries are shown in the middle and right part of Figure 2.

Note that the middle query has  $A_i$  at both ends of the chain, and the right one has  $\neg A_i$  at the ends. According to our above argumentation, the original query  $q_{\mathcal{D},c}^i$  has a match in the grid representation iff one of these two collapsings has a match. Thus, every match  $\pi$  of  $q_{\mathcal{D},c}^i$  in the grid representation is such that  $\pi(v)$  and  $\pi(v')$  are (not necessarily distinct) instances of  $G$  that agree on the value of  $A_i$ . Informally, we say that  $q_{\mathcal{D},c}^i$  connects  $G$ -nodes that have the same  $A_i$ -value.

At this point, a technical remark is in order. Observe that the two relevant collapsings of  $q_{\mathcal{D},c}^i$  are such that the nodes next to the outer nodes are labelled dually w.r.t.  $A_i$  compared to the outer nodes. This is an artifact of query construction and cannot be avoided. It is the reason for introducing the  $F$ -nodes into our grid representation, and for ensuring that they satisfy Property (T1) from above.

Now set  $q_{\text{cnt}} := \bigcup_{i < m} q_{\mathcal{D},c}^i$ . It is easy to see that  $q_{\text{cnt}}$  connects  $G$ -nodes that have the same  $A_i$ -value, for all  $i < m$ . The query  $q_{\text{cnt}}$  is almost the desired query  $q_{\mathcal{D},c}$ . Recall that we want to enforce Condition (\*) from above, and thus need to talk about tile types in the query. The query  $q_{\text{tile}}$  is given in the left-hand side of Figure 3 for the



**Fig. 3.** The query  $q_{\text{tile}}$  (left) and one of its collapsings (right).

case of three tiles, i.e.,  $T = \{0, 1, 2\}$ . In general, for  $T = \{1, \dots, k-1\}$ , we define

$$\begin{aligned}
 q_{\text{tile}} := & \bigcup_{i < k} \{s(w_{i,0}, w_{i,1}), \dots, s(w_{i,2m+2}, w_{i,2m+3}), \\
 & s(w_{\text{ans}}, w_{i,m+1}), s(w_{\text{ans}}, w_{i,m+2}), \\
 & s(v, w_{i,0}), s(v', w_{i,2m+3}), \\
 & D_i(w_{i,0}), D_i(w_{i,2m+3})\} \\
 \cup & \bigcup_{i < j < k} \{s(w_{i,0}, w_{j,0}), s(w_{i,2m+3}, w_{j,2m+3})\} \\
 \cup & \{G(v), G(v')\}
 \end{aligned}$$

Observe that  $q_{\text{cnt}}$  and  $q_{\text{tile}}$  share the variables  $v, v'$ , and  $v_{\text{ans}}$ . Also observe that  $q_{\text{tile}}$  is very similar to the queries  $q_{\mathcal{D},c}^i$ , the main difference being the number of vertical chains. Whereas the queries  $q_{\mathcal{D},c}^i$  have two collapsings that are cycle-free and can have matches in the grid representation,  $q_{\text{tile}}$  has  $k \cdot (k-1)$  such collapsings: for all  $i, j \in T$  with  $i \neq j$ , there is a collapsing into a linear chain of length  $2m+4$  whose end nodes are labelled  $D_i$  and  $D_j$ . An example of such a collapsing is presented on the right-hand side of Figure 3. The arguments for how to obtain these collapsings and why other collapsings have no matches in the grid representation are very similar to the line of argumentation used for  $q_{\mathcal{D},c}^i$ . We only give a brief walkthrough. First, the cycle  $v, w_{0,0}, \dots, w_{k-1,0}$  can be eliminated by identifying  $v$  with one of the  $w_{i,0}$ . Note that we cannot eliminate the cycle by identifying all of  $w_{0,0}, \dots, w_{k-1,0}$ , because then there would be no match in the grid representation. Similarly, the cycle  $v', w_{0,2m+3}, \dots, w_{k-1,2m+3}$  can be eliminated by identifying  $v'$  with one of the  $w_{i,2m+3}$ . We can show that  $i \neq j$  by analyzing the two cases of  $v_{\text{ans}}$  being identified with  $w_{i,m+1}$  or  $w_{i,m+2}$ . In the first case, there is no match in the grid representation because the path from  $v$  to  $w_{i,m+1}$  is too short, and in the second case the same holds for the path from  $w_{i,m+2}$  to  $v'$ . Thus,  $i \neq j$  is shown. Also

because of paths lengths, we have to identify  $v_{\text{ans}}$  with  $v_{i,m+1}$  and  $v_{j,m+2}$ . Next, we consider the cycle  $v = w_{i,0}, \dots, w_{i,2m+3}, v' = w_{j,2m+3}, \dots, w_{j,0}$ . As in the case of  $q_w^i$ , there is only one way to eliminate this cycle such that the result is a chain of length  $2m + 4$  with the  $G$ -variables at both ends and the answer variable exactly in the middle, and any other way to collapse means that there are no matches. It remains to eliminate the cycles  $v = w_{i,0}, \dots, w_{i,2m+3}, v', w_{\ell,2m+3}, \dots, w_{\ell,0}$  with  $\ell \neq j$ . What is important here is that we have to identify  $w_{i,1}$  with  $w_{\ell,0}$  and  $w_{i,2m+3}$  with  $w_{\ell,2m+3}$ . This is the case since the alternative (identifying  $w_{i,0}$  with  $w_{\ell,0}$  or  $v' = 2_{j,2m+2}$  with  $w_{\ell,2m+3}$ ) leads to a variable labelled with  $G$ ,  $D_\ell$ , and  $D_i$  (resp.  $D_j$ ), and thus there is no match. Once these two identifications have been done, there is more than one way to identify the remaining nodes on the mentioned cycle, but the resulting query is always the same.

In summary, it is not hard to see that  $q_{\text{tile}}$  connects those  $G$ -nodes that are labelled by different tile types. Observe that we need property (T2) for this query to match at all.

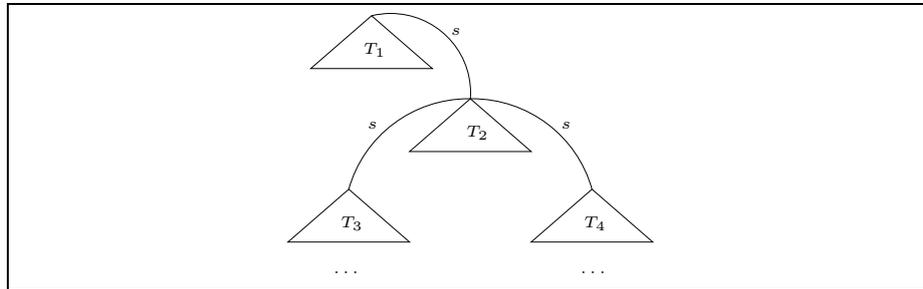
Now, the desired query  $q_{\mathcal{D},c}$  is simply the union of  $q_{\text{cnt}}$  and  $q_{\text{tile}}$ . From what was already said about  $q_{\text{cnt}}$  and  $q_{\text{tile}}$ , it is easily derived that  $q_{\mathcal{D},c}$  does not match the grid representation iff Property (\*) is satisfied. It is possible to show that there is a solution for  $\mathcal{D}$  and  $c$  iff  $(\emptyset, \mathcal{A}_{\mathcal{D},c}) \not\models q_{\mathcal{D},c}$ . We have thus proved that rooted query entailment in  $\mathcal{ALCC}$  is co-NEXPTIME-hard. A matching upper bound can be obtained by adapting the techniques in [6]. More details are given in the full version of this paper.

**Theorem 1.** *Rooted query entailment in  $\mathcal{ALCC}$  is co-NEXPTIME-complete. This holds even w.r.t. knowledge bases in which the TBox is empty and the ABox is a singleton.*

#### 4 Boolean Query Entailment in $\mathcal{ALCC}$ is 2-EXPTIME-complete

If we drop the requirement that queries are connected or that they have at least one answer variable, query entailment in  $\mathcal{ALCC}$  becomes 2-EXPTIME-complete. An upper bound can be taken e.g. from [6]. The lower bound can be proved by a reduction of the word problem of exponentially space bounded alternating Turing machines (ATMs) [4]. Because of space limitations, we can only give a very rough sketch of this result here. More details can be found in the extended version of this paper [10].

The main idea is to represent each configuration of an ATM by the leafs of a tree of depth  $n$ , very similar to the grid representation in Section 3. The trees representing configurations are then interconnected to a tree representing the computation. This is illustrated in Figure 4, where each of the  $T_i$  is a tree of depth  $n$  that is built using the role name  $s$ . The leafs of each  $T_i$  represent a configuration. The tree  $T_1$  represents an existential configuration, and thus has only one successor configuration, which is represented by  $T_2$  and connected via the same role name  $s$  also used inside the  $T_i$  trees. In contrast, the tree  $T_2$  represents a universal configuration with two successor configurations  $T_2$  and  $T_3$ . The crucial point in the reduction is to relate the content of tape cells in one configuration to the content of the corresponding cells in the successor configurations. In principle, this is achieved using queries that are very similar to the query  $q_{\mathcal{D},c}$  employed in the previous section. A few additional technical tricks are needed to achieve directedness (i.e., talking only about successor configurations, but not about predecessor configurations) since we work with symmetric roles.



**Fig. 4.** Representing ATM computations.

**Theorem 2.** *Query entailment in  $\mathcal{ALCI}$  is 2-EXPTIME-complete. This holds even for queries without answer variables and w.r.t. knowledge bases in which the ABox is a singleton.*

## 5 Conclusion

We have shown that in the presence of inverse roles, conjunctive query answering is computationally more costly than instance checking. A Corresponding NEXPTIME upper bound for Theorem 1 and containment of conjunctive query entailment in EXPTIME for  $\mathcal{ALC}$  will be shown elsewhere. As (almost) remarked by a reviewer, the proof of Theorem 2 can easily be adapted to rooted query entailment if transitive roles and role hierarchies are present. Details on this will also be given elsewhere.

**Acknowledgement** We thanks the anonymous reviewers for valuable remarks on the submitted version of this paper.

## References

1. F. Baader, D. L. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.
2. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.
3. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*. AAAI Press, 2006.
4. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.
5. B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for description logics with transitive roles. In B. Parsia, U. Sattler, and D. Toman, editors, *Proceedings of the 2006 International Workshop on Description Logics (DL'06)*, volume 189 of CEUR-WS, 2006.

6. B. Glimm, C. Lutz, I. Horrocks, and U. Sattler. Answering conjunctive queries in the *SHIQ* description logic. In M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 299–404. AAAI Press, 2007.
7. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic *SHIQ*. In D. MacAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, number 1831 in Lecture Notes in Computer Science, Germany, 2000. Springer Verlag.
8. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In L. P. Kaelbling and A. Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 466–471. Professional Book Center, 2005.
9. C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002.
10. C. Lutz. Inverse Roles Make Conjunctive Queries Hard. Available from <http://lat.inf.tu-dresden.de/~clu/papers/>
11. C. Lutz, C. Areces, I. Horrocks, and U. Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research (JAIR)*, 23:667–726, 2005.
12. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in *SHIQ*. In B. Parsia, U. Sattler, and D. Toman, editors, *Proceedings of the 2006 International Workshop on Description Logics (DL'06)*, volume 189 of *CEUR-WS*, 2006.
13. A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

## A From $\mathcal{ALC}^{\text{rs}}$ to $\mathcal{ALCI}$ without TBoxes

We show that rooted query entailment in  $\mathcal{ALC}^{\text{rs}}$  w.r.t. the empty TBox can be polynomially reduced to rooted query entailment in  $\mathcal{ALCI}$  w.r.t. the empty TBox.

As already explained, the main idea behind the reduction is to replace each symmetric role  $r$  with the composition of  $r^-$  and  $r$ . Let  $\mathcal{A}$  be an  $\mathcal{ALC}^{\text{rs}}$  ABox and  $q$  a conjunctive query. We assume w.l.o.g. that all concepts in  $\mathcal{A}$  are in negation normal form (NNF), i.e., that negation is applied only to concept names. Let  $\text{Ind}(\mathcal{A})$  denote the set of all individual names occurring in  $\mathcal{A}$ ,  $\text{rol}(\mathcal{A})$  be the set of role names used in  $\mathcal{A}$ , and let  $\text{rol}(q)$  be defined analogously. Fix a fresh concept name  $R$ . Intuitively, the purpose of  $R$  is to distinguish “real” domain elements from the auxiliary ones that serve as intermediate points in the composition of  $r^-$  and  $r$ . Also, define  $X$  as an abbreviation for  $\bigwedge_{r \in \text{rol}(\mathcal{A}) \cup \text{rol}(q)} \exists r^- . \top$ . We will enforce that  $X$  is satisfied by all relevant real individuals, thus achieving reflexivity.

We now present the details of the reduction. For each concept  $C$  in NNF, let  $\delta(C)$  denote the result of replacing

- every subconcept  $\exists r.C$  with  $\exists r^- . \exists r.(C \sqcap R \sqcap X)$ , and
- every subconcept  $\forall r.C$  with  $\forall r^- . \forall r.C$ ;

Now define an  $\mathcal{ALCI}$  ABox  $\mathcal{A}'$  and a query  $q'$  by manipulating  $\mathcal{A}$  and  $q$  as follows:

1. replace every concept assertion  $C(a) \in \mathcal{A}$  with  $\delta(C)(a)$ ;
2. for all  $a \in \text{Ind}(\mathcal{A})$ , add a concept assertion  $R \sqcap X(a)$  to  $\mathcal{A}$ ;

3. replace every role assertion  $r(a, b) \in \mathcal{A}$  with  $r(c, a)$  and  $r(c, b)$ , where  $c$  is a fresh individual name;
4. for every variable  $v$  in  $q$ , add  $R(v)$  to  $q$ ;
5. replace every role atom  $r(v, v') \in q$  with  $r(v^*, v)$  and  $r(v^*, v)$ , where  $v^*$  is a fresh variable.

The following lemma shows that our reduction is correct.

**Lemma 2.**  $\mathcal{A} \not\models q$  iff  $\mathcal{A}' \not\models q'$ .

**Proof.** “ $\Rightarrow$ ”. If  $\mathcal{A} \not\models q$ , then there is a model  $\mathcal{I}$  of  $\mathcal{A}$  such that  $\mathcal{I} \not\models q$ . Define a model  $\mathcal{I}'$  as follows:

- $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \{x_{d,r,e} \mid r \in \text{rol}(\mathcal{A}) \cup \text{rol}(q) \text{ and } (d, e) \in r^{\mathcal{I}}\}$ ;
- $r^{\mathcal{I}'} = \{(x_{d,r,e}, d), (x_{d,r,e}, e) \mid (d, e) \in r^{\mathcal{I}}\}$
- $A^{\mathcal{I}'} = A^{\mathcal{I}}$  for all concept names  $A$  except  $R$ ;
- $R^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ ;
- $a^{\mathcal{I}'} = a^{\mathcal{I}}$  for all  $a \in \text{Ind}(\mathcal{A})$ ;
- if  $c$  was introduced into  $\mathcal{A}'$  to split the assertion  $r(a, b) \in \mathcal{A}$ , set  $c^{\mathcal{I}'} = x_{a^{\mathcal{I}}, r, b^{\mathcal{I}}}$ .

It is readily checked that  $\mathcal{I}'$  is a model of  $\mathcal{A}'$ . In particular,  $X^{\mathcal{I}'} = \Delta^{\mathcal{I}'}$  since roles are interpreted reflexively in  $\mathcal{I}$ . Furthermore, since  $\mathcal{I} \not\models q$ , we have  $\mathcal{I}' \not\models q'$ : suppose to the contrary that  $\mathcal{I}' \models^{\pi} q'$  for some match  $\pi$ . Since  $q'$  contains the atom  $R(v)$  for every variable  $v \in \text{Var}(q)$ , we have  $\pi(v) \in \Delta^{\mathcal{I}'}$  for all  $v \in \text{Var}(q)$ . Let  $\pi'$  be the restriction of  $\pi$  to the variables in  $\text{Var}(q)$ . It is readily checked that  $\mathcal{I} \models^{\pi'} q$ , which is a contradiction.

“ $\Leftarrow$ ”. If  $\mathcal{A}' \not\models q'$ , then there is a model  $\mathcal{I}'$  of  $\mathcal{A}'$  such that  $\mathcal{I}' \not\models q'$ . Define a model  $\mathcal{I}$  as follows:

- $\Delta^{\mathcal{I}} = (R \sqcap X)^{\mathcal{I}'}$ ;
- $r^{\mathcal{I}} = \{(d, e) \mid \exists f. (f, d) \in r^{\mathcal{I}'} \wedge (f, e) \in r^{\mathcal{I}'}\}$ ;
- $A^{\mathcal{I}} = A^{\mathcal{I}'} \cap \Delta^{\mathcal{I}'}$ ;
- $a^{\mathcal{I}} = a^{\mathcal{I}'}$  for all  $a \in \text{Ind}(\mathcal{A})$ .

Observe that  $r^{\mathcal{I}}$  is reflexive (due to the choice of  $\Delta^{\mathcal{I}}$  as a subset of  $X^{\mathcal{I}'}$ ) and symmetric. Also observe that the interpretation of the individual names is well-defined: since  $\mathcal{A}'$  contains  $R \sqcap X(a)$  for all  $a \in \text{Ind}(\mathcal{A})$ ,  $a^{\mathcal{I}'} \in \Delta^{\mathcal{I}'}$ . Since  $\mathcal{I}' \not\models q'$  and it is easily seen that  $\mathcal{I} \models q$  would imply  $\mathcal{I}' \models q'$ , we have  $\mathcal{I} \not\models q$ . It remains to show that  $\mathcal{I}$  is a model of  $\mathcal{A}$ . This is a consequence of the following claim, which is easily proved by induction on the structure of  $C$ .

**Claim.** For all  $d \in \Delta^{\mathcal{I}}$  and all  $C \in \text{sub}(\mathcal{A})$ ,  $d \in \delta(C)^{\mathcal{I}'}$  implies  $d \in C^{\mathcal{I}}$ .

We only do the two interesting cases.

- Let  $C = \forall r. D$ . Then  $\delta(C) = \forall r^-. \forall r. \delta(D)$ . Let  $(d, e) \in r^{\mathcal{I}}$ . We have to show that  $e \in D^{\mathcal{I}}$ . Since  $(d, e) \in r^{\mathcal{I}}$ , by definition of  $\mathcal{I}$  we have  $(d, e) \in (r^-)^{\mathcal{I}'} \circ r^{\mathcal{I}'}$ . Since  $d \in \delta(C)^{\mathcal{I}'}$ , we have  $e \in D^{\mathcal{I}'}$  and it remains to apply the induction hypothesis.
- Let  $C = \exists r. D$ . Then  $\delta(C) = \exists r^-. \exists r. (\delta(D) \sqcap R \sqcap X)$ . Since  $d \in \delta(C)^{\mathcal{I}'}$ , there is an  $e \in \Delta^{\mathcal{I}'}$  such that (i)  $(d, e) \in (r^-)^{\mathcal{I}'} \circ r^{\mathcal{I}'}$  and (ii)  $e \in (\delta(D) \sqcap R \sqcap X)^{\mathcal{I}'}$ . By (ii),  $d \in \Delta^{\mathcal{I}}$ . By (i) and definition of  $\mathcal{I}$ ,  $(d, e) \in r^{\mathcal{I}}$ . By (ii) and induction hypothesis,  $d \in D^{\mathcal{I}}$  and we are done.  $\square$

# Planning in Action Formalisms based on DLs: First Results

Maja Miličić

Institut für Theoretische Informatik  
TU Dresden, Germany  
maja@tcs.inf.tu-dresden.de

**Abstract.** In this paper, we continue the recently started work on integrating action formalisms with description logics (DLs), by investigating planning in the context of DLs. We prove that the plan existence problem is decidable for actions described in fragments of *ALCQIO*. More precisely, we show that its computational complexity coincides with the one of projection for DLs between *ALC* and *ALCQIO*.

## 1 Introduction

The idea to investigate action formalisms based on description logics was inspired by the expressivity gap between existing action formalisms: they were either based on FO logic and undecidable, like the Situation Calculus [12] and the Fluent Calculus [14], or decidable but only propositional.

First results on integrating DLs with action formalisms from [2] show that reasoning remains decidable even if an action formalism is based on the expressive DL *ALCQIO*. In [2], ABox assertions are used for describing the current state of the world, and the pre- and post-conditions of actions. Domain constraints are captured by *acyclic* TBoxes, and post-conditions may contain only atomic concept and role assertions. It is shown in [2] that the projection and executability problem for actions can be reduced to standard DL reasoning problems. Further papers in this line [11, 10] treat the problem of computing ABox updates and the ramification problem induced by GCIs.

However, in the mentioned DL-action-framework, planning, an important reasoning task, has not yet been considered. Intuitively, given an initial state  $\mathcal{A}$ , final state  $I$  and a final set of actions  $\text{Op}$ , the *plan existence problem* is the following: “is there a plan (a sequence of actions from  $\text{Op}$ ) which transforms  $\mathcal{A}$  into  $I$ ?”. It is known that, already in the propositional case, planning is a hard problem. For example, the plan existence problem for propositional STRIPS-style actions is PSPACE-complete [5, 7].

The planning problem in DL action formalisms is not only interesting from the theoretical point of view. It is well known that web ontology languages for the Semantic Web are based on description logics; thus actions described in DLs can be viewed as simple semantic web services. In this context, planning is a very important reasoning task as it supports, e.g., web service discovery which is needed for an automatic service execution.

This paper is, to our best knowledge, the first try to formally define the planning problem in the context of description logics. It is based on the action formalism from [2]. We investigate the computational complexity of the plan existence problem for the description logics “between”  $\mathcal{ALC}$  and  $\mathcal{ALCQIO}$ . We show that, in these logics, the plan existence problem is decidable, and of the same computational complexity as projection. In the last section we discuss possible ways of developing practical planning algorithms for DLs.

## 2 Preliminaries

In this paper we will use a slightly modified version of the action formalism from [2]. We disallow occlusions, a source of a limited non-determinism in [2]. Moreover, we introduce parameterised actions (operators). The formalism is not restricted to a particular DL, but for our complexity results we will consider the DL  $\mathcal{ALCQIO}$  and its fragments. We refrain from introducing the syntax and semantics of  $\mathcal{ALCQIO}$  in full detail, referring instead to [1].

We give only the definition of ABoxes, as it slightly differs from the one from [1]. An *ABox assertion* is of the form  $C(a)$ ,  $r(a, b)$ , or  $\neg r(a, b)$  where  $a, b$  are individual names,  $C$  is a concept, and  $r$  a role name. An *ABox* is a finite set of ABox assertions.

The main ingredients of our framework are operators and actions (as defined below), ABoxes for describing the current knowledge about the state of affairs in the application domain, and acyclic TBoxes for describing general knowledge about the application domain similar to state constraints in the SitCalc and Fluent Calculus.

**Definition 1 (Action, operator).** Let  $N_X$  and  $N_I$  be disjoint and countably infinite sets of variables and individual names. Moreover, let  $\mathcal{T}$  be an acyclic TBox. A primitive literal for  $\mathcal{T}$  is an ABox assertion

$$A(a), \neg A(a), r(a, b), \text{ or } \neg r(a, b)$$

with  $A$  a primitive concept name in  $\mathcal{T}$ ,  $r$  a role name, and  $a, b \in N_I$ . An atomic  $\alpha = (\text{pre}, \text{post})$  for  $\mathcal{T}$  consists of

- a finite set **pre** of ABox assertions, the pre-conditions;
- a finite set **post** of conditional post-conditions of the form  $\varphi/\psi$ , where  $\varphi$  is an ABox assertion and  $\psi$  is a primitive literal for  $\mathcal{T}$ .

A composite action for  $\mathcal{T}$  is a finite sequence  $\alpha_1, \dots, \alpha_k$  of atomic actions for  $\mathcal{T}$ .

An operator for  $\mathcal{T}$  is a parametrised atomic action for  $\mathcal{T}$ , i.e., an action in which definition variables from  $N_X$  may occur in place of individual names.

We call post-conditions of the form  $\top(t)/\psi$  *unconditional* and write just  $\psi$  instead.

Applying an action changes the state of affairs, and thus transforms an interpretation  $\mathcal{I}$  into an interpretation  $\mathcal{I}'$ . Intuitively, the pre-conditions specify under which conditions the action is applicable. The post-condition  $\varphi/\psi$  says that, if  $\varphi$  is true in the original interpretation  $\mathcal{I}$ , then  $\psi$  is true in the interpretation  $\mathcal{I}'$  obtained by applying the action.

**Definition 2.** Let  $\mathcal{T}$  be an acyclic TBox,  $\alpha = (\text{pre}, \text{post})$  an atomic action for  $\mathcal{T}$ , and  $\mathcal{I}, \mathcal{I}'$  models of  $\mathcal{T}$  respecting the unique name assumption (UNA) and sharing the same domain and interpretation of all individual names. We say that  $\alpha$  may transform  $\mathcal{I}$  to  $\mathcal{I}'$  ( $\mathcal{I} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}'$ ) iff, for each primitive concept  $A$  and role name  $r$ , we have

$$\begin{aligned} A^{\mathcal{I}'} &:= (A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid \varphi/A(a) \in \text{post and } \mathcal{I} \models \varphi\}) \setminus \\ &\quad \{a^{\mathcal{I}} \mid \varphi/\neg A(a) \in \text{post and } \mathcal{I} \models \varphi\} \\ r^{\mathcal{I}'} &:= (r^{\mathcal{I}} \cup \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/r(a, b) \in \text{post and } \mathcal{I} \models \varphi\}) \setminus \\ &\quad \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/\neg r(a, b) \in \text{post and } \mathcal{I} \models \varphi\}. \end{aligned}$$

The composite action  $\alpha_1, \dots, \alpha_k$  may transform  $\mathcal{I}$  to  $\mathcal{I}'$  ( $\mathcal{I} \Rightarrow_{\alpha_1, \dots, \alpha_k}^{\mathcal{T}} \mathcal{I}'$ ) iff there are models  $\mathcal{I}_0, \dots, \mathcal{I}_k$  of  $\mathcal{T}$  with  $\mathcal{I} = \mathcal{I}_0$ ,  $\mathcal{I}' = \mathcal{I}_k$ , and  $\mathcal{I}_{i-1} \Rightarrow_{\alpha_i}^{\mathcal{T}} \mathcal{I}_i$  for  $1 \leq i \leq k$ .

Note that this definition does not check whether the action is indeed executable, i.e., whether the pre-conditions are satisfied. It just says what the result of applying the action is, irrespective of whether it is executable or not. Since we use acyclic TBoxes to describe background knowledge, there cannot exist more than one  $\mathcal{I}'$  such that  $\mathcal{I} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}'$ . Thus, actions are deterministic.

Like in [2], we assume that actions  $\alpha = (\text{pre}, \text{post})$  are *consistent with  $\mathcal{T}$*  in the following sense: for every model  $\mathcal{I}$  of  $\mathcal{T}$ , there exists  $\mathcal{I}'$ , such that  $\mathcal{I} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}'$ . It is not difficult to see that this is the case iff  $\{\varphi_1/\psi, \varphi_2/\neg\psi\} \subseteq \text{post}$  implies that the ABox  $\{\varphi_1, \varphi_2\}$  is inconsistent w.r.t.  $\mathcal{T}$ .

Two standard reasoning problems about actions, *projection* and *executability*, are thoroughly investigated in [2] in the context of DLs. Executability is the problem of whether an action can be applied in a given situation, i.e. if pre-conditions are satisfied in the states of the world considered possible.

Formally, let  $\mathcal{T}$  be an acyclic TBox,  $\mathcal{A}$  an ABox, and let  $\alpha_1, \dots, \alpha_n$  be a composite action with  $\alpha_i = (\text{pre}_i, \text{post}_i)$  atomic actions for  $\mathcal{T}$  for  $i = 1, \dots, n$ .

We say that  $\alpha_1, \dots, \alpha_n$  is *executable in  $\mathcal{A}$  w.r.t.  $\mathcal{T}$*  iff the following conditions are true for all models  $\mathcal{I}$  of  $\mathcal{A}$  and  $\mathcal{T}$ :

- $\mathcal{I} \models \text{pre}_1$
- for all  $i$  with  $1 \leq i < n$  and all interpretations  $\mathcal{I}'$  with  $\mathcal{I} \Rightarrow_{\alpha_1, \dots, \alpha_i}^{\mathcal{T}} \mathcal{I}'$ , we have  $\mathcal{I}' \models \text{pre}_{i+1}$ .

Projection is the problem of whether applying an action achieves the desired effect, i.e., whether an assertion that we want to make true really holds after executing the action. Formally, the assertion  $\varphi$  is a *consequence of applying  $\alpha_1, \dots, \alpha_n$  in  $\mathcal{A}$  w.r.t.  $\mathcal{T}$*  iff for all models  $\mathcal{I}$  of  $\mathcal{A}$  and  $\mathcal{T}$  and for all  $\mathcal{I}'$  with  $\mathcal{I} \Rightarrow_{\alpha_1, \dots, \alpha_n}^{\mathcal{T}} \mathcal{I}'$ , we have  $\mathcal{I}' \models \varphi$ .

In [2] it was shown that projection and executability are decidable for the logics between  $\mathcal{ALC}$  and  $\mathcal{ALCQIO}$ . More precisely, projection in  $\mathcal{L}$  can be reduced to (in)consistency of an ABox relative to an acyclic TBox in  $\mathcal{LO}$ . The following theorem from [2] states that upper complexity bounds obtained in this way are optimal:

**Theorem 1.** ([2]) *Projection and executability of composite actions are:*

- (a) PSPACE-complete in  $\mathcal{ALC}, \mathcal{ALCO}, \mathcal{ALCQ}$ , and  $\mathcal{ALCQO}$ ;
- (b) EXPTIME-complete in  $\mathcal{ALCI}$  and  $\mathcal{ALCIO}$ ;
- (c) co-NEXPTIME-complete in  $\mathcal{ALCQI}$  and  $\mathcal{ALCQIO}$ .

Looking carefully at the reduction of projection in  $\mathcal{L}$  to ABox inconsistency in  $\mathcal{LO}$  from [2, 3], we conclude that the upper complexity bounds from Theorem 1 hold even for the “stronger” projection problem, namely the one where the assertion  $\varphi$  is a disjunction of ABox assertions. We will need this strengthened complexity result in the coming sections.

### 3 Planning problem

We continue by defining the plan existence problem in our framework. As in the previous section, we do not fix the DL, but assume it to be a sublogic of  $\mathcal{ALCQIO}$ .

First we introduce a bit of notation. If  $o$  is an operator (for a TBox  $\mathcal{T}$ ), we use  $\text{var}(o)$  to denote the set of variables in  $o$ . A substitution  $v$  for  $o$  is a mapping  $v : \text{var}(o) \rightarrow \mathbf{N}_I$ . An action  $\alpha$  that is obtained by applying a substitution  $v$  to  $o$  is denoted as  $\alpha := o[v]$ . Intuitively, the plan existence problem is: given an acyclic TBox  $\mathcal{T}$  which describes the background knowledge, ABoxes  $\mathcal{A}$  and  $\Gamma$  describing respectively the initial and the goal state, and a set of operators  $\text{Op}$ , is there a plan (sequence of actions obtained by instantiating operators from  $\text{Op}$ ) which “transforms”  $\mathcal{A}$  into  $\Gamma$ ?

In this paper, we assume that operators can be instantiated with individuals from a finite set  $\text{Ind} \subset \mathbf{N}_I$ . Moreover, we assume that  $\mathcal{T}$ ,  $\mathcal{A}$  and  $\Gamma$  contain only individuals from  $\text{Ind}$  (we say that they are based on  $\text{Ind}$ ). For an operator  $o$ , we set  $o[\text{Ind}] := \{o[v] \mid v : \text{var}(o) \rightarrow \text{Ind}\}$  and for  $\text{Op}$  a set of operators, we set  $\text{Op}[\text{Ind}] := \{o[\text{Ind}] \mid o \in \text{Op}\}$ . In the following definition, we formally introduce the notion of a planing task:

**Definition 3 (Planning task).** *A planning task is a tuple  $\Pi = (\text{Ind}, \mathcal{T}, \text{Op}, \mathcal{A}, \Gamma)$ , where*

- $\text{Ind}$  is a finite set of individual names;
- $\mathcal{T}$  is an acyclic TBox based on  $\text{Ind}$ ;
- $\text{Op}$  is a finite set of atomic operators for  $\mathcal{T}$ ;
- $\mathcal{A}$  (initial state) is an ABox based on  $\text{Ind}$ ;
- $\Gamma$  (goal) is an ABox based on  $\text{Ind}$ .

A plan in  $\Pi$  is a composite action  $\alpha = \alpha_1, \dots, \alpha_k$ , such that  $\alpha_i \in \text{Op}[\text{Ind}]$ ,  $i = 1..k$ . A plan  $\alpha = \alpha_1, \dots, \alpha_k$  in  $\Pi$  is a solution to the planning task  $\Pi$  iff:

1.  $\alpha$  is executable in  $\mathcal{A}$  w.r.t.  $\mathcal{T}$ ; and
2. for all interpretations  $\mathcal{I}$  and  $\mathcal{I}'$  such that  $\mathcal{I} \models \mathcal{A}, \mathcal{T}$  and  $\mathcal{I} \Rightarrow_{\alpha}^{\mathcal{T}} \mathcal{I}'$ , it holds that  $\mathcal{I}' \models \Gamma$ .

Two common planning problems, PLANEX and PLANLEN, c.f. [7], are defined below:

**Definition 4 (Planning problems).** Plan existence problem (PLANEX): Does a given planning task  $\Pi$  have a solution?

Bounded plan existence problem (PLANLEN): For a planning task  $\Pi$  and a natural number  $n$ , is there a plan of length at most  $2^n$  which is a solution to  $\Pi$ ?

## 4 Complexity of planning

In this section, we will present a decision procedure for the plan existence problem. It turns out that PLANEX is not more difficult, at least in theory, than projection in the DLs from Theorem 1.

In what follows, for the sake of simplicity we assume that  $\mathcal{T} = \emptyset$ . It is not difficult to show that the complexity results from this section hold in the case of non-empty acyclic TBoxes.

Obviously, the plan existence problem is closely related to projection and executability. First we introduce some notation. Let  $\mathcal{A}$  be an ABox,  $\alpha$  a (possibly composite) action, and  $\varphi$  an ABox assertion or a disjunction of ABox assertions. We will write  $\mathcal{A}^{\alpha} \models \varphi$  iff  $\varphi$  is a consequence of applying  $\alpha$  in  $\mathcal{A}$ . For an ABox  $\mathcal{B}$ , we write  $\mathcal{A}^{\alpha} \models \mathcal{B}$  iff  $\mathcal{A}^{\alpha} \models \varphi$  for all  $\varphi \in \mathcal{B}$ .

Let  $\Pi = (\text{Ind}, \emptyset, \text{Op}, \mathcal{A}, \Gamma)$  be a planning task, for which we want to decide if it has a solution. This means that we want to check if there is a sequence of actions from  $\text{Op}[\text{Ind}]$  which transform the initial state (described by  $\mathcal{A}$ ) into the goal state (described by  $\Gamma$ ). In the propositional case, planning is based on step-wise computation of the next state – which corresponds to computing updated ABoxes. However, in [11], it is shown that an updated ABox may be exponentially large in the size of the initial ABox and the update, which makes this approach unsuitable. We base our approach in this paper on the following observation: possible worlds obtained by applying (composite) actions in the initial world  $\mathcal{A}$  can be implicitly described by  $\mathcal{A}$  together with the list of applied atomic changes (intuitively, this is an accumulated list of the triggered post-conditions).

We define the set of possible (negated) atomic changes as:

$$\mathcal{L} := \{\psi, \neg\psi \mid \varphi/\psi \in \alpha, \alpha \in \text{Op}[\text{Ind}]\}$$

An *update* for  $\Pi$  is a consistent subset of  $\mathcal{L}$ . Let  $\mathcal{U}$  be a set of all updates for  $\Pi$ . Then  $\mathcal{U}$  is our search space, the size of which  $|\mathcal{U}|$  is exponential in the

size of  $|\mathcal{L}|$  (and  $\Pi$ ). For a  $\mathcal{U} \in \mathfrak{U}$ , we set  $\neg\mathcal{U} := \{\neg l \mid l \in \mathcal{U}\}$ . Intuitively,  $\mathcal{U}_0 := \{l \in \mathcal{L} \mid \mathcal{A} \models l\}$  represents the initial state, and all updates  $\mathcal{U} \in \mathfrak{U}$  such that  $\mathcal{A}^{\mathcal{U}} \models \Gamma$  represent goal states<sup>1</sup>.

In the next step, we define the transition relation “ $\xrightarrow{\alpha}_{\mathcal{A}}$ ” between updates. Let  $\mathcal{U}$  and  $\mathcal{V}$  be two updates. For  $\alpha = (\text{pre}, \text{post})$ , we say that  $\mathcal{U} \xrightarrow{\alpha}_{\mathcal{A}} \mathcal{V}$  iff:

- (i)  $\mathcal{A}^{\mathcal{U}} \models \text{pre}$
- (ii)  $\mathcal{V} = (\mathcal{U} \setminus \neg\text{post}_{\alpha}^{\mathcal{U}}) \cup \text{post}_{\alpha}^{\mathcal{U}}$ , where  $\text{post}_{\alpha}^{\mathcal{U}} = \{\psi \mid \mathcal{A}^{\mathcal{U}} \models \bigvee_{\varphi/\psi \in \text{post}} \varphi\}$

Obviously, the relation “ $\xrightarrow{\alpha}_{\mathcal{A}}$ ” is a partial function for every  $\alpha$ . In the following lemma, we show that “ $\xrightarrow{\alpha}_{\mathcal{A}}$ ” simulates “ $\Rightarrow_{\alpha}$ ”<sup>2</sup> on the level of updates.

**Lemma 1.** *Let  $\mathcal{A}$  be an ABox, and  $\alpha = \alpha_1, \dots, \alpha_k$  a composite action, with  $\alpha_i = (\text{pre}_i, \text{post}_i) \in \text{Op}[\text{Ind}]$ . Let  $\mathcal{U}_0 := \{l \in \mathcal{L} \mid \mathcal{A} \models l\}$ . Then the following holds:*

- (a) *There exist unique  $\mathcal{U}_1, \dots, \mathcal{U}_k$  such that  $\mathcal{U}_0 \xrightarrow{\alpha_1}_{\mathcal{A}} \mathcal{U}_1 \cdots \xrightarrow{\alpha_k}_{\mathcal{A}} \mathcal{U}_k$  iff  $\alpha_1, \dots, \alpha_k$  is executable in  $\mathcal{A}$ ;*
- (b) *Let  $\mathcal{U}_k$  be defined as in (a). Then for all interpretations  $\mathcal{I}, \mathcal{I}'$  such that  $\mathcal{I} \models \mathcal{A}$ , we have that  $\mathcal{I} \Rightarrow_{\alpha_1, \dots, \alpha_k} \mathcal{I}'$  iff  $\mathcal{I} \Rightarrow_{\mathcal{U}_k} \mathcal{I}'$ .*

*Proof.* Proof by induction on  $k$ . For  $k = 0$ , trivially true. Assume that the claim holds for  $k = m$ , and let us prove that it implies the same for  $k = m + 1$ .

(a) follows directly from the point (i) of the definition of  $\xrightarrow{\alpha_{m+1}}_{\mathcal{A}}$ . As for (b), let  $\mathcal{I} \models \mathcal{A}$  and let  $\mathcal{I} \Rightarrow_{\alpha_1, \dots, \alpha_{m+1}} \mathcal{I}'$ . The latter holds iff there exists  $\mathcal{I}''$  such that  $\mathcal{I} \Rightarrow_{\alpha_1, \dots, \alpha_m} \mathcal{I}''$  and  $\mathcal{I}'' \Rightarrow_{\alpha_{m+1}} \mathcal{I}'$ . By I.H., we have that for  $\mathcal{I} \models \mathcal{A}$  it holds that  $\mathcal{I} \Rightarrow_{\alpha_1, \dots, \alpha_m} \mathcal{I}''$  iff  $\mathcal{I} \Rightarrow_{\mathcal{U}_m} \mathcal{I}''$ . Finally, the point (ii) of the definition of  $\xrightarrow{\alpha_{m+1}}_{\mathcal{A}}$  implies that there exists  $\mathcal{I}''$  such that  $\mathcal{I} \Rightarrow_{\mathcal{U}_m} \mathcal{I}''$  and  $\mathcal{I}'' \Rightarrow_{\alpha_{m+1}} \mathcal{I}'$  iff  $\mathcal{I} \Rightarrow_{\mathcal{U}_m} \mathcal{I}''$  and  $\mathcal{I}'' \Rightarrow_{\text{post}_{\alpha_{m+1}}^{\mathcal{U}_m}} \mathcal{I}'$ . It is not difficult to see that the latter holds iff  $\mathcal{I} \Rightarrow_{\mathcal{U}_{m+1}} \mathcal{I}'$ .

We now present a procedure which decides if a state  $\mathcal{V} \in \mathfrak{U}$  is reachable from  $\mathcal{U} \in \mathfrak{U}$  by executing a sequence of actions from  $\text{Op}[\text{Ind}]$  (an adaption of the reachability algorithm from [13]). Since the search space  $\mathfrak{U}$  is of size  $3^{\frac{|\mathcal{L}|}{2}} (< 2^{|\mathcal{L}|})$ , there is no need to check for the existence of longer paths.

```

reachable( $\Pi, \mathcal{U}, \mathcal{V}$ )
  if path( $\Pi, \mathcal{U}, \mathcal{V}, |\mathcal{L}|$ )
    then return TRUE
  return FALSE
  
```

**path**( $\Pi, \mathcal{U}, \mathcal{V}, i$ ) checks if  $\mathcal{V}$  is reachable from  $\mathcal{U}$  by a path of length at most  $2^i$ :

<sup>1</sup> Starting from here, we will sometimes write  $\mathcal{U}$  as short for the action  $(\emptyset, \mathcal{U})$ . Please note that  $\mathcal{A}^{\mathcal{U}} \models \varphi$  is only an abbreviation for “ $\varphi$  is a consequence of applying  $(\emptyset, \mathcal{U})$  in  $\mathcal{A}$ ”, and does not imply computing the update of the ABox  $\mathcal{A}$  with  $\mathcal{U}$  as in [11].  
<sup>2</sup>  $\Rightarrow_{\alpha}$  is short for  $\Rightarrow_{\emptyset_{\alpha}}$

```

path( $\Pi, \mathcal{U}, \mathcal{V}, i$ )
  if ( $i = 0$  and ( $\mathcal{U} = \mathcal{V}$  or one_step( $\Pi, \mathcal{U}, \mathcal{V}$ )))
    then return TRUE
  for all ( $\mathcal{W} \in \mathfrak{U}$ )
    if (path( $\Pi, \mathcal{U}, \mathcal{W}, i - 1$ ) and path( $\Pi, \mathcal{W}, \mathcal{V}, i - 1$ ))
      then return TRUE
  return FALSE
    
```

The predicate **one\_step**( $\Pi, \mathcal{U}, \mathcal{V}$ ) checks if  $\mathcal{V}$  can be reached from  $\mathcal{U}$  in exactly one step by applying an action  $\alpha \in \text{Op}[\text{Ind}]$ .

```

one_step( $\Pi, \mathcal{U}, \mathcal{V}$ )
  for all  $\alpha \in \text{Op}[\text{Ind}]$ 
    if ( $\mathcal{U} \xrightarrow{\alpha}_{\mathcal{A}} \mathcal{V}$ )
      then return TRUE;
  return FALSE;
    
```

**Lemma 2.** *Let  $\Pi = (\text{Ind}, \mathcal{T}, \text{Op}, \mathcal{A}, \Gamma)$  be a planning task and let  $\mathcal{U}_0 := \{l \in \mathcal{L} \mid \mathcal{A} \models l\}$ . Then  $\Pi$  has a solution iff there exists an  $\mathcal{U}_\Gamma \in \mathfrak{U}$  such that  $\mathcal{A}^{\mathcal{U}_\Gamma} \models \Gamma$  and **reachable**( $\Pi, \mathcal{U}_0, \mathcal{U}_\Gamma$ ) returns TRUE.*

*Proof.* “ $\Rightarrow$ ” Let the plan  $\alpha_1, \dots, \alpha_k$  be a solution to  $\Pi$  such that  $k < 2^{|\mathcal{L}|}$ . This means that (i)  $\alpha_1, \dots, \alpha_k$  is executable w.r.t.  $\mathcal{A}$  and (ii)  $\mathcal{A}^{\alpha_1, \dots, \alpha_k} \models \Gamma$ . By Lemma 1 (a), there exist unique  $\mathcal{U}_i, 1 \leq i \leq k$ , such that  $\mathcal{U}_0 \xrightarrow{\alpha_1}_{\mathcal{A}} \mathcal{U}_1 \cdots \xrightarrow{\alpha_k}_{\mathcal{A}} \mathcal{U}_k$ . Thus, **reachable**( $\Pi, \mathcal{U}_0, \mathcal{U}_k$ ) returns TRUE. Let  $\mathcal{U}_\Gamma = \mathcal{U}_k$ . By Lemma 1 (b), we have that  $\mathcal{A}^{\alpha_1, \dots, \alpha_k} \models \Gamma$  implies  $(\mathcal{A}^{\mathcal{U}_\Gamma} =) \mathcal{A}^{\mathcal{U}_k} \models \Gamma$ .

“ $\Leftarrow$ ” Let  $\mathcal{U}_\Gamma \in \mathfrak{U}$  be such that  $\mathcal{A}^{\mathcal{U}_\Gamma} \models \Gamma$  and **reachable**( $\Pi, \mathcal{U}_0, \mathcal{U}_\Gamma$ ) returns TRUE. Then there exists a sequence of actions  $\alpha_1, \dots, \alpha_k$  such that  $\mathcal{U}_0 \xrightarrow{\alpha_1}_{\mathcal{A}} \mathcal{U}_1 \cdots \xrightarrow{\alpha_k}_{\mathcal{A}} \mathcal{U}_k (= \mathcal{U}_\Gamma)$ . By Lemma 1, we have that  $\alpha_1, \dots, \alpha_k$  is executable w.r.t.  $\mathcal{A}$  and  $\mathcal{A}^{\alpha_1, \dots, \alpha_k} \models \Gamma$ . Thus,  $\alpha_1, \dots, \alpha_k$  is a solution to  $\Pi$ .

The previous lemma tells us that the plan existence problem can be decided by checking if **reachable**( $\mathcal{U}_0, \mathcal{U}_F, \Pi$ ) returns TRUE for some final state  $\mathcal{U}_F \in \mathfrak{U}$ . If **one\_step** could be decided in a constant time, **reachable** would require PSPACE. However, **one\_step** relies on polynomially many projection calls, which means that both **one\_step** and **reachable** belong to the same complexity class as projection in DLs from Theorem 1, i.e. they are in PSPACE (EXPTIME, co-NEXPTIME) if projection is in PSPACE (EXPTIME, co-NEXPTIME).

- (i) If projection is in PSPACE, then we can decide the plan existence problem in PSPACE: obviously, guessing a state (update)  $\mathcal{U}_F$  from  $\mathfrak{U}$ , and checking whether  $\mathcal{A}^{\mathcal{U}_F} \models \Gamma$  and **reachable**( $\mathcal{U}_0, \mathcal{U}_F, \Pi$ ) returns TRUE, can be done in NPSpace. Finally, we use the result by Savitch [13] that PSPACE = NPSpace.
- (ii) If projection is in EXPTIME (co-NEXPTIME), then we can check for all  $\mathcal{U}_F$  from  $\mathfrak{U}$ , if  $\mathcal{A}^{\mathcal{U}_F} \models \Gamma$  and **reachable**( $\mathcal{U}_0, \mathcal{U}_F, \Pi$ ) returns TRUE. Since  $\mathfrak{U}$  is

exponentially big in the size of  $\Pi$ , plan existence problem can thus be decided in EXPTIME (co-NEXPTIME).

We obtained the following lemma:

**Lemma 3.** *Let  $\mathcal{L} \in \{\mathcal{ALC}, \mathcal{ALCO}, \mathcal{ALCT}, \mathcal{ALCQ}, \mathcal{ALCTO}, \mathcal{ALCQO}, \mathcal{ALCQT}, \mathcal{ALCQTO}\}$ . The plan existence problem in  $\mathcal{L}$  has the same upper complexity bound as projection in  $\mathcal{L}$ .*

We show that the upper complexity bounds established in Lemma 1 are tight by the following easy reduction of projection to PLANEX. Let  $\mathcal{A}$  be an ABox,  $\alpha$  an action without pre-conditions and only with unconditional post-conditions, and  $\varphi$  an assertion. We define the planning task  $\Gamma_{\mathcal{A},\alpha,\varphi}$  as  $\Gamma_{\mathcal{A},\alpha,\varphi} := (\emptyset, \emptyset, \{\alpha\}, \mathcal{A}, \{\varphi\})$ . It is not difficult to see that  $\mathcal{A}^\alpha \models \varphi$  iff  $\Gamma_{\mathcal{A},\alpha,\varphi}$  has a solution.

Since the lower bounds for projection from Theorem 1 hold already in the case of the empty TBox and an atomic action with no pre-conditions and no occlusions and only with unconditional post-conditions [2], we conclude that the complexity bounds from Lemma 1 are optimal, i.e., plan existence problem is of exactly the same computational complexity as projection. Moreover, the afore presented reduction implies that the same hardness results hold for the the bounded plan existence problem.

**Theorem 2.** *The planning problems PLANEX and PLANLEN are:*

- (a) PSPACE-complete in  $\mathcal{ALC}$ ,  $\mathcal{ALCO}$ ,  $\mathcal{ALCQ}$ , and  $\mathcal{ALCQO}$ ;
- (b) EXPTIME-complete in  $\mathcal{ALCT}$  and  $\mathcal{ALCTO}$ ;
- (c) co-NEXPTIME-complete in  $\mathcal{ALCQT}$  and  $\mathcal{ALCQTO}$ .

## 5 Extended Planning

The previous decidability and complexity results are obtained under assumption that the set of individuals  $\text{Ind}$  used to instantiate operators is finite and a part of the input. This assumption is rather natural and in the line with the standard definitions of planning tasks for STRIPS operators from [5, 7]. Intuitively,  $\text{Ind}$  is a set of individuals the planning agent has control over.

Alternatively, one can omit individuals from the input and define a planning task  $\Pi$  as  $\Pi = (\mathcal{T}, \text{Op}, \mathcal{A}, \Gamma)$ . The *extended plan existence problem* is the one of whether there is a solution for  $\Pi$ , defining a plan for  $\Pi$  to be a sequence of actions  $\alpha_1, \dots, \alpha_k$ , where each  $\alpha_i$  is obtained by instantiating an operator from  $\text{Op}$  with individuals from an infinitely countable set  $\mathbb{N}_1$ .

The extended planning raises new interesting questions:

- Q1 In order to solve  $\Pi$ , do we have to use infinitely many individuals?
- Q2 If the number of needed individuals can be shown to be bounded by  $f(|\Pi|)$ , is  $f$  a polynomial (exponential, double-exponential,...) function?

In the case of the datalog STRIPS, it is shown that the extended plan existence problem is undecidable [7, 6]. However, this undecidability result does not automatically carry over to the action formalism used in this paper. Indeed, the undecidability result from [7, 6] relies on the closed world assumption and negative pre-conditions. By using these two, one can define operators which are applicable only if instantiated with “unused” individuals. Such operators would have  $\neg\text{Used}(x)$  among its pre-conditions, and  $\text{Used}(x)$  in the list of post-conditions. Like this, one can enforce a usage of infinitely many individuals.

In the case of DLs considered in the previous sections, due to the open world assumption, it is not possible to state that all individuals not appearing in the initial ABox are instances of the concept  $\neg\text{Used}$ .

However, in the presence of the universal role  $U$ , we can make assertions over the whole domain. For example, the assertion  $\forall U. \neg\text{Used}(a)$  can ensure that all element domains are unused in the initial state. We will show that extended planning in  $\mathcal{ALC}_U$  (extension of  $\mathcal{ALC}$  with the universal role) is undecidable. Undecidability is shown by reducing the halting problem of a deterministic Turing machine to the extended plan existence problem, similar to [6].

Let  $M = (Q, \Sigma, \delta, q_0, q_f)$  be a deterministic Turing machine, where

- $Q = \{q_0, \dots, q_n\}$  a finite set of states;
- $\Sigma = \{\text{blank}, a_1, \dots, a_m\}$  a finite alphabet;
- $\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\}$  is a transition function;
- $q_0$  is the initial state;
- $q_f \in Q$  is the final state.

Let  $a = a_{i_0}, \dots, a_{i_k} \in \Sigma^*$  be an input word. We will define a planning task  $\Pi_{M,a} = (\emptyset, \text{Op}_{M,a}, \mathcal{A}_{M,a}, \Gamma_{M,a})$  such that a planner for  $\Pi$  simulates moves of the Turing Machine  $M$ .

In the reduction, we use concept names  $Q_0, \dots, Q_n, \text{Blank}, A_1, \dots, A_m, \text{Used}, \text{Last}, M, \text{Done}$ , and a role name  $\text{right}$ . We define the initial state  $\mathcal{A}_{M,a}$ , the goal  $\Gamma_{M,a}$ , and the set of operators  $\text{Op}_{M,a}$  as:

$$\begin{aligned} \mathcal{A}_{M,a} &:= \{(M \sqcap \forall U. \neg\text{Used})(t_0)\} \cup \{A_{i_0}(t_0), \dots, A_{i_k}(t_k)\} \\ &\quad \cup \{\text{right}(t_0, t_1), \dots, \text{right}(t_{k-1}, t_k)\} \\ \Gamma_{M,a} &:= \{\text{Done}(t_0)\} \\ \text{Op}_{M,a} &:= \{\text{start}, \text{create\_succ}(x.y), \text{done}(x), \text{done\_to\_left}(x, y)\} \cup \\ &\quad \bigcup_{\delta(q,a)=(q',b,R)} \{\text{right}_{q,a,q',b}(x, y)\} \cup \bigcup_{\delta(q,a)=(q',b,L)} \{\text{left}_{q,a,q',b}(x, y)\} \end{aligned}$$

where the single operators are defined as follows:

$$\begin{aligned} \text{start} &:= (\{M(t_0)\}, \{\text{Used}(t_0), \dots, \text{Used}(t_k), \text{Last}(t_k), \neg M(t_0), Q_0(t_0)\}) \\ \text{create\_succ}(x.y) &:= (\{\text{Last}(x), \neg\text{Used}(y)\}, \\ &\quad \{\text{right}(x, y), \neg\text{Last}(x), \text{Last}(y), \text{Used}(y), \text{Blank}(y)\}) \end{aligned}$$

$$\begin{aligned}
 \text{right}_{q,a,q',b}(x,y) &:= (\{Q(a), A(x), \text{right}(x,y)\}, \{\neg Q(x), \neg A(x), B(x), Q'(y)\}) \\
 \text{left}_{q,a,q',b}(x,y) &:= (\{Q(a), A(x), \text{right}(y,x)\}, \{\neg Q(x), \neg A(x), B(x), Q'(y)\}) \\
 \text{done}(x) &:= (\{Q_f(x)\}, \{\text{Done}(x)\}) \\
 \text{done\_to\_left}(x,y) &:= (\{\text{Done}(x), \text{right}(y,x)\}, \{\text{Done}(y)\})
 \end{aligned}$$

It is not difficult to show that the following lemma holds:

**Lemma 4.** *The Turing machine  $M$  halts for the input  $a$  iff there is a solution to the planning task  $\Pi_{M,a} = (\emptyset, \text{Op}_{M,a}, \mathcal{A}_{M,a}, \Gamma_{M,a})$ .*

Thus, we obtained the following theorem:

**Theorem 3.** *The extended plan existence problem is undecidable in  $\mathcal{ALC}_U$ .*

To conclude this section, we are leaving questions Q1 and Q2 open for the description logics between  $\mathcal{ALC}$  and  $\mathcal{ALCQIO}$ . We conjecture that, without the universal role, it is not possible to enforce introduction of an unbounded number of individuals. It seems to be difficult even to enforce an exponential number of new individuals.

## 6 Conclusion and Future Work

In this paper, we have shown that the planning problems PLANEX and PLANLEN are decidable in action formalisms based on fragments of  $\mathcal{ALCQIO}$ . More precisely, both PLANEX and PLANLEN are of the same computational complexity as projection in the logics between  $\mathcal{ALC}$  and  $\mathcal{ALCQIO}$ . It is not difficult to show that the same complexity results apply to the unrestricted version of the action formalism from [2], the one with occlusions. We conjecture that the extended plan existence problem for DLs without universal role is also decidable, but a proof is yet to be done.

A future work will include a development and implementation of efficient planners for description logics. Unfortunately, the complexity results we obtained are quite discouraging. Even in the propositional case, planning is a very hard combinatorial problem. An advantage in the propositional case is that, although PLANEX is PSPACE-complete [5, 7], if we are only interested in polynomial-length plans (which is the case in practice), then planning becomes NP-complete. On the contrary, for DLs between  $\mathcal{ALC}$  and  $\mathcal{ALCQIO}$ , looking for polynomial-length plans is not easier than PLANEX, since the hardness results from Theorem 2 hold already for the plans of constant length. Thus it looks reasonable to start with “small” DLs, like  $\mathcal{EL}$  or  $\mathcal{EL}^{(\neg)}$  and try to adapt one of the well-known strategies from propositional planning: reduction to SAT [9]; planning based on planning graphs [4]; or a combination of the previous two [8].

There seem to be two possible methods for reducing projection and planning in  $\mathcal{EL}$  to SAT. One would require “pre-computing” relevant consequences  $C(a)$  of the initial ABox, where  $C(a)$  is relevant if  $C$  is a sub-concept of the goal or

of the concepts appearing in the pre-conditions, while the other one would use propositional formulae to describe models of the initial ABox.

**Acknowledgements:** The author wants to thank Carsten Lutz for inspiring discussions.

## References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, USA, 2005.
3. F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms for reasoning about web services. LTCS-Report 05-02, TU Dresden, Germany, 2005. See <http://lat.inf.tu-dresden.de/research/reports.html>.
4. A. Blum and M. Furst. Fast planning through planning graph analysis. In *Proceedings of Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1636–1642, 1995.
5. T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.
6. K. Erol, D. S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning: A detailed analysis. Technical Report CS-TR-2797, University of Maryland College Park, Maryland 20742, USA, 1991.
7. K. Erol, D. S. Nau, and V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence*, 76(1-2):75–88, 1995.
8. H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of In Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-96)*, pages 1194–1201, Menlo Park, California, 1996.
9. H. A. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363, 1992.
10. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Reasoning about actions using description logics with general TBoxes. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA 2006)*, volume 4160 of *Lecture Notes in Artificial Intelligence*, pages 266–279, 2006.
11. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 46–56. AAAI Press, 2006.
12. R. Reiter. *Knowledge in Action*. MIT Press, 2001.
13. W. J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.
14. M. Thielscher. Introduction to the Fluent Calculus. *Electronic Transactions on Artificial Intelligence*, 2(3-4):179–192, 1998.

## On Ordering Descriptions in a Description Logic

Jeffrey Pound<sup>‡</sup>, Lubomir Stanchev<sup>†</sup>, David Toman<sup>‡,b</sup>, and Grant Weddell<sup>‡</sup>

<sup>†</sup>Department of Computer Science  
Purdue University Fort Wayne, U.S.A.

<sup>‡</sup>David R. Cheriton School of Computer Science  
University of Waterloo, Canada

<sup>b</sup>Faculty of Computer Science  
Free University of Bozen-Bolzano, Italy

**Abstract.** We introduce a description language for specifying partial ordering relations over concept descriptions in description logics, and show how the language can be used in combination with binary trees to efficiently search a database that corresponds to a finite set of concept descriptions. The language consists of a pair of *ordering constructors* that support a form of exogenous indexing in which search criteria is independent of data, and a form of endogenous indexing in which the data itself provides search criteria. Our language can be refined in the same way as a description logic in that greater expressiveness and consequent richer search capability is achieved by adding additional ordering constructors.

### 1 Introduction

In earlier work, Stanchev and Weddell have shown how description logics can play a role in searching among objects in an object-oriented database [1]. In this paper, we introduce a description language deriving from their notion of an *extended index* that can be used to specify what we call *ordering descriptions*. Such descriptions correspond to strict partial orders over concept descriptions in a given description logic, in this paper  $\mathcal{ALCQ}(\mathcal{D})$ . We also show how an ordering description can be used in combination with binary trees to efficiently query a database of  $\mathcal{ALCQ}(\mathcal{D})$  concept descriptions.

To illustrate, consider the case of an online supplier of photography equipment. As part of a web presence, the supplier maintains a binary tree of (descriptions of) items available for purchase, and maintains the tree in such a way that a traversal of the tree will visit items in a sequence compatible with a partial order defined by the following ordering description:

$$ProductCode : SaleItem(DisPrice : Un, RegPrice : Un).$$

Intuitively, the ordering description specifies a non-descending major sort on the concrete feature *ProductCode*. Items having the same value for this feature then appear in two consecutive groups. The first group consists of items on sale

that in turn occur in non-descending order of their discount price. The second group consists of items not on sale that in turn occur in non-descending order of their regular price. The supplier also maintains a terminology that includes the following constraints:

$$\begin{aligned} \text{SaleItem} &\sqsubseteq (\geq 3 \text{Suppliers } \top), \\ \text{SaleItem} &\sqsubseteq (\text{Price} = \text{DisPrice}), \\ \neg\text{SaleItem} &\sqsubseteq (\text{Price} = \text{RegPrice}), \text{ and} \\ &\top \sqsubseteq (\text{DisPrice} < \text{RegPrice}). \end{aligned}$$

Now consider a request by an online user for *all descriptions of single-sourced digital cameras, retrieved in non-descending order of their price*. This query can be captured in our formalism by a concept description/ordering description pair as follows:

$$\langle \text{ProductCode} = \text{“digicam”} \sqcap \neg(\geq 2 \text{Suppliers } \top), \text{Price} : \text{Un} \rangle.$$

Our results enable a procedure to reason that the tree maintained by the supplier refines the sort order for the query. To paraphrase, the query can be evaluated by an in-order search of the tree only, during which items will be returned in the order requested by the user. However, to avoid sorting, items will need to satisfy a property called *descriptive sufficiency* that relates to the above ordering description for the search tree, e.g., that each item description “supplies” a value for *ProductCode*. Note that this property will also make it possible to perform arbitrary rotations to ensure that the tree is balanced following the insertion of a new item.

Suppose the supplier decides that being single-sourced is a necessary condition for an item to not be on sale. Suppose in particular that the supplier adds the following constraint to the terminology:

$$\neg\text{SaleItem} \sqsubseteq \neg(\geq 2 \text{Suppliers } \top).$$

Our results enable a further procedure to reason that the tree maintained by the supplier will now *support* the above query. This means not only that the same in-order search of the tree will suffice, but also that the number of subsumption checks in  $\mathcal{ALCQ}(\mathcal{D})$  will be bounded both by the size of the result and by the logarithm (base two) of the number of items occurring in the tree.

The example illustrates the use of the two kinds of *ordering constructors* that make up our initial version of this ordering language. The separation of sale and non-sale items is an example of exogenous indexing in which search criteria is independent of data, while the major sort on *ProductCode* and minor independent sorts on *DisPrice* and *RegPrice* are examples of endogenous indexing in which the data itself provides search criteria. However, the language can be refined in the same way as a description logic in that greater expressiveness and consequent richer search capability becomes possible by adding additional ordering constructors. We suggest examples in our summary comments. Also, the exogenous constructor imposes no conditions on the selection of a particular

dialect of description logic, while the endogenous constructor requires only that the dialect has a linearly ordered concrete domain.

The remainder of the paper is organized as follows. A formal definition of our ordering language, a refinement relationship among ordering descriptions, and a study of some of their properties follows in Section 2. We also include a REF procedure for reasoning about refinement relationships. Although not known to be a complete reasoner at this time, REF is easily able to recognize the above example case regarding sort order. In Section 3, we show how ordering descriptions define pruning criteria during search in a binary tree of concept descriptions. To ensure efficiency, such criteria will depend on the above-mentioned notion of descriptive sufficiency for concept descriptions occurring in the tree. Our main results complete the section in which we characterize *supported queries* for a given ordering condition and terminology. A summary and discussion follow in Section 4.

## 2 Ordering Descriptions

Our language for specifying partial orders over concept descriptions depends on the choice of underlying description logic. In this paper, we use  $\mathcal{ALCQ}(\mathcal{D})$ , a dialect that satisfies our illustrative requirements. However, our results apply to any choice of description logic that includes a linearly-ordered concrete domain.

**Definition 1 (Description Logic  $\mathcal{ALCQ}(\mathcal{D})$ )** *Let  $\{C, C_1, \dots\}$ ,  $\{R, S, \dots\}$ ,  $\{f, g, \dots\}$  and  $\{k, k_1, \dots\}$  denote sets of primitive concept names, roles, concrete features, and constants respectively. A concept description is then defined by the grammar:*

$$D, E ::= f < g \mid f < k \mid C \mid D \sqcap E \mid \neg D \mid (\geq n R D).$$

An inclusion dependency is an expression of the form  $D \sqsubseteq E$ . A terminology  $\mathcal{T}$  is a finite set of inclusion dependencies.

An interpretation  $\mathcal{I}$  is a 3-tuple  $\langle \Delta_I, \Delta_C, \cdot^{\mathcal{I}} \rangle$  where  $\Delta_I$  is an arbitrary abstract domain,  $\Delta_C$  a linearly ordered concrete domain, and  $\cdot^{\mathcal{I}}$  an interpretation function that maps each concrete feature  $f$  to a total function  $f^{\mathcal{I}} : \Delta_I \rightarrow \Delta_C$ , each role  $R$  to a relation  $R^{\mathcal{I}} \subseteq \Delta_I \times \Delta_I$ , each primitive concept  $C$  to a set  $C^{\mathcal{I}} \subseteq \Delta_I$ , the  $<$  symbol to the binary relation for the linear order on  $\Delta_C$ , and  $k$  to a constant in  $\Delta_C$ . The interpretation function is extended to arbitrary concepts in the standard way.

An interpretation  $\mathcal{I}$  satisfies an inclusion dependency  $D \sqsubseteq E$  if  $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$ .  $\mathcal{T} \models D \sqsubseteq E$  if  $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$  for all interpretations  $\mathcal{I}$  that satisfy all inclusion dependencies in  $\mathcal{T}$ .

For the remainder of the paper, we also use standard abbreviations, e.g.,  $D \sqcup E$  for  $\neg(\neg D \sqcap \neg E)$ , as well as the derived comparisons  $\leq, >, \geq$ , and  $=$  on the concrete domain.

**Notation 2** We write  $D^*$  to denote a description obtained from  $D$  by replacing all features  $f$  by  $f^*$ , roles  $R$  by  $R^*$ , and concepts  $C$  by  $C^*$ , and extend this notation in the obvious way to apply to inclusion dependencies and terminologies.

A formal definition of our description language for specifying partial orders over concept descriptions in  $\mathcal{ALCQ}(\mathcal{D})$  now follows. As our introductory comments illustrate, we use the language in the next section for two purposes:

1. to define the relative positions of descriptions occurring in a search tree, and
2. as part of a query specifying the order in which such descriptions are to be presented to a user.

**Definition 3 (Ordering Description)** Let  $D$  be an  $\mathcal{ALCQ}(\mathcal{D})$  concept description, and  $f$  a concrete feature. An ordering description is defined by the grammar:

$$Od ::= Un \mid f : Od \mid D(Od, Od).$$

An instance of the first (resp. second and third) production is called the null ordering (resp. feature value ordering and description ordering).

For a given terminology  $\mathcal{T}$  and concept descriptions  $D$  and  $E$ ,  $D$  is ordered before  $E$  by ordering description  $Od$  with respect to  $\mathcal{T}$ , denoted  $(Od)_{\mathcal{T}}(D, E)$ , if  $\mathcal{T} \not\models D \sqsubseteq \perp$ ,  $\mathcal{T} \not\models E \sqsubseteq \perp$ , and at least one of the following conditions holds:

- $Od = “f : Od_1”$  and  $(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq (f < f^*)$ ,
- $Od = “f : Od_1”$ ,  $(Od_1)_{\mathcal{T}}(D, E)$  and  $(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq (f = f^*)$ ,
- $Od = “D'(Od_1, Od_2)”$ ,  $\mathcal{T} \models D \sqsubseteq D'$  and  $\mathcal{T} \models E \sqsubseteq \neg D'$ ,
- $Od = “D'(Od_1, Od_2)”$ ,  $(Od_1)_{\mathcal{T}}(D, E)$  and  $\mathcal{T} \models (D \sqcup E) \sqsubseteq D'$ , or
- $Od = “D'(Od_1, Od_2)”$ ,  $(Od_2)_{\mathcal{T}}(D, E)$  and  $\mathcal{T} \models (D \sqcup E) \sqsubseteq \neg D'$ .

Two descriptions  $D$  and  $E$  are said to be incomparable with respect to an ordering  $Od$  and terminology  $\mathcal{T}$  if  $\neg(Od)_{\mathcal{T}}(D, E)$  and  $\neg(Od)_{\mathcal{T}}(E, D)$ , or simply incomparable when  $Od$  and  $\mathcal{T}$  are clear from context.

Note that the null ordering denotes an unspecified or unknown ordering, and is used to capture circumstances when no (possibly residual) ordering relationship between descriptions is either sensible or needed.

Important properties of ordering descriptions are given by the following lemma.

**Lemma 4** For any terminology  $\mathcal{T}$ , ordering description  $Od$ , and concept descriptions  $D_1, D_2$ , and  $D_3$ :

1. If  $(Od)_{\mathcal{T}}(D_1, D_2)$ , then  $\neg(Od)_{\mathcal{T}}(D_2, D_1)$ ;
2. If  $(Od)_{\mathcal{T}}(D_1, D_2)$  and  $(Od)_{\mathcal{T}}(D_2, D_3)$ , then  $(Od)_{\mathcal{T}}(D_1, D_3)$ ;
3. If  $(Od)_{\mathcal{T}}(D_1, D_2)$ , then  $\mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \perp$ ;
4. If  $(Od)_{\mathcal{T}}(D_1, D_2)$ ,  $\mathcal{T} \models D_3 \sqsubseteq D_1$  and  $\mathcal{T} \not\models D_3 \sqsubseteq \perp$ , then  $(Od)_{\mathcal{T}}(D_3, D_2)$ ;  
and
5. If  $(Od)_{\mathcal{T}}(D_1, D_2)$ ,  $\mathcal{T} \models D_3 \sqsubseteq D_2$  and  $\mathcal{T} \not\models D_3 \sqsubseteq \perp$ , then  $(Od)_{\mathcal{T}}(D_1, D_3)$ .

Properties 1 and 2 establish the basic requirement that any ordering description will define a strict (or irreflexive) partial order over descriptions in  $\mathcal{ALCQ}(\mathcal{D})$ . It turns out that these properties are sufficient conditions for pruning subtrees during search.

We now introduce the notion of *order refinement* that can be used, for example, to characterize order optimization—to formally define when sorting can be avoided when evaluating a query. Some properties of order refinement and an outline of procedures for reasoning about order refinement then follow.

**Definition 5 (Order Refinement)** *Assume a given terminology  $\mathcal{T}$ , concept description  $D$  and pair of ordering descriptions  $Od_1$  and  $Od_2$ . Then,  $Od_1$  refines  $Od_2$  with respect to  $\mathcal{T}$  and  $D$ , written  $Od_1 \prec_{\mathcal{T},D} Od_2$ , if, for all concept descriptions  $E_1$  and  $E_2$  such that  $\mathcal{T} \models (E_1 \sqcup E_2) \sqsubseteq D$ :*

$$(Od_2)_{\mathcal{T}}(E_1, E_2) \text{ implies } (Od_1)_{\mathcal{T}}(E_1, E_2).$$

$Od_1$  is equivalent to  $Od_2$  with respect to  $\mathcal{T}$  and  $D$ , written  $Od_1 \approx_{\mathcal{T},D} Od_2$ , when  $Od_1 \prec_{\mathcal{T},D} Od_2$  and  $Od_2 \prec_{\mathcal{T},D} Od_1$ . In all cases,  $D$  is called a parameter description.

Consider again an application in order optimization. One can, for example, avoid sorting a query result if the order in which the descriptions are retrieved is the same as the sort order specified by the query. Our notion of order refinement, however, is much more general, and the procedures that are outlined below for reasoning about order refinement can easily handle the case given in our introductory comments.

To further illustrate some basic capabilities of these procedures, assume that the order in which descriptions occur in a binary tree are defined by the ordering description “ $f : g : \text{Un}$ ”, in particular, that an in-order traversal of the tree satisfies a major sort on concrete feature  $f$  and a minor sort on concrete feature  $g$ . Also assume that a user submits a query with the sort order “ $f : \text{Un}$ ”. The procedures will deduce an order refinement between these ordering descriptions. Should the query also stipulate that any retrieved description should be subsumed by the concept description “ $f = 27$ ”, then the procedures will also deduce an order refinement between “ $f : g : \text{Un}$ ” and “ $g : \text{Un}$ ”. Further details on query evaluation will be given in Section 3. Also note that an ability to reason about order refinement will have many related applications, e.g., in query optimization [2] and in index selection [1].

The following lemma establishes a number of equivalence properties of ordering descriptions that are independent of parameter descriptions.

**Lemma 6** *For any terminology  $\mathcal{T}$ , ordering descriptions  $Od_1, Od_2$ , and  $Od_3$  and concept descriptions  $D_1$  and  $D_2$ :*

1. If  $\mathcal{T} \models D_2 \sqsubseteq D_1$ , then  $(D_2(Od_1, D_1(Od_2, Od_3))) \approx_{\mathcal{T},\top} (D_1(D_2(Od_1, Od_2), Od_3))$ ;
2. If  $\mathcal{T} \models D_1 \sqsubseteq D_2$ , then  $(D_1(D_2(Od_1, Od_2), Od_3)) \approx_{\mathcal{T},\top} (D_1(Od_1, Od_3))$ ;
3. If  $\mathcal{T} \models D_1 \sqsubseteq \neg D_2$ , then  $(D_1(D_2(Od_1, Od_2), Od_3)) \approx_{\mathcal{T},\top} (D_1(Od_2, Od_3))$ ;

$$\begin{aligned}
 \text{REF}(Od, \text{Un}, \mathcal{T}, D) &= \text{true}. \\
 \text{REF}(\text{Un}, g : Od, \mathcal{T}, D) &= (\mathcal{T} \models D \sqsubseteq (g = k) \text{ for some } k, \text{ and } \text{REF}(\text{Un}, Od, \mathcal{T}, D)). \\
 \text{REF}(\text{Un}, D'(Od_1, Od_2), \mathcal{T}, D) &= (\mathcal{T} \models D \sqsubseteq D' \text{ and } \text{REF}(\text{Un}, Od_1, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq \neg D' \text{ and } \text{REF}(\text{Un}, Od_2, \mathcal{T}, D)). \\
 \text{REF}(f : Od', g : Od, \mathcal{T}, D) &= (\mathcal{T} \models D \sqsubseteq (f = g) \text{ and } \text{REF}(Od', Od, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq (f = k) \text{ for some } k, \text{ and } \text{REF}(Od', g : Od, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq (g = k) \text{ for some } k, \text{ and } \text{REF}(f : Od', Od, \mathcal{T}, D)). \\
 \text{REF}(f : Od, D'(Od_1, Od_2), \mathcal{T}, D) &= \\
 &\quad (\mathcal{T} \models D \sqsubseteq (f = k) \text{ for some } k, \text{ and } \text{REF}(Od, D'(Od_1, Od_2), \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq \neg D' \text{ and } \text{REF}(f : Od, Od_2, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq D' \text{ and } \text{REF}(f : Od, Od_1, \mathcal{T}, D)). \\
 \text{REF}(D'(Od_1, Od_2), g : Od, \mathcal{T}, D) &= \\
 &\quad (\mathcal{T} \models D \sqsubseteq (g = k) \text{ for some } k, \text{ and } \text{REF}(D'(Od_1, Od_2), Od, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq \neg D' \text{ and } \text{REF}(Od_2, g : Od, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq D' \text{ and } \text{REF}(Od_1, g : Od, \mathcal{T}, D)). \\
 \text{REF}(D_1(Od_1, Od_2), D_2(Od_3, Od_4), \mathcal{T}, D) &= \\
 &\quad (\mathcal{T} \models D \sqsubseteq D_2 \text{ and } \text{REF}(D_1(Od_1, Od_2), Od_3, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq \neg D_2 \text{ and } \text{REF}(D_1(Od_1, Od_2), Od_4, \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq D_1 \text{ and } \text{REF}(Od_1, D_2(Od_3, Od_4), \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models D \sqsubseteq \neg D_1 \text{ and } \text{REF}(Od_2, D_2(Od_3, Od_4), \mathcal{T}, D)) \text{ or} \\
 &\quad (\mathcal{T} \models (D \sqcap D_1) \equiv (D \sqcap D_2) \text{ and} \\
 &\quad \quad \text{REF}(Od_1, Od_3, \mathcal{T}, D \sqcap D_1 \sqcap D_2) \text{ and} \\
 &\quad \quad \text{REF}(Od_2, Od_4, \mathcal{T}, D \sqcap \neg D_1 \sqcap \neg D_2)).
 \end{aligned}$$

**Fig. 1.** AN APPROXIMATE STRUCTURAL REFINEMENT PROCEDURE.

4. If  $\mathcal{T} \models D_2 \sqsubseteq D_1$ , then  $(D_1(Od_1, D_2(Od_2, Od_3))) \approx_{\mathcal{T}, \top} (D_1(Od_1, Od_3))$ ;
5. If  $\mathcal{T} \models \neg D_1 \sqsubseteq D_2$ , then  $(D_1(Od_1, D_2(Od_2, Od_3))) \approx_{\mathcal{T}, \top} (D_1(Od_1, Od_2))$ ;
6. If  $\mathcal{T} \models D \sqsubseteq (f < k)$  and  $\mathcal{T} \models \neg D \sqsubseteq \neg(f < k)$ , for some  $k \in \Delta_C$ , then  $f : D(Od_1, Od_2) \approx_{\mathcal{T}, \top} D(f : Od_1, f : Od_2)$ ; and
7. If  $\mathcal{T} \models (f = g)$  and  $Od_3 = “g : Od_2”$ , then  $f : Od_1 \approx_{\mathcal{T}, \top} f : Od_1[Od_3/Od_2]$  for any occurrence of  $Od_3$  in  $Od_1$ .

Observe that the first is an associativity condition for nested instances of the description ordering constructor. This allows balancing a large number of occurrences of this constructor that might hypothetically comprise a (very large) ordering description.

It is possible to define a canonical form for ordering descriptions based on the above properties only, and to devise an effective procedure for computing this form, say  $\text{CAN}(Od, \mathcal{T})$ , by a careful search for constants  $k$ , by orienting the

equations for the first five properties, and so on. To determine if  $Od_1 \prec_{\mathcal{T},D} Od_2$ , we present the sound structural algorithm REF ( $Od_1, Od_2, \mathcal{T}, D$ ) in Figure 1.

**Lemma 7** *For any terminology  $\mathcal{T}$ , ordering descriptions  $Od_1$  and  $Od_2$ , and concept description  $D$ , it follows that  $Od_1 \prec_{\mathcal{T},D} Od_2$  holds if*

$$\text{REF}(\text{CAN}(Od_1, \mathcal{T}), \text{CAN}(Od_2, \mathcal{T}), \mathcal{T}, D).$$

### 3 Indexing Descriptions

Our goal in this section is to show how ordering descriptions can be used to efficiently search an index consisting of a finite collection of concept descriptions in  $\mathcal{ALCQ}(\mathcal{D})$ . We begin with a formal definition of an underlying tree for an index.

**Definition 8 (Description Tree)** *Let  $D$  denote an arbitrary concept description in  $\mathcal{ALCQ}(\mathcal{D})$ . A description tree is an ordered rooted binary tree conforming to the grammar:*

$$Tr, L, R ::= \langle \rangle \mid \langle D, L, R \rangle.$$

*An instance of the first production denotes an empty tree, while an instance of the second production denotes a node at the root of a tree with left subtree  $L$ , right subtree  $R$ , and labelled by  $D$ . We write  $\langle D, L, R \rangle \in Tr$  if  $\langle D, L, R \rangle$  is a node occurring in  $Tr$ , and call any tree of the form  $\langle D, \langle \rangle, \langle \rangle$  a leaf node.*

*A description tree  $Tr$  is well formed for ordering description  $Od$  with respect to terminology  $\mathcal{T}$  if, for all  $\langle D, L, R \rangle \in Tr$ ,*

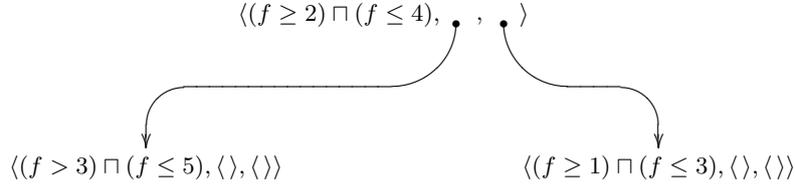
- $\mathcal{T} \not\models D \sqsubseteq \perp$ ,
- $\neg(Od)_{\mathcal{T}}(D, D')$  for all  $\langle D', L', R' \rangle \in L$ , and
- $\neg(Od)_{\mathcal{T}}(D', D)$  for all  $\langle D', L', R' \rangle \in R$ .

*When  $Od$  and  $\mathcal{T}$  are clear from context, we say simply that  $Tr$  is well formed.*

For a given ordering description  $Od$ , the conditions for  $Tr$  to be well formed provide the invariants for insertions of new nodes. For example, when inserting a new node for description  $D'$  in description tree  $\langle D, L, R \rangle$ , a new leaf node  $\langle D', \langle \rangle, \langle \rangle$  must be added in subtree  $L$  if  $(Od)_{\mathcal{T}}(D', D)$ .

**Definition 9 (Description Index)** *Let  $\mathcal{T}$  be a terminology,  $Od$  an ordering description, and  $Tr$  a well formed description tree with respect to  $Od$  and  $\mathcal{T}$ . A description index is a 3-tuple  $\langle Tr, Od, \mathcal{T} \rangle$ .*

We consider queries  $Q$  of the form  $\langle D_Q, Od_Q \rangle$ , where  $D_Q$  is a concept description in  $\mathcal{ALCQ}(\mathcal{D})$  and  $Od_Q$  is an ordering description. A user presumes that query  $Q$  is evaluated with respect to an index  $\langle Tr, Od, \mathcal{T} \rangle$  by first finding all concept descriptions  $E_i$  labelling nodes in  $Tr$  for which  $\mathcal{T} \models E_i \sqsubseteq D_Q$  and then sorting the descriptions  $E_i$  according to  $Od_Q$ . In concrete terms, the first operation is accomplished by standard in-order tree traversal algorithms, assuming the following pruning conditions for a subtree  $\langle E_i, L, R \rangle$  in  $Tr$ :



**Fig. 2.** A WELL FORMED DESCRIPTION TREE WITH RESPECT TO “ $f : \text{Un}$ ”.

1. prune  $L$  if  $(Od)_{\mathcal{T}}(E_i, D_Q)$ , and
2. prune  $R$  if  $(Od)_{\mathcal{T}}(D_Q, E_i)$ .

If  $\mathcal{T} \models E_i \sqsubseteq D_Q$  then  $E_i$  is included in the query result.

The correctness of this procedure is a simple consequence of the following lemma showing that no query result can exist in a pruned subtree. Note that its proof is a straightforward consequence of Lemma 4.

**Lemma 10** For any description index  $\langle Tr, Od, \mathcal{T} \rangle$ , node  $\langle D, L, R \rangle \in Tr$ , and concept description  $E$ :

1.  $(Od)_{\mathcal{T}}(E, D)$  implies  $\mathcal{T} \not\models D' \sqsubseteq E$  for any node  $\langle D', L', R' \rangle \in R$ , and
2.  $(Od)_{\mathcal{T}}(D, E)$  implies  $\mathcal{T} \not\models D' \sqsubseteq E$  for any node  $\langle D', L', R' \rangle \in L$ .

Unfortunately, the conditions for a description tree to be well formed are not strong enough to prevent a concept description labelling a  $Tr$  node to be ordered by  $Od$  prior to a concept description labelling a previous node in an in-order traversal of  $Tr$ . Thus, there is no guarantee that the descriptions returned during an in-order traversal will satisfy the ordering description for the index. Worse, rotations to ensure balance in  $Tr$  will not in general be possible. The following example illustrates these problems.

**Example 11** Consider the description index  $\langle Tr, f : \text{Un}, \emptyset \rangle$  in which  $Tr$  consists of the three nodes illustrated in Figure 2. Note that  $Tr$  is well formed since each description is satisfiable and since the root node is incomparable to both of the child nodes. If one considers a query that retrieves all descriptions, then descriptions will be retrieved out of order by an in-order traversal since the right child compares *left* of the left child with respect to  $f : \text{Un}$ .

Now consider what happens when a user submits a query  $Q$  for which  $D_Q$  is the same as the description labelling the right child, and when  $Tr$  itself is rotated right to make the left child the new root node. In this new circumstance, the above procedure for evaluating a query, in particular the strategy for pruning, is now incorrect in the sense that the description labelling the rightmost node will not be returned. This problem is caused by the rotation producing a description tree that is not well formed.

One way to overcome these problems is to introduce limitations on the concept descriptions that can label nodes in a description tree.

**Definition 12 (Descriptive Sufficiency)** *A concept description  $D$  is sufficiently descriptive for ordering description  $Od$  with respect to terminology  $\mathcal{T}$ , written  $SD_{\mathcal{T}}(D, Od)$ , if at least one of the following conditions hold:*

- $Od = \text{“Un”}$ ,
- $Od = \text{“}f : Od_1\text{”}$ ,  $SD_{\mathcal{T}}(D, Od_1)$ , and  $\mathcal{T} \models D \sqsubseteq (f = k)$ ,
- $Od = \text{“}D'(Od_1, Od_2)\text{”}$ ,  $SD_{\mathcal{T}}(D, Od_1)$ , and  $\mathcal{T} \models D \sqsubseteq D'$ , or
- $Od = \text{“}D'(Od_1, Od_2)\text{”}$ ,  $SD_{\mathcal{T}}(D, Od_2)$ , and  $\mathcal{T} \models D \sqsubseteq \neg D'$ ,

for some  $k \in \Delta_C$ . When  $Od$  and  $\mathcal{T}$  are clear from context, we say simply that  $D$  is sufficiently descriptive.

Again, it is possible to devise an effective procedure for deciding if  $SD_{\mathcal{T}}(D, Od)$  holds, primarily by reusing the careful search for constants  $k$  presumed by the CAN procedure mentioned at the end of the previous section.

Descriptive sufficiency now enables us to say when rotations can be used to balance a description tree  $Tr$ .

**Lemma 13** *Let  $Od$  be an ordering description and  $\mathcal{T}$  a terminology. If concept descriptions  $D_1$  and  $D_2$  are sufficiently descriptive, then, for any description trees  $Tr_1$ ,  $Tr_2$ , and  $Tr_3$  that are well formed,  $\langle D_1, \langle D_2, Tr_1, Tr_2 \rangle, Tr_3 \rangle$  is well formed if and only if  $\langle D_2, Tr_1, \langle D_1, Tr_2, Tr_3 \rangle \rangle$  is well formed.*

Lemma 14 that follows defines the additional properties of ordering descriptions that are needed to ensure the above procedure for searching a description index will return descriptions in non-descending order of the ordering description for the index. A proof of this is now a simple consequence of this lemma together with Lemma 4. As we show in the remainder of this section, it also becomes possible to ensure that query evaluation can be accomplished very efficiently in terms of the number of calls to a DL reasoner.

**Lemma 14** *Let  $Od$  be an ordering description and  $\mathcal{T}$  a terminology. For any concept description  $D$  and pair of concept descriptions  $E_1$  and  $E_2$  that are incomparable and sufficiently descriptive:*

1. If  $(Od)_{\mathcal{T}}(D, E_1)$ , then  $(Od)_{\mathcal{T}}(D, E_2)$ ; and
2. If  $(Od)_{\mathcal{T}}(E_1, D)$ , then  $(Od)_{\mathcal{T}}(E_2, D)$ .

**Definition 15 (Order Preserving Description Index)** *An index  $\langle Tr, Od, \mathcal{T} \rangle$  is order preserving if any concept description labelling any node in  $Tr$  is sufficiently descriptive.*

To summarize, adding a new concept description  $D$  to an order preserving index  $\langle Tr, Od, \mathcal{T} \rangle$  is only possible if  $D$  is sufficiently descriptive, and is accomplished by adding  $\langle D, \langle \rangle, \langle \rangle \rangle$  as a new leaf node in  $Tr$  and then performing rotations

to ensure balance. If  $\langle D', L, R \rangle \in Tr$ , then the new node must be inserted in  $L$  when  $(Od)_{\mathcal{T}}(D, D')$  holds, in  $R$  if  $(Od)_{\mathcal{T}}(D', D)$  holds, and in either  $L$  or  $R$  otherwise.

Our main result now follows in which we characterize a number of the standard cases of queries for which an initial search followed by an index scan (and without a subsequent sort) will suffice to efficiently evaluate the query.

**Definition 16 (Supported Query)** *A description  $D$  is sufficiently selective for ordering description  $Od$  with respect to terminology  $\mathcal{T}$ , denoted  $SS_{\mathcal{T}}(D, Od)$ , if at least one of the following conditions hold:*

- $\mathcal{T} \models \top \sqsubseteq D$ ,
- $Od = “f : Od_1”$ ,  $SS_{\mathcal{T}}(E, Od_1)$  and  $\mathcal{T} \models D \equiv ((f = k) \sqcap E)$ ,
- $Od = “f : Od_1”$  and  $\mathcal{T} \models D \equiv (f < k)$ ,
- $Od = “f : Od_1”$  and  $\mathcal{T} \models D \equiv \neg(f < k)$ ,
- $Od = “f : Od_1”$  and  $\mathcal{T} \models D \equiv (\neg(f < k) \sqcap (f < k'))$ ,
- $Od = “D'(Od_1, Od_2)”$ ,  $SS_{\mathcal{T}}(E, Od_1)$  and  $\mathcal{T} \models D \equiv (D' \sqcap E)$ ,
- $Od = “D'(Od_1, Od_2)”$ ,  $SS_{\mathcal{T}}(E, Od_2)$  and  $\mathcal{T} \models D \equiv (\neg D' \sqcap E)$ , or
- $\mathcal{T} \models D \equiv (E \sqcup E')$ ,  $SS_{\mathcal{T}}(E, Od)$  and  $SS_{\mathcal{T}}(E', Od)$ ,

for some constants  $k$  and  $k'$  in  $\Delta_C$  and descriptions  $E$  and  $E'$ .

A query  $\langle D_Q, Od_Q \rangle$  is supported by an order preserving description index  $\langle Tr, Od', \mathcal{T} \rangle$  if and only if  $D_Q$  is sufficiently selective for ordering description  $Od'$  with respect to  $\mathcal{T}$ , and  $Od' \prec_{\mathcal{T}, D_Q} Od_Q$ .

A procedure for deciding if the concept description of a query is sufficiently selective is an open problem at this time. However, an approximate procedure easily capable of recognizing our introductory example is straightforward, e.g., one that uses the above-mentioned careful search for constants  $k$  and  $k'$  to recognize the range query cases.

**Theorem 17** *Let  $\langle Tr, Od, \mathcal{T} \rangle$  be an order preserving description index and  $Q$  a supported query. Then  $Q$  can be evaluated in  $O(\log(n) + k)$  subsumption tests of  $\mathcal{ALCQ}(\mathcal{D})$ , where  $n$  is the number of nodes in  $Tr$  and  $k$  is the number of descriptions in the result.*

## 4 Summary and Discussion

We have proposed a language for specifying partial orders over concept descriptions that consists of an initial selection of two ordering constructors. The first is called feature value ordering, and is the standard notion of ordering supported by SQL and relational databases. The second is called description ordering, and can be viewed as a way of capturing indexing based on grid file techniques in which the focus is on organizing the data space in which data resides [3]. Multidimensional indices such as quad trees [4] are examples.

There are also a number of other ordering constructors that one might consider. Two possibilities for additional endogenous indexing are given by the following productions for our ordering language.

$$Od ::= f \text{ desc} : Od \mid \sqsubseteq$$

The first is an obvious extension that would enable (sub)orders satisfying non-ascending values for concrete features in descriptions, while the second appeals directly to subsumption checking in a description logic. More formally, the necessary revision to Definition 3 requires adding three conditions:

- $Od = “f \text{ desc} : Od_1”$  and  $(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq (f^* < f)$ ,
- $Od = “f \text{ desc} : Od_1”$ ,  $(Od_1)_{\mathcal{T}}(D, E)$  and  $(\mathcal{T} \cup \mathcal{T}^*) \models (D \sqcap E^*) \sqsubseteq (f = f^*)$ ,
- or
- $Od = “\sqsubseteq”$  and  $\mathcal{T} \models D \sqsubseteq E$ .

For the first constructor, it is also straightforward to extend proofs for Lemmas 4 and 10, and to extend the definition of description sufficiency in a way that will preserve Lemmas 13 and 14, which will then allow arbitrary rotations in a description tree and the possibility of removing a sort operator from a query plan, respectively. The same is not true, however, for the second constructor. In this case, Property 5 of Lemma 4 and Property 2 of Lemma 10 will no longer hold. Thus, pruning during search will only remain possible for right subtrees in a description index. There is also no obvious way to repair the definition of descriptive sufficiency in a way that will also preserve Lemmas 13 and 14.

We have demonstrated how our ordering language can support retrieving descriptions in response to queries. In an algebraic sense, the query language we have considered is very simple, consisting only of a selection operation for finding descriptions subsumed by a given “selection” concept, and a sort operation for ordering the set of descriptions produced by selection. DL systems such as Racer [5] that implement ABox reasoning already support the first of these operations. We believe our results provide some guidance on how DL systems can incorporate better support for sorting, for order optimization in ABox querying, and for ABox indexing.

There are several open problems and many possible avenues of further research. Finding a complete REF procedure and either adding further operators to our query language or a sort capability to existing query languages such as EQL [6] are respective examples. Finally, an expanded version of this paper containing complete proofs is available as a technical report [7].

### Acknowledgments

We thank Peter Tarle and Nortel for many valuable discussions on this work and for financial support. We also acknowledge the financial support in part by grants from NSERC, Canada.

## References

1. Stanchev, L., Weddell, G.: Index Selection for Embedded Control Applications using Description Logics. In: Description Logics 2003, CEUR-WS vol.81 (2003) 9–18
2. Simmen, D.E., Shekita, E.J., Malkemus, T.: Fundamental Techniques for Order Optimization. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. (1996) 57–67
3. Nievergelt, J., Hinterberger, H., Sevcik, K.C.: The grid file: An adaptable, symmetric multikey file structure. *ACM Trans. Database Syst.* **9**(1) (1984) 38–71
4. Samet, H.: The quadtree and related hierarchical data structures. *ACM Comput. Surv.* **16**(2) (1984) 187–260
5. Volker Haarslev and Ralf Moller: Racer system description. In: International Joint Conference on Automated Reasoning, IJCAR. (2001)
6. Diego Calvanese and Giuseppe De Giacomo and Domenico Lembo and Maurizio Lenzerini and Riccardo Rosati: Epistemic First-Order Queries over Description Logic Knowledge Bases. In: Description Logics 2006, CEUR-WS vol.189 (2006)
7. Pound, J., Stanchev, L., Toman, D., Weddell, G.: On ordering descriptions in a description logic. Technical Report CS-2007-16, David R. Cheriton School of Computer Science, University of Waterloo (2007)

## Deciding $\mathcal{ALBO}$ with Tableau

Renate A. Schmidt<sup>1</sup> and Dmitry Tishkovsky<sup>1</sup>

School of Computer Science, The University of Manchester  
`{renate.schmidt,dmitry.tishkovsky}@manchester.ac.uk`

**Abstract.** This paper presents a tableau approach for deciding description logics outside the scope of OWL DL and current state-of-the-art tableau-based description logic systems. In particular, we define a sound and complete tableau calculus for the description logic  $\mathcal{ALBO}$  and show that it provides a basis for decision procedures for this logic and numerous other description logics.  $\mathcal{ALBO}$  is the extension of  $\mathcal{ALC}$  with the Boolean role operators, inverse of roles, domain and range restriction operators and it includes full support for objects (nominals).  $\mathcal{ALBO}$  is a very expressive description logic which is NExpTime complete and subsumes Boolean modal logic and the two-variable fragment of first-order logic. An important novelty is the use of a versatile, unrestricted blocking rule as a replacement for standard loop checking mechanisms implemented in description logic systems. Our decision procedure is implemented in the METTEL system.

### 1 Introduction

The description logic  $\mathcal{ALBO}$  is an extension of the description logic  $\mathcal{ALB}$  introduced in [7] with singleton concepts, called nominals in modal logic.  $\mathcal{ALB}$  is the extension of  $\mathcal{ALC}$ , in which concepts and roles form a Boolean algebra, and additional operators include inverse of roles and a domain restriction operator.  $\mathcal{ALBO}$  extends  $\mathcal{ALC}$  by union of roles, negation of roles, inverse of roles, and domain as well as range restriction. In addition, it provides full support for ABox objects and singleton concepts.

None of the current state-of-the-art tableau-based description logic systems are able to handle  $\mathcal{ALBO}$ . Because  $\mathcal{ALBO}$  allows full negation of roles, it is out of the scope of OWL DL and most description logic systems including FACT++, KAON2, PELLET, and RACERPRO. A tableau decision procedure for the description logic  $\mathcal{ALCQIb}$  which allows for Boolean combinations of ‘safe’ occurrences of negated roles is described in [14]. Safeness essentially implies a ‘guardedness’ property which is violated for unsafe occurrences of role negation. Description logics with full, i.e. safe and unsafe, role negation can be decided however by translation to first-order logic and first-order resolution theorem provers such as MSPASS, SPASS and VAMPIRE. The paper [7] shows that the logic  $\mathcal{ALB}$  can be decided by translation to first-order logic and ordered resolution. This result is extended in [3] to  $\mathcal{ALB}$  with positive occurrences of composition of roles.  $\mathcal{ALBO}$  can be embedded into the two-variable fragment

of first-order logic with equality which can be decided with first-order resolution methods [2]. This means that  $\mathcal{ALBO}$  is decidable and can be decided using first-order resolution methods.

$\mathcal{ALBO}$  is a very expressive description logic. It subsumes the Boolean modal logic [4, 5] and tense, hybrid versions of Boolean modal logic with the @ operator and nominals.  $\mathcal{ALBO}$  can also be shown to subsume the two-variable fragment of first-order logic (without equality) [8]. The following constructs and statements can be handled in  $\mathcal{ALBO}$ .

- Role negation, the universal role, the sufficiency or window operator, Peirce sum, domain restriction, cross product, and cylindrification.
- Role inclusion axioms and role equivalence axioms in the language of  $\mathcal{ALBO}$ .
- Role assertions in the language of  $\mathcal{ALBO}$ .
- Boolean combinations of both concept and role inclusion and equivalence axioms.
- Boolean combinations of concept and role assertions, including negated role assertions.
- Disjoint roles, symmetric roles and serial roles.<sup>1</sup>

Since  $\mathcal{ALBO}$  subsumes Boolean modal logic it follows from [10] that the satisfiability problem in  $\mathcal{ALBO}$  is  $\text{NExpTime-hard}$ . In [6] it is shown that the two variable first-order fragment with equality is  $\text{NExpTime-complete}$ . It follows therefore that the computational complexity of  $\mathcal{ALBO}$ -satisfiability is  $\text{NExpTime-complete}$ . This follows also from a (slight extension of a) result in [14].

In this paper we present a tableau approach which decides the description logic  $\mathcal{ALBO}$ . The tableau calculi we define for  $\mathcal{ALBO}$  are ground semantic tableau calculi which work on ground labelled expressions. In contrast to the tableau calculi for description logics with role operators presented in [3, 11–13] (including  $\mathcal{ALC}(\sqcup, \sqcap, ^{-1})$ ,  $\mathcal{ALC}(\sqcup, \sqcap, ^{-1}, \upharpoonright)$ , and Peirce logic, or the equivalent modal versions, where  $\upharpoonright$  denotes the domain restriction operator), our tableau calculi operate only on ground labelled concept expressions. As a consequence the calculi can be implemented as extensions of existing tableau-based description logic systems which can handle singleton concepts.

In order to limit the number of objects in the tableau we need a mechanism for detecting periodicity in the underlying interpretations (models). *Standard loop checking* mechanisms are based on comparing sets of (labelled or unlabelled) concept expressions such as subset blocking or equality blocking. Instead of using the standard loop checking mechanisms our procedure uses a new inference rule, the *unrestricted blocking* rule, and equality reasoning. Our approach has the following advantages over standard loop checking.

- It is conceptually simple and easy to implement.
- It is universal and does not depend on the notion of a type.

<sup>1</sup> It is not difficult to extend our method and results to include full equality handling including reflexive roles, identity and diversity roles, and the test operator.

- It is versatile and enables more controlled model construction in a tableau procedure. For instance, it can be used to construct small models for a satisfiable concept, e.g. domain minimal models.
- It generalises to other logics, including full first-order logic.
- It can be simulated in first-order logic provers.

The unrestricted blocking rule corresponds to an unrestricted version of the first-order blocking rule invented by [1], simply called the *blocking rule*. The blocking rule is constrained to objects  $l$  and  $l'$  such that the object  $l'$  is a successor of the object  $l$ . I.e. in the common branch of  $l$  and  $l'$  the object  $l'$  is obtained from  $l$  as a result of a sequence of applications of the existential restriction rule. In this form the rule can be used to simulate standard blocking mechanisms.

The structure of the paper is as follows. The syntax and semantics of  $\mathcal{ALBO}$  is defined in Section 2. In Section 3 we prove that  $\mathcal{ALBO}$  has the finite model property. The constructions used in the proof are also used in Section 4, where we define a tableau calculus for  $\mathcal{ALBO}$  and prove that it is sound and complete without the unrestricted blocking rule. Section 5 introduces the (unrestricted) blocking mechanism and proves soundness, completeness and termination of the extended tableau calculus. This allows us to define general decision procedures for  $\mathcal{ALBO}$  and its sublogics which is discussed in Section 6. We conclude with Section 7. Due to lack of space some proofs are omitted or only sketched.

## 2 Syntax and semantics of $\mathcal{ALBO}$

The syntax of  $\mathcal{ALBO}$  is defined over the signature  $\sigma = (O, C, R)$  of three disjoint alphabets:  $O = \{\ell_0, \ell_1, \dots\}$  the alphabet of object symbols,  $C = \{p_0, p_1, \dots\}$  the alphabet of concept symbols, and  $R = \{r_0, r_1, \dots\}$  the alphabet of role symbols. The logical connectives are:  $\neg$ ,  $\sqcup$ ,  $\exists$ ,  $^{-1}$  (role *inverse*),  $\upharpoonright$  (*domain restriction*),  $\downarrow$  (*range restriction*). *Concept expressions* (or *concepts*) and *role expressions* (or *roles*) are defined as follows:

$$\begin{aligned} C &\stackrel{\text{def}}{=} p \mid \{\ell\} \mid \neg C \mid C \sqcup D \mid \exists R.C, \\ R &\stackrel{\text{def}}{=} r \mid R^{-1} \mid \neg R \mid R \sqcup S \mid R \upharpoonright C \mid R \downarrow C. \end{aligned}$$

$p$  ranges over the set  $C$ ,  $\ell$  ranges over  $O$ , and  $r$  ranges over  $R$ . The  $\sqcup$  connective on concepts and roles is defined as usual in terms of  $\neg$  and  $\sqcup$ , and the top and bottom concepts are defined by  $\top \stackrel{\text{def}}{=} p \sqcup \neg p$  and  $\perp \stackrel{\text{def}}{=} p \sqcap \neg p$ , respectively, for some concept name  $p$ . The universal restriction operator  $\forall$  is a dual to the existential restriction operator  $\exists$ , specified by  $\forall R.C \stackrel{\text{def}}{=} \neg \exists R. \neg C$ .

Next, we define the semantics of  $\mathcal{ALBO}$ . A *model* (or an *interpretation*)  $\mathcal{I}$  of  $\mathcal{ALBO}$  is a tuple  $\mathcal{I} = (\Delta^{\mathcal{I}}, p_0^{\mathcal{I}}, \dots, \ell_0^{\mathcal{I}}, \dots, r_0^{\mathcal{I}}, \dots)$ , where  $\Delta^{\mathcal{I}}$  is a non-empty set,  $p_i^{\mathcal{I}}$  is a subset of  $\Delta^{\mathcal{I}}$ ,  $\ell_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ , and  $r_0^{\mathcal{I}}$  is a binary relation over  $\Delta^{\mathcal{I}}$ . The semantics of concepts and roles in the model  $\mathcal{I}$ , i.e.  $C^{\mathcal{I}}$  and  $R^{\mathcal{I}}$ , is specified in Figure 1. A *TBox* (respectively *RBox*), is a (finite) set of inclusion statements  $C \sqsubseteq D$  (respectively  $R \sqsubseteq S$ ) which are interpreted in any model  $\mathcal{I}$  as subset

$$\begin{aligned}
 \{\ell\}^{\mathcal{I}} &\stackrel{\text{def}}{=} \{\ell^{\mathcal{I}}\}, & (R^{-1})^{\mathcal{I}} &\stackrel{\text{def}}{=} (R^{\mathcal{I}})^{-1} = \{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}, \\
 (\neg C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, & (\neg R)^{\mathcal{I}} &\stackrel{\text{def}}{=} (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}, \\
 (C \sqcup D)^{\mathcal{I}} &\stackrel{\text{def}}{=} C^{\mathcal{I}} \cup D^{\mathcal{I}}, & (R \sqcup S)^{\mathcal{I}} &\stackrel{\text{def}}{=} R^{\mathcal{I}} \cup S^{\mathcal{I}}, \\
 (R \upharpoonright C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \{(x, y) \mid x \in C^{\mathcal{I}} \text{ and } (x, y) \in R^{\mathcal{I}}\}, \\
 (R \downharpoonright C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \{(x, y) \mid y \in C^{\mathcal{I}} \text{ and } (x, y) \in R^{\mathcal{I}}\}, \\
 (\exists R.C)^{\mathcal{I}} &\stackrel{\text{def}}{=} \{x \mid \exists y \in C^{\mathcal{I}} (x, y) \in R^{\mathcal{I}}\}.
 \end{aligned}$$

**Fig. 1.** Definition of  $\cdot^{\mathcal{I}}$

relationships, namely  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$  (respectively  $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ ). An *ABox* is a (finite) set of statements of the form  $\ell : C$  or  $(\ell, \ell') : R$ , called concept assertions or role assertions. A *knowledge base* is a tuple  $(T, R, A)$  of a TBox  $T$ , an RBox  $R$ , and an ABox  $A$ .

The *top role*  $\nabla$  and the *empty role*  $\Delta$  are definable in  $\mathcal{ALBO}$  as  $r \sqcup \neg r$  and  $r \sqcap \neg r$ , respectively, for some role symbol  $r$ . As a consequence any concept assertion  $\ell : C$  can be expressed as a concept expression as follows:  $\ell : C \stackrel{\text{def}}{=} \exists \nabla. (\{\ell\} \sqcap C)$ . It is clear that  $(\ell : C)^{\mathcal{I}} = \Delta^{\mathcal{I}}$  iff  $\ell^{\mathcal{I}} \in C^{\mathcal{I}}$  in every model  $\mathcal{I}$ . In  $\mathcal{ALBO}$  a role assertion  $(\ell, \ell') : R$  can also be expressed as a concept assertion and concept expression, namely  $(\ell, \ell') : R \stackrel{\text{def}}{=} \ell : \exists R. \{\ell'\}$ . Moreover, concept and role inclusion axioms are definable as concept expressions, too. We let  $C \sqsubseteq D \stackrel{\text{def}}{=} \forall \nabla. (\neg C \sqcup D)$  and  $R \sqsubseteq S \stackrel{\text{def}}{=} \forall \nabla. \exists \neg (\neg R \sqcup S)$ , respectively. Thus, Boolean combinations of inclusion and assertion statements of concepts and roles are also expressible in  $\mathcal{ALBO}$  as the corresponding Boolean combinations of the concepts which represent these statements. As usual, concept satisfiability in  $\mathcal{ALBO}$  with respect to any knowledge base can be reduced to concept satisfiability with respect to a knowledge base where all TBox, RBox, and ABox are empty. Without loss of generality we therefore focus on the problem of concept satisfiability in  $\mathcal{ALBO}$ .

### 3 Finite model property

Let  $\prec$  be the smallest transitive ordering on the set of all  $\mathcal{ALBO}$  expressions (concepts and roles) satisfying:

- |  |  |  |
|--|--|--|
| (s1) $C \prec \neg C$ ,                | (s7) $R \prec \exists R.C$ ,           | (s13) $\neg C \prec R \upharpoonright C$ ,   |
| (s2) $C \prec C \sqcup D$ ,            | (s8) $\neg C \prec \neg \exists R.C$ , | (s14) $R \prec R \upharpoonright C$ ,        |
| (s3) $D \prec C \sqcup D$ ,            | (s9) $R \prec \neg \exists R.C$ ,      | (s15) $\neg C \prec R \downharpoonright C$ , |
| (s4) $\neg C \prec \neg(C \sqcup D)$ , | (s10) $R \prec R \sqcup S$ ,           | (s16) $R \prec R \downharpoonright C$ ,      |
| (s5) $\neg D \prec \neg(C \sqcup D)$ , | (s11) $S \prec R \sqcup S$ ,           | (s17) $R \prec \neg R$ .                     |
| (s6) $C \prec \exists R.C$ ,           | (s12) $R \prec R^{-1}$ ,               |  |

It is easy to see that  $\prec$  is a well-founded ordering. Let  $\preceq$  be the reflexive closure of  $\prec$ .

Let  $\mathcal{I}$  be any  $\mathcal{ALBO}$  model and  $C$  be a concept. A  $C$ -type  $\tau^C(x)$  of an element  $x$  of the model  $\mathcal{I}$  is defined by  $\tau^C(x) \stackrel{\text{def}}{=} \{D \mid D \preceq C \text{ and } x \in D^{\mathcal{I}}\}$ . Let  $\sim$  be an equivalence relation on  $\Delta^{\mathcal{I}}$  such that  $x \sim y$  implies  $\tau^C(x) = \tau^C(y)$ . Let  $\|x\| \stackrel{\text{def}}{=} \{y \in \Delta^{\mathcal{I}} \mid x \sim y\}$ .

Given a model  $\mathcal{I}$  we define the *filtrated* model  $\bar{\mathcal{I}}$  (through  $\sim$ ), as follows. Let  $\Delta^{\bar{\mathcal{I}}} \stackrel{\text{def}}{=} \{\|x\| \mid x \in \Delta^{\mathcal{I}}\}$ . For every  $r \in R$  let  $r^{\bar{\mathcal{I}}} \stackrel{\text{def}}{=} \{(\|x\|, \|y\|) \mid (x, y) \in r^{\mathcal{I}}\}$ . For every  $p \in C$  let  $p^{\bar{\mathcal{I}}} \stackrel{\text{def}}{=} \{\|x\| \mid x \in p^{\mathcal{I}}\}$  and for every  $\ell \in O$  let  $\ell^{\bar{\mathcal{I}}} \stackrel{\text{def}}{=} \|\ell^{\mathcal{I}}\| = \{\ell^{\mathcal{I}}\}$ .

The following lemma can be proved by induction on the ordering  $\prec$ .

**Lemma 1 (Filtration Lemma).**

- (1)  $x \in D^{\mathcal{I}}$  iff  $\|x\| \in D^{\bar{\mathcal{I}}}$  for any  $D \preceq C$  and  $x \in \Delta^{\mathcal{I}}$ ,
- (2)  $(x, y) \in R^{\mathcal{I}}$  iff  $(\|x\|, \|y\|) \in R^{\bar{\mathcal{I}}}$  for every  $R \prec C$  and  $x, y \in \Delta^{\mathcal{I}}$ .

A corollary of this lemma is the finite model property.

**Theorem 1 (Finite Model Property).**  *$\mathcal{ALBO}$  has the finite model property, i.e. if a concept  $C$  is satisfiable then it has a finite model.*

*Proof.* Let  $\mathcal{I}$  be a model for a concept  $C$  and  $\sim$  is an equivalence relation on  $\Delta^{\mathcal{I}}$  defined by  $x \sim y \iff \tau^C(x) = \tau^C(y)$ . Then, by the Filtration Lemma, the model  $\bar{\mathcal{I}}$  filtrated through  $\sim$  is also a model for  $C$ . Moreover the domain of  $\bar{\mathcal{I}}$  is finite because the number of  $C$ -types in every model is finite.

## 4 Tableau calculus

Let  $T$  denote a tableau calculus and  $C$  a concept. We denote by  $T(C)$  a finished tableau built using the rules of the calculus  $T$  starting with the concept  $C$  as input. I.e. we assume that all branches in the tableau are expanded and all applicable rules of  $T$  have been applied in  $T(C)$ . As usual we assume that all the rules of the calculus are applied *non-deterministically, to a tableau*. A branch of a tableau is *closed* if a contradiction has been derived in this branch, otherwise the branch is called *open*. The tableau  $T(C)$  is *closed* if all its branches are closed and  $T(C)$  is *open* otherwise. We say that  $T$  is *terminating* iff for every concept  $C$  either  $T(C)$  is finite whenever  $T(C)$  is closed or  $T(C)$  has a finite open branch if  $T(C)$  is open.  $T$  is *sound* iff  $C$  is unsatisfiable whenever  $T(C)$  is closed for all concepts  $C$ .  $T$  is *complete* iff for any concept  $C$ ,  $C$  is satisfiable (has a model) whenever  $T(C)$  is open.

Let  $T_{\mathcal{ALBO}}$  be the tableau calculus consisting of the rules listed in Table 1. Given an input concept  $C$ , preprocessing is performed which pushes the role inverse operators toward atomic concepts by exhaustively applying the following role equivalences from left to right.

$$\begin{aligned} (-R)^{-1} &= \neg(R^{-1}), & (R \sqcup S)^{-1} &= R^{-1} \sqcup S^{-1}, \\ (R \uparrow C)^{-1} &= R^{-1} \uparrow C, & (R \downarrow C)^{-1} &= R^{-1} \downarrow C, & (R^{-1})^{-1} &= R. \end{aligned}$$

$(\perp): \frac{\ell : C, \ell : \neg C}{\perp}$ $(\neg\sqcup): \frac{\ell : \neg(C \sqcup D)}{\ell : \neg C, \ell : \neg D}$ $(\text{sym}): \frac{\ell : \{\ell'\}}{\ell' : \{\ell\}} \quad (\text{sym-}): \frac{\ell : \neg\{\ell'\}}{\ell' : \neg\{\ell\}}$ $(\exists): \frac{\ell : \exists R.C}{\ell : \exists R.\{\ell'\}, \ell' : C} (\ell' \text{ is new})$ $(\text{bridge}): \frac{\ell : \exists R.\{\ell'\}, \ell' : \{\ell''\}}{\ell : \exists R.\{\ell''\}}$ $(\exists\sqcup): \frac{\ell : \exists(R \sqcup S).\{\ell'\}}{\ell : \exists R.\{\ell'\} \mid \ell : \exists S.\{\ell'\}}$ $(\exists^{-1}): \frac{\ell : \exists R^{-1}.\{\ell'\}}{\ell' : \exists R.\{\ell\}}$ $(\exists\mid): \frac{\ell : \exists(R \mid C).\{\ell'\}}{\ell : C, \ell : \exists R.\{\ell'\}}$ $(\exists\mid): \frac{\ell : \exists(R \mid C).\{\ell'\}}{\ell' : C, \ell : \exists R.\{\ell'\}}$ $(\exists\neg): \frac{\ell : \exists\neg R.\{\ell'\}}{\ell : \neg\exists R.\{\ell'\}}$	$(\neg\neg): \frac{\ell : \neg\neg C}{\ell : C}$ $(\sqcup): \frac{\ell : (C \sqcup D)}{\ell : C \mid \ell : D}$ $(\text{mon}): \frac{\ell : \{\ell'\}, \ell' : C}{\ell : C} \quad (\text{id}): \frac{\ell : C}{\ell : \{\ell\}}$ $(\neg\exists): \frac{\ell : \neg\exists R.C, \ell : \exists R.\{\ell'\}}{\ell' : \neg C}$ $(\neg\exists\sqcup): \frac{\ell : \neg\exists(R \sqcup S).C}{\ell : \neg\exists R.C, \ell : \neg\exists S.C}$ $(\neg\exists^{-1}): \frac{\ell : \neg\exists R^{-1}.C, \ell' : \exists R.\{\ell\}}{\ell' : \neg C}$ $(\neg\exists\mid): \frac{\ell : \neg\exists(R \mid C).D}{\ell : \neg C \mid \ell : \neg\exists R.D}$ $(\neg\exists\mid): \frac{\ell : \neg\exists(R \mid C).D}{\ell : \neg\exists R.\neg(C \sqcup \neg D)}$ $(\neg\exists\neg): \frac{\ell : \neg\exists\neg R.C, \ell' : D}{\ell : \exists R.\{\ell'\} \mid \ell : \exists\neg R.\{\ell'\}}$
---	--

**Table 1.** Tableau calculus  $T_{\mathcal{ALBO}}$  for  $\mathcal{ALBO}$ .

Next, the (preprocessed) input concept  $C$  is tagged with a fresh object name  $\ell$  which does not occur in  $C$ . Then we build a complete tableau  $T_{\mathcal{ALBO}}(C)$  as usual by applying the rules of  $T_{\mathcal{ALBO}}$  to the concept assertion  $\ell : C$ . It is however important to note that  $\ell : C$  and all labelled expressions and assertions really denote concept expressions.

Because every rule preserves the satisfiability of concept assertions, it is easy to see that the calculus  $T_{\mathcal{ALBO}}$  is sound for  $\mathcal{ALBO}$ .

We turn to proving completeness of the calculus. Suppose that a tableau  $T_{\mathcal{ALBO}}(C)$  for the given concept  $C$  is open, i.e. it contains an open branch  $\mathcal{B}$ . We construct a model  $\mathcal{I}$  for the satisfiability of  $C$  as follows. By definition, let  $\ell \sim \ell' \stackrel{\text{def}}{\iff} \ell : \{\ell'\} \in \mathcal{B}$ . It is clear that the rules (sym), (mon), and (id) ensure that  $\sim$  is an equivalence relation on objects. The equivalence class  $\|\ell\|$  of a representative  $\ell$  is defined as usual by:  $\|\ell\| \stackrel{\text{def}}{=} \{\ell' \mid \ell \sim \ell'\}$ . We set

$$\Delta^{\mathcal{I}} \stackrel{\text{def}}{=} \{\|\ell\| \mid \ell : \{\ell\} \in \mathcal{B}\}, \quad r^{\mathcal{I}} \stackrel{\text{def}}{=} \{(\|\ell\|, \|\ell'\|) \mid \ell : \exists r.\{\ell'\} \in \mathcal{B}\},$$

$$p^{\mathcal{I}} \stackrel{\text{def}}{=} \{\|\ell\| \mid \ell : p \in \mathcal{B}\}, \quad \ell^{\mathcal{I}} \stackrel{\text{def}}{=} \begin{cases} \|\ell\|, & \text{if } \ell : \{\ell\} \in \mathcal{B}, \\ \|\ell'\| & \text{for some } \|\ell'\| \in \Delta^{\mathcal{I}}, \text{ otherwise.} \end{cases}$$

It is easy to show that, using the rules (sym), (mon), and (id), the definition of  $\mathcal{I}$  does not depend on representatives of the equivalence classes.

**Lemma 2.** (1) If  $\ell : D \in \mathcal{B}$  then  $\|\ell\| \in D^{\mathcal{I}}$  for any concept  $D$ .

(2) For every role  $R$  and every concept  $D$

(2a)  $\ell : \exists R.\{\ell'\} \in \mathcal{B}$  implies  $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}}$ ,

(2b) if  $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}}$  and  $\ell : \neg\exists R.D \in \mathcal{B}$  then  $\ell' : \neg D \in \mathcal{B}$ .

*Proof.* We prove both properties simultaneously by induction on the ordering  $\prec$ . The induction hypothesis is: for an arbitrary  $\mathcal{ALBCO}$  expression  $E$  for every expression  $F$  such that  $F \prec E$  if  $F$  is a concept then property (1) holds with  $D = F$ . Otherwise (i.e. if  $F$  is a role), property (2) holds with  $R = F$ . To prove property (1) we consider the following cases.

$D = p, D = \{\ell'\}$ . These cases follow from the definitions of  $p^{\mathcal{I}}$  and  $\ell^{\mathcal{I}}$ .

$D = \neg D_0$ . If  $\ell : \neg D_0 \in \mathcal{B}$  then  $\ell : D_0 \notin \mathcal{B}$  because otherwise  $\mathcal{B}$  would have been closed by the  $(\perp)$  rule. We have the following subcases.

$D_0 = p$ . We have  $\ell : p \in \mathcal{B} \iff \|\ell\| \in p^{\mathcal{I}}$  by the definition of  $p^{\mathcal{I}}$ .

$D_0 = \{\ell'\}$ . As the rules (sym-) and (id) have been applied in  $\mathcal{B}$  it is clear that  $\ell' : \{\ell'\}$  is in  $\mathcal{B}$  and  $\ell'^{\mathcal{I}} = \|\ell'\|$ . Similarly,  $\ell^{\mathcal{I}} = \|\ell\|$ . Furthermore, because  $\ell : \{\ell'\} \notin \mathcal{B}$  we have  $\ell \not\prec \ell'$ , i.e.  $\ell \notin \|\ell'\| = \ell'^{\mathcal{I}}$ . That is,  $\|\ell\| \notin \{\|\ell'\|\} = \{\ell'\}^{\mathcal{I}}$ .

$D_0 = \neg D_1, D_0 = D_1 \sqcup D_2$ . The proofs of these cases are easy.

$D_0 = \exists R.D_1$ . Let  $\|\ell'\|$  be an arbitrary element of  $\Delta^{\mathcal{I}}$  such that  $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}}$  (trivially, if there is no such element then there is nothing to prove).

By the induction hypothesis the property (2b) holds for  $R \prec D_0$ . Thus,  $\ell' : \neg D_1 \in \mathcal{B}$ . The induction hypothesis for the property (1) gives us  $\|\ell'\| \notin D_1^{\mathcal{I}}$ . Finally, we obtain  $\|\ell\| \in (\neg\exists R.D_1)^{\mathcal{I}}$  because  $\ell'$  was chosen arbitrarily.

$D = D_0 \sqcup D_1$ . The proof of this case is easy.

$D = \exists R.D_0$ . If  $\ell : \exists R.D_0 \in \mathcal{B}$  then  $\ell' : D_0 \in \mathcal{B}$  and  $\ell : \exists R.\{\ell'\} \in \mathcal{B}$  for some object  $\ell'$  by the  $(\exists)$  rule. By the induction hypothesis the properties (1) and (2a) hold for  $D_0 \prec D$  and  $R \prec D$  respectively. Hence,  $\|\ell'\| \in D_0^{\mathcal{I}}$  and  $(\|\ell\|, \|\ell'\|) \in R^{\mathcal{I}}$ . That is,  $\|\ell\| \in (\exists R.D_0)^{\mathcal{I}}$ .

To prove property (2) we consider all cases corresponding to the possible forms of a role  $R$ .

$R = r$ . This case easily follows from the definition of  $r^{\mathcal{I}}$ .

$R = S^{-1}$ . For the property (2a) let  $\ell : \exists S^{-1}.\{\ell'\} \in \mathcal{B}$ . Then  $\ell' : \exists S.\{\ell\} \in \mathcal{B}$  by the rule  $(\exists^{-1})$ . By the induction hypothesis for  $S \prec R$  we have  $(\|\ell'\|, \|\ell\|) \in S^{\mathcal{I}}$ . Consequently,  $(\|\ell\|, \|\ell'\|) \in (S^{-1})^{\mathcal{I}}$ . For (2b) suppose that  $(\|\ell\|, \|\ell'\|) \in (S^{-1})^{\mathcal{I}}$  and  $\ell : \neg\exists S^{-1}.D \in \mathcal{B}$ . As all the occurrences of the inverse operator have been pushed through other role connectives and double occurrences of  $^{-1}$  have been removed<sup>2</sup> we can assume that  $S = r$  for some role name  $r$ . Hence,  $(\|\ell'\|, \|\ell\|) \in r^{\mathcal{I}}$  and, consequently,  $\ell' : \exists r.\{\ell\} \in \mathcal{B}$  by the definition of  $r^{\mathcal{I}}$ . Finally, by the  $(\neg\exists^{-1})$  rule,  $\ell' : \neg D$  is in the branch  $\mathcal{B}$ .

$R = S_0 \sqcup S_1, R = S \downarrow D, R = S \uparrow D$ . The proofs of these cases are easy.

<sup>2</sup> Removal of the double occurrences of the inverse operator in front of atoms is not essential for the proof but it simplifies the proof a bit.

$R = \neg S$ . For (2a) suppose  $\ell : \exists \neg S.\{\ell'\} \in \mathcal{B}$ . Then  $\ell : \neg \exists S.\{\ell'\} \in \mathcal{B}$  is obtained with the  $(\exists \neg)$  rule. If  $(\|\ell\|, \|\ell'\|) \notin (\neg S)^{\mathcal{I}}$  then  $(\|\ell\|, \|\ell'\|) \in S^{\mathcal{I}}$  and by property (2b) which holds by the induction hypothesis for  $S \prec R$  we have that  $\ell' : \neg\{\ell'\}$  is in  $\mathcal{B}$ . This concept together with  $\ell' : \{\ell'\}$  implies the branch is closed. We reach a contradiction, so  $(\|\ell\|, \|\ell'\|) \in (\neg S)^{\mathcal{I}}$ .

For property (2b) suppose that  $(\|\ell\|, \|\ell'\|) \in (\neg S)^{\mathcal{I}}$  and  $\ell : \neg \exists \neg S.D$  are in the branch  $\mathcal{B}$ . Then we have  $(\|\ell\|, \|\ell'\|) \notin S^{\mathcal{I}}$  and, hence, by the contra-positive of property (2a) for  $S \prec R$ ,  $\ell : \exists S.\{\ell'\}$  is not in  $\mathcal{B}$ . Applying the  $(\neg \exists \neg)$  rule to  $\ell : \neg \exists \neg S.D$  we get  $\ell : \exists \neg S.\{\ell'\} \in \mathcal{B}$ . Therefore, by the  $(\neg \exists)$  rule,  $\ell' : \neg D$  is in the branch too.

A consequence of this lemma is completeness of the tableau calculus. Hence, we can state:

**Theorem 2.**  $T_{\mathcal{ALBO}}$  is a sound and complete tableau calculus for  $\mathcal{ALBO}$ .

## 5 Blocking

The calculus  $T_{\mathcal{ALBO}}$  is non-terminating for  $\mathcal{ALBO}$ . There are satisfiable concepts which result in an infinite  $T_{\mathcal{ALBO}}$ -tableau where all open branches are infinite. All the rules respect the well-founded ordering  $\prec$  of expressions under labels, i.e. in every rule the main symbol of the concept above the line is strictly greater w.r.t.  $\prec$  than the main symbol(s) of the expression(s) below the line of the rule. Furthermore, it is easy to see that only applications of the  $(\exists)$  rule generate new symbols in the branch. Thus, the reason that a branch can be infinite is the unlimited application of the  $(\exists)$  rule. As a consequence, the following lemma holds, where  $\#\exists(\mathcal{B})$  denotes the number of applications of the  $(\exists)$  rule in a branch  $\mathcal{B}$ .

**Lemma 3.** If  $\#\exists(\mathcal{B})$  is finite then  $\mathcal{B}$  is finite.

In order to avoid infinite derivations we restrict the application of the  $(\exists)$  rule by the following *blocking* mechanism.

Let  $<$  be an ordering on objects in the branch which is a linear extension of the order in which the objects are introduced during a derivation. I.e. let  $\ell < \ell'$  whenever the first appearance of object  $\ell'$  in the branch is strictly later than the first appearance of the object  $\ell$ . We add the following rule, called the *unrestricted blocking* rule, to the calculus.

$$(ub): \frac{\ell : C, \ell' : D}{\ell : \{\ell'\} \mid \ell : \neg\{\ell'\}}$$

Moreover, we require the following conditions to hold.

- (c1) Any rule is applied at most once to the same set of premises.
- (c2) The  $(\exists)$  rule is applied only to expressions of the form  $\ell : \exists R.C$  when  $C$  is not a singleton, i.e.  $C \neq \{\ell''\}$  for some object  $\ell''$ .

- (c3) If  $\ell : \{\ell'\}$  appears in a branch and  $\ell < \ell'$  then all further applications of the  $(\exists)$  rule to expressions of the form  $\ell' : \exists R.C$  are not performed within the branch.
- (c4) In every open branch there is some node from which point onwards before any application of the  $(\exists)$  rule all possible applications of the (ub) rule must have been performed.

We use the notation  $T_{\mathcal{ALBO}} + (\text{ub})$  for the extension of  $T_{\mathcal{ALBO}}$  with this rule and this blocking mechanism.

**Theorem 3.**  $T_{\mathcal{ALBO}} + (\text{ub})$  is a sound and complete tableau calculus for  $\mathcal{ALBO}$ .

*Proof.* The blocking requirements (c1)–(c4) are sound in the sense that they cannot cause an open branch to become closed. The (ub) rule is sound in the usual sense. Thus, we can safely add the blocking requirements and the blocking rule to a sound and complete tableau without endangering soundness or completeness. Hence,  $T_{\mathcal{ALBO}} + (\text{ub})$  is sound and complete.

Let  $\mathcal{B}$  be the *leftmost* open branch with respect to the rule (ub) in the  $T_{\mathcal{ALBO}} + (\text{ub})$  tableau for a given concept  $C$ . Assume that  $\mathcal{I}$  is a model constructed from  $\mathcal{B}$  as in the completeness section above.

**Lemma 4.** If  $\tau^C(\|\ell\|) = \tau^C(\|\ell'\|)$  in  $\mathcal{I}$  then  $\ell : \{\ell'\} \in \mathcal{B}$ .

*Proof.* Suppose that  $\tau^C(\|\ell\|) = \tau^C(\|\ell'\|)$ . Therefore,  $\ell : \{\ell'\}$  is consistent with  $C$ . By (c4) the rule (ub) has been applied to the objects  $\ell$  and  $\ell'$  in  $\mathcal{B}$ . As  $\ell : \{\ell'\}$  is consistent with  $C$ , the left branch (containing  $\ell : \{\ell'\}$ ) of this application of (ub) is open and, by the choice of the branch  $\mathcal{B}$ , coincides with  $\mathcal{B}$ .

**Corollary 1.** The model  $\bar{\mathcal{I}}$  obtained from  $\mathcal{I}$  by filtration with respect to  $C$  is isomorphic to  $\mathcal{I}$ . In particular,  $\Delta^{\mathcal{I}}$  is finite.

For every  $\|\ell\| \in \Delta^{\mathcal{I}}$ , let  $\#\exists(\|\ell\|)$  denote the number of applications of the  $(\exists)$  rule to concepts of the form  $\ell' : \exists R.D$  with  $\ell' \in \|\ell\|$ .

**Lemma 5.**  $\#\exists(\|\ell\|)$  is finite for every  $\|\ell\| \in \Delta^{\mathcal{I}}$ .

*Proof.* Suppose not, i.e.  $\#\exists(\|\ell\|)$  is infinite. The number of concepts of the shape  $\exists R.D$  under labels in the branch is finitely bounded. By requirements (c1) and (c2) there is a sequence of objects  $\ell_0, \ell_1, \dots$  such that every  $\ell_i \in \|\ell\|$  and the  $(\exists)$  rule has been applied to concepts  $\ell_0 : \exists R.D, \ell_1 : \exists R.D, \dots$  for some  $\exists R.D \prec C$ . However, such a situation is impossible because of requirements (c4) and (c3). Indeed, without loss of generality we can assume that  $\ell < \ell_0 < \ell_1 < \dots$ . Then, by requirement (c4), starting from some node of  $\mathcal{B}$ , as soon as  $\ell_i$  appears in  $\mathcal{B}$ , it is detected that  $\ell_i \in \|\ell\|$  before any next application of the rule  $(\exists)$  and, hence,  $\ell_i$  is immediately blocked for any application of the rule  $(\exists)$ , by requirement (c3).

**Lemma 6.**  $\#\exists(\mathcal{B})$  is finite.

*Proof.* Clearly  $\#\exists(\mathcal{B}) \leq \max\{\#\exists(\|\ell\|) \mid \|\ell\| \in \Delta^T\} \times \text{Card}(\Delta^T)$ . The rest follows from Corollary 1 and Lemma 5.

**Corollary 2.** *If the leftmost branch with respect to the rule (ub) in a  $T_{\mathcal{ALBO}} + (\text{ub})$  tableau is open then the branch is finite.*

**Theorem 4 (Termination).**  *$T_{\mathcal{ALBO}} + (\text{ub})$  is a terminating tableau calculus for  $\mathcal{ALBO}$ .*

*Proof.* Termination of  $T_{\mathcal{ALBO}} + (\text{ub})$  follows from Corollary 2. Indeed, every closed branch of a  $T_{\mathcal{ALBO}} + (\text{ub})$ -tableau is trivially finite and by Corollary 2 the length of the leftmost open branch with respect to the rule (ub) is finite, too.

Notice that condition (c4) is essential for ensuring termination of a  $T_{\mathcal{ALBO}} + (\text{ub})$  derivation. Indeed, it is easy to see that in absence of (c4) the  $T_{\mathcal{ALBO}} + (\text{ub})$  tableau for the concept  $\neg(\exists(s \sqcup \neg s). \neg \exists r.p \sqcup \exists(s \sqcup \neg s). \neg \exists r. \neg p)$  does not terminate because new objects are generated more often than their equality check via the rule (ub) is performed in the tableau.

## 6 Decision procedures

When turning the presented calculus  $T_{\mathcal{ALBO}} + (\text{ub})$  into a deterministic decision procedure it is crucial that this is done in a *fair* way. A procedure is fair if, when if an inference is possible forever then it is performed eventually. In other words a deterministic tableau algorithm based on  $T_{\mathcal{ALBO}} + (\text{ub})$  may not defer the use of an applicable rule indefinitely. Note that understand fairness in a ‘global’ sense. That is, a tableau algorithm has to be fair not only to expressions in a particular branch but to expressions in all branches of a tableau. In another words, the algorithm is fair if it fairly chooses a branch and expression(s) in it to apply a rule.

**Theorem 5.** *Any fair tableau procedure based on  $T_{\mathcal{ALBO}} + (\text{ub})$  is a decision procedure for  $\mathcal{ALBO}$  and all its sublogics.*

Note that we do not assume that the branches are expanded in a depth-first left-to-right order. However it also follows from our results that:

**Theorem 6.** *Any fair tableau procedure based on  $T_{\mathcal{ALBO}} + (\text{ub})$  which uses a depth-first and left-to-right strategy, with respect to branch selection of the (ub) rule, is a decision procedure for  $\mathcal{ALBO}$  and all its sublogics.*

To illustrate the importance of fairness we give an example. The concept

$$C \stackrel{\text{def}}{=} \neg(\exists(s \sqcup \neg s). \neg \exists r.p \sqcup \neg \exists t. \neg \exists r.p)$$

is not satisfiable. Figure 2 gives a depth-first left-to-right derivation which is unfair and does not terminate. Each line in the derivation is numbered on the left. The rule applied and the number of the premise(s) to which it was applied to

1. $\ell_0 : C$ .....given	20. Unsatisfiable.....( $\perp$ ),16,19
2. $\ell_0 : \neg\exists(s \sqcup \neg s).\neg\exists r.p$ .....( $\neg\sqcup$ ),1	21. $\blacktriangleright\ell_0 : \exists\neg s.\{\ell_2\}$ .....( $\neg\exists\neg$ ),6,16
3. $\ell_0 : \neg\neg\exists t.\neg\exists r.p$ .....( $\neg\sqcup$ ),1	22. $\ell_2 : \neg\neg\exists r.p$ .....( $\neg\exists$ ),6,21
4. $\ell_0 : \exists t.\neg\exists r.p$ .....( $\neg\neg$ ),3	23. Unsatisfiable.....( $\perp$ ),16,22
5. $\ell_0 : \neg\exists s.\neg\exists r.p$ .....( $\neg\exists\sqcup$ ),2	24. $\blacktriangleright\ell_0 : \neg\{\ell_1\}$ .....(ub)
6. $\ell_0 : \neg\exists\neg s.\neg\exists r.p$ .....( $\neg\exists\sqcup$ ),2	25. $\ell_2 : p$ .....( $\exists$ ),14
7. $\blacktriangleright\ell_0 : \exists s.\{\ell_0\}$ .....( $\neg\exists\neg$ ),6	26. $\ell_1 : \exists r.\{\ell_2\}$ .....( $\exists$ ),14
8. $\ell_0 : \neg\neg\exists r.p$ .....( $\neg\exists$ ),7,5	27. $\blacktriangleright\ell_0 : \exists s.\{\ell_2\}$ .....( $\neg\exists\neg$ ),6,25
9. $\ell_0 : \exists r.p$ .....( $\neg\neg$ ),8	28. $\ell_2 : \neg\neg\exists r.p$ .....( $\neg\exists$ ),27,5
10. $\ell_1 : p$ .....( $\exists$ ),9	29. $\ell_2 : \exists r.p$ .....( $\neg\neg$ ),28
11. $\ell_0 : \exists r.\{\ell_1\}$ .....( $\exists$ ),9	30. <b>Non-terminating</b> .....
12. $\blacktriangleright\ell_0 : \exists s.\{\ell_1\}$ .....( $\neg\exists\neg$ ),6,10	.....Repetition of 14–29
13. $\ell_1 : \neg\neg\exists r.p$ .....( $\neg\exists$ ),12,5	31. $\blacktriangleright\ell_0 : \exists\neg s.\{\ell_2\}$ .....( $\neg\exists\neg$ ),6,25
14. $\ell_1 : \exists r.p$ .....( $\neg\neg$ ),13	32. ....Similarly to 27–30
15. $\blacktriangleright\ell_0 : \{\ell_1\}$ .....(ub)	33. $\blacktriangleright\ell_0 : \exists\neg s.\{\ell_1\}$ .....( $\neg\exists\neg$ ),6,10
16. $\ell_2 : \neg\exists r.p$ .....( $\exists$ ),4	34. ....Similarly to 12–32
17. $\ell_0 : \exists t.\{\ell_2\}$ .....( $\exists$ ),4	35. $\blacktriangleright\ell_0 : \exists\neg s.\{\ell_0\}$ .....( $\neg\exists\neg$ ),6
18. $\blacktriangleright\ell_0 : \exists s.\{\ell_2\}$ .....( $\neg\exists\neg$ ),6,16	36. ....Similarly to 7–34
19. $\ell_2 : \neg\neg\exists r.p$ .....( $\neg\exists$ ),5,18	

Fig. 2. An infinite, unfair derivation

produce the labelled concept expression (assertion) in each line is specified on the right. The black triangles denote branching points in the derivation. A branch expansion after a branching point is indicated by appropriate indentation. We observe that the derivation is infinite because the application of the ( $\exists$ ) rule to  $\ell_0 : \exists t.\neg\exists r.p$  is deferred forever and, consequently, a contradiction is not found. The example illustrates the importance of fairness to completeness.

Without giving further details we observe that the calculi are compatible with standard optimisations such as backjumping, simplification, different strategies for branch selection and rule selection, etc provided that the fairness condition is not violated.

We have implemented the unrestricted blocking rule as a plug-in to the METTEL tableau prover [9], and tested it on various description logics.

## 7 Conclusion

We have presented a new, general tableau approach for deciding description logics with complex role operators, including especially ‘non-safe’ occurrences of role negation. The tableau decision procedures found in the description logic literature, and implemented in existing tableau-based description logic systems, can handle a large class of description logics but cannot currently handle description logics with full role negation such as  $\mathcal{ALB}$  or  $\mathcal{ALBO}$ . An important novelty of our approach is the use of a blocking mechanism based on the use of inference rules rather than standard loop checking mechanisms which are based on tests performed on sets of expressions or assertions which may need to be

tailored toward specific logics. Our techniques are versatile and are not limited to  $\mathcal{ALBO}$  or its sublogics, but carry over to all description logics and also other logics including first-order logic. We are optimistic that the ideas of this paper can be taken a lot further.

## References

1. P. Baumgartner and R. A. Schmidt. Blocking and other enhancements for bottom-up model generation methods. In *Proc. IJCAR 2006*, vol. 4130 of *LNAI*, pp. 125–139. Springer, 2006.
2. H. De Nivelle and I. Pratt-Hartmann. A resolution-based decision procedure for the two-variable fragment with equality. In *Proc. IJCAR 2001*, vol. 2083 of *LNAI*, pp. 211–225. Springer, 2001.
3. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic J. IGPL*, 8(3):265–292, 2000.
4. G. Gargov and S. Passy. A note on Boolean modal logic. In *Mathematical Logic: Proc. 1988 Heyting Summerschool*, pp. 299–309. Plenum Press, 1990.
5. G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In *Mathematical Logic and its Applications: Proc. 1986 Gödel Conf.*, pp. 253–263. Plenum Press, 1987.
6. E. Grädel, P. G. Kolaitis, and M. Y. Vardi. On the decision problem for two-variable first-order logic. *Bull. Symbolic Logic*, 3:53–69, 1997.
7. U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In *Automated Deduction in Classical and Non-Classical Logics*, vol. 1761 of *LNAI*, pp. 191–205. Springer, 2000.
8. U. Hustadt, R. A. Schmidt, and L. Georgieva. A survey of decidable first-order fragments and description logics. *J. Relational Methods Comput. Sci.*, 1:251–276, 2004.
9. U. Hustadt, D. Tishkovsky, F. Wolter, and M. Zakharyashev. Automated reasoning about metric and topology (System description). In *Proc. JELIA’06*, vol. 4160 of *LNAI*, pp. 490–493. Springer, 2006.
10. C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logics. In *Advances in Modal Logics, Vol. 3*. CSLI Publications, 2002.
11. R. A. Schmidt. Developing modal tableaux and resolution methods via first-order resolution. In *Advances in Modal Logic, Vol. 6*, pp. 1–26. College Publications, 2006.
12. R. A. Schmidt and U. Hustadt. First-order resolution methods for modal logics. In *Volume in memoriam of Harald Ganzinger*, LNCS. Springer, 2006. To appear.
13. R. A. Schmidt, E. Orłowska, and U. Hustadt. Two proof systems for Peirce algebras. In *Proc. RelMiCS 7*, vol. 3051 of *LNCS*, pp. 238–251. Springer, 2004.
14. S. Tobies. *Decidability Results and Practical Algorithms for Logics in Knowledge Representation*. Phd dissertation, RWTH Aachen, Germany, 2001.

## Description Logic vs. Order-Sorted Feature Logic

Hassan Ait-Kaci

ILOG, Inc.  
hak@ilog.com

**Abstract.** We compare and contrast Description Logic ( $\mathcal{DL}$ ) and Order-Sorted Feature ( $\mathcal{OSF}$ ) Logic from the perspective of using them for expressing and reasoning with knowledge structures of the kind used for the Semantic Web.

### Introduction

The advent of the Semantic Web has spurred a great deal of interest in various Knowledge Representation formalisms for expressing, and reasoning with, so-called formal *ontologies*. Such ontologies are essentially sets of expressions describing data and properties thereof. Data is generally organized into ordered hierarchies of set-denoting concepts where the order denotes set inclusion. Properties are binary relations involving these concepts. This set-theoretic semantics is amenable to a crisp formal rendering based on first-order logic, and therefore to proof-theoretic operational semantics.

Description Logic ( $\mathcal{DL}$ ) and Order-Sorted Feature ( $\mathcal{OSF}$ ) logic are two mathematical formalisms that possess such proof-theories. Both are direct descendants of Ron Brachman’s original ideas [1]. This inheritance goes through my own early work formalizing Brachman’s ideas [2], which in turn inspired the work of Gert Smolka, who pioneered the use of constraints *both* for the  $\mathcal{DL}$  [3] and  $\mathcal{OSF}$  [4] formalisms. While the  $\mathcal{DL}$  approach has become the mainstream of research on the Semantic Web, the lesser known  $\mathcal{OSF}$  formalisms have evolved out of Unification Theory [5], and been used in Constraint-Logic Programming and Computational Linguistics [6–19].

In this short communication (extracted from [20]), we compare and contrast  $\mathcal{DL}$  and  $\mathcal{OSF}$  logics with the purpose of using them effectively for ontological representation and reasoning.

### Relation between $\mathcal{DL}$ and $\mathcal{OSF}$ Formalisms

The two formalisms for describing attributed typed objects of interest—*viz.*,  $\mathcal{DL}$  and  $\mathcal{OSF}$ —have several common, as well as distinguishing, aspects. Thanks to both formalisms using the common language of  $\mathcal{FOL}$  for expressing semantics, they may thus be easily compared—see, for example, [21, 22]. We here brush on some essential points of comparison and contrast.<sup>1</sup>

**Common Aspects**  $\mathcal{DL}$  reasoning is generally carried out using (variations on) Deductive Tableau methods [23].<sup>2</sup> This is also the case of the constraint propagation rules of Fig. 1, which simply mimick a Deductive Tableau decision procedure [24].<sup>3</sup>  $\mathcal{OSF}$  reasoning is performed by the  $\mathcal{OSF}$ -constraint normalization rules of Figs. 2 and 3, which implement a logic of sorted-feature equality.

<sup>1</sup> Due to severe, and strictly enforced, space limitation in these proceedings, most of the points we make here are further elaborated for the interested reader in [20].

<sup>2</sup> Although one can find some publications on Description Logics that do not (fully) use Tableaux reasoning for their operational semantics and mix it with resolution (*i.e.*, Prolog technology), the overwhelming majority follow the official W3C recommendations based on Tableaux methods for TBox reasoning.

<sup>3</sup> The constraint-rule notation we use is Plotkin’s SOS style [30]. The constraint system  $\mathcal{ALCN}\mathcal{R}$  is given here as an exemplar of a DL Tableaux-based reasoning system. It is neither the most expressive nor the most efficient. However, it uses the same style of formula-expansion rules used by all Tableaux-based DL systems such as, in particular, the ever-growing family of esoterically-named Description Logics  $\mathcal{SHIQ}$ ,  $\mathcal{SHOIN}$ ,  $\mathcal{SHOIQ}$ ,  $\mathcal{SHOQ(D)}$ ,  $\mathcal{SRIQ}$ , and other  $\mathcal{SROIQ}$ , which underlie all the official nocturnal bird languages promoted by the W3C to enable the Semantic Web—see for example the “official” DL site (<http://dl.kr.org/>) as well as the output of one of its most prolific spokespersons (<http://www.cs.man.ac.uk/~horrocks/Publications/>).

$(\mathcal{C}_\cap) \text{ CONJUNCTIVE CONCEPT:$ $\left[ \begin{array}{l} \text{if } x : (C_1 \sqcap C_2) \in S \\ \text{and } \{x : C_1, x : C_2\} \not\subseteq S \end{array} \right]$	$\frac{S}{S \cup \{x : C_1, x : C_2\}}$
$(\mathcal{C}_\sqcup) \text{ DISJUNCTIVE CONCEPT:$ $\left[ \begin{array}{l} \text{if } x : (C_1 \sqcup C_2) \in S \\ \text{and } x : C_i \notin S \ (i = 1, 2) \end{array} \right]$	$\frac{S}{S \cup \{x : C_1\}}$
$(\mathcal{C}_\forall) \text{ UNIVERSAL ROLE:$ $\left[ \begin{array}{l} \text{if } x : (\forall R.C) \in S \\ \text{and } y \in R_S[x] \\ \text{and } y : C \notin S \end{array} \right]$	$\frac{S}{S \cup \{y : C\}}$
$(\mathcal{C}_\exists) \text{ EXISTENTIAL ROLE:$ $\left[ \begin{array}{l} \text{if } x : (\exists R.C) \in S \text{ s.t. } R \stackrel{\text{DEF}}{=} (\prod_{i=1}^m R_i) \\ \text{and } z : C \in S \implies z \notin R_S[x] \\ \text{and } y \text{ is new} \end{array} \right]$	$\frac{S}{S \cup \{xR_iy\}_{i=1}^m \cup \{y : C\}}$
$(\mathcal{C}_{\geq}) \text{ MIN CARDINALITY:$ $\left[ \begin{array}{l} \text{if } x : (\geq n.R) \in S \text{ s.t. } R \stackrel{\text{DEF}}{=} (\prod_{i=1}^m R_i) \\ \text{and }  R_S[x]  \neq n \\ \text{and } y_i \text{ is new } (0 \leq i \leq n) \end{array} \right]$	$\frac{S}{S \cup \{xR_iy_j\}_{i,j=1,1}^{m,n} \cup \{y_i \neq y_j\}_{1 \leq i < j \leq n}}$
$(\mathcal{C}_{\leq}) \text{ MAX CARDINALITY:$ $\left[ \begin{array}{l} \text{if } x : (\leq n.R) \in S \\ \text{and }  R_S[x]  > n \\ \text{and } y, z \in R_S[x] \\ \text{and } y \neq z \notin S \end{array} \right]$	$\frac{S}{S \cup S[y/z]}$

**Fig. 1.** Some  $\mathcal{DL}$ -constraint propagation rules ( $\mathcal{ALCN}$ )

§ **Object Descriptions**—Both the  $\mathcal{DL}$  and  $\mathcal{OSF}$  formalisms describe typed attributed objects. In each, objects are data structures described by combining set-denoting concepts and relation-denoting roles.

§ **Logic-Based Semantics**—Both  $\mathcal{DL}$  and  $\mathcal{OSF}$  logic are syntactic formalisms expressing meaning using conventional logic styles. In other words, both formalisms take their meaning in a common universal language—viz., (elementary) Set Theory. This is good since it eases understanding each formalism in relation to the other thanks to their denotations in the common language.

§ **Proof-Theoretic Semantics**—Both  $\mathcal{DL}$  and  $\mathcal{OSF}$  logics have their corresponding proof theory. Indeed, since both formalisms are syntactic variants of fragments of  $\mathcal{FOL}$ , proving theorems in each can always rely on  $\mathcal{FOL}$  mechanized theorem proving.

§ **Constraint-Based Formalisms**—Even further, both  $\mathcal{DL}$  and  $\mathcal{OSF}$  logic are operationalized using a constraint-based decision procedure. As we have expounded, this makes both paradigms amenable to being manipulated by rule-based systems such as based on  $\mathcal{CLP}$ , rewrite rules, or production rules.

§ **Concept Definitions**—Both  $\mathcal{DL}$  and  $\mathcal{OSF}$  provide a means for defining concepts in terms of other concepts. This enables a rich framework for expressing recursive data structures.

**Distinguishing Aspects** There are also aspects in each that distinguish the  $\mathcal{DL}$  and  $\mathcal{OSF}$  formalisms apart. However, several of these distinguishing features are in fact cosmetic—*i.e.*, are simply equivalent notation for the same meaning. Remaining non-cosmetic differences are related to the nature of the deductive processes enabled out by each formalism.

§ **Functional Features vs. Relational Roles**—The  $\mathcal{OSF}$  formalism uses *functions* to denote attributes while the  $\mathcal{DL}$  formalism uses *binary relations* for the same purpose. Many have argued that this difference is fundamental and restricts the expressivity of  $\mathcal{OSF}$  vs.  $\mathcal{DL}$ . This, however, is only a cosmetic difference as we have already explained. First of all, a function  $f : A \mapsto B$  is a binary relation since  $f \in A \times B$ . It a *functional* relation because it obeys the axiom of functionality; namely,  $\langle a, b \rangle \in f \ \& \ \langle a, b' \rangle \in f \Rightarrow b = b'$ . In other words, a function is a binary relation that associates at most one range element to any domain element. This axiom is fundamental as it is used in basic  $\mathcal{OSF}$  unification “*Feature Functionality*” shown in Fig. 2. Indeed, the correctness of this rule relies on the semantics of features as functions, not as relations.

---

$(\mathcal{O}_1)$ <b><u>SORT INTERSECTION:</u></b>	$\phi \ \& \ X : s \ \& \ X : s'$
	$\phi \ \& \ X : s \wedge s'$
$(\mathcal{O}_2)$ <b><u>INCONSISTENT SORT:</u></b>	$\phi \ \& \ X : \perp$
	$X : \perp$
$(\mathcal{O}_3)$ <b><u>FEATURE FUNCTIONALITY:</u></b>	$\phi \ \& \ X.f \doteq X' \ \& \ X.f \doteq X''$
	$\phi \ \& \ X.f \doteq X' \ \& \ X' \doteq X''$
$(\mathcal{O}_4)$ <b><u>VARIABLE ELIMINATION:</u></b>	$\phi \ \& \ X \doteq X'$
[if $X \neq X'$ and $X \in \text{VAR}(\phi)$ ]	$\phi[X/X'] \ \& \ X \doteq X'$
$(\mathcal{O}_5)$ <b><u>VARIABLE CLEANUP:</u></b>	$\phi \ \& \ X \doteq X$
	$\phi$

---

**Fig. 2.** Basic  $\mathcal{OSF}$ -constraint normalization rules

However, a relation  $R \in A \times B$  is equivalent to either of a pair of set-denoting functions—*viz.*, either the function  $R[\_ ] : A \mapsto \mathbf{2}^B$ , returning the *R-object* (or *R-image*) set  $R[x] \subseteq B$  of an element  $x \in A$ ; or, dually, the function  $R^{-1}[\_ ] : B \mapsto \mathbf{2}^A$ , returning the *R-subject* (or *R-antecedent*) set  $R^{-1}[y] \subseteq A$  of an element  $y \in B$ . Indeed, the following statements (s<sub>1</sub>)–(s<sub>3</sub>) are equivalent:

$$\begin{aligned} \forall \langle x, y \rangle \in A \times B, \quad \langle x, y \rangle \in R & \quad (s_1) \\ y \in R[x] & \quad (s_2) \\ x \in R^{-1}[y] & \quad (s_3) \end{aligned}$$

Therefore, it is a simple matter for the  $\mathcal{OSF}$  formalism to express relational attributes (or roles) with features taking values as sets. This is trivially done as a special case of the “*Value Aggregation*”  $\mathcal{OSF}$  unification rule shown in Fig. 3, using a set data constructor—*i.e.*, a commutative idempotent monoid.

$$\begin{array}{l}
 (\mathcal{O}_6) \text{ **PARTIAL FEATURE:** } \\
 \left[ \text{if } s \in \text{DOM}(f) \text{ and } \text{RAN}_s(f) = s' \right] \frac{\phi \ \& \ X.f \doteq X'}{\phi \ \& \ X.f \doteq X' \ \& \ X : s \ \& \ X' : s'} \\
 \\
 (\mathcal{O}_7) \text{ **WEAK EXTENSIONALITY:** } \\
 \left[ \text{if } s \in \mathcal{E} \text{ and } \forall f \in \text{ARITY}(s) : \right. \\
 \left. \{X.f \doteq Y, X'.f \doteq Y\} \subseteq \phi \right] \frac{\phi \ \& \ X : s \ \& \ X' : s}{\phi \ \& \ X : s \ \& \ X \doteq X'} \\
 \\
 (\mathcal{O}_8) \text{ **VALUE AGGREGATION:** } \\
 \left[ \text{if } s \text{ and } s' \text{ are both subsorts of} \right. \\
 \left. \text{commutative monoid } \langle \star, \mathbf{1}_\star \rangle \right] \frac{\phi \ \& \ X = e : s \ \& \ X = e' : s'}{\phi \ \& \ X = e \star e' : s \wedge s'}
 \end{array}$$

**Fig. 3.** Additional  $\mathcal{OSF}$ -constraint normalization rules

<sup>§</sup>**Sets vs. Individuals**—Because the  $\mathcal{OSF}$  formalism has only set-denoting sorts, it is often misconstrued as unable to deal with individual elements of these sets. However, as explained in [20], this is again an innocuous cosmetic difference since elements are simply assimilated to singleton-denoting sorts.

<sup>§</sup>**No Number Restrictions vs. Number Restrictions**—Strictly speaking, the  $\mathcal{OSF}$  formalism has no special constructs for number restrictions as they exist in  $\mathcal{DL}$ . Now, this does not mean that it lacks the power to enforce such constraints. Before we show how this may be done, however, it important to realize that it may not always be a good idea to use the  $\mathcal{DL}$  approach to do so.

Indeed, as can be seen in Fig. 1, the “*Min Cardinality*” rule ( $\mathcal{C}_{\leq}$ ) will introduce  $n(n-1)/2$  new disequality constraints for each such constraint of cardinality  $n$ . Clearly, this is a source of gross inefficiency a  $n$  increases. Similarly, the “*Existential Role*” rule ( $\mathcal{C}_{\exists}$ ) will systematically introduce a new variable for a role, *even when this role is never accessed!* It does so because, it materializes the full extent of role value sets. In other words,  $\mathcal{C}$  constraint-propagation rules flesh out complete skeletons for attributed data structures whether or not the actual attribute values are needed.

By contrast, it is simple and efficient to accommodate cardinality constraints in the  $\mathcal{OSF}$  calculus with value aggregation using a set constructor (*i.e.*, an idempotent commutative monoid  $M = \langle \star, \mathbf{1}_\star \rangle$ ), and a function  $\text{CARD} : M \mapsto \mathbb{N}$  that returns the number of elements in a set. Then, imposing a role cardinality constraint for a role  $r$  in a feature term  $t = X : s(r \Rightarrow S = \{e_1, \dots, e_n\} : m)$ , where sort  $m$  denotes  $M$ 's domain, is achieved by the constraint  $\varphi(t) \ \& \ \text{CARD}(S) \leq n$ —or  $\varphi(t) \ \& \ \text{CARD}(S) \geq n$ . If the set contains variables, these constraints will residuate as needed pending the complete evaluation of the function  $\text{CARD}$ . However, as soon as enough non-variable elements have materialized in the set that enable the decision, the constraint will be duly enforced. Clearly, this “lazy” approach saves the time and space wasted by  $\mathcal{DL}$ -propagation rules, while fully enforcing the needed cardinalities.

Incidentally, note also that this principle allows not only min and max cardinality, but any constraints on a set, whether cardinality or otherwise. Importantly, this foregoing method works not only for sets, but can be used with arbitrary aggregations using other monoids.

<sup>§</sup>**Greatest Fix-Point vs. Least Fix-Point**—It is well known that unfolding recursive definitions of all kinds (be it function, relation, or sort) is precisely formalized as computing a fix-point in some information-theoretic lattice. Indeed, given a complete lattice  $\mathcal{L} \stackrel{\text{DEF}}{=} \langle D^{\mathcal{L}}, \sqsubseteq^{\mathcal{L}}, \sqcap^{\mathcal{L}}, \sqcup^{\mathcal{L}}, \top^{\mathcal{L}}, \perp^{\mathcal{L}} \rangle$  and a monotone function<sup>4</sup>  $\mathcal{F} : D^{\mathcal{L}} \mapsto D^{\mathcal{L}}$ , Tarski's fix-point theorem<sup>5</sup> states that the set  $\text{FP}(\mathcal{F}) \stackrel{\text{DEF}}{=} \{x \in D^{\mathcal{L}} \mid \mathcal{F}(x) = x\}$  of

<sup>4</sup> That is, such that:  $\forall x, y \in D^{\mathcal{L}}, x \sqsubseteq^{\mathcal{L}} y \implies \mathcal{F}(x) \sqsubseteq^{\mathcal{L}} \mathcal{F}(y)$ .

<sup>5</sup> See, e.g., [25].

fix-points of  $\mathcal{F}$  is itself a complete sublattice of  $\mathcal{L}$ . Moreover, its bottom element is called  $\mathcal{F}$ 's *least fix-point* (LFP), written  $\mathcal{F}^\uparrow$ , defined by Equation (1):

$$\mathcal{F}^\uparrow \stackrel{\text{DEF}}{=} \bigsqcup_{n \in \mathbb{N}} \mathcal{F}^n(\perp^{\mathcal{L}}) \tag{1}$$

and its top element is called  $\mathcal{F}$ 's *greatest fix-point* (GFP), written  $\mathcal{F}^\downarrow$ , defined by Equation (2):

$$\mathcal{F}^\downarrow \stackrel{\text{DEF}}{=} \bigsqcap_{n \in \mathbb{N}} \mathcal{F}^n(\top^{\mathcal{L}}) \tag{2}$$

where:

$$\mathcal{F}^n(x) = \begin{cases} x & \text{if } n = 0, \\ \mathcal{F}(\mathcal{F}^{n-1}(x)) & \text{otherwise.} \end{cases}$$

Informally,  $\mathcal{F}^\uparrow$  is the *upward* iterative limit of  $\mathcal{F}$  starting from the least element in  $D^{\mathcal{L}}$ , while  $\mathcal{F}^\downarrow$  is its *downward* iterative limit starting from the greatest element in  $D^{\mathcal{L}}$ . One can easily show that  $\mathcal{F}(\mathcal{F}^\uparrow) = \mathcal{F}^\uparrow$  [resp.,  $\mathcal{F}(\mathcal{F}^\downarrow) = \mathcal{F}^\downarrow$ ], and that no element of  $D^{\mathcal{L}}$  lesser than  $\mathcal{F}^\uparrow$  [resp., greater than  $\mathcal{F}^\downarrow$ ] is a fix-point of  $\mathcal{F}$ .

One may wonder when one, or the other, kind of fix-point captures the semantics intended for a set of recursive definitions. Intuitively, LFP semantics is appropriate when inference proceeds by deriving *necessary consequences* from facts that hold true, and GFP semantics is appropriate when inference proceeds by deriving *sufficient conditions* for facts to hold true.<sup>6</sup> Therefore, LFP computation can model only well-founded (*i.e.*, terminating) recursion, while GFP computation can also model non well-founded (*i.e.*, not necessarily terminating) recursion. Hence, typically, LFP computation is naturally described as a *bottom-up* process, while GFP computation is naturally described as a *top-down* process.

An example of GFP semantics is given by the Herbrand-term unification. Indeed, this process transforms a set of equations into an equivalent one using sufficient conditions by processing the terms top-down from roots to leaves. The problem posed is to find sufficient conditions for a term equation to hold on the constituents (*i.e.*, the subterms) of both sides of the equation. For first-order terms, this process converges to either failure or producing a most general sufficient condition in the form of a variable substitution, or equation set in solved form (the MGU). Similarly, the  $\mathcal{OSF}$ -constraint normalization rules of Figs. 2, 4, 5, and 3 also form an example of converging GFP computation for the same reasons. Yet another example of GFP computation where the process may diverge is the lazy recursive sort definition unfolding described in [26].

On the other hand, constraint-propagation rules based on Deductive Tableau methods such as used in [3] or shown in Fig. 1 are LFP computations. Indeed, they proceed bottom-up by building larger and larger constraint sets by completing them with additional (and often redundant) constraints. In short,  $\mathcal{OSF}$ -constraint normalization follows a reductive semantics (it eliminates constraints) while  $\mathcal{DL}$ -constraint propagation follows an inflationary semantics (it introduces constraints). As a result,  $\mathcal{DL}$ 's tableau-style reasoning method is *expansive*—therefore, *expensive* in time and space. One can easily see this simply by realizing that each rule in Fig. 1 builds a larger set  $S$  as it keeps adding more constraints and more variables to  $S$ . Only the “*Max Cardinality*” rule ( $C_{<}$ ) may reduce the size of  $S$  to enforce upper limits on a concept's extent's size by merging two variables. Finally, it requires that the constraint-solving process be decidable.

By contrast, the  $\mathcal{OSF}$  labelled-graph unification-style reasoning method is more efficient both in time and space. Moreover, it can accommodate semi-decidable—*i.e.*, undecidable, though recursively enumerable—constraint-solving. Indeed, no rule in Figs. 2, 4, 5, and 3 ever introduces a new variable. Moreover, all the rules in Fig. 2 as well as the rule 3, except for the “*Partial Feature*” rule, all eliminate constraints. Even this latter rule introduces no more constraints than the number of features in the whole constraint. The rules in Figs. 4 and 5 may replace some constraints with more constraints, but the introduced constraints are all more restrictive than those eliminated.

<sup>§</sup> **Coinduction vs. Induction**—Remarkably, the interesting duality between least and greatest fix-point computations is in fact equivalent to another fundamental one; namely, *induction vs. coinduction* in computation

<sup>6</sup> One might also say that LFP is *deductive* since it moves from premiss to consequent, and that GFP is *abductive* since it moves from consequent to premiss.

$$\begin{array}{l}
 (\mathcal{O}_9) \text{ NON-UNIQUE GLB:} \\
 \left[ \begin{array}{l} \text{if } \{s_1\}_{i=0}^n = \max_{\leq} \{t \in \mathcal{S} \mid t \leq s\} \\ \text{and } t \leq s' \end{array} \right] \frac{\phi \ \& \ X : s \ \& \ X : s'}{\phi \ \& \ (X : s_1 \parallel \dots \parallel X : s_n)} \\
 \\
 (\mathcal{O}_{10}) \text{ DISTRIBUTIVITY:} \\
 \frac{\phi \ \& \ (\phi' \parallel \phi'')}{(\phi \ \& \ \phi') \parallel (\phi \ \& \ \phi'')} \\
 \\
 (\mathcal{O}_{11}) \text{ DISJUNCTION:} \\
 \frac{\phi \parallel \phi'}{\phi}
 \end{array}$$

Fig. 4. Disjunctive  $\mathcal{OSF}$ -constraint normalization

$$\begin{array}{l}
 (\mathcal{O}_{12}) \text{ DISEQUALITY:} \\
 \frac{\phi \ \& \ X \neq X}{\perp} \\
 \\
 (\mathcal{O}_{13}) \text{ COMPLEMENT:} \\
 \left[ \begin{array}{l} \text{if } s' \in \max_{\leq} \{t \in \mathcal{S} \mid t \not\leq s\} \\ \text{and } t \not\leq s \end{array} \right] \frac{\phi \ \& \ X : \bar{s}}{\phi \ \& \ X : s'}
 \end{array}$$

Fig. 5. Negative  $\mathcal{OSF}$ -constraint normalization

and logic, as nicely explained in [27]. Indeed, while induction allows to derive a whole entity from its constituents, coinduction allows to derive the constituents from the whole. Thus, least fix-point computation is induction, while greatest fix-point computation is coinduction. Indeed, coinduction is invaluable for reasoning about non well-founded computations such as those carried out on potentially infinite data structures [28], or (possibly infinite) process bisimulation [29].

This is a fundamental difference between  $\mathcal{DL}$  and  $\mathcal{OSF}$  formalisms:  $\mathcal{DL}$  reasoning proceeds by actually building a model’s domain verifying a TBox, while  $\mathcal{OSF}$  reasoning proceeds by eliminating impossible values from the domains. Interestingly, this was already surmised in [3] where the authors state:

*“[...] approaches using feature terms as constraints [...] use a lazy classification and can thus tolerate undecidable subproblems by postponing the decision until further information is available. [...] these] approaches are restricted to feature terms; however, an extension to KL-ONE-like concept terms appears possible.”*

Indeed, the extended  $\mathcal{OSF}$  formalism overviewed in [20] is a means to achieve precisely this.

## Conclusion

We have briefly reviewed two well-known data description formalisms based on constraints, Description Logic and Order-Sorted Feature Logic, explicating how they work and how they are formally related. We

have identified similarities and differences by studying their set-theoretic semantics and first-order logic proof-theory based on constraint-solving. In so doing, we identified that the two formalisms differ essentially as they follow dual constraint-based reasoning strategies,  $\mathcal{DL}$  constraint-solving being inductive (or eager), and  $\mathcal{OSF}$  constraint-solving being coinductive (or lazy). This has as consequence that  $\mathcal{OSF}$  logic is more effective at dealing with infinite data structures and semi-decidable inference.

It seems therefore evident that, since the  $\mathcal{DL}$  and  $\mathcal{OSF}$  formalisms are one another’s formal *duals*, both semantically and pragmatically, we should be well-advised to know precisely *when* one or the other technology is more appropriate for *what* Semantic Web reasoning task.

## References

1. Brachman, R.: A Structural Paradigm for Representing Knowledge. PhD thesis, Harvard University, Cambridge, MA, USA (1977)
2. Ait-Kaci, H.: A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially-Ordered Types. PhD thesis, University of Pennsylvania, Philadelphia, PA (1984)
3. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. *Artificial Intelligence* **48** (1991) 1–26
4. Smolka, G.: A feature logic with subsorts. LILOG Report 33, IWBS, IBM Deutschland, Stuttgart, Germany (1988)
5. Schmidt-Schauß, M., Siekmann, J.: Unification algebras: An axiomatic approach to unification, equation solving and constraint solving. Technical Report SEKI-report SR-88-09, FB Informatik, Universität Kaiserslautern (1988) [Available online <sup>7</sup>].
6. Ait-Kaci, H.: An introduction to LIFE—Programming with Logic, Inheritance, Functions, and Equations. In Miller, D., ed.: *Proceedings of the International Symposium on Logic Programming*, MIT Press (1993) 52–68 [Available online <sup>8</sup>].
7. Ait-Kaci, H., Dumant, B., Meyer, R., Podelski, A., Van Roy, P.: The Wild LIFE handbook. [Available online <sup>9</sup>] (1994)
8. Ait-Kaci, H., Podelski, A.: Towards a meaning of LIFE. *Journal of Logic Programming* **16**(3-4) (1993) 195–234 [Available online <sup>10</sup>].
9. Carpenter, B.: The Logic of Typed Feature Structures. Volume 32 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK (1992)
10. Dörre, J., Rounds, W.C.: On subsumption and semiunification in feature algebras. In: *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA)*, Washington, DC, IEEE, Computer Society Press (1990) 301–310
11. Dörre, J., Seiffert, R.: Sorted feature terms and relational dependencies. In Nebel, B., von Luck, K., Peltason, C., eds.: *Proceedings of the International Workshop on Terminological Logics*, DFKI (1991) 109–116 [Available online <sup>11</sup>].
12. Emele, M.C.: Unification with lazy non-redundant copying. In: *Proceedings of the 29th annual meeting of the ACL*, Berkeley, California, Association for Computational Linguistics (June 1991)
13. Emele, M.C., Zajac, R.: A fixed point semantics for feature type systems. In: *Proceedings of the 2nd International CTRS Workshop*, Montreal (June 1990). Number 516 in *Lecture Notes in Computer Science*, Springer-Verlag (1992) 383–388
14. Fischer, B.: Resolution for feature logics. In: *GI-Fachgruppe über Alternative Konzepte für Sprachen und Rechner*, GI Softwaretechnik Trends (1993) 23–34 [Available online <sup>12</sup>].
15. Smolka, G.: Feature constraint logic for unification grammars. *Journal of Logic Programming* **12** (1992) 51–87

<sup>7</sup> <http://www.ki.informatik.uni-frankfurt.de/papers/schauss/unif-algebr.pdf>

<sup>8</sup> <http://koala.ilog.fr/wiki/pub/Main/HassanAitKaci/ilps93.ps.gz>

<sup>9</sup> <http://citeseer.ist.psu.edu/134450.html>

<sup>10</sup> <http://www.hpl.hp.com/techreports/Compaq-DEC/PRL-RR-11.pdf>

<sup>11</sup> <http://elib.uni-stuttgart.de/opus/volltexte/1999/93/>

<sup>12</sup> <http://www.infosun.fmi.uni-passau.de/st/papers/resolution/>

16. Zajac, R.: Towards object-oriented constraint logic programming. In: Proceedings of the ICLP'91 Workshop on Advanced Logic Programming Tools and Formalisms for Natural Language Processing, Paris (1991)
17. Treinen, R.: Feature trees over arbitrary structures. In Blackburn, P., de Rijke, M., eds.: Specifying Syntactic Structures. CSLI Publications and FoLLI (1997) 185–211 [Available online <sup>13</sup>].
18. Müller, M., Niehren, J., Treinen, R.: The first-order theory of ordering constraints over feature trees. *Discrete Mathematics & Theoretical Computer Science* **4**(2) (2001) 193–234 [Available online <sup>14</sup>].
19. Müller, M., Niehren, J., Podelski, A.: Ordering constraints over feature trees. *Constraints* **5**(1–2) (2000) 7–42 Special issue on CP'97, Linz, Austria. [Available online <sup>15</sup>].
20. Ait-Kaci, H.: Data models as constraint systems: A key to the semantic web. Research paper, submitted for publication, ILOG, Inc. (2007) [Available online <sup>16</sup>].
21. Nebel, B., Smolka, G.: Representation and reasoning with attributive descriptions. In Blasius, K., Hedtstück, U., Rollinger, C.R., eds.: *Sorts and Types in Artificial Intelligence*. Volume 418 of *Lecture Notes in Artificial Intelligence*., Springer-Verlag (1990) 112–139
22. Nebel, B., Smolka, G.: Attributive description formalisms and the rest of the world. In Herzog, O., Rollinger, C.R., eds.: *Text Understanding in LILOG: Integrating Computational Linguistics and Artificial Intelligence*. Volume 546 of *Lecture Notes in Artificial Intelligence*., Springer-Verlag (1991) 439–452
23. Manna, Z., Waldinger, R.: Fundamentals of deductive program synthesis. In Apostolico, A., Galil, Z., eds.: *Combinatorial Algorithms on Words*. NATO ISI Series. Springer-Verlag (1991) [Available online <sup>17</sup>].
24. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: Reasoning in description logics. In Brewka, G., ed.: *Principles of Knowledge Representation*. CSLI Publications, Stanford, CA (1996) 191–236 [Available online <sup>18</sup>].
25. Birkhoff, G.: *Lattice Theory*. 3rd edn. Volume 25 of *Colloquium Publications*. American Mathematical Society, Providence, RI, USA (1979)
26. Ait-Kaci, H., Podelski, A., Goldstein, S.C.: Order-sorted feature theory unification. *Journal of Logic Programming* **30**(2) (1997) 99–124 [Available online <sup>19</sup>].
27. Sangiorgi, D.: Coinduction in programming languages. Invited Lecture ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages—POPL'04 (2004) [Available online <sup>20</sup>].
28. Aczel, P.: *Non Well-Founded Sets*. Center for the Study of Language and Information, Stanford, CA, USA (1988) [Available online <sup>21</sup>].
29. Baeten, J.C.M., Weijland, W.P.: *Process Algebra*. Volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK (1990)
30. Plotkin, G.D.: A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Århus, Århus, Denmark (1981) [Available online <sup>22</sup>].

<sup>13</sup> <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/FeatArbStruct.ps>

<sup>14</sup> <http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/dm040211.ps>

<sup>15</sup> <http://www.ps.uni-sb.de/Papers/abstracts/ftsub-constraints-99.html>

<sup>16</sup> <http://koala.ilog.fr/wiki/bin/view/Main/HassanAitKaci/semwebclp.pdf>

<sup>17</sup> <http://citeseer.ist.psu.edu/manna92fundamentals.html>

<sup>18</sup> <http://citeseer.ist.psu.edu/article/donini97reasoning.html>

<sup>19</sup> <http://www.hpl.hp.com/techreports/Compaq-DEC/PRL-RR-32.pdf>

<sup>20</sup> [http://www.cs.unibo.it/~sangiorgio/DOC\\_public/TalkPOPL.ps.gz](http://www.cs.unibo.it/~sangiorgio/DOC_public/TalkPOPL.ps.gz)

<sup>21</sup> <http://standish.stanford.edu/pdf/00000056.pdf>

<sup>22</sup> <http://citeseer.ist.psu.edu/plotkin81structural.html>

## SEMilarity: Towards a Model-Driven Approach to Similarity

Rudi Araújo and H. Sofia Pinto

INESC-ID

Rua Alves Redol 9, Apartado 13069, 1000-029 Lisboa, Portugal  
{`rna`, `sofia`}@`algos.inesc-id.pt`

**Abstract.** Enabling the Semantic Web requires solving the semantic heterogeneity problem, for which ontology matching methods have been proposed. These methods rely on similarity measures that are mainly focused on terminological, structural and extensional properties of the ontologies. Semantics rarely play a direct role on the ontology matching process, albeit some algorithms have been proposed. On the other hand, many ontology engineers choose representation languages that have an underlying formal logic, providing well-defined model-theoretic semantics. Since semantics are a key advantage of ontologies, we believe that semantics-based similarity measures are crucial. In this paper, we present a novel approach to semantic similarity.

**Key words:** ontology matching, semantic similarity

### 1 Introduction

Given the semi-anarchic organisation of the current World Wide Web, it is unrealistic to expect that the Semantic Web, its envisioned evolution, will not suffer from semantic heterogeneity, which can, if it is not properly tackled, hinder its acceptance and consequently its growth and, in the worst case, preclude its development. Ontology matching and alignment is an area that deals with this problem by establishing relations (usually equivalence and subsumption relations) between elements in different ontologies. According to [1], ontology alignment techniques can be categorised in two major groups: *local*, which focuses on similarities of individual elements and/or their relations to other elements, and *global*, dealing with the whole ontology or parts of it. The local alignment techniques are further classified as *terminological*, *structural*, *extensional* or *semantics*. Terminological methods are twofold: many rely in string-matching techniques, such as sub-string matching, Jaccard Distance, Edit Distance, etc; others use external linguistic resources, such as dictionaries or thesauri. Structural techniques rely on the structure of the elements, and their relations to other elements, recurring, for example, to graph matching techniques. Extensional techniques focus on the extensions (instances) of concepts to assess their likelihood. Finally, semantics-based alignment approaches are aware and make use of the semantics underlying

the representation language, which enables them to resort to deduction services, such as subsumption and consistency checking.

Similarity measures are used to assess the likelihood of elements of ontologies or the ontologies themselves. In this paper we present our preliminary work on defining an ontology similarity measure that is purely based in the semantics of concepts. We should note at this point that we are committing to the notion of semantics as defined by a formal logic system, and not as its pragmatical meaning as approached in [2]. For the target representation language, we chose a Description Logics formalism for mainly three reasons: it is the backbone of the current most prominent ontology representation language for the Semantic Web – the OWL language –, it is the most active family of languages in the community and it provides well-defined model-theoretic semantics. Our target representation language is  $\mathcal{ALC}$  without roles. Although this is a rather inexpressive logic, we stress that this work is only preliminary and that we plan to extend it towards more expressive languages. Note that this logic is equivalent to propositional logic, which, inexpressive as it is, can still find application in the real world, since it allows to describe taxonomies (web directories are examples of this). In the following, we assume that a TBox is a set of subsumption and equivalence axioms, that relate atomic and complex concepts. The concepts we are considering are  $\perp$ ,  $\top$ ,  $A$ ,  $C \sqcap D$ ,  $C \sqcup D$ ,  $\neg C$ , where  $A$  is a concept name, and  $C$  and  $D$  are concepts. Their semantics are defined as usually [3]. The term *ontology* is often used to refer to a number of different artifacts that may include, for example, a glossary of terms, the conceptual and coded model and the documentation. For simplicity, in this paper we will restrict the notion of *ontology* to an  $\mathcal{ALC}$  TBox without roles. In the following, it is assumed that the set of concepts  $\mathcal{C}$  contained in an ontology is finite.

The paper is organised as follows: section 2 presents the theoretical underpinning of the work presented here, followed by a toy example demonstrating how it works in practice. The implementation of the algorithm is the subject of section 3. We conducted an experiment with average-sized ontologies, using the proposed similarity measure, described in section 4. Section 5 comprises an evaluation and discussion of the proposed measure. We summarise related work in section 6 and finish the paper with conclusions and future directions in section 7.

## 2 Theory

Given the set of possible ontologies in the language we are considering,  $\mathcal{O}$ , our aim is to define a similarity measure  $\sigma : \mathcal{O} \times \mathcal{O} \rightarrow [0, 1]$ , which is purely based on semantics. This similarity measure is required to take the highest value for equivalent ontologies, i.e. given three ontologies  $\mathcal{T}_1$ ,  $\mathcal{T}_2$  and  $\mathcal{T}_3$ , if  $\mathcal{T}_2 \equiv \mathcal{T}_3$ :<sup>1</sup>

1.  $\sigma(\mathcal{T}_1, \mathcal{T}_2) = \sigma(\mathcal{T}_1, \mathcal{T}_3)$ .
2.  $\sigma(\mathcal{T}_2, \mathcal{T}_3) = 1$ .

<sup>1</sup> Assume that  $\mathcal{T}_1 \equiv \mathcal{T}_2$  is equivalent to  $\mathcal{T}_1 \models \mathcal{T}_2$  and  $\mathcal{T}_2 \models \mathcal{T}_1$ .

Our approach starts by considering a simple version of the aimed similarity function, defined as follows:

$$\sigma'(\mathcal{T}_1, \mathcal{T}_2) = \begin{cases} 1 & \text{if } \mathcal{T}_1 \equiv \mathcal{T}_2 \\ 0.5 & \text{else if } \mathcal{T}_1 \models \mathcal{T}_2 \text{ or } \mathcal{T}_2 \models \mathcal{T}_1 \\ 0 & \text{otherwise .} \end{cases} \quad (1)$$

This similarity function is not sufficiently discriminative, which is due to the fact that the definition of the entailment operator requires *every* model of  $\mathcal{T}_1$  to be a model of  $\mathcal{T}_2$  so that  $\mathcal{T}_1 \models \mathcal{T}_2$ . What we wish to achieve is a similarity operator that is a function of the *quantity* of models of  $\mathcal{T}_1$  and  $\mathcal{T}_2$ . However, the amount of models of an ontology is usually infinite. Our approach is to consider a kind of Herbrand interpretation to get around this problem, but instead of redefining the whole logic system, as it is done with Herbrand logic to deal with Herbrand models, we choose to define syntactic elements (concepts), rather than semantic ones (interpretations). Consider the following definitions.

**Definition 1 (Characteristic Concept).** *Let  $\mathcal{C}$  be a set of DL concept names. A characteristic concept wrt  $\mathcal{C}$  is a concept conjunction of the form  $C_1 \sqcap \dots \sqcap C_n$ , where  $C_i$  is either  $A$  or  $\neg A$ , with  $A \in \mathcal{C}$ ,  $n = |\mathcal{C}|$ , and for every  $i \neq j$ ,  $C_i \neq C_j$  and  $C_i \neq \neg C_j$ .  $\zeta(\mathcal{C})$  is the set of all possible characteristic concepts wrt  $\mathcal{C}$ .*

**Definition 2 (Characteristic Disjunction and Axiom).** *Let  $\mathcal{C}$  be a set of DL concept names and  $S \subseteq \zeta(\mathcal{C})$ . The characteristic disjunction of  $S$ ,  $U(S)$ , is the concept  $\bigsqcup_{C \in S} C$ . The characteristic axiom of  $S$ ,  $\theta(U(S))$ , is the axiom  $\top \sqsubseteq U(S)$ .*

**Definition 3 (Characteristic Acceptance Set).** *Let  $\mathcal{T}$  be an ontology containing the set of DL concept names  $\mathcal{C}$ . The characteristic acceptance set of  $\mathcal{T}$ , written  $Z(\mathcal{T})$ , is such that  $Z(\mathcal{T}) \subseteq \zeta(\mathcal{C})$  and  $\mathcal{T} \equiv \theta(Z(\mathcal{T}))$ .*

In other words, a characteristic concept wrt a set of concept names is one of the most specific concepts that is possible to build from them. The characteristic disjunction is the concept disjunction of all the characteristic concepts. Finally, the characteristic acceptance set of an ontology is the set of all characteristic concepts consistent in that ontology. We should note that the characteristic disjunction can also be interpreted as a formula in the disjunctive normal form (DNF). Although normal forms are usually very large, we show how to circumvent this in section 3. Note that the elements of the characteristic acceptance set can be thought of as Herbrand models (with an arbitrary constant). Since the similarity measure we present is heavily based on the characteristic acceptance set, the following lemmas must hold.

**Lemma 1.** *Let  $\mathcal{T}$  be a consistent ontology containing the DL concept names  $\mathcal{C}$ .  $Z(\mathcal{T})$  exists and is unique.*

**Lemma 2.** *Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be consistent ontologies containing the concepts  $\mathcal{C}$ .*

- i.  $\mathcal{T}_1 \models \mathcal{T}_2$  iff  $Z(\mathcal{T}_1) \subseteq Z(\mathcal{T}_2)$ ;*

ii.  $\mathcal{T}_1 \equiv \mathcal{T}_2$  iff  $Z(\mathcal{T}_1) = Z(\mathcal{T}_2)$ .

The proofs of these lemmas can be found in [4]. The acceptance set depends on the number of models of the ontology, since it is the set of most specific consistent concepts in the ontology (i.e., for which there are at least one model). Given these definitions, we are now able to expand the definition of our similarity measure.

**Definition 4 (Semantic Similarity).** Let  $\mathcal{T}_1$  and  $\mathcal{T}_2$  be consistent ontologies containing the concepts  $\mathcal{C}$ . Let  $Z_1 = Z(\mathcal{T}_1)$  and  $Z_2 = Z(\mathcal{T}_2)$ . The semantic similarity measure  $\sigma : \mathcal{O} \times \mathcal{O} \rightarrow [0, 1]$  is defined as follows:

$$\sigma(\mathcal{T}_1, \mathcal{T}_2) = 1 - (|Z_1 - Z_2| + |Z_2 - Z_1|) / 2^{|\mathcal{C}|} . \quad (2)$$

Intuitively, equation 2 measures the accordance of characteristic concepts between both ontologies.

**Theorem 1.** Let  $\mathcal{T}_1$ ,  $\mathcal{T}_2$  and  $\mathcal{T}_3$  be consistent ontologies containing the DL concept names  $\mathcal{C}$ . If  $\mathcal{T}_2 \equiv \mathcal{T}_3$  then  $\sigma(\mathcal{T}_1, \mathcal{T}_2) = \sigma(\mathcal{T}_1, \mathcal{T}_3)$  (i.e.,  $\sigma$  is purely based on semantics).

*Proof.* The result follows immediately from Lemma 2.

*Example 1.* Consider the following ontologies:

$\mathcal{T}_1$	$\mathcal{T}_2$
$\neg \text{Male} \sqsubseteq \text{Female}$	$\text{Person} \sqsubseteq \text{Male} \sqcup \text{Female}$
$\text{Man} \doteq \text{Person} \sqcap \text{Male}$	$\text{Man} \doteq \text{Person} \sqcap \text{Male}$
$\text{Woman} \doteq \text{Person} \sqcap \text{Female}$	$\text{Female} \doteq \neg \text{Male}$
$\text{MaleCat} \doteq \text{Cat} \sqcap \text{Male}$	$\text{Woman} \doteq \text{Person} \sqcap \neg \text{Man}$
	$\text{MaleCat} \sqsubseteq \text{Cat}$

Although similar, these two ontologies display subtle differences. In particular, **Male** or **Female** are necessary in  $\mathcal{T}_1$  (each individual has to be either one or the other, *or both*), but in  $\mathcal{T}_2$  the definition is stricter: each individual is exclusively one or the other. Furthermore, in  $\mathcal{T}_2$  we define **Woman** as a **Person** and not a **Man**. In both ontologies, the concept of **Man** is defined as the intersection of **Person** and **Male**, but in  $\mathcal{T}_1$ , some members of **Man** can also be **Female**. However, in  $\mathcal{T}_2$  it is forbidden for a **Male** to be **Female**, so it restricts the concept of **Man** to individuals who are **Male** and, consequently, not **Female**. Finally, **MaleCat**'s definition in  $\mathcal{T}_2$  is incomplete wrt  $\mathcal{T}_1$ .

Table 1 shows the  $Z(\mathcal{T}_1)$  and  $Z(\mathcal{T}_2)$  sets. Each row is a concept name and each column is a characteristic concept, in such a way that if + (resp. -) is in the intersection of a concept name  $C$  and a characteristic concept  $D$ , then  $C$  appears in  $D$  as a positive (resp. negative) literal.

As can be seen from the table,  $|Z_1 - Z_2| = |Z_2 - Z_1| = 4$ . Equation 2 yields:

$$\sigma(\mathcal{T}_1, \mathcal{T}_2) = 1 - (|Z_1 - Z_2| + |Z_2 - Z_1|) / 2^{|\mathcal{C}|} = 1 - (4 + 4) / 128 = 93.75\% .$$

**Table 1.** The characteristic acceptance sets for  $\mathcal{T}_1$  and  $\mathcal{T}_2$ .

Concept	$Z_1 - Z_2$	$Z_1 \cap Z_2$	$Z_2 - Z_1$
Man	+ - - +	+ + - - - - -	- - + -
Male	+ + + +	+ + - - - - + +	- - + +
Person	+ - - +	+ + + + - - -	- + + -
Cat	+ + - -	+ - - + - + - +	+ + + +
Female	+ + + +	- - + + + + - -	+ + - -
Woman	+ - - +	- - + + - - - -	- + - -
MaleCat	+ + - -	+ - - - - - - +	+ + - -

### 3 Implementation

A naive implementation of this theory could potentially be very inefficient, since  $Z(\mathcal{T})$  grows exponentially in proportion to  $|\mathcal{C}|$ . However, we only need the size of a sub-set of  $Z(\mathcal{T})$ . Given that the characteristic disjunction of a sub-set of  $Z(\mathcal{T})$  is equivalent to a DNF formula, we can use #SAT, which computes the size of the set.

Given two ontologies  $\mathcal{T}_1$  and  $\mathcal{T}_2$ , our implementation starts by computing the concepts  $C_1$  and  $C_2$  such that  $\mathcal{T}_1 \equiv \top \sqsubseteq C_1$  and  $\mathcal{T}_2 \equiv \top \sqsubseteq C_2$ . The purpose is to count the characteristic concepts that are subsumed by  $C_1 \sqcap C_2$  (i.e., that are both in  $Z(\mathcal{T}_1)$  and  $Z(\mathcal{T}_2)$ ) and the ones that are subsumed by  $\neg C_1 \sqcap \neg C_2$  (i.e., that are neither in  $Z(\mathcal{T}_1)$  nor  $Z(\mathcal{T}_2)$ ). To achieve this, we represent the concept  $C_1 \sqcap C_2 \sqcup \neg C_1 \sqcap \neg C_2$  as a CNF formula and feed it to a #SAT solver. To transform the concept into CNF we use the Definitional CNF Transformation algorithm (CNF with naming). Note that the following holds:

$$\sigma(\mathcal{T}_1, \mathcal{T}_2) = \text{mc}(\text{cnf}(C_1 \sqcap C_2 \sqcup \neg C_1 \sqcap \neg C_2)) / 2^{|\mathcal{C}|}, \tag{3}$$

where mc is the model count and cnf is the CNF representation of the formula. To perform model counting we use the RELSAT tool [5]. We should note that the computation is not performed exactly as defined in equation 3. We observed that RELSAT performed considerably faster using the following equivalent equation:

$$\sigma(\mathcal{T}_1, \mathcal{T}_2) = \left( (2^{|\mathcal{C}|} - \text{mc}(\text{cnf}(C_1))) - \text{mc}(\text{cnf}(C_2)) + 2 \times \text{mc}(\text{cnf}(C_1 \sqcap C_2)) \right) / 2^{|\mathcal{C}|}.$$

### 4 Experiment

In this experiment, we were aiming at evaluating our similarity measure against an intuition of similarity. This measure is only applicable to ontologies sharing the same concept names, but the lack of such ontologies thwarts the direct employment of the measure. It is thus necessary to map a set of ontologies in the same domain. Then, the set of concepts involved in the mapping are cropped, so that the ontologies that are to be compared contain the same set of concept names.

As dataset we used three ontologies in the cooking domain. The first one is called ONTOCHEF<sub>GS</sub> ( $O_{GS}$ ) and can be seen as a gold standard, as its development was carried out more zealously and by a bigger team than the others [6]. The other two, ONTOCHEF<sub>1</sub> and ONTOCHEF<sub>2</sub> ( $O_1$  and  $O_2$ ), were developed by students at an undergraduate course on Knowledge Representation. The selection of these ontologies was based on their correctness and thoroughness. Many contained axioms such as Preparation  $\sqsubseteq$  Recipe, using subsumption incorrectly and were ruled out. The ontologies were required to define at least: recipes, measurements, (kitchen) tools and ingredients/food, so we ruled out the ones that were not sufficiently thorough on (or completely neglected) these topics. Figure 1 shows a section of each ontology.  $O_1$  has 49 concept names, while  $O_2$  has 167 and  $O_{GS}$  has 571.

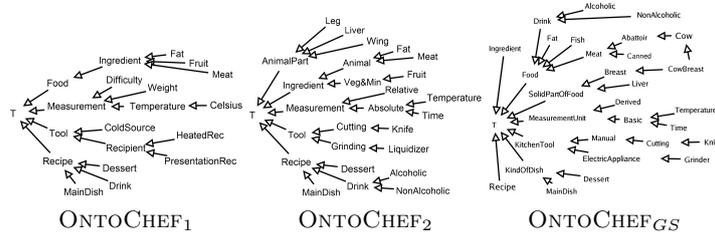


Fig. 1. A relevant part of the ontologies.

Despite obvious dissimilarities, there is an overlap of concepts in the ontologies. For example, both  $O_1$  and  $O_2$  characterise dishes as Recipes, while in  $O_{GS}$  these are subsumed by KindOfDish, which intuitively makes more sense. Also, in  $O_{GS}$ , Salad is subsumed by Starters, but in  $O_1$  and  $O_2$  they are at the same level as the other kinds of dishes, which shows that even the gold standard can be (and usually is) less than perfect, since salads are not necessarily starters.

The results of applying the similarity measure are as follows:

$$\sigma(O_1, O_2) = 98.1134\%, \sigma(O_1, O_{GS}) = 94.8180\%, \sigma(O_2, O_{GS}) = 94.0139\%$$

## 5 Evaluation and Discussion

Although it is not clear from figure 1 that these ontologies are as similar as assessed by the proposed measure, their cropped sections display many similarities. Thus, we can say that the measure is on a par with our intuition of similarity. We can also observe that when a set of values is available, comparing the different values is a reasonable way to establish which ontologies are more or less similar to an ontology. In the previous section we observed that the two ontologies built by the undergraduate students were more similar, which is an intuitive outcome. This is mostly due to the fact that each kind of dish is considered as a Recipe in  $O_1$  and  $O_2$ , and also that the kinds of Ingredient in these ontologies are considered as Food in  $O_{GS}$ . Furthermore, the similarity between  $O_1$  and  $O_{GS}$  is slightly higher than the similarity between  $O_2$  and  $O_{GS}$ . This

happens mainly because `Cup`, `TeaSpoon` and `SoupSpoon` are represented in  $O_1$  as volume measurement units and in  $O_2$  they are tools.

A possible use-case we envision for our similarity measure is the automatic assessment of a learnt ontology against a gold standard, assuming the learnt ontology has the same concepts as the gold standard, or a sub-set of them. Also, our measure could be used in an ontology merging system that would search an ontology library for similar ontologies and propose extending the source ontology with axioms and concepts from the most similar ontologies.

In [7], the authors present a set of reasonable criteria for assessing the quality of a similarity measure. It can be shown that our measure respects the *proportional error effect* and the *usage of interval* criteria. It is also worth mentioning that the measure is, indeed, a similarity measure as it is usually defined (e.g. [8]).

Although our implementation is based on  $\#SAT$ , which is NP-HARD, the use of heuristics boost the efficiency of the  $\#SAT$  solvers, and can deliver results for ontologies containing more than 500 concepts, in less than 10 seconds. We consider this to be acceptable.

## 6 Related Work

Some alignment algorithms and tools have been developed, many of which are described in [1]. In this survey it is mentioned that only 4 out of the 21 systems analysed rely directly on semantic properties of the ontologies: S-Match [9], Buster [10], Chimarae [11] and KILT [12]. There are also approaches to similarity in DL formalisms, such as [13]. In this work, Hu *et al.* present a method for calculating distances between concepts based on their signatures. A concept signature is the set of elements that a concept is dependent of, which is determined using tableaux-like reasoning rules. Their approach starts by computing the signatures of concepts and counting the times each element (atomic concept and role) appears in the signature and fine-tuning it using information retrieval techniques. They define the distance between ontologies by aggregating the distances between their different components. Herein lies an advantage of their work: they define similarity on many levels; our work focuses on ontologies as wholes. An advantage of our work is that our measure is bounded between 0 and 1, as opposed to their work which can yield any (possibly negative) number, and thus cannot be strictly considered as a similarity measure since it does not hold the positive definiteness condition. Other work in DL similarity can be found in [8, 14, 15].

## 7 Conclusions and Future Work

In this paper we present a semantic similarity measure for a sub-set of the  $\mathcal{ALC}$  Description Logic. We show some properties of the measure, how it can be applied to average-sized ontologies and that the results yielded roughly correspond to our intuitive notion of similarity.

In the future, we would like to tackle the efficiency and expressiveness problems. A formal analysis of the algorithm should be done in order to provide a deeper understanding of the limitations of the current implementation. We would like to add the possibility of having a weighing factor in the form of a probability distribution over concepts. Finally, we should apply it to a use-case, namely in the automatic assessment of learnt ontologies against a gold standard.

## References

1. Ehrig, M., Euzenat, J.: State of the art on ontology alignment. Knowledge Web Deliverable 2.2.3, University of Karlsruhe (2004)
2. Hliaoutakis, A., Varelas, G., Voutsakis, E., Petrakis, E.G.M., Milios, E.: Information retrieval by semantic similarity. *Int'l Journal on Semantic Web & Information Systems* **2**(3) (2006) 55–73
3. Nardi, D., Brachman, R.J.: An introduction to description logics. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: *Description Logic Handbook*, Cambridge University Press (2003) 1–40
4. Araújo, R., Pinto, H.S.: SEMilarity theory. Technical report, INESC-ID (2007) See <http://algos.inesc-id.pt/~rnna/semilarity-theory.pdf>.
5. Jr., R.J.B., Pehoushek, J.D.: Counting models using connected components. In: *AAAI/IAAI*. (2000) 157–162
6. Ribeiro, R., Batista, F., Pardal, J.P., Mamede, N.J., Pinto, H.S.: Cooking an ontology. In Euzenat, J., Domingue, J., eds.: *AIMSA*. Volume 4183 of *LNCS.*, Springer (2006) 213–221
7. Dellschaft, K., Staab, S.: On how to perform a gold standard based evaluation of ontology learning. In Cruz, I.F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L., eds.: *Proc. of ISWC 2006*. Volume 4273 of *LNCS.*, Springer (2006) 228–241
8. d’Amato, C., Fanizzi, N., Esposito, F.: A dissimilarity measure for ALC concept descriptions. In Haddad, H., ed.: *Proc. of the 2006 ACM Symposium on Applied Computing (SAC)*, ACM (2006) 1695–1699
9. Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-match: an algorithm and an implementation of semantic matching. Technical report, University of Trento (2004)
10. Vögele, T., Hübner, S., Schuster, G.: BUSTER - an information broker for the semantic web. *KI* **17**(3) (2003) 31
11. McGuinness, D., Fikes, R., Rice, J., Wilder, S.: An environment for merging and testing large ontologies. In: *Proc. of KR 2000*, Morgan Kaufmann (2000) 483–493
12. d’Aquin, Mathieu, Bouthier, C., Brachais, S., Lieber, J., Napoli, A.: Knowledge editing and maintenance tools for a semantic portal in oncology. *Int'l Journal of Human-Computer Studies* **62**(5) (2005) 619–638
13. Hu, B., Kalfoglou, Y., Alani, H., Dupplaw, D., Lewis, P.H., Shadbolt, N.: Semantic metrics. In Staab, S., Svátek, V., eds.: *Proc. of EKAW 2006*. Volume 4248 of *LNCS.*, Springer (2006) 166–181
14. Borgida, A., Walsh, T., Hirsh, H.: Towards measuring similarity in description logics. In Horrocks, I., Sattler, U., Wolter, F., eds.: *Proc. of DL2005*. Volume 147 of *CEUR Workshop Proceedings.*, CEUR-WS.org (2005)
15. Janowicz, K.: Sim-DL: Towards a semantic similarity measurement theory for the description logic ALCNR in geographic information retrieval. In Meersman, R., Tari, Z., Herrero, P., eds.: *On the Move to Meaningful Internet Systems. Proc., Part II*. Volume 4278 of *LNCS.*, Springer (2006) 1681–1692

## Complexity of Reasoning over Entity-Relationship Models<sup>\*</sup>

A. Artale<sup>1</sup>, D. Calvanese<sup>1</sup>, R. Kontchakov<sup>2</sup>, V. Ryzhikov<sup>1</sup> and  
M. Zakharyashev<sup>2</sup>

<sup>1</sup> Faculty of Computer Science  
Free University of Bozen-Bolzano  
I-39100 Bolzano, Italy  
`lastname@inf.unibz.it`

<sup>2</sup> School of Comp. Science and Inf. Sys.  
Birkbeck College  
London WC1E 7HX, UK  
`{roman,michael}@dcs.bbk.ac.uk`

**Abstract.** We investigate the complexity of reasoning over various fragments of the Extended Entity-Relationship (EER) language, which include different combinations of the constructors for ISA between concepts and relationships, disjointness, covering, cardinality constraints and their refinement. Specifically, we show that reasoning over EER diagrams with ISA between relationships is EXPTIME-complete even when we drop both covering and disjointness for relationships. Surprisingly, when we also drop ISA between relations, reasoning becomes NP-complete. If we further remove the possibility to express covering between entities, reasoning becomes polynomial. Our lower bound results are established by direct reductions, while the upper bounds follow from correspondences with expressive variants of the description logic *DL-Lite*. The established correspondence shows also the usefulness of *DL-Lite* as a language for reasoning over conceptual models and ontologies.

### 1 Introduction

Conceptual modelling formalisms, such as the Entity-Relationship model [1], are used in the phase of conceptual database design where the aim is to capture at best the semantics of the modelled application. This is achieved by expressing constraints that hold on the concepts, attributes and relations representing the domain of interest through suitable constructors provided by the conceptual modelling language. Thus, on the one hand it would be desirable to make such a language as expressive as possible in order to represent as many aspects of the modelled reality as possible. On the other hand, when using an expressive language, the designer faces the problem of understanding the complex interactions between different parts of the conceptual model under construction and the constraints therein. Such interactions may force, e.g., some class (or even all classes) in the model to become inconsistent in the sense that there cannot be any database state satisfying all constraints in which the class (respectively, all classes) is populated by at least one object. Or a class may be implied to be a subclass of another one, even if this is not explicitly asserted in the model.

<sup>\*</sup> Authors partially supported by the U.K. EPSRC grant GR/S63175, Tones, KnowledgeWeb and InterOp EU projects, and by the PRIN project funded by MIUR.

To understand the consequences, both explicit and implicit, of the constraints in the conceptual model being constructed, it is thus essential to provide for an automated reasoning support.

In this paper, we address these issues and investigate the complexity of reasoning in conceptual modelling languages equipped with various forms of constraints. We carry out our analysis in the context of the Extended Entity-Relationship (EER) language [2], where the domain of interest is represented via *entities* (representing sets of objects), possibly equipped with *attributes*, and *relationships* (representing relations over objects)<sup>1</sup>. Specifically, the kind of constraints that will be taken into account in this paper are the ones typically used in conceptual modelling, namely:

- *is-a* relations between both entities and relationships;
- *disjointness* and *covering* (referred to as the *Boolean* constructors in what follows) between both entities and relationships;
- *cardinality* constraints for participation of entities in relationships;
- *refinement* of cardinalities for sub-entities participating in relationships; and
- *multiplicity* constraints for attributes.

The hierarchy of EER languages we consider here is shown in the table below together with the complexity results for reasoning in these languages (all our languages include cardinality, refinement and multiplicity constraints).

lang.	entities			relationships			complexity
	ISA	disjoint	covering	ISA	disjoint	covering	
	$C_1 \sqsubseteq C_2$	$C_1 \sqcap C_2 \sqsubseteq \perp$	$C = C_1 \sqcup C_2$	$R_1 \sqsubseteq R_2$	$R_1 \sqcap R_2 \sqsubseteq \perp$	$R = R_1 \sqcup R_2$	
$ER_{full}$	+	+	+	+	+	+	EXPTIME [3]
$ER_{isaR}$	+	+	+	+	–	–	EXPTIME
$ER_{bool}$	+	+	+	–	–	–	NP
$ER_{ref}$	+	+	–	–	–	–	NLOGSPACE

According to [3] reasoning over UML class diagrams is EXPTIME-complete, and it is easy to see that the same holds for  $ER_{full}$  diagrams as well (cf. e.g., [4]). Here we strengthen this result by showing (using reification) that reasoning is still EXPTIME-complete for its sublanguage  $ER_{isaR}$ . The NP upper bound for  $ER_{bool}$  is proved by embedding  $ER_{bool}$  into  $DL-Lite_{bool}$ , the *Boolean extension* of the tractable DL  $DL-Lite$  [5, 6]. Thus, quite surprisingly, ISA between relationships alone is a major source of complexity of reasoning over conceptual schemas. Finally, we show that  $ER_{ref}$  is closely related to  $DL-Lite_{krom}$ , the Krom fragment of  $DL-Lite_{bool}$ , and that reasoning in it is polynomial. The correspondence between modelling languages like  $ER_{bool}$  and DLs like  $DL-Lite_{bool}$  shows that the  $DL-Lite$  family are useful languages for reasoning over conceptual models and ontologies, even though they are not equipped with all the constructors that are typical of rich ontology languages such as OWL and its variants [7].

Our analysis is in spirit similar to [8], where the consistency checking problem for an EER model equipped with forms of inclusion and disjointness constraints is studied and a polynomial-time algorithm for the problem is given (assuming constant arities of relationships). Such a polynomial-time result is incomparable

<sup>1</sup> Our results can be adapted to other modelling formalisms, such as UML diagrams.

with the one for  $ER_{ref}$ , since  $ER_{ref}$  lacks both ISA and disjointness for relationships (both present in [8]); on the other hand, it is equipped with cardinality and multiplicity constraints. We also mention [9], where reasoning over cardinality constraints in the basic ER model is investigated and a polynomial-time algorithm for strong schema consistency is given, and [10], where the study is extended to the case where ISA between entities is also allowed and an exponential algorithm for entity consistency is provided. Note, however, that in [9, 10] the reasoning problem is analysed under the assumption that databases are finite, whereas we do not require finiteness in this paper.

## 2 The *DL-Lite* Language

We consider the extension *DL-Lite<sub>bool</sub>* [6] of the description logic *DL-Lite* [11, 5]. The language of *DL-Lite<sub>bool</sub>* contains *concept names*  $A_0, A_1, \dots$  and *role names*  $P_0, P_1, \dots$ . Complex *roles*  $R$  and *concepts*  $C$  of *DL-Lite<sub>bool</sub>* are defined as follows:

$$\begin{aligned} R & ::= P_i \mid P_i^-, \\ B & ::= \perp \mid A_i \mid \geq q R, \\ C & ::= B \mid \neg C \mid C_1 \sqcap C_2, \end{aligned}$$

where  $q \geq 1$ . Concepts of the form  $B$  are called *basic concepts*. A *DL-Lite<sub>bool</sub> knowledge base* is a finite set of axioms of the form  $C_1 \sqsubseteq C_2$ . A *DL-Lite<sub>bool</sub> interpretation*  $\mathcal{I}$  is a structure  $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where  $\Delta^{\mathcal{I}} \neq \emptyset$  and  $\cdot^{\mathcal{I}}$  is a function such that  $A_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ , for all  $A_i$ , and  $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ , for all  $P_i$ . The role and concept constructors are interpreted in  $\mathcal{I}$  as usual. We also make use of the standard abbreviations:  $\top := \neg \perp$ ,  $\exists R := (\geq 1 R)$  and  $\leq q R := \neg(\geq q + 1 R)$ . We say that  $\mathcal{I}$  *satisfies* an axiom  $C_1 \sqsubseteq C_2$  if  $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ . A knowledge base  $\mathcal{K}$  is *satisfiable* if there is an interpretation  $\mathcal{I}$  that satisfies all the axioms of  $\mathcal{K}$  (such an  $\mathcal{I}$  is called a *model* of  $\mathcal{K}$ ). A concept  $C$  is *satisfiable w.r.t.*  $\mathcal{K}$  if there is a model  $\mathcal{I}$  of  $\mathcal{K}$  such that  $C^{\mathcal{I}} \neq \emptyset$ .

We also consider a sub-language *DL-Lite<sub>krom</sub>* of *DL-Lite<sub>bool</sub>*, called the *Krom fragment*, where only axioms of the following form are allowed (with  $B_i$  basic concepts):

$$B_1 \sqsubseteq B_2, \quad B_1 \sqsubseteq \neg B_2, \quad \neg B_1 \sqsubseteq B_2,$$

**Theorem 1 ([6]).** *Concept and KB satisfiability are NP-complete for DL-Lite<sub>bool</sub> KBs and NLOGSPACE-complete for DL-Lite<sub>krom</sub> KBs.*

## 3 The Conceptual Modelling Language

In this section, we define the notion of a *conceptual schema* by providing its syntax and semantics for the fully-fledged conceptual modelling language *ER<sub>full</sub>*. First citizens of a conceptual schema are *entities*, *relationships* and *attributes*. Arguments of relationships—specifying the role played by an entity when participating in a particular relationship—are called *roles*. Given a conceptual schema, we make the following assumptions: relationship and entity names are unique; attribute names are local to entities (i.e., the same attribute may be used by

different entities; its type, however, must be the same); role names are local to relationships (this freedom will be limited when considering conceptual models without sub-relationships).

Given a finite set  $X = \{x_1, \dots, x_n\}$  and a set  $Y$ , an  $X$ -labelled tuple over  $Y$  is a (total) function  $T: X \rightarrow Y$ . The element  $T[x] \in Y$  is said to be *labelled* by  $x$ ; we also write  $(x, y) \in T$  if  $y = T[x]$ . The set of all  $X$ -labelled tuples over  $Y$  is denoted by  $T_Y(X)$ . For  $y_1, \dots, y_n \in Y$ , the expression  $\langle x_1: y_1, \dots, x_n: y_n \rangle$  denotes  $T \in T_Y(X)$  such that  $T[x_i] = y_i$ , for  $1 \leq i \leq n$ .

**Definition 1 ( $ER_{full}$  syntax).** An  $ER_{full}$  conceptual schema  $\Sigma$  is a tuple of the form  $(\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}_R, \text{CARD}_A, \text{REF}, \text{ISA}, \text{DISJ}, \text{COV})$ , where

- $\mathcal{L}$  is the disjoint union of alphabets  $\mathcal{E}$  of *entity* symbols,  $\mathcal{A}$  of *attribute* symbols,  $\mathcal{R}$  of *relationship* symbols,  $\mathcal{U}$  of *role* symbols and  $\mathcal{D}$  of *domain* symbols; the tuple  $(\mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{U}, \mathcal{D})$  is called the *signature* of the schema  $\Sigma$ .
- REL is a function assigning to every relationship symbol  $R \in \mathcal{R}$  a tuple  $\text{REL}(R) = \langle U_1: E_1, \dots, U_m: E_m \rangle$  over the entity symbols  $\mathcal{E}$  labelled with a non-empty set  $\{U_1, \dots, U_m\}$  of role symbols;  $m$  is called the *arity* of  $R$ .
- ATT is a function that assigns to every entity symbol  $E \in \mathcal{E}$  a tuple  $\text{ATT}(E)$ ,  $\text{ATT}(E) = \langle A_1: D_1, \dots, A_h: D_h \rangle$ , over the domain symbols  $\mathcal{D}$  labelled with some (possibly empty) set  $\{A_1, \dots, A_h\}$  of attribute symbols.
- $\text{CARD}_R: \mathcal{R} \times \mathcal{U} \times \mathcal{E} \rightarrow \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$  is a *partial* function (called *cardinality constraints*);  $\text{CARD}_R(R, U, E)$  may be defined only if  $(U, E) \in \text{REL}(R)$ .
- $\text{CARD}_A: \mathcal{A} \times \mathcal{E} \rightarrow \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$  is a *partial* function (called *multiplicity of attributes*);  $\text{CARD}_A(A, E)$  may be defined only if  $(A, D) \in \text{ATT}(E)$ , for some  $D \in \mathcal{D}$ .
- $\text{REF}: \mathcal{R} \times \mathcal{U} \times \mathcal{E} \rightarrow \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$  is a *partial* function (called *refinement of cardinality constraints*);  $\text{REF}(R, U, E)$  may be defined only if  $E \text{ ISA } E'$  and  $(U, E') \in \text{REL}(R)$ ; note that REF subsumes cardinality constraints  $\text{CARD}_R$ .
- $\text{ISA} = \text{ISA}_E \cup \text{ISA}_R$ , where  $\text{ISA}_R \subseteq \mathcal{E} \times \mathcal{E}$  and  $\text{ISA}_R \subseteq \mathcal{R} \times \mathcal{R}$ .
- $\text{DISJ} = \text{DISJ}_E \cup \text{DISJ}_R$  and  $\text{COV} = \text{COV}_E \cup \text{COV}_R$ , where  $\text{DISJ}_E, \text{COV}_E \subseteq 2^{\mathcal{E}} \times \mathcal{E}$  and  $\text{DISJ}_R, \text{COV}_R \subseteq 2^{\mathcal{R}} \times \mathcal{R}$ .

$\text{ISA}_R$ ,  $\text{DISJ}_R$  and  $\text{COV}_R$  may only be defined for relationships of the same arity. In what follows we also use infix notation for relations  $\text{ISA}$ ,  $\text{ISA}_E$ , etc.

**Definition 2 ( $ER_{full}$  semantics).** Let  $\Sigma$  be an  $ER_{full}$  conceptual schema and  $B_D$ , for  $D \in \mathcal{D}$ , a collection of disjoint countable sets called *basic domains*. An *interpretation* of  $\Sigma$  is a pair  $\mathcal{B} = (\Delta^{\mathcal{B}} \cup \Lambda^{\mathcal{B}}, \cdot^{\mathcal{B}})$ , where  $\Delta^{\mathcal{B}} \neq \emptyset$  is the *interpretation domain*;  $\Lambda^{\mathcal{B}} = \bigcup_{D \in \mathcal{D}} \Lambda_D^{\mathcal{B}}$ , with  $\Lambda_D^{\mathcal{B}} \subseteq B_D$  for each  $D \in \mathcal{D}$ , is the *active domain* such that  $\Delta^{\mathcal{B}} \cap \Lambda^{\mathcal{B}} = \emptyset$ ;  $\cdot^{\mathcal{B}}$  is a function such that  $E^{\mathcal{B}} \subseteq \Delta^{\mathcal{B}}$ , for each  $E \in \mathcal{E}$ ,  $A^{\mathcal{B}} \subseteq \Delta^{\mathcal{B}} \times \Lambda^{\mathcal{B}}$ , for each  $A \in \mathcal{A}$ ,  $R^{\mathcal{B}} \subseteq T_{\Delta^{\mathcal{B}}}(\mathcal{U})$ , for each  $R \in \mathcal{R}$ ; and  $D^{\mathcal{B}} = \Lambda_D^{\mathcal{B}}$ , for each  $D \in \mathcal{D}$ . An interpretation  $\mathcal{B}$  of  $\Sigma$  is called a *legal database state* if the following holds:

1. for each  $R \in \mathcal{R}$  with  $\text{REL}(R) = \langle U_1: E_1, \dots, U_m: E_m \rangle$  and each  $1 \leq i \leq m$ ,
  - for all  $r \in R^{\mathcal{B}}$ ,  $r = \langle U_1: e_1, \dots, U_m: e_m \rangle$  and  $e_i \in E_i^{\mathcal{B}}$ ;
  - if  $\text{CARD}_R(R, U_i, E_i) = (\alpha, \beta)$  then  $\alpha \leq \#\{r \in R^{\mathcal{B}} \mid (U_i, e_i) \in r\} \leq \beta$ , for all  $e_i \in E_i^{\mathcal{B}}$ ;

- if  $\text{REF}(R, U_i, E) = (\alpha, \beta)$ , for  $E \in \mathcal{E}$  with  $E \text{ ISA } E_i$ , then, for all  $e \in E^{\mathcal{B}}$ ,  $\alpha \leq \#\{r \in R^{\mathcal{B}} \mid (U_i, e) \in r\} \leq \beta$ ;
- 2. for each  $E \in \mathcal{E}$  with  $\text{ATT}(E) = \langle A_1: D_1, \dots, A_h: D_h \rangle$  and each  $1 \leq i \leq h$ ,
  - for all  $(e, a) \in \Delta^{\mathcal{B}} \times A^{\mathcal{B}}$ , if  $(e, a) \in A_i^{\mathcal{B}}$  then  $a \in D_i^{\mathcal{B}}$ ;
  - if  $\text{CARD}_A(A_i, E) = (\alpha, \beta)$  then  $\alpha \leq \#\{(e, a) \in A_i^{\mathcal{B}}\} \leq \beta$ , for all  $e \in E^{\mathcal{B}}$ ;
- 3. for all  $E_1, E_2 \in \mathcal{E}$ , if  $E_1 \text{ ISA}_E E_2$  then  $E_1^{\mathcal{B}} \subseteq E_2^{\mathcal{B}}$  (similarly for relationships);
- 4. for all  $E, E_1, \dots, E_n \in \mathcal{E}$ , if  $\{E_1, \dots, E_n\} \text{ DISJ}_E E$  then  $E_i^{\mathcal{B}} \subseteq E^{\mathcal{B}}$ , for every  $1 \leq i \leq n$ , and  $E_i^{\mathcal{B}} \cap E_j^{\mathcal{B}} = \emptyset$ , for  $1 \leq i < j \leq n$  (similarly for relationships);
- 5. for all  $E, E_1, \dots, E_n \in \mathcal{E}$ ,  $\{E_1, \dots, E_n\} \text{ COV}_E E$  implies  $E^{\mathcal{B}} = \bigcup_{i=1}^n E_i^{\mathcal{B}}$  (similarly for relationships).

Reasoning tasks over conceptual schemas include verifying whether an entity, a relationship, or a schema is *consistent*, or checking whether an entity (or a relationship) *subsumes* another entity (relationship, respectively):

**Definition 3 (Reasoning services).** Let  $\Sigma$  be an  $ER_{full}$  schema.

- $\Sigma$  is *consistent (strongly consistent)* if there exists a legal database state  $\mathcal{B}$  for  $\Sigma$  such that  $E^{\mathcal{B}} \neq \emptyset$ , for some (*every*, respectively) entity  $E \in \mathcal{E}$ .
- An entity  $E \in \mathcal{E}$  (relationship  $R \in \mathcal{R}$ ) is *consistent w.r.t.  $\Sigma$*  if there exists a legal database state  $\mathcal{B}$  for  $\Sigma$  such that  $E^{\mathcal{B}} \neq \emptyset$  ( $R^{\mathcal{B}} \neq \emptyset$ , respectively).
- An entity  $E_1 \in \mathcal{E}$  (relationship  $R_1 \in \mathcal{R}$ ) *subsumes* an entity  $E_2 \in \mathcal{E}$  (relationship  $R_2 \in \mathcal{R}$ ) w.r.t.  $\Sigma$  if  $E_2^{\mathcal{B}} \subseteq E_1^{\mathcal{B}}$  ( $R_2^{\mathcal{B}} \subseteq R_1^{\mathcal{B}}$ , respectively), for every legal database state  $\mathcal{B}$  for  $\Sigma$ .

One can show that the reasoning tasks of schema/entity/relationship consistency and entity subsumption are reducible to each other. (Note that in the absence of the covering constructor schema consistency cannot be reduced to a single instance of entity consistency, though it can be reduced to several entity consistency checks.) Due to these equivalences, in the following we will consider entity consistency as the main reasoning service.

## 4 Complexity of Reasoning in EER Languages

This section shows the complexity results obtained in this paper for reasoning over different EER languages (All proofs can be found at <http://www.inf.unibz.it/~artale/papers/dl07-full.pdf>.)

*Reasoning over  $ER_{isaR}$  schemas.* The modelling language  $ER_{isaR}$  is the subset of  $ER_{full}$  without the Booleans between relationships (i.e.,  $\text{DISJ}_R = \emptyset$  and  $\text{COV}_R = \emptyset$ ) but with the possibility to express ISA between them. We establish an EXPTIME lower bound for satisfiability of  $ER_{isaR}$  conceptual schemas by reduction of the satisfiability problem for  $\mathcal{ALC}$  knowledge bases. It is easy to show (see, e.g., [3, Lemma 5.1]) that one can convert, in a satisfiability preserving way, an  $\mathcal{ALC}$  KB  $\mathcal{K}$  into a *primitive* KB  $\mathcal{K}'$  that contains only axioms of the form:  $A \sqsubseteq B, A \sqsubseteq \neg B, A \sqsubseteq B \sqcup B', A \sqsubseteq \forall R.B, A \sqsubseteq \exists R.B$ , where  $A, B, B'$  are concept



$\Sigma$  is defined as follows:

$$\begin{aligned} \tau(\Sigma) = & \tau_{dom} \cup \bigcup_{R \in \mathcal{R}} [\tau_{rel}^R \cup \tau_{card_R}^R \cup \tau_{ref}^R] \cup \bigcup_{E \in \mathcal{E}} [\tau_{att}^E \cup \tau_{card_A}^E] \cup \\ & \bigcup_{\substack{E_1, E_2 \in \mathcal{E} \\ E_1 \text{ ISA } E_2}} \tau_{isa}^{E_1, E_2} \cup \bigcup_{\substack{E_1, \dots, E_n, E \in \mathcal{E} \\ \{E_1, \dots, E_n\} \text{ DISJ } E}} \tau_{disj}^{\{E_1, \dots, E_n\}, E} \cup \bigcup_{\substack{E_1, \dots, E_n, E \in \mathcal{E} \\ \{E_1, \dots, E_n\} \text{ COV } E}} \tau_{cov}^{\{E_1, \dots, E_n\}, E}, \end{aligned}$$

where

- $\tau_{dom} = \{\bar{D} \sqsubseteq \neg \bar{X} \mid D \in \mathcal{D}, X \in \mathcal{E} \cup \mathcal{R} \cup \mathcal{D}, D \neq X\}$ ;
- $\tau_{rel}^R = \{\bar{R} \sqsubseteq \exists \bar{U}, \geq 2 \bar{U} \sqsubseteq \perp, \exists \bar{U} \sqsubseteq \bar{R}, \exists \bar{U}^- \sqsubseteq \bar{E} \mid (U, E) \in \text{REL}(R)\}$ ;
- $\tau_{card_R}^R = \{\bar{E} \sqsubseteq \geq \alpha \bar{U}^- \mid (U, E) \in \text{REL}(R), \text{CARD}_R(R, U, E) = (\alpha, \beta), \alpha \neq 0\}$   
 $\cup \{\bar{E} \sqsubseteq \leq \beta \bar{U}^- \mid (U, E) \in \text{REL}(R), \text{CARD}_R(R, U, E) = (\alpha, \beta), \beta \neq \infty\}$ ;
- $\tau_{ref}^R = \{\bar{E} \sqsubseteq \geq \alpha \bar{U}^- \mid (U, E) \in \text{REL}(R), \text{REF}(R, U, E) = (\alpha, \beta), \alpha \neq 0\}$   
 $\cup \{\bar{E} \sqsubseteq \leq \beta \bar{U}^- \mid (U, E) \in \text{REL}(R), \text{REF}(R, U, E) = (\alpha, \beta), \beta \neq \infty\}$ ;
- $\tau_{att}^E = \{\exists \bar{A}^- \sqsubseteq \bar{D} \mid (A, D) \in \text{ATT}(E)\}$ ;
- $\tau_{card_A}^E = \{\bar{E} \sqsubseteq \geq \alpha \bar{A} \mid (A, D) \in \text{ATT}(E), \text{CARD}_A(A, E) = (\alpha, \beta), \alpha \neq 0\}$   
 $\cup \{\bar{E} \sqsubseteq \leq \beta \bar{A} \mid (A, D) \in \text{ATT}(E), \text{CARD}_A(A, E) = (\alpha, \beta), \beta \neq \infty\}$ ;
- $\tau_{isa}^{E_1, E_2} = \{\bar{E}_1 \sqsubseteq \bar{E}_2\}$ ;
- $\tau_{disj}^{\{E_1, \dots, E_n\}, E} = \{\bar{E}_i \sqsubseteq \bar{E} \mid 1 \leq i \leq n\} \cup \{\bar{E}_i \sqsubseteq \neg \bar{E}_j \mid 1 \leq i < j \leq n\}$ ;
- $\tau_{cov}^{\{E_1, \dots, E_n\}, E} = \{\bar{E}_i \sqsubseteq \bar{E} \mid 1 \leq i \leq n\} \cup \{\bar{E} \sqsubseteq \bar{E}_1 \sqcup \dots \sqcup \bar{E}_n\}$ .

Clearly, the size of  $\tau(\Sigma)$  is polynomial in the size of  $\Sigma$ .

**Lemma 2.** *An entity  $E$  is consistent w.r.t. an  $ER_{bool}$  schema  $\Sigma$  iff the concept  $\bar{E}$  is satisfiable w.r.t. the  $DL\text{-Lite}_{bool}$  KB  $\tau(\Sigma)$ .*

**Theorem 3.** *Reasoning over  $ER_{bool}$  conceptual schemas is NP-complete.*

The upper bound is proved by Lemma 2 and Theorem 1; the lower one is by reduction of the NP-complete 3SAT problem to entity consistency for  $ER_{bool}$  schemas.

*Reasoning over  $ER_{ref}$  schemas.* Denote by  $ER_{ref}$  the modelling language without the Booleans and ISA between relationships, but with the possibility to express ISA and disjointness between entities (i.e.,  $\text{DISJ}_R = \emptyset$ ,  $\text{COV}_R = \emptyset$ ,  $\text{ISA}_R = \emptyset$  and  $\text{COV}_E = \emptyset$ ). Thus,  $ER_{ref}$  is essentially  $ER_{bool}$  without covering.

**Theorem 4.** *The entity consistency problem for  $ER_{ref}$  is NLOGSPACE-complete.*

The upper bound follows from the fact that for any  $ER_{ref}$  schema,  $\Sigma$ ,  $\tau(\Sigma)$  is a  $DL\text{-Lite}_{krom}$  KB ( $\tau_{cov} = \emptyset$ ). Thus, by Lemma 2, the entity consistency problem for  $ER_{ref}$  can be reduced to concept satisfiability for  $DL\text{-Lite}_{krom}$  KBs, which is NLOGSPACE-complete (see Theorem 1), while the reduction can be proved

to be computed in logspace. The lower bound is obtained by reduction of the non-reachability problem in oriented graphs (the non-reachability problem is known to be  $\text{CONLOGSPACE}$ -complete and so, it is  $\text{NLOGSPACE}$ -complete as these classes coincide by the Immerman-Szelepcsényi theorem; see, e.g., [12]).

## 5 Conclusions

This paper provides new complexity results for reasoning over Extended Entity-Relationship (EER) models with different modelling constructors. Starting from the  $\text{EXPTIME}$  result [3] for reasoning over the fully-fledged EER language, we prove that the same complexity holds even if the Boolean constructors (disjointness and covering) on relationships are dropped. This result shows that ISA between relationships (with the Booleans on entities) is powerful enough to capture  $\text{EXPTIME}$ -hard problems. To illustrate that the presence of relationship hierarchies is a major source of complexity in reasoning we show that avoiding them makes reasoning in  $ER_{bool}$  an NP-complete problem. Another source of complexity is the covering constraint. Indeed, without relationship hierarchies and covering constraints reasoning problem for  $ER_{ref}$  is  $\text{NLOGSPACE}$ -complete.

The paper also provides a tight correspondence between conceptual modelling languages and the *DL-Lite* family of description logics and shows the usefulness of *DL-Lite* in representing and reasoning over conceptual models and ontologies.

## References

1. Batini, C., Ceri, S., Navathe, S.B.: Conceptual Database Design, an Entity-Relationship Approach. Benjamin and Cummings Publ. Co. (1992)
2. ElMasri, R.A., Navathe, S.B.: Fundamentals of Database Systems. 5th edn. Addison Wesley Publ. Co. (2007)
3. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. *Artificial Intelligence* **168**(1–2) (2005) 70–118
4. Calvanese, D., Lenzerini, M., Nardi, D.: Unifying class-based representation formalisms. *J. of Artificial Intelligence Research* **11** (1999) 199–240
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: KR’06. (2006) 260–270
6. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: *DL-Lite* in the light of first-order logic. In: AAAI’07. (2007)
7. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language reference. W3C Recommendation (February 2004) Available at <http://www.w3.org/TR/owl-ref/>.
8. Di Battista, G., Lenzerini, M.: Deductive entity-relationship modeling. *IEEE Trans. on Knowledge and Data Engineering* **5**(3) (1993) 439–450
9. Lenzerini, M., Nobili, P.: On the satisfiability of dependency constraints in entity-relationship schemata. *Information Systems* **15**(4) (1990) 453–461
10. Calvanese, D., Lenzerini, M.: On the interaction between ISA and cardinality constraints. In: ICDE’94. (1994) 204–213
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: AAAI’05. (2005) 602–607
12. Kozen, D.: Theory of Computation. Springer (2006)

## Pinpointing in the Description Logic $\mathcal{EL}$

Franz Baader<sup>1</sup>, Rafael Peñaloza<sup>2\*</sup>, and Boontawee Suntisrivaraporn<sup>1</sup>

<sup>1</sup> Theoretical Computer Science, TU Dresden, Germany  
 {baader,meng}@tcs.inf.tu-dresden.de

<sup>2</sup> Intelligent Systems, University of Leipzig, Germany  
 penaloza@informatik.uni-leipzig.de

### 1. Introduction

For a developer or user of a DL-based ontology, it is often quite hard to understand why a certain consequence holds, and even harder to decide how to change the ontology in case the consequence is unwanted. For example, in the current version of the medical ontology SNOMED [16], the concept *Amputation-of-Finger* is classified as a subconcept of *Amputation-of-Arm*. Finding the axioms that are responsible for this among the more than 350,000 terminological axioms of SNOMED without support by an automated reasoning tool is not easy.

As a first step towards providing such support, Schlobach and Cornet [14] describe an algorithm for computing all the *minimal subsets* of a given knowledge base that *have* a given *consequence*. In the following, we call such a set a *minimal axiom set (MinA)*. It helps the user to comprehend why a certain consequence holds. The knowledge bases considered in [14] are so-called unfoldable  $\mathcal{ALC}$ -terminologies, and the unwanted consequences are the unsatisfiability of concepts. The algorithm is an extension of the known tableau-based satisfiability algorithm for  $\mathcal{ALC}$  [15], where labels keep track of which axioms are responsible for an assertion to be generated during the run of the algorithm. The authors also coin the name “axiom pinpointing” for the task of computing these minimal subsets.

The problem of computing MinAs of a DL knowledge base was actually considered earlier in the context of extending DLs by default rules. In [2], Baader and Hollunder solve this problem by introducing a labeled extension of the tableau-based consistency algorithm for  $\mathcal{ALC}$ -ABoxes [9], which is very similar to the one described later in [14]. The main difference is that the algorithm described in [2] does not directly compute minimal subsets that have a consequence, but rather a monotone Boolean formula whose variables correspond to the axioms of the knowledge bases and whose minimal satisfying valuations correspond to the MinAs.

The approach of Schlobach and Cornet [14] was extended by Parsia et al. [12] to more expressive DLs, and the one of Baader and Hollunder [2] was extended by Meyer et al. [11] to the case of  $\mathcal{ALC}$ -terminologies with general concept inclusions (GCIs), which are no longer unfoldable. Axiom pinpointing has also been considered in other research areas, though usually not under this name.

\* Funded by the German Research Foundation (DFG) under grant GRK 446.

For example, in the SAT community, people have considered the problem of computing minimally unsatisfiable (and maximally satisfiable) subsets of a set of propositional formulae. The approaches for computing these sets developed there include special purpose algorithms that call a SAT solver as a black box [10, 5], but also algorithms that extend a resolution-based SAT solver directly [7, 18].

Whereas the previous work on pinpointing in DLs considered fairly expressive DLs that contain at least  $\mathcal{ALC}$ , this work is concerned with pinpointing in the inexpressive DL  $\mathcal{EL}$ , which has recently drawn considerable attention. On the one hand, several bio-medical ontologies such as SNOMED [16], the Gene Ontology [17], and large parts of Galen [13] can be expressed in  $\mathcal{EL}$ . On the other hand, reasoning in  $\mathcal{EL}$  and some of its extensions remains polynomial even in the presence of GCIs [6, 1]. Although the polynomial-time subsumption algorithm for  $\mathcal{EL}$  described in [6, 1] is not tableau-based, the ideas for extending tableau-based algorithms to pinpointing algorithms employed in [2, 14] can also be applied to this algorithm. However, we will see that the normalization phase employed by this algorithm introduces an additional problem. We will also consider the complexity of pinpointing in  $\mathcal{EL}$ . In contrast to the case of  $\mathcal{ALC}$ , where the subsumption problem is already highly complex, subsumption in  $\mathcal{EL}$  is polynomial, which makes it easier to analyze the extent to which pinpointing is a source of additional complexity. Not surprisingly, it turns out that there may be exponentially many MinAs. In addition, even testing whether there is a MinA of cardinality  $\leq n$  for a given natural number  $n$  is an NP-complete problem. However, one MinA can always be computed in polynomial time. Finally, we will provide some experimental results regarding the performance of a practical algorithm that computes one (not necessarily minimal) set that has a given consequence.

## 2. A pinpointing algorithm for $\mathcal{EL}$

Recall that  $\mathcal{EL}$  allows for conjunction ( $\sqcap$ ), existential restrictions ( $\exists r.C$ ), and the top concept ( $\top$ ). We consider general  $\mathcal{EL}$ -TBoxes  $\mathcal{T}$  consisting of GCIs  $C \sqsubseteq D$ , where  $C, D$  are arbitrary  $\mathcal{EL}$ -concept descriptions, and are interested in the subsumption relation between concept names occurring in  $\mathcal{T}$ , which is denoted as  $\sqsubseteq_{\mathcal{T}}$ . Given such a TBox  $\mathcal{T}$  and concept names  $A, B$  occurring in it, a *MinA* for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$  is a subset  $\mathcal{S}$  of  $\mathcal{T}$  such that  $A \sqsubseteq_{\mathcal{S}} B$ , but  $A \not\sqsubseteq_{\mathcal{S}'} B$  for all strict subsets  $\mathcal{S}' \subset \mathcal{S}$ . For example, consider the TBox  $\mathcal{T}$  consisting of the GCIs

$$\text{ax}_1: A \sqsubseteq \exists r.A, \quad \text{ax}_2: A \sqsubseteq Y, \quad \text{ax}_3: \exists r.Y \sqsubseteq B, \quad \text{ax}_4: Y \sqsubseteq B. \quad (1)$$

We have  $A \sqsubseteq_{\mathcal{T}} B$ , and it is easy to see that  $\{\text{ax}_2, \text{ax}_4\}$  and  $\{\text{ax}_1, \text{ax}_2, \text{ax}_3\}$  are the MinAs of  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$ .

In the following, we show how the polynomial-time subsumption algorithm for  $\mathcal{EL}$  with GCIs [6, 1] can be modified to a pinpointing algorithm. However, instead of computing the MinAs directly, we follow the approach introduced in [2] that computes a monotone Boolean formula from which the MinAs can

be derived. To define this formula, which we will call a pinpointing formula in the following, we assume that every GCI  $t \in \mathcal{T}$  is labeled with a unique propositional variable,  $lab(t)$ . Let  $lab(\mathcal{T})$  be the set of all propositional variables labeling GCIs in  $\mathcal{T}$ . A *monotone Boolean formula* over  $lab(\mathcal{T})$  is a Boolean formula using (some of) the variables in  $lab(\mathcal{T})$  and only the binary connectives conjunction and disjunction and the nullary connective  $\top$  (for truth). As usual, we identify a propositional *valuation* with the set of propositional variables it makes true. For a valuation  $\mathcal{V} \subseteq lab(\mathcal{T})$ , let  $\mathcal{T}_{\mathcal{V}} := \{t \in \mathcal{T} \mid lab(t) \in \mathcal{V}\}$ .

**Definition 1 (pinpointing formula).** *Given an  $\mathcal{EL}$ -TBox  $\mathcal{T}$  and concept names  $A, B$  occurring in it, the monotone Boolean formula  $\phi$  over  $lab(\mathcal{T})$  is a pinpointing formula for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$  if the following holds for every valuation  $\mathcal{V} \subseteq lab(\mathcal{T})$ :  $A \sqsubseteq_{\mathcal{T}_{\mathcal{V}}} B$  iff  $\mathcal{V}$  satisfies  $\phi$ .*

In our example, we can take  $lab(\mathcal{T}) = \{ax_1, \dots, ax_4\}$  as set of propositional variables. It is easy to see that  $ax_2 \wedge (ax_4 \vee (ax_1 \wedge ax_3))$  is a pinpointing formula for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$ .

Let  $\phi$  be a pinpointing formula for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$ . If we order valuations by set inclusion, then we obviously have that

$$\{T_{\mathcal{V}} \mid \mathcal{V} \text{ is a minimal valuation satisfying } \phi\}$$

is the set of all MinAs for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$ . This shows that it is enough to design an algorithm for computing a pinpointing formula to obtain all MinAs. For example, one possibility is to bring  $\phi$  into disjunctive normal form and then remove disjuncts implying other disjuncts. Note that this may cause an exponential blowup, which means that, in some cases, the pinpointing formula provides us with a compact representation of the set of all MinAs. Also note that this blowup is not really in the size of the pinpointing formula but rather in the number of variables. Thus, if the size of the pinpointing formula is already exponential in the size of the TBox  $\mathcal{T}$  (which may well happen), computing all MinAs from it is still “only” exponential in the size of  $\mathcal{T}$ .

In order to describe our algorithm for computing pinpointing formulae for subsumption in  $\mathcal{EL}$ , we must briefly recall the subsumption algorithm for  $\mathcal{EL}$  [6, 1]. First, this algorithm transforms a given TBox into a *normal form* where all GCIs have one of the following forms:  $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B$ ,  $A \sqsubseteq \exists r.B$ ,  $\exists r.A \sqsubseteq B$ , where  $n \geq 1$  and  $A, A_1, \dots, A_n, B$  are concept names (including  $\top$ ). This transformation can be achieved in linear time using simple transformation rules, which basically break down complex GCIs into simpler ones (see [6, 1] for details). For the pinpointing extension it is relevant that the relationship between original axioms and normalized axioms is many to many: one axiom in the original TBox can give rise to several axioms in the normalized one, and one axiom in the normalized TBox can come from several axioms in the original TBox. For example, consider the GCIs  $A \sqsubseteq B_1 \sqcap B_2$ ,  $A \sqsubseteq B_2 \sqcap B_3$ , which are normalized to  $A \sqsubseteq B_1$ ,  $A \sqsubseteq B_2$ ,  $A \sqsubseteq B_3$ . Each original GCI gives rise to two normalized ones, and the normalized GCI  $A \sqsubseteq B_2$  has two sources, i.e., it is present in the normalized TBox if the first *or* the second original GCI is present in the input TBox.

- If**  $A_1 \sqcap \dots \sqcap A_n \sqsubseteq B \in \mathcal{T}$  **and**  $\{(X, A_1), \dots, (X, A_n)\} \subseteq \mathcal{A}$  **then** add  $(X, B)$  to  $\mathcal{A}$ .  
**If**  $A \sqsubseteq \exists r.B \in \mathcal{T}$  **and**  $(X, A) \in \mathcal{A}$  **then** add  $(X, r, B)$  to  $\mathcal{A}$ .  
**If**  $\exists r.A \sqsubseteq B \in \mathcal{T}$  **and**  $\{(X, r, Y), (Y, A)\} \subseteq \mathcal{A}$  **then** add  $(X, B)$  to  $\mathcal{A}$ .

**Fig. 1.** Completion rules for subsumption in  $\mathcal{EL}$ .

Given a TBox  $\mathcal{T}$  in normal form, *the subsumption algorithm for  $\mathcal{EL}$*  employs completion rules to extend an initial set of assertions until no more assertions can be added. Assertions are of the form  $(A, B)$  or  $(A, r, B)$  where  $A, B$  are concept names occurring in  $\mathcal{T}$  or  $\top$ , and  $r$  is a role name occurring in  $\mathcal{T}$ . Intuitively, the assertion  $(A, B)$  expresses that  $A \sqsubseteq_{\mathcal{T}} B$  holds and  $(A, r, B)$  expresses that  $A \sqsubseteq_{\mathcal{T}} \exists r.B$  holds. The algorithm starts with a set of assertions  $\mathcal{A}$  that contains  $(A, \top)$  and  $(A, A)$  for every concept name  $A$ , and then uses the rules shown in Fig. 1 to extend  $\mathcal{A}$ . Note that such a rule is only applied if it really extends  $\mathcal{A}$ , i.e., if the assertion added by the rule is not yet contained in  $\mathcal{A}$ . As shown in [6, 1], rule application terminates after a polynomial number of steps, and the set of assertions  $\mathcal{A}$  obtained after termination satisfies  $A \sqsubseteq_{\mathcal{T}} B$  iff  $(A, B) \in \mathcal{A}$  for all concept names  $A, B$  occurring in  $\mathcal{T}$ .

In *the pinpointing extension* of this algorithm, assertions  $a$  are also labeled with monotone Boolean formulae  $lab(a)$ . The initial assertions  $(A, \top)$  and  $(A, A)$  receive label  $\mathbf{t}$ . The definition of rule application is modified as follows. Assume that the precondition of a rule from Fig. 1 are satisfied for the set of assertions  $\mathcal{A}$  w.r.t. the TBox  $\mathcal{T}$ . Let  $\phi$  be the conjunction of the labels of the GCIs from  $\mathcal{T}$  and the assertions from  $\mathcal{A}$  occurring in the precondition. If the assertion in the consequence of the rule does not yet belong to  $\mathcal{A}$ , then it is added with label  $\phi$ . If the assertion is already there with label  $\psi$ , then its label is changed to  $\psi \vee \phi$  if this formula is not equivalent to  $\psi$ ; otherwise (i.e., if  $\phi$  implies  $\psi$ ) the rule is not applied.

It is easy to see that this modified algorithm always terminates, though not necessarily in polynomial time. In fact, there are polynomially many assertions that can be added to  $\mathcal{A}$ . If the label of an assertion is changed, then the new label is a more general monotone Boolean formula, i.e., it has more models than the original label. Since there are only exponentially many models, the label of a given assertion can be changed only exponentially often. In addition, the set of assertions  $\mathcal{A}$  obtained after termination is identical to the one obtained by the unmodified algorithm, and for all assertions  $(A, B) \in \mathcal{A}$  we have that  $lab((A, B))$  is a pinpointing formula for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$ .

As an example, consider the TBox consisting of the GCIs given in (1). The pinpointing algorithm proceeds as follows. Since  $ax_2 : A \sqsubseteq Y \in \mathcal{T}$  and  $(A, A) \in \mathcal{A}$  with label  $\mathbf{t}$ , the assertion  $(A, Y)$  is added to  $\mathcal{A}$  with label  $ax_2$  (actually with label  $ax_2 \wedge \mathbf{t}$ , which is equivalent to  $ax_2$ ). Since  $ax_1 : A \sqsubseteq \exists r.A \in \mathcal{T}$  and  $(A, A) \in \mathcal{A}$  with label  $\mathbf{t}$ ,  $(A, r, A)$  is added to  $\mathcal{A}$  with label  $ax_1$ . Since  $ax_4 : Y \sqsubseteq B \in \mathcal{T}$  and  $(A, Y) \in \mathcal{A}$  with label  $ax_2$ ,  $(A, B)$  is added to  $\mathcal{A}$  with label  $ax_2 \wedge ax_4$ . Finally, since  $ax_3 : \exists r.Y \sqsubseteq B \in \mathcal{T}$ ,  $(A, Y) \in \mathcal{A}$  with label  $ax_2$ , and  $(A, r, A) \in \mathcal{A}$  with label  $ax_1$ , the label of  $(A, B) \in \mathcal{A}$  is modified from  $ax_2 \wedge ax_4$

---

**Algorithm 1** Compute one MinA for  $\mathcal{T} = \{t_1, \dots, t_n\}$  w.r.t.  $A \sqsubseteq B$ .

---

```

1: if  $A \not\sqsubseteq_{\mathcal{T}} B$  then
2:   return no MinA
3:  $\mathcal{S} := \mathcal{T}$ .
4: for  $1 \leq i \leq n$  do
5:   if  $A \sqsubseteq_{\mathcal{S} \setminus \{t_i\}} B$  then
6:      $\mathcal{S} := \mathcal{S} \setminus \{t_i\}$ 
7: return  $\mathcal{S}$ 

```

---

to  $(ax_2 \wedge ax_4) \vee (ax_1 \wedge ax_2 \wedge ax_3)$ . This final label of  $(A, B)$  is a pinpointing formula for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$ .

As described until now, our pinpointing algorithm for  $\mathcal{EL}$  can only deal with normalized TBoxes, i.e., the pinpointing formula  $\phi$  it yields contains propositional variables corresponding to the normalized GCIs. However, modifying  $\phi$  to a *pinpointing formula for the original TBox* is quite simple. Assume that the original GCIs are also associated with propositional variables. Each normalized GCI has a finite number of original GCIs as sources. We modify  $\phi$  by replacing the propositional variable for each normalized axiom by the disjunction of the propositional variables of its sources.

### 3. The complexity of pinpointing in $\mathcal{EL}$

If we want to compute all MinAs, then in the worst case an exponential runtime cannot be avoided since there may be *exponentially many MinAs* for a given TBox. This is shown by the following example.

*Example 1.* For all  $n \geq 1$ , the size of the TBox

$$\mathcal{T}_n := \{B_{i-1} \sqsubseteq P_i \sqcap Q_i, P_i \sqsubseteq B_i, Q_i \sqsubseteq B_i \mid 1 \leq i \leq n\}$$

is linear in  $n$ , and we have  $B_0 \sqsubseteq_{\mathcal{T}_n} B_n$ . There are  $2^n$  MinAs for  $\mathcal{T}_n$  w.r.t.  $B_0 \sqsubseteq B_n$  since, for each  $i, 1 \leq i \leq n$ , it is enough to have  $P_i \sqsubseteq B_i$  or  $Q_i \sqsubseteq B_i$  in the set.

On the other hand, a *single MinA can be computed in polynomial time* by the simple Algorithm 1, which goes through all GCIs (in a given fixed order) and throws away those that are not needed to obtain the desired subsumption relationship. Since the algorithm performs  $n + 1$  subsumption tests (where  $n$  is the cardinality of  $\mathcal{T}$ ), and each such test takes only polynomial time, the overall complexity of this algorithm is polynomial. It is easy to see that its output (in case  $A \sqsubseteq_{\mathcal{T}} B$ ) is indeed a MinA for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$ .

However, as soon as we want to know more about the properties of the set of *all* MinAs, this cannot be achieved in polynomial time (unless  $P=NP$ ). For example, the following *minimum cardinality problem* is NP-complete: given an  $\mathcal{EL}$ -TBox  $\mathcal{T}$ , concept names  $A, B$  occurring in  $\mathcal{T}$ , and a natural number  $n$ , is there a MinA for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$  of cardinality  $\leq n$ ? The problem is in NP, since one can simply guess a subset  $\mathcal{S}$  of  $\mathcal{T}$  cardinality  $n$ , and then check in

polynomial time whether  $A \sqsubseteq_S B$ . Clearly, such a set exists iff there is a MinA of cardinality  $\leq n$ .

NP-hardness can be shown by a reduction of the NP-hard *hitting set problem* [8]: given a collection  $S_1, \dots, S_k$  of sets and a natural number  $n$ , is there a set  $S$  of cardinality  $\leq n$  such that  $S \cap S_i \neq \emptyset$  for  $i = 1, \dots, k$ . Such a set  $S$  is called a *hitting set*. In the reduction, we use a concept name  $P$  for every element  $p \in S_1 \cup \dots \cup S_n$  as well as the additional concept names  $A, B, Q_1, \dots, Q_k$ . Given  $S_1 = \{p_{11}, \dots, p_{1\ell_1}\}, \dots, S_k = \{p_{k1}, \dots, p_{k\ell_k}\}$ , we define the TBox

$$\mathcal{T} := \{P_{ij} \sqsubseteq Q_i \mid 1 \leq i \leq k, 1 \leq j \leq \ell_i\} \cup \{A \sqsubseteq P_{ij} \mid 1 \leq i \leq k, 1 \leq j \leq \ell_i\} \cup \{Q_1 \sqcap \dots \sqcap Q_k \sqsubseteq B\}.$$

It is easy to see that  $S_1, \dots, S_k$  has a hitting set of cardinality  $\leq n$  iff there is a MinA for  $\mathcal{T}$  w.r.t.  $A \sqsubseteq B$  of cardinality  $\leq n + k + 1$ .

#### 4. A practical algorithm for computing one MinA

Although it requires only polynomial time, computing one MinA using Algorithm 1 may still be impractical for very large TBoxes like SNOMED. In fact, the algorithm has to make as many calls of the subsumption algorithm as there are axioms in the TBox (in the case of SNOMED, more than 350,000). Here we propose an *improved algorithm* that proceeds in two steps: (i) first compute a (not necessarily minimal) subset of the TBox from which the subsumption relationship follows; (ii) then minimize this set using Algorithm 1. Of course, this approach makes sense only if the algorithm used in step (i) is efficient and produces fairly small sets. It would not help to use the trivial algorithm that always produces the whole TBox.

An algorithm that realizes step (i) and runs in polynomial time can easily be obtained from the pinpointing algorithm sketched in Section 2. The only modification is the following: if an assertion in the consequence of a rule already belongs to the current set of assertions, then this rule is not applied, i.e., once an assertion is there with some label, the label remains unchanged. Thus, every assertion  $(A, B)$  in the final set has a conjunction of propositional variables as its label, which clearly corresponds to a subset of the TBox from which the subsumption relationship  $A \sqsubseteq B$  follows. In general, this subset is not minimal, however. (Because of the space constraints, we cannot give an example demonstrating this.)

As described until now, this modified algorithm works on normalized TBoxes. To get an appropriate subset of the original axioms, one can use a greedy strategy for producing a set of original axioms that covers a given set  $\mathcal{S}$  of normalized axioms in the following sense. For each original axiom  $t$ , let  $\mathcal{S}_t$  be the set of normalized axioms  $t$  gives rise to. The set  $\mathcal{T}'$  of original axioms *covers*  $\mathcal{S}$  if  $\mathcal{S} \subseteq \bigcup_{t \in \mathcal{T}'} \mathcal{S}_t$ . The use of a greedy strategy adds another possible source of non-minimality. (We use a non-optimal greedy strategy to keep the algorithm polynomial. In fact, even determining whether there is a cover set of size  $\leq n$  is another NP-complete problem [8].)

Our preliminary experimental results confirm that this algorithm is indeed more practical than Algorithm 1. Based on the CEL reasoner [3], we have implemented a slightly extended version of the practical algorithm for computing one MinA in  $\mathcal{EL}^+$ , which is  $\mathcal{EL}$  extended with complex role inclusions, and thus can express role hierarchies and transitive roles. The experiments were run on a variant of the Galen Medical Knowledge Base [13],<sup>3</sup> which is a TBox consisting of more than 4,000 axioms. On the normalized version of this TBox, CEL needs about 14 sec to compute all subsumption relationships between concept names occurring in this TBox. Overall, more than 27,000 subsumption relationships are computed. The overhead for computing for all of these subsumption relationships (possibly non-minimal) subsets from which they already follow was a bit more than 50%: the modified pinpointing algorithm described above needed about 23 sec. Going from the subsets of the normalized TBox to the corresponding (still possibly non-minimal) subsets of the original Galen TBox took 0.27 sec. Finally, the overall time required for minimizing these sets using Algorithm 1 (with CEL as the subsumption reasoner) was 9:45 min. For these last two numbers one should take into account, however, that these involved treating more than 27,000 such sets. For a single such set, the average post-processing time was negligible (on average 21 milliseconds). Also note that applying Algorithm 1 directly to the whole TBox for just one subsumption relationship (between *Renal-Artery* and *Artery-Which-Has-Laterality*) took more than 7 hours.

Thus, from the point of view of runtime, our practical algorithm behaves quite well on Galen. The same can be said about the quality of its results. The average size of an axiom set computed by the algorithm before using Algorithm 1 to minimize it was 5 (with maximum size 31), which is quite small and thus means that this set can directly be given to the user as an explanation for the subsumption relationship. Also, the computed sets were almost minimal: on average, the possibly non-minimal sets computed by the algorithm were only 2.59% larger than the minimal ones. When considering the normalized TBox (i.e., without translating back to the original TBox), this number was even better (0.1%). This means that in most cases it is probably not necessary to further minimize the sets using Algorithm 1. If demanded by the user for a specific subsumption relationship it can still be done without taking much time.

## 5. Additional work on pinpointing

The pinpointing extension of the subsumption algorithm for  $\mathcal{EL}$  described in Section 2 as well as the pinpointing algorithm for  $\mathcal{ALC}$  described in [2] are instances of a general approach for modifying “tableau-like” reasoning procedures to pinpointing procedures [4].

Instead of computing minimal subsets that have a given consequence, one sometimes also wants to compute maximal subsets that do not have a given consequence. Given the pinpointing formula  $\phi$ , these sets correspond to maximal

<sup>3</sup> Since Galen uses expressivity not available in  $\mathcal{EL}^+$ , we have simplified it by removing inverse role axioms and treating functional roles as ordinary ones.

valuations that do not satisfy  $\phi$ . The complexity results from Section 3 hold accordingly for such maximal sets. However, we currently do not know how to obtain a practical algorithm computing one such set (i.e., the results of Section 4 cannot be transferred to the case of maximal sets).

## References

- [1] F. Baader, S. Brandt, and C. Lutz. Pushing the  $\mathcal{EL}$  envelope. In *Proc. of IJCAI 2005*, pages 364–369. Morgan Kaufmann, Los Altos, 2005.
- [2] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. of Automated Reasoning*, 14:149–180, 1995.
- [3] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In *Proceedings of IJCAR’06*, volume 4130 of *LNAI*, pages 287–291. Springer-Verlag, 2006.
- [4] F. Baader and R. Penaloza. Axiom pinpointing in general tableaux. LTCS-Report 07-01, Germany, 2007. See <http://lat.inf.tu-dresden.de/research/reports.html>.
- [5] J. Bailey and P.J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *Proc. of PADL’05*, volume 3350 of *LNCS*, pages 174–186. Springer, 2005.
- [6] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. of ECAI 2004*, pages 298–302.
- [7] G. Davydov, I. Davydova, and H. Kleine Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. *Ann. of Mathematics and Artificial Intelligence*, 23(3–4):229–245, 1998.
- [8] M. R. Garey and D. S. Johnson. *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (CA, USA), 1979.
- [9] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of German Workshop on AI*, pages 38–47. Springer, 1990.
- [10] M.H. Liffiton and K.A. Sakallah. On finding all minimally unsatisfiable subformulas. In *Proc. of SAT’05*, volume 3569 of *LNCS*, pages 173–186. Springer, 2005.
- [11] T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic  $\mathcal{ALC}$ . In *Proc. of AAAI 2006*. AAAI Press.
- [12] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW’05*, pages 633–640. ACM, 2005.
- [13] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of AAAI’97*, Stanford, CA, 1997. AAAI Press.
- [14] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI 2003*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.
- [15] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
- [16] K.A. Spackman, K.E. Campbell, and R.A. Cote. SNOMED RT: A reference terminology for health care. *J. of the Am. Med. Inf. Ass.*, pages 640–644, 1997.
- [17] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
- [18] L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *Proc. of DATE’03*, pages 10880–10885. IEEE Computer Society Press, 2003.

## Modal vs. Propositional Reasoning for Model Checking with Description Logic

Shoham Ben-David    Richard Trefler    Grant Weddell

David R. Cheriton School of Computer Science  
University of Waterloo

### 1 Introduction

Model checking ([7, 13], c.f.[6]) is a technique for verifying finite-state concurrent systems that has proven effective in the verification of industrial hardware and software programs. In model checking, a model  $M$ , given as a set of state variables  $V$  and their next-state relations, is verified against a temporal logic formula  $\varphi$ . In this work we consider only safety formulas of the form  $AG(b)$ , with  $b$  being a Boolean expression over the state variables of the model, meaning that  $b$  is an invariant of  $M$ .

The main challenge in model checking is known as the *state space explosion* problem, where the number of states in the model grows exponentially in the number of program variables. To cope with this problem, model checking is done *symbolically*, by representing the system under verification as sets of states and transitions, and by using Boolean functions to manipulate those sets. Two main symbolic methods are used to perform model checking. The first, implemented in *SMV* [10], is based on Binary Decision Diagrams (BDDs) [5]. The second is known as Bounded Model Checking (*BMC*) [4]. Using this method, the model under verification and its specification are unfolded to depth  $k$  (for a given bound  $k$ ), and translated into a propositional CNF formula. This results in an encoding of the model checking problem that is essentially  $k$  times the size of the textual description of  $M$ . A SAT solver is then applied to the formula to find a satisfying assignment. Such an assignment, if found, demonstrates an error in the model.

We investigate the possibility of using a Description Logic reasoner for bounded model checking. Recent work [3] showed how to embed BMC problems as concept consistency problems in the DL dialect *ALCT*. The encoding as a terminology resulted in a natural *symbolic* representation of the sets of states and transitions that is significantly smaller than the one obtained by translating a model into a CNF formula. This translation works as follows.

Let  $M$  be a model defined by a set  $V$  of Boolean state variables and their next-state transitions  $R$ . We represent each variable  $v_i \in V$  as a concept  $V_i$ , and the transition relation as a single role  $R$ . We then introduce concept inclusions of the type

$$C_0 \sqsubseteq \forall R.C_1$$

stating that if the current state satisfies the condition represented by  $C_0$  then all the next-states that can be reached in one step by  $R$  must satisfy the condition  $C_1$ . Note that interpretations for this set of concept inclusions correspond to sub-models of  $M$ .

Let the concept  $S_0$  represent the set of initial states of  $M$ , and let  $S_1$  be a new concept. If the concept inclusion  $S_1 \sqsubseteq \exists R^-.S_0$  holds in the interpretation, then the set  $S_1$  is a subset of all the states that can be reached from  $S_0$  by going one step forward

using the relation  $R$ . Similarly, we denote by  $S_i$  a subset of the states that can be reached after  $i$  steps, and introduce the inclusions

$$S_i \sqsubseteq \exists R^-.S_{i-1}$$

for  $0 < i \leq k$ . Let  $\varphi = AG(b)$  be the specification to be verified, and let  $B$  be the concept representing  $b$  (composed of a Boolean combination of the concepts  $V_i$  representing the state variables). Model checking is then carried out by asking the query: “does there exist an interpretation for the above set of concept inclusions such that  $\neg B \sqcap S_i$  is not empty for some  $S_i$ ?”. A positive answer from the DL reasoner indicates an error in  $M$ .

Experimental results comparing this method to SAT-based model checking showed that SAT solving outperformed DL reasoning, especially as the bound  $k$  increased [3], despite the significantly smaller DL encoding of the model checking problem —  $k$  times the size of  $M$  in the SAT case vs.  $k$  plus the size of  $M$  in the DL case.

In this paper we report on an experiment aimed at determining whether it is the *modal* reasoning (that involved taking backward steps through the role  $R$ ) that caused the problem, or the propositional reasoning that is more efficient in the SAT solvers. For this we produced a series of translations of the model into a terminology. Instead of one set of concepts corresponding to the variables of the given model, we introduce  $l$  copies to represent states of increasing distance from the initial state. The number of *modal* traversals through  $R$  is then reduced by a factor of  $l$  while increasing the size of the DL encoding by a factor of  $l$ . We expected the reduced modality translations would outperform the original. It was surprising to us that the results suggested the opposite.

The rest of the paper is organized as follows. In the next section we give definitions and present bounded model checking using DL [3]. Section 3 is the main section of the paper where the new translation is presented. An evaluation and discussion follow in Section 4.

## 2 Background and Definitions

**Definition 1 (Description Logic  $\mathcal{ALCI}$ )** Let  $NC$  and  $NR$  be sets of atomic concepts  $\{A_1, A_2, \dots\}$  and atomic roles  $\{R_1, R_2, \dots\}$  respectively. The set of concepts  $C$  of the description logic  $\mathcal{ALCI}$  is the smallest set including  $NC$  that satisfies the following.

- If  $C_1, C_2 \in C$  then so are  $\neg C_1$  and  $C_1 \sqcap C_2$
- If  $C \in C$  and  $R \in R$  then so are  $\exists R.C$  and  $\exists R^-.C$

Additional concepts are defined as syntactic sugaring of those above:

- $\top = A \sqcup \neg A$  for some  $A$
- $\forall R.C = \neg \exists R.\neg C$
- $C_1 \sqcup C_2 = \neg(\neg C_1 \sqcap \neg C_2)$

A *General Concept Inclusion Axiom* is an expression of the form  $C_1 \sqsubseteq C_2$ . A *TBox*  $\mathcal{T}$  consists of a finite set of inclusion dependencies.

The *semantics* of expressions is defined with respect to a structure  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where  $\Delta^{\mathcal{I}}$  is a non-empty set, and  $\cdot^{\mathcal{I}}$  is a function mapping every concept to a subset of  $\Delta^{\mathcal{I}}$  and every role to a subset of  $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$  such that the following conditions are satisfied.

- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \sqcap C_2^{\mathcal{I}}$
- $\exists R.C = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} \text{ s.t. } (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
- $\exists R^{-}.C = \{y \in \Delta^{\mathcal{I}} \mid \exists x \in \Delta^{\mathcal{I}} \text{ s.t. } (x, y) \in R^{\mathcal{I}} \wedge x \in C^{\mathcal{I}}\}$

A structure *satisfies an inclusion dependency*  $C_1 \sqsubseteq C_2$  if  $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ . The *consistency problem for  $\mathcal{ALCC}$*  asks if  $\mathcal{T} \models_{dl} C$  holds;<sup>1</sup> that is, if there exists  $\mathcal{I}$  such that  $C^{\mathcal{I}}$  is non-empty and such that  $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$  holds for each  $C_1 \sqsubseteq C_2$  in  $\mathcal{T}$ .

## 2.1 Symbolic Model Checking

**Definition 2 (Kripke Structure)** Let  $V$  be a set of Boolean variables. A Kripke structure  $M$  over  $V$  is a four tuple  $M = (S, I, R, L)$  where

1.  $S$  is a finite set of states.
2.  $I \subseteq S$  is the set of initial states.
3.  $R \subseteq S \times S$  is a transition relation that must be total, that is, for every state  $s \in S$  there is a state  $s' \in S$  such that  $R(s, s')$ .
4.  $L : S \rightarrow 2^V$  is a function that labels each state with the set of variables true in that state.

We view each state  $s$  as a truth assignment to the variables  $V$ . We view a set of states as a Boolean function over  $V$ , characterizing the set. For example, The set of initial states  $I$  is considered as a Boolean function over  $V$ . Thus, if a state  $s$  belongs to  $I$ , we write  $s \models I$ . Similarly, if  $v_i \in L(s)$  we write  $s \models v_i$ , and if  $v_i \notin L(s)$  we write  $s \models \neg v_i$ . We say that  $w = s_0, s_1, \dots, s_k$  is a path in  $M$  if  $\forall i, 0 \leq i < k, (s_i, s_{i+1}) \in R$  and  $s_0 \models I$ .

In practice, the full Kripke structure of a system is not explicitly given. Rather, a model is given as a set of Boolean variables  $V = \{v_1, \dots, v_n\}$ , their initial values and their next-state assignments. The definition we give below is an abstraction of the input language of *SMV* [10].

**Definition 3 (Model Description)** Let  $V = \{v_1, \dots, v_n\}$  be a set of Boolean variables. A tuple  $MD = (I_{MD}, [\langle c_1, c'_1 \rangle, \dots, \langle c_n, c'_n \rangle])$  is a Model Description over  $V$  where  $I_{MD}, c_i, c'_i$  are Boolean expressions over  $V$ .

The semantics of a model description is a Kripke structure  $M_{MD} = (S, I_M, R, L)$ , where  $S = 2^V$ ,  $L(s) = s$ ,  $I_M = \{s \mid s \models I_{MD}\}$ , and  $R = \{(s, s') : \forall 1 \leq i \leq n, s \models c_i \text{ implies } s' \models \neg v_i \text{ and } s \models c'_i \wedge \neg c_i \text{ implies } s' \models v_i\}$ .

Intuitively, a pair  $\langle c_i, c'_i \rangle$  defines the next-state assignment of variable  $v_i$  in terms of the current values of  $\{v_1, \dots, v_n\}$ . That is,

$$\text{next}(v_i) = \begin{cases} 0 & \text{if } c_i \\ 1 & \text{if } c'_i \wedge \neg c_i \\ \{0, 1\} & \text{otherwise} \end{cases}$$

where the assignment  $\{0, 1\}$  indicates that for every possible next-state value of variables  $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n$  there must exist a next-state with  $v_i = 1$ , and a next-state with  $v_i = 0$ .

<sup>1</sup> We write “ $\models_{dl}$ ” to distinguish the use of the double turnstyle symbol by both description logic and model checking communities.

**Safety Formulas** The formulas we consider are *safety* formulas, given as  $AG(b)$  in *CTL* [7], or  $G(b)$  in *LTL* [12]. Such formulas state that the Boolean expression  $b$  holds on all reachable states of the model under verification. We note that a large and useful subset of *CTL* and *LTL* can be translated into  $AG(b)$  type formulas [2].

**Bounded Model Checking** Given a Kripke structure  $M = (S, I, R, L)$ , a formula  $\varphi$ , and a bound  $k$ , Bounded Model Checking (BMC) tries to refute  $M \models \varphi$  by proving the existence of a witness to the negation of  $\varphi$  of length  $k$  or less. For  $\varphi = AG(b)$ , we say that  $M^k \not\models \varphi$  if and only if there exists a path  $w = s_0, \dots, s_j$  in  $M$  such that  $j \leq k$  and  $s_j \models \neg b$ .

## 2.2 Bounded Model Checking Using Description Logic Reasoning

We briefly describe how bounded model checking can be achieved using description logic reasoning. For a detailed explanation and proof of correctness, refer to [3].

Let  $MD = (I, [\langle c_1, c'_1 \rangle, \dots, \langle c_n, c'_n \rangle])$  be a model description for the model  $M_{MD} = (S, I, R, L)$ , over  $V = \{v_1, \dots, v_n\}$ . Let  $k$  be the bound and let  $\varphi = AG(b)$  be a safety formula. We generate a terminology  $\mathcal{T}_{MD}^k$ , linear in the size of  $MD$ , and a concept  $C_\varphi$  such that  $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$  if and only if  $M_{MD}^k \not\models \varphi$ .

For each variable  $v_i \in V$  we introduce one primitive concept  $V_i$ , where  $V_i$  denotes  $v_i = 1$  and  $\neg V_i$  denotes  $v_i = 0$ . We introduce one primitive role  $R$  corresponding to the transition relation of the model. We define the concept  $S_0$  to represent  $I$ , by replacing each  $v_i$  in  $I$  with the concept  $V_i$ , and the connectives  $\wedge, \vee, \neg$  with  $\sqcap, \sqcup, \neg$ . The concepts  $C_i, C'_i$  correspond to the Boolean conditions  $c_i, c'_i$  in the same way. We then introduce three types of concept inclusions:

1. (*inclusions describing the model*) For each pair  $\langle c_i, c'_i \rangle$  add the pair of inclusions

$$\begin{aligned} C_i &\sqsubseteq \forall R. \neg V_i \\ (\neg C_i \sqcap C'_i) &\sqsubseteq \forall R. V_i. \end{aligned}$$

2. (*inclusions describing sets of reachable states, of distance  $i$  from the initial set*) For a bound  $k$ , add  $k$  primitive concepts,  $S_1, \dots, S_k$ , and for  $1 \leq i \leq k$ , add the  $k$  inclusions

$$S_i \sqsubseteq \exists R. \neg. S_{i-1}.$$

3. (*inclusion to describe the specification*) Let  $\varphi = AG(b)$  be the specification to be verified. The Boolean formula  $b$  is translated to a concept  $B$  in the usual way; in particular, each variable  $v_i$  is mapped to the concept  $V_i$ , and the Boolean connectives  $\vee, \wedge, \neg$  into their corresponding concept constructors  $\sqcup, \sqcap, \neg$ .

We define the concept  $C_\varphi \equiv \neg B \sqcap (S_0 \sqcup S_1 \sqcup \dots \sqcup S_k)$ . If  $C_\varphi$  is consistent with respect to the terminology  $\mathcal{T}_{MD}^k$  then  $\neg b$  must hold in some state with distance less than  $k$  from the initial state. Verification is therefore reduced to the query:  $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$ .

**Theorem 4 (from [3]).**  $M_{MD}^k \not\models \varphi$  if and only if  $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$ .

Let  $|\mathcal{T}_{MD}^k|$  represent the number of concept inclusions in  $\mathcal{T}_{MD}^k$ , and let  $n$  be the number of state variables in the model  $M_{MD}$ . The following proposition is discussed in [3].

**Proposition 5. (size of translation)**  $|\mathcal{T}_{MD}^k| = 2 \cdot n + k + 1$ .

### 3 On Controlling Propositional vs. Modal Reasoning

The above translation of a BMC problem into a terminology, denoted  $\mathcal{T}_{MD}^k$ , uses one set of primitive concepts corresponding to the state variables, and  $k$  concepts  $S_1, \dots, S_k$ , where  $S_i$  represents the set of states of distance  $i$  from the initial state. Thus, reaching a state that is  $i$  steps from an initial state will require a DL reasoner to build an  $R$ -chain consisting of  $i$  nodes. On the suspicion that reducing the length of this chain might improve performance, we generalize the above terminological embedding of model descriptions with an ability to supply an additional parameter  $l$  so that the resulting terminology, denoted  $\mathcal{T}_{MD}^{k/l}$  would entail a reduction of the length of the chain by a factor of  $l$ . We present the details of this more elaborate embedding in the remainder of this section. For simplicity, we assume that the original bound  $k$  is devisable by  $l$ .

The initial set  $S_0$  does not change, and corresponds to  $I$  as before. If  $k/l > 1$ , we introduce a role  $R$ . For each variable  $v_i \in \{v_1, \dots, v_n\}$ , we introduce  $l$  primitive concepts  $V_i^0, \dots, V_i^{l-1}$ , and for each pair  $\langle c_i, c'_i \rangle$ , we introduce  $2 * l$  concept inclusions of the following form.

$$\begin{aligned} C_i^0 &\sqsubseteq \neg V_i^1 \\ (\neg C_i^0 \sqcap C_i^{0'}) &\sqsubseteq V_i^1 \\ &\vdots \\ C_i^{l-2} &\sqsubseteq \neg V_i^{l-1} \\ (\neg C_i^{l-2} \sqcap C_i^{l-2'}) &\sqsubseteq V_i^{l-1} \\ C_i^{l-1} &\sqsubseteq \forall R. \neg V_i^0 \\ (\neg C_i^{l-1} \sqcap C_i^{l-1'}) &\sqsubseteq \forall R. V_i^0 \end{aligned}$$

If  $k/l = 1$  there is no need for the role  $R$ , and the last pair of concept inclusions would therefore be omitted. We introduce the concepts  $S_l, S_{2*l}, \dots, S_k$ , and the following  $k/l$  concept inclusions.

$$\begin{aligned} S_l &\sqsubseteq \exists R^- . S_0 \\ S_{2*l} &\sqsubseteq \exists R^- . S_l \\ &\vdots \\ S_k &\sqsubseteq \exists R^- . S_{k-l} \end{aligned}$$

For a specification  $\varphi = AG(b)$ , let  $B^j$  be the correspondent Boolean expression over the concepts  $V_i^j$  for all  $0 \leq j < l$ . We then define the concept

$$C_\varphi^{0..l-1} \equiv (\neg B^0 \sqcup \dots \sqcup \neg B^{l-1}) \sqcap (S_0 \sqcup S_l \sqcup S_{2*l} \sqcup \dots \sqcup S_k).$$

**Proposition 6.** (Size of encoding)  $|\mathcal{T}_{MD}^{k/l}| = 2 \cdot n \cdot l + k/l + 1$ .

The following proposition relates the terminology  $\mathcal{T}_{MD}^k$  to the reduced modality terminology  $\mathcal{T}_{MD}^{k/l}$ .

**Proposition 7.**  $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$  if and only if  $\mathcal{T}_{MD}^{k/l} \models_{dl} C_\varphi^{0..l}$ .

*Proof (sketch).* ( $\implies$ ) Suppose there exists an interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  for  $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$ . Since  $C_\varphi$  is not empty in this interpretation, there must exist a  $j$  such that  $S_j^{\mathcal{I}}$  is not

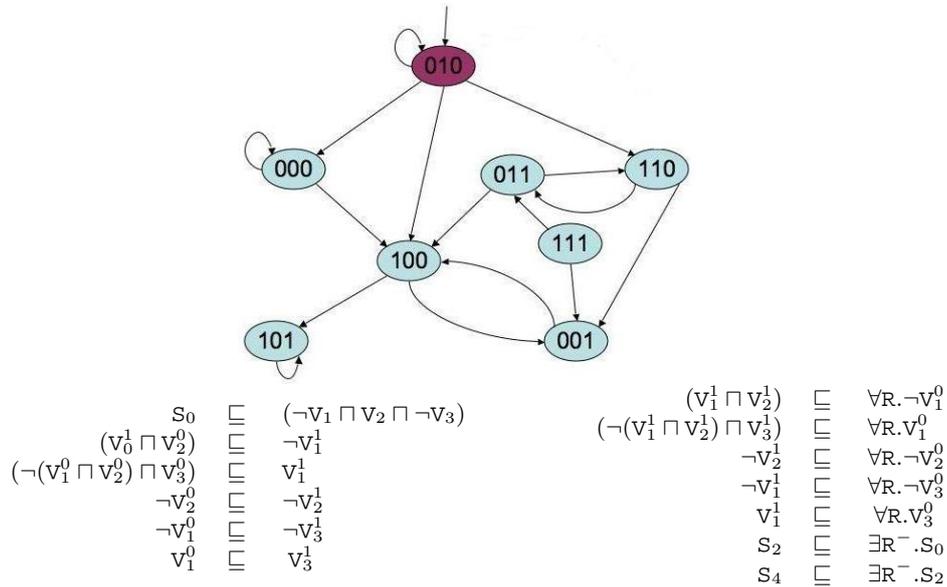
empty. Let  $\sigma_j \in S_j^{\mathcal{I}}$ . Since the concept inclusion  $S_j \sqsubseteq \exists R^-.S_{j-1}$  holds in the interpretation, and  $S_j^{\mathcal{I}}$  is not empty, we deduce that  $S_{j-1}^{\mathcal{I}}$  is not empty, and that  $\exists \sigma_{j-1} \in S_{j-1}^{\mathcal{I}}$ , such that  $(\sigma_{j-1}, \sigma_j) \in R^{\mathcal{I}}$ . By similar considerations, there must exist a sequence of elements  $\sigma_0, \dots, \sigma_j \in \Delta^{\mathcal{I}}$ , such that for  $0 \leq i < j$ ,  $(\sigma_i, \sigma_{i+1}) \in R^{\mathcal{I}}$ , and  $\sigma_0 \in S_0^{\mathcal{I}}$ . We build an interpretation  $\mathcal{I}_l = (\Delta^{\mathcal{I}_l}, \cdot^{\mathcal{I}_l})$  for  $\mathcal{T}_{MD}^{k/l}$ .  $\Delta^{\mathcal{I}_l}$  will consist of  $j/l + 1$  elements,  $\gamma_0, \dots, \gamma_{j/l}$ , where each  $\gamma_i$  corresponds to  $l$  consequent elements from  $\Delta^{\mathcal{I}}$ . The mapping  $\cdot^{\mathcal{I}_l}$  will be defined according to  $\cdot^{\mathcal{I}}$ . Thus,  $\forall 1 \leq i \leq n, \forall 0 \leq j < l, \gamma_0 \in V_i^j$  if and only if  $\sigma_j \in V_i$ . In a similar manner,  $\gamma_1$  will be mapped according to  $\sigma_l, \dots, \sigma_{2 \cdot l - 1}$  and  $\gamma_{k/l}$  according to  $\sigma_{k-l}, \dots, \sigma_j$ . It remains to show that the concept inclusions of  $\mathcal{T}_{MD}^{k/l}$  hold under the interpretation  $\mathcal{I}_l$ , and that the interpretation of  $C_\varphi^{0..l}$  is not empty. These follow easily from the definitions, given that all concept inclusions of  $\mathcal{T}_{MD}^k$  hold under  $\mathcal{I}$ .

The opposite direction proceeds in a similar way. □

**Example** Consider the model description

$$\text{Exmp} = (I, [\langle v_1 \wedge v_2, v_3 \rangle, \langle \neg v_2, v_1 \wedge \neg v_1 \rangle, \langle \neg v_1, v_1 \rangle])$$

over  $V = \{v_1, v_2, v_3\}$  with  $I = \neg v_1 \wedge v_2 \wedge \neg v_3$ . Figure 1 draws the states and transitions of the Kripke structure  $M_{\text{Exmp}}$  described by  $\text{Exmp}$ , where the label of each state is the value of the vector  $(v_1, v_2, v_3)$ . Let the formula to be verified be  $\varphi = AG(\neg v_2 \vee$



**Fig. 1.** Kripke Structure and Terminology for “Exmp”

$\neg v_3)$ . Note that  $M_{\text{Exmp}} \not\models \varphi$ , as can be seen in Figure 1, since the state  $(0, 1, 1)$ , that

contradicts  $\varphi$ , can be reached in two steps from the initial state. We choose the bound to be  $k = 4$ , and the reduction factor 2.

We build the terminology  $\mathcal{T}_{\text{Exmp}}^{4/2}$  for Exmp. We introduce one primitive role R and two sets of primitive concepts:  $V_1^0, V_2^0, V_3^0$  and  $V_1^1, V_2^1, V_3^1$ . The initial state, represented by the concept  $S_0$ , depends only on the set  $V_i^0$ :  $S_0 \sqsubseteq (\neg V_1^0 \sqcap V_2^0 \sqcap \neg V_3^0)$ . The rest of  $\mathcal{T}_{\text{Exmp}}^{4/2}$  is composed of the transition relation of the model, as given in Figure 1. For the specification  $\varphi = AG(\neg v_2 \vee \neg v_3)$  we have  $B^0 \equiv \neg V_2^0 \sqcup \neg V_3^0$  and  $B^1 \equiv \neg V_2^1 \sqcup \neg V_3^1$ . The concept  $C_\varphi^{0..1}$  is then defined as  $C_\varphi^{0..1} \equiv (\neg B^0 \sqcup \neg B^1) \sqcap (S_0 \sqcup S_2 \sqcup S_4)$ .

Verification is carried out by asking the query:  $\mathcal{T}_{\text{Exmp}}^{4/2} \models_{dl} C_\varphi^{0..1}$ .

## 4 Evaluation

We conducted an experiment on a model derived from the NuSMV example “dme1-16”, taken from [11], to test our hypothesis that reducing the number of nodes created during model building would improve performance when using a DL system for bounded model checking. The original model from [11] was composed of 16 symmetric “cells”, each consisting of 17 propositional variables. We reduced the model to have only 2 cells, in order to get a reasonable run-time. The formula verified expresses a safely condition that is satisfied in the model. We used the DL reasoner *FaCT++* [9], and as expected according to our translation, all runs returned an “unsatisfiable” result. Table 4 reports on the length of time required to determine this for reduction factors of 1 (no reduction), 2 and 4. We believe the times are clear evidence that what really happens is contrary to our hypothesis.

**Table 1.** Modal vs. Propositional Reasoning

Variables	Bound ( $k$ )	Reduction Factor	Time (m)
34	8	1	140
34	8	2	197
34	8	4	686

### 4.1 Discussion

Highly successful SAT solvers such as Minisat [8] use model building algorithms that operate by progressively refining an understanding of a “possible world”. DL systems such as *FaCT++* also use model building algorithms, but, to relate their behavior to typical SAT solvers, must deal with the added complication of modal reasoning in which a potentially large number of possible worlds are involved. For applications such as fixed depth model checking, DL systems therefore enable a tradeoff between the complexity of particular worlds and the number of worlds.

We were quite surprised that moving towards fewer but more complicated worlds would have the negative impact on performance that it did in our experiment, which prompted a lot of reflection on why this happens. We conclude with some suggestions on directions of future research on how DL technology might be adapted in order to improve its performance on applications like model checking.

Part of this reflection was to conduct a small literature survey on how modern SAT solvers and DL reasoners are implemented [1, 8]. It became quickly apparent that, e.g.,

Minisat relies heavily on using arrays to encode knowledge about a particular world, and that DL technology is more likely to encode similar knowledge in separate records, and to navigate among the records via pointers. It is common folklore that, when feasible, replacing pointer navigation with array indexing will improve the performance of algorithms, which suggests one possible avenue for improving the performance of propositional reasoning in DL systems.

A non-trivial problem for DL systems relates to *blocking*. In particular, such systems must frequently compare different possible worlds to ensure that model building will terminate. This prompted a more careful consideration of the structure of the terminology encoding a model description. We noticed that it might be straightforward to recognize that the “schema” underlying all occurrences of the “ $\exists R.C$ ” concept constructor was acyclic, which suggests a possible extension to, e.g., preprocessing in *FaCT++* in which such acyclic (sub)schema are recognized, and consequently that blocking activity during model building is disabled for possible worlds “within” acyclic schema. Indeed, any terminology generated by our reductions will always satisfy a global acyclicity property that would allow disabling any processing relating to blocking.

### Acknowledgements

We thank Dmitry Tsarkov for his support in the installation of *FaCT++*, and Peter Tarle and Nortel for many valuable discussions on this work and for financial support. The authors are also supported in part by grants from NSERC of Canada.

### References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. I. Beer, S. Ben-David, and A. Landver. On-the-fly model checking of RCTL formulas. In *Proc. 10<sup>th</sup> International Conference on Computer Aided Verification (CAV'98)*, LNCS 1427, pages 184–194. Springer-Verlag, 1998.
3. S. Ben-David, R. Treffer, and G. Weddell. Bounded model checking with description logic reasoning. In *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, pages 60–72, July 2007.
4. A. Biere, A. Cimatti, E. Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *TACAS'99*, 1999.
5. Randy Bryant. Graph-based algorithms for boolean function manipulation. In *IEEE Transactions on Computers*, volume c-35 no. 8.
6. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 2000.
7. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logics of Programs*, LNCS 131, pages 52–71. Springer-Verlag, 1981.
8. Niklas Een and Niklas Sorensson. An Extensible SAT-solver. In *Theory and Applications of Satisfiability Testing*, pages 502–518. Springer Berlin/Heidelberg, LNCS 2919, 2004.
9. I. Horrocks. The FaCT system. pages 307–312, 1998.
10. K. McMillan. *Symbolic model checking*, 1993.
11. NuSMV examples collection. <http://nusmv.irst.itc.it/examples/examples.html>.
12. Amir Pnueli. The temporal logic of programs. In *18th IEEE Symposium on Foundation of Computer Science*.
13. J. Quielle and J. Sifakis. Specification and verification of concurrent systems in cesar. In *5th International Symposium on Programming*, 1982.

## A general framework for covering concepts using terminologies

B. Benatallah<sup>1</sup>, M-S. Hacid<sup>2</sup>, A. Léger<sup>3</sup>, C. Rey<sup>4</sup>, and F. Toumani<sup>4</sup>

<sup>1</sup> SCSE, University of New South Wales, Sydney, [boualem@cse.unsw.edu.au](mailto:boualem@cse.unsw.edu.au)

<sup>2</sup> LIRIS, Université Lyon I, France, [mshacid@bat710.univ-lyon1.fr](mailto:mshacid@bat710.univ-lyon1.fr)

<sup>3</sup> France Telecom R&D, Rennes, France, [alain.leger@francetelecom.com](mailto:alain.leger@francetelecom.com)

<sup>4</sup> LIMOS, CNRS, Université Blaise Pascal, [{rey,ftoumani}@isima.fr](mailto:{rey,ftoumani}@isima.fr)

**Abstract.** We describe a general framework for covering concepts using terminologies and briefly present the already investigated instances of this framework. Then, we formalize the best covering problem in the context of the  $\mathcal{ALN}$  language, in which the difference operation is not semantically unique, and sketches the technique to solve the underlying computation problems.

### 1 Introduction

In [2] a general framework for *rewriting using terminologies* is defined as follows:

- given a terminology  $\mathcal{T}$  expressed using a language  $\mathcal{L}_t$ ,
- given a (query) concept description  $Q$ , expressed using a language  $\mathcal{L}_s$  and that does not contain concept names defined in  $\mathcal{T}$ ,
- given a binary relation  $\rho : \mathcal{L}_s \times \mathcal{L}_t$ , between  $\mathcal{L}_s$  and  $\mathcal{L}_t$  descriptions.

Can  $Q$  be rewritten into a description  $E$ , expressed using a language  $\mathcal{L}_d$  and built using (some) of the names defined in  $\mathcal{T}$ , such that  $Q\rho E$  ?

Additionally, some optimality criterion may be used in order to select the relevant rewritings.

Existing instances of this general framework can be distinguished with respect to the nature of the relation  $\rho$ , the optimality criterion as well as the languages  $\mathcal{L}_t$ ,  $\mathcal{L}_s$  and  $\mathcal{L}_d$ , respectively used to describe the terminology  $\mathcal{T}$  the (query) concept  $Q$  and the rewriting  $E$ . Examples of such instances are (i) the minimal rewriting problem [2], where  $\rho$  is instantiated by equivalence modulo  $\mathcal{T}$  while the size of the rewriting is used as the optimality criterion and (ii) rewriting queries using views [5], while  $\rho$  is instantiated by subsumption and the optimality criterion is the inverse subsumption [2].

Subsumption relation plays a central role in the existing rewriting approaches. Indeed, all the mentioned instances of the general framework aim at reformulating a given query  $Q$  into a description which is equivalent or subsumed by  $Q$ . The intuition here is that a given rewriting must capture all the ‘information’ conveyed by a query  $Q$ . However, in many application contexts (e.g., see [3, 4]) it is not realistic to assume that such a rewriting always exists and it may be

interesting to look for a rewriting that *approximates* a given query. This observation motivates our work on a new instance of the general framework for rewriting, namely *covering concepts using terminologies* [3]. The salient feature of our approach is to use a measure of a *semantic distance* between concepts, instead of subsumption, to define rewritings, thereby enabling a more *flexible* rewriting process. More precisely, our aim is to reformulate a query  $Q$  into a description that contain as much as possible of *common information* with  $Q$ . We call such a reformulation a *cover* of  $Q$ .

A key step toward handling such a problem is a precise definition of the measure used to compute the semantic proximity between concepts. We rely on a non standard operation in description logics, namely the (semantic) *difference or subtraction* operation, in order to define such a measure. The difference of two descriptions is defined in [6] as being a description containing all information which is a part of one argument but not a part of the other one. An interesting feature of the difference operation comes from its ability to produce a (set of) concept description(s) as output. In the sequel, we refer to descriptions obtained by the difference operation as *difference descriptions*. Note that, if the difference is not semantically unique, a difference description is in fact a set of descriptions. Difference descriptions characterize the notion of ‘*extra information*’, i.e., the information contained in one description and not contained in the other, thereby providing a means to measure a semantic distance between concepts.

With the difference operation at hand, and given an appropriate ordering  $\prec_d$  on difference descriptions, we are then interested by the problem of rewriting a query  $Q$  into a description  $E$  such that the difference between  $Q$  and  $E$  is minimal w.r.t.  $\prec_d$ . In our previous work [3, 4], we investigated this problem in a restricted framework of languages in which the difference is semantically unique. In such a framework it turns out to be sufficient to specify  $\prec_d$  as an ordering on the size of descriptions to capture the intuition behind the notion of best covers (i.e., covers whose descriptions contain as much as possible of *common information* with the original query). However, in the cases where the difference produces a set of descriptions, for example in the  $\mathcal{ALN}$  language, a more subtle ordering is required. This paper extends our previous work in the following directions: (i) we define a general framework for covering concepts using terminologies and point out how the previous investigated instances can be formalized in this context, and (ii) we formalize the best covering problem in the context of the  $\mathcal{ALN}$  language, in which the difference operation is not semantically unique, and sketches the technique to solve the underlying computation problems. Technical details regarding this new result are available in [1].

## 2 The general best covering problem

This section introduces some basic definitions to formally define the *general best covering problem*.

**Definition 1. (Cover)** Let  $\mathcal{L}$  be a DL in which the difference operation is computable,  $\mathcal{T}$  (respectively,  $Q$ ) be an  $\mathcal{L}$ -terminology (respectively, an  $\mathcal{L}$ -concept) and

let  $E$  be a conjunction of some defined concepts from  $\mathcal{T}$ .  $E$  is a **cover** of  $Q$  using  $\mathcal{T}$  iff: (i)  $E$  is consistent with  $Q$ , i.e.,  $Q \sqcap E \neq \perp$ , and (ii)  $E$  shares some information with  $Q$ , i.e.,  $Q \notin_{\equiv} Q - lcs_{\mathcal{T}}(Q, E)$ , where  $\in_{\equiv}$  stands for set membership modulo equivalence.

Hence, a cover of a concept  $Q$  using  $\mathcal{T}$  is defined as being a conjunction of defined concepts occurring in  $\mathcal{T}$  which is consistent with  $Q$  and that share some information with  $Q$ . We use the expression  $rest_E(Q) = Q - lcs_{\mathcal{T}}(Q, E)$ , called the *rest* of a cover, to denote the part of a query  $Q$  that is not captured by the cover  $E$ . In practical situations, however, we are not interested in all kinds of covers. Therefore, we define additional criteria to characterize the notion of *relevant covers*. For example, it is clearly not interesting to consider those covers that do not minimize the rest. Then, given an appropriate ordering  $\prec_d$  on cover rests, the notion of closest covers is defined below.

**Definition 2. (Closest cover w.r.t.  $\prec_d$ ).** Let  $\mathcal{L}$  be a DL in which the difference operation is computable,  $\mathcal{T}$  (respectively,  $Q$ ) be an  $\mathcal{L}$ -terminology (respectively, an  $\mathcal{L}$ -concept) and let  $E$  be a conjunction of some defined concepts from  $\mathcal{T}$ .  $E$  is a **closest cover** of  $Q$  using  $\mathcal{T}$  w.r.t.  $\prec_d$  (or simply, **closest cover** of  $Q$  using  $\mathcal{T}$ ) iff: (i)  $E$  is a **cover** of  $Q$  using  $\mathcal{T}$  and (ii) it does not exist a cover  $E'$  of  $Q$  using  $\mathcal{T}$  such that  $rest_{E'}(Q) \prec_d rest_E(Q)$ .

Closest covers correspond to those covers that minimize part of a query  $Q$  not captured in a cover. Hence, they are clearly relevant rewritings in practical situations. For example, when contained or equivalent rewritings of  $Q$  using  $\mathcal{T}$  exist, they constitute the closest covers of  $Q$ .

However, usually it may not be interesting or efficient to compute all the possible closest covers. For example, in existing rewriting approaches, it does not make a lot of sense to compute all the rewritings contained in a given query. Usually, one is interested by either maximally-contained or equivalent rewritings. Similarly to the general framework introduced above, an additional optimality criterion can be used to select among the closest covers of a query  $Q$ , the most relevant ones. To abstract from particular optimality criterion, assume that we are provided with an ordering, noted  $\prec_c$ , on concept covers such that  $E' \prec_c E$  means that the cover  $E$  is better (or of higher quality) than the cover  $E'$ . As will be seen later, when defined appropriately the ordering  $\prec_c$  can be used for example to capture the semantics of maximally-contained rewritings or, more interestingly, to maximize the user satisfaction with respect to a given set of Quality of Service (QoS) criteria.

Definition 3 given below characterizes the notion of *best covers* w.r.t.  $\prec_d$ , i.e., a closest cover that is considered as *optimal* according to the ordering  $\prec_c$ .

**Definition 3. (Best cover w.r.t.  $(\prec_d, \prec_c)$ ).** Let  $\mathcal{L}$  be a DL in which the difference operation is computable,  $\mathcal{T}$  (respectively,  $Q$ ) be an  $\mathcal{L}$ -terminology (respectively, an  $\mathcal{L}$ -concept) and let  $E$  be a conjunction of some defined concepts from  $\mathcal{T}$ . Given two orderings  $\prec_d$  and  $\prec_c$ ,  $E$  is a **best cover** of  $Q$  using  $\mathcal{T}$  w.r.t.  $(\prec_d, \prec_c)$  iff: (i)  $E$  is a **closest cover** of  $Q$  using  $\mathcal{T}$  w.r.t.  $\prec_d$ , and (ii) it does not exist a closest cover  $E'$  of  $Q$  using  $\mathcal{T}$  w.r.t.  $\prec_d$  such that  $E' \prec_c E$ .

Finally, we are now able to provide a precise definition for the general *best covering problem*.

*Problem 1. (GBCP( $\mathcal{T}, Q$ )).* Let  $\mathcal{L}$  be a DL in which the difference operation is computable,  $\mathcal{T}$  (respectively,  $Q$ ) be an  $\mathcal{L}$ -terminology (respectively, an  $\mathcal{L}$ -concept) and let  $\prec_d$  be an ordering on difference descriptions and  $\prec_c$  be an ordering on concept covers. The general best covering problem, denoted  $GBCP(\mathcal{T}, Q)$ , is the problem of computing all the best covers of  $Q$  using  $\mathcal{T}$  w.r.t.  $(\prec_d, \prec_c)$ .

Note that, problem 1 provides a general framework for covering concepts using terminologies. This framework can have different instantiations depending on the precise language  $\mathcal{L}$  used to describe  $\mathcal{T}$  and  $Q$  as well as the precise definition of the orderings  $\prec_d$  and  $\prec_c$ .

### 3 Investigated instances

Motivated by different application contexts, we have investigated three instances of the general best covering problem. A first line of demarcation between the studied instances comes from the properties of the difference operation used in each setting. We considered two cases in our work:

- **Languages in which the difference is semantically unique.** As described in [3, 4], in this case it is enough to consider  $\prec_d$  as an ordering on the size of descriptions. In the sequel, we use the notation  $\prec_d^{\parallel}$  to denote that the ordering  $\prec_d$  is defined on the size of descriptions.
- **The  $\mathcal{ALN}$  language.** In the setting of  $\mathcal{ALN}$  the difference operation is not semantically unique and produces a set of descriptions. In this case a more subtle definition of the ordering  $\prec_d$  is required. In Section 4, we provide a definition for such an ordering based on an extension of the subsumption relation to sets of descriptions. To differentiate with the first case, we note such an ordering  $\prec_d^S$  where the superscript  $S$  is used to recall that the ordering  $\prec_d$  is defined on sets of descriptions.

Table 1 presents the three instances of the best covering problem we have investigated in our work. The first two instances, respectively called  $BCOV(\mathcal{T}, Q)$  and  $QoS-BCOV(\mathcal{T}, Q)$ , consider the family of languages equipped with structural subsumption<sup>5</sup>. Such a property ensures that the difference operation is semantically unique and can be determined using the *structural difference operation* [6]. Hence, both  $BCOV(\mathcal{T}, Q)$  and  $QoS-BCOV(\mathcal{T}, Q)$  use the ordering  $\prec_d^{\parallel}$  to characterize closest covers (i.e., they define  $\prec_d = \prec_d^{\parallel}$ ). However, these two instances differ in the specification of the ordering  $\prec_c$  which was, in each case, motivated by the application context. In  $BCOV(\mathcal{T}, Q)$ , the purpose was to select only the closest covers that contain as less as possible of *extra information* with

<sup>5</sup> Note that we use here the definition of structural subsumption in the sense of [6] which is different from the one usually used in the literature.

Instance Name	$\mathcal{L}$	Ordering $\prec_d$	Ordering $\prec_c$	Applications	Refs
$BCOV(\mathcal{T}, Q)$	Structural subsumption	$\prec_d^  $ an ordering on size of the rest	$\prec_c^m$ an ordering on size of the missing information	Service discovery	[3]
$QoS-BCOV(\mathcal{T}, Q)$	Structural subsumption	$\prec_d^  $ an ordering on size of the rest	$\prec_c^q$ an ordering w.r.t. a QoS function	Querying e-catalogs	[4]
$\mathcal{ALN}-BCOV(\mathcal{T}, Q)$	$\mathcal{ALN}$	$\prec_d^S$ an ordering on size of the rest	$\prec_c^S$ an ordering on size of the missing information	Service discovery	[1]

Table 1. Investigated instances of the general best covering problem.

respect to a query  $Q$ . We call the part of a cover  $E$  that is not contained in the description of the query  $Q$  the *missing information*. Hence, the ordering  $\prec_c^m$ , used to instantiate  $\prec_c$  in the context of  $BCOV(\mathcal{T}, Q)$ , is defined as an ordering on the size of the missing information. In  $QoS-BCOV(\mathcal{T}, Q)$ , however, the purpose was to select only the closest covers that maximize the user satisfaction with respect to a given set of Quality of Service (QoS) criteria (e.g., price, execution time, reliability, etc). Hence, in the context of  $QoS-BCOV(\mathcal{T}, Q)$ ,  $\prec_c$  is instantiated as an ordering, noted  $\prec_c^q$ , that sorts the covers of a query  $Q$  w.r.t. their *quality scores*.

Finally, the third instance, called  $\mathcal{ALN}-BCOV(\mathcal{T}, Q)$ , is an extension of  $BCOV(\mathcal{T}, Q)$  to the language  $\mathcal{ALN}$  in which the difference operation is not semantically unique (i.e., it produces sets of descriptions). Hence,  $\mathcal{ALN}-BCOV(\mathcal{T}, Q)$  uses the ordering  $\prec_d^S$  to define the closest covers (i.e.,  $\prec_d = \prec_d^S$ ) and the ordering  $\prec_c^S$  to define the best covers (i.e.,  $\prec_c = \prec_c^S$ ). The ordering  $\prec_c^S$  sorts covers of a query  $Q$  by order of their missing information in the case where missing information are expressed as sets of descriptions.

#### 4 The problem $\mathcal{ALN}-BCOV(\mathcal{T}, Q)$

We consider the problem  $\mathcal{ALN}-BCOV(\mathcal{T}, Q)$ , an extension of  $BCOV(\mathcal{T}, Q)$  to the language  $\mathcal{ALN}$ . A main feature that sets the  $\mathcal{ALN}$  setting apart from the two previous ones lies in the possibility of nontrivial decomposition of the inconsistent concept  $\perp$ . As an example, consider the following two decompositions of  $\perp$ :  $\perp \equiv (\leq 2R) \sqcap (\geq 4R) \equiv P \sqcap \neg P$ , where  $P$  denotes an atomic concept. Consequently, as highlighted below, new difficulties arise when dealing with the best covering problem in this context: (i) the difference operation is not semantically unique and produces sets of descriptions as a result. Therefore, to handle the best covering problem in this context there is a need for:

- an effective procedure to compute difference descriptions in  $\mathcal{ALN}$ , and
- a new formalization of the best covering problem. This is because the *rest* of a cover as well as the missing information are now expressed as sets of

descriptions and hence the orderings  $\prec_d^{\parallel}$  and  $\prec_c^m$ , respectively used in the case of  $\mathcal{BCOV}(\mathcal{T}, Q)$  to instantiate  $\prec_d$  and  $\prec_c$ , are no longer valid in this context.

(ii) The possibility to obtain inconsistent conjunctions of consistent concepts. Therefore, when computing best covers as conjunction of (consistent) defined concepts we have to ensure that only 'consistent' covers are generated.

Altogether these points make the best covering problem much more complex to solve in the context of  $\mathcal{ALN}$ . We describe below a formalization of the best covering problem in this setting, then we sketch an approach to solve it.

#### 4.1 Problem statement

Analogous to  $\mathcal{BCOV}(\mathcal{T}, Q)$ , in  $\mathcal{ALN}\text{-}\mathcal{BCOV}(\mathcal{T}, Q)$  we are interested by the computation of covers that contain as much as possible of *common information* with  $Q$  and as less as possible of *extra information* with respect to  $Q$ . The main difficulty encountered when formalizing  $\mathcal{ALN}\text{-}\mathcal{BCOV}(\mathcal{T}, Q)$  lies in the specification of the orderings  $\prec_d$  and  $\prec_c$  which, in this context, will be respectively used for minimizing the rest and the missing information. To this end, we introduce below a slight extension of the subsumption relation to sets of descriptions and then we show how it can be used for specifying the orderings  $\prec_d$  and  $\prec_c$  in the context of  $\mathcal{ALN}$ .

##### Definition 4. (Subsumption between sets of descriptions)

Let  $C = \{c_1, \dots, c_n\}$  and  $D = \{d_1, \dots, d_m\}$  be two sets of  $\mathcal{ALN}$ -descriptions. The set  $C$  is subsumed by  $D$ , noted  $C \sqsubseteq_S D$  iff  $\forall c_i \in C, \exists d_j \in D | c_i \sqsubseteq d_j$

The orderings  $\prec_d$  and  $\prec_c$  of the general best covering problem  $GBCP(\mathcal{T}, Q)$  are respectively instantiated in the setting of  $\mathcal{ALN}$  using the orderings  $\prec_d^S$  and  $\prec_c^S$  defined below.

##### Definition 5. (The orderings $\prec_d^S$ and $\prec_c^S$ )

Let  $Q$  be an  $\mathcal{L}$ -concept description and  $E$  and  $E'$  two covers of  $Q$  using  $\mathcal{T}$ .

- The ordering  $\prec_d^S$  is defined as follows  
 $E \prec_d^S E'$  iff  $rest_{E'}(Q) \sqsubseteq_S rest_E(Q)$
- The ordering  $\prec_c^S$  is defined as follows  
 $E \prec_c^S E'$  iff  $Miss_{E'}(Q) \sqsubseteq_S Miss_E(Q)$

$\mathcal{ALN}\text{-}\mathcal{BCOV}(\mathcal{T}, Q)$  is then obtained from the general covering problem, by instantiating the language  $\mathcal{L} = \mathcal{ALN}$  and the orderings  $\prec_d = \prec_d^S$  and  $\prec_c = \prec_c^S$ . As in the case of  $\mathcal{BCOV}(\mathcal{T}, Q)$ , in  $\mathcal{ALN}\text{-}\mathcal{BCOV}(\mathcal{T}, Q)$  we are interested by the computation of the non redundant best covers.

##### Problem 2. ( $\mathcal{ALN}\text{-}\mathcal{BCOV}(\mathcal{T}, Q)$ ).

Let  $\mathcal{T}$  (respectively,  $Q$ ) be an  $\mathcal{ALN}$ -terminology (respectively, an  $\mathcal{ALN}$ -concept).  $\mathcal{ALN}\text{-}\mathcal{BCOV}(\mathcal{T}, Q)$  is the problem of computing all the non redundant best covers of  $Q$  using  $\mathcal{T}$  w.r.t.  $(\prec_d^S, \prec_c^S)$ .

The problem  $\mathcal{ALN}\text{-}\mathcal{BCOV}(\mathcal{T}, Q)$  is NP-hard. This complexity result is easily derived from the complexity of the  $\mathcal{BCOV}(\mathcal{T}, Q)$  problem [3].

#### 4.2 Dealing with $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$

In this section, we turn our attention to the computational problem underlying  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$ . The aforementioned features of  $\mathcal{ALN}$  make this problem much more complex to solve than the previous instances. A complete description of the solution to  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  is lengthy and technical. We sketch below the main steps of our approach for handling  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$ . Technical details are given in [1] where an algorithm, called *computeALNBCov*, to solve  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  is proposed.

The first step in dealing with  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  consists in the design of an algorithm that implements the difference operation between  $\mathcal{ALN}$  descriptions [1]. Then, with such an algorithm at hand, we investigated the computational problem underlying  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$ . Let  $\mathcal{C}_{\mathcal{T}}$  be the set of defined concept names that appear in  $\mathcal{T}$ . The search space of  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  is the power set of  $\mathcal{C}_{\mathcal{T}}$ . Unfortunately, a similar approach to the one used for  $\text{BCOV}(\mathcal{T}, Q)$  cannot be exploited here to avoid an exhaustive exploration of this search space. Indeed, in the case of  $\text{BCOV}(\mathcal{T}, Q)$  [3], a full characterization of best covers in terms of hypergraph transversals yields to a reduction of  $\text{BCOV}(\mathcal{T}, Q)$  to a computation of the minimal transversals with minimal cost of an associated hypergraph. The characterization of best covers in  $\mathcal{ALN}$  context is more complex.

The approach we have developed to cope with  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  embody a *divide-and-conquer* strategy that enables to progressively reduce the search space. We decompose the  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  problem into a set of smaller tasks by considering separately each criterion that must be satisfied by best covers in this setting. Then, for each criterion we provide a characterization that enables to confine the search space.

Observe that a solution  $E$  of an  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  problem must satisfy the following conditions: (Cond-1:)  $E \sqcap Q \not\equiv \perp$ , and (Cond-2:)  $E$  is a cover of  $Q$  (i.e.,  $Q \not\equiv Q - lcs(Q, E)$ ), and (Cond-3:)  $E$  is a closest cover of  $Q$  (i.e.,  $Rest_E(Q)$  is minimal w.r.t.  $\prec_d^S$ ).

For each of these conditions we provide a characterization that yields to an upper/lower bound in the search space. More precisely, given an  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  problem and let  $\mathcal{C}_{\mathcal{T}}$  be the set of defined concept names appearing in  $\mathcal{T}$ , we provide full characterizations for the following borders:

- $S_{cons}$ : the greatest subsets of  $\mathcal{C}_{\mathcal{T}}$  that satisfy Cond-1.
- $S_{couv}$ : the greatest subsets of  $\mathcal{C}_{\mathcal{T}}$  that satisfy Cond-1 and Cond-2.
- $I_{couv}$ : the smallest subsets of  $\mathcal{C}_{\mathcal{T}}$  that satisfy Cond-1 and Cond-2.
- $S_{rest}$ : the greatest subsets of  $\mathcal{C}_{\mathcal{T}}$  that satisfy Cond-1, Cond-2 and Cond-3.
- $I_{rest}$ : the smallest subsets of  $\mathcal{C}_{\mathcal{T}}$  that satisfy Cond-1, Cond-2 and Cond-3.

Based on the characterizations of the aforementioned borders, the proposed algorithm *computeALNBCov* breaks  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$  into the following subproblems: (1) Computation of  $S_{cons}$ , (2) Computation of  $S_{couv}$  and  $I_{couv}$ , (3) Computation of  $S_{rest}$  and  $I_{rest}$ , and (4) Computation of the best covers.

Briefly stated, the algorithm *computeALNBCov* proceeds in four steps each of which consisting in the resolution of one subproblem. The first three steps

exploit the provided characterizations of the aforementioned borders to progressively confine the search space of  $\mathcal{ALN}\text{-BCOV}(\mathcal{T}, Q)$ . Unfortunately, due to the non-monotonic nature of the ordering  $\prec_c^S$ , we were not able to provide a full characterization for the last step. Consequently, to handle step 4, the algorithm *computeALNBCov* enumerates all the covers confined between  $S_{rest}$  and  $I_{rest}$  and test for minimality w.r.t.  $\prec_c^S$ . Due to previous reductions of the search space, we expect step 4 to be still handled efficiently in practical cases. The implementation of *computeALNBCov* is an ongoing work. In our future work, we plan to conduct experiments on real cases as well as synthetic ontologies in order to evaluate the performance of this algorithm.

## 5 Conclusion

This paper focuses on the problem of covering concepts using terminologies. This work has relation with exiting works on concept/query rewriting. The salient feature of our approach is to use a measure of *semantic distance* between concepts, instead of subsumption relation, to define rewritings, thereby enabling a more *flexible* rewriting process. We provided a formal definition of a general framework for covering concepts using terminologies and described some already investigated instances of this problem. We then studied a new instance of the covering problem in the context of the  $\mathcal{ALN}$  language, in which the difference operation is not semantically unique.

## References

1. C. Rey B. Benatallah, A. Leger and F. Toumani. Covering concepts using terminologies in the ALN language. Technical report, LIMOS, Clermont-Ferrand, France, <http://www.isima.fr/rey/alnBcov.pdf>, 2007.
2. F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies – revisited. Report 00-04, LTCS, RWTH Aachen, Germany, 2000.
3. B. Benatallah, M-S. Hacid, A. Leger, C. Rey, and F. Toumani. On automating Web services discovery. *VLDB J.*, 14(1):84–96, 2005.
4. B. Benatallah, M-S. Hacid, H-Y. Paik, C. Rey, and F. Toumani. Towards semantic-driven, flexible and scalable framework for peering and querying e-catalog communities. *Information Systems*, 31(4-5), 2006.
5. Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
6. G. Teege. Making the difference: A subtraction operation for description logics. In J. Doyle, E. Sandewall, and P. Torasso, editors, *KR'94*, San Francisco, CA, 1994. Morgan Kaufmann.

## Expressing DL-Lite Ontologies with Controlled English

Raffaella Bernardi, Diego Calvanese, Camilo Thorne

Faculty of Computer Science  
Free University of Bozen-Bolzano  
Piazza Domenicani 3, Bolzano, Italy  
{bernardi, calvanese, cthorne}@inf.unibz.it

**Abstract.** We are interested in providing natural language front-ends to databases upon which an ontology layer has been added. Specifically, here we deal with how to express ontologies formalized in Description Logics in a controlled language, i.e., a fragment of natural language tailored to compositionally translate into a knowledge representation (KR) language. As KR language we have chosen  $DL-Lite_{R,\sqcap}$ , a representative of the well-known  $DL-Lite$  family [3, 4], and we aim at understanding the kind of English constructs the controlled language can and cannot have to correspond to  $DL-Lite_{R,\sqcap}$ . Hence, we compare the expressive power of  $DL-Lite_{R,\sqcap}$  to that of various fragments of FOL identified by Pratt and Third as corresponding to fragments of English [8]. Our analysis shows that  $DL-Lite_{R,\sqcap}$ , though itself tractable, is incomparable in expressive power with respect to tractable fragments of English. Interestingly, it allows one to represent a restricted form of relative clauses, which lead to intractability when used without restrictions on the occurrences of negations, and existential quantifiers.

### 1 Introduction

The importance of using an ontology to facilitate the access of users to structured data is well established [2, 3]. Having an ontology as support for querying a database (DB) will allow the user to find the relevant answers without knowing about the structure of the DB itself. Though having an ontology will provide support to users able to use query languages, it will still fail to make the data accessible to non expert users. These could instead benefit from using a natural language interface to the ontology and the DB, both for querying the DB and for entering knowledge, either intensional (i.e., ontology assertions) or extensional (i.e., DB facts) one. Therefore, we are interested in looking at the *query entailment problem*, i.e.,  $\mathcal{T} \cup \mathcal{D} \models \varphi$ , for an ontology  $\mathcal{T}$ , database  $\mathcal{D}$ , and query  $\varphi$ , but from a natural language perspective.

We know that query entailment can be done efficiently (i.e., in LOGSPACE in the size of the DB  $\mathcal{D}$ ), if the ontology  $\mathcal{T}$  is expressed in a Description Logic (DL) of the  $DL-Lite$  family [3, 4] and the query  $\varphi$  is a (union of) conjunctive queries (CQs). When resorting to natural language interfaces, we aim at preserving this efficiency. Thus, we are interested in understanding (i) which fragments of natural language correspond to the two fragments of First-Order Logic (FOL) we need, viz.  $DL-Lite$  and CQs, and (ii) whether these two fragments will be suitable for non expert users to accomplish the tasks we are interested in, entering intensional and extensional knowledge into an ontology and querying a DB.

Roughly, with respect to FOL, CQs lack negation and universal quantification. This might seem too restrictive when interested in expressing natural language questions as DB queries. However, an analysis of several corpora of real life users’ questions<sup>1</sup> has shown that the use of those operators in questions is rather limited. Similarly, we are now trying to understand how far *DL-Lite* is from the linguistic structures that domain experts would naturally use to describe their intensional knowledge. To this end, as a preliminary study, we have started looking at the answers provided by domain experts to FAQs<sup>2</sup>. Again, the first results are rather promising, showing that domain experts, when allowed to freely use natural language, write rather simple structures with only few occurrences of those operators “forbidden” by *DL-Lite* definition, e.g., universal quantifiers in non subject position. Similarly, the use of negation and disjunction is rather limited and controlled while relative pronouns instead are rather common in these corpora and they are usually used to further specify properties of the nearest noun. As will become clearer in the next section, these operators are relevant to understand the connection between *DL-Lite* and natural language fragments since their corresponding logical operators are the major players in determining the complexity property of the entailment problem above.<sup>3</sup>

Against this background, our research line is as follows. We propose to study the problem of accessing structured data via an ontology by moving back and forth between logic and natural language: on the one hand, by studying the expressivity of suitable logic fragments and identifying the corresponding natural language fragments, and on the other hand, by analysing natural language structures used in real life applications and trying to extend the corresponding logic fragments to better suit users’ needs, but without paying in terms of computational complexity.

In this paper, we concentrate on *DL-Lite*<sub>R,∩</sub>, which is the DL that stays as close as possible to the expressive power required to capture natural language constructs, while still preserving the nice computational properties of the *DL-Lite* family. As a first step towards understanding the relationship between ontology languages and natural language constructs, we compare *DL-Lite*<sub>R,∩</sub> with the expressive power of fragments of FOL studied by Pratt and Third [8, 10] and defined starting from natural language.

## 2 The Description Logic *DL-Lite*<sub>R,∩</sub>

In this work, we consider a DL belonging to the *DL-Lite* family [3, 4], and specifically, we consider *DL-Lite*<sub>R,∩</sub>, in which the TBox is constituted by a set of (concept and role) *inclusion assertions* of the form  $Cl \sqsubseteq Cr$  and  $R_1 \sqsubseteq R_2$ , where  $Cl$  and  $Cr$  denote concepts that may occur respectively on the left and right-hand side of inclusion assertions, and  $R_1, R_2$  denote roles, constructed according to the following syntax:

$$\begin{array}{ll} Cl \longrightarrow A \mid \exists R \mid Cl_1 \sqcap Cl_2 & R \longrightarrow P \mid P^- \\ Cr \longrightarrow A \mid \exists R \mid Cr_1 \sqcap Cr_2 \mid \exists R.A \mid \neg A \mid \neg \exists R \end{array}$$

<sup>1</sup> [http://wiki.answers.com/Q/WikiFAQs:Finding\\_Questions\\_to\\_Answer](http://wiki.answers.com/Q/WikiFAQs:Finding_Questions_to_Answer)  
<http://clinques.nlm.nih.gov/JitSearch.html> (clinical questions)

<sup>2</sup> <http://www.unibz.it/library/faq/>

<sup>3</sup> What seems to cause a lack of expressivity is the limitation on the occurrences of qualified existential, viz., the fact that they cannot occur in the left concepts of TBox statements, in all of the logics of the *DL-Lite* family. This will be the topic of further studies.

where  $A$  denotes an atomic concept, and  $P$  denotes an atomic role.

For convenience w.r.t. what we need in the following sections, we formally specify the semantics of  $DL-Lite_{R,\sqcap}$ , by providing its translation to FOL. Specifically, we map each concept  $C$  (we use  $C$  to denote an arbitrary concept, constructed applying the rules above) to a FOL formula  $\varphi(C, x)$  with one free variable  $x$  (i.e., a unary predicate), and each role  $R$  to a binary predicate  $\varphi(R, x, y)$  as follows:

$$\begin{array}{ll} \varphi(A, x) = A(x) & \varphi(\exists R, x) = \exists y(\varphi(R, x, y)) \\ \varphi(\neg C, x) = \neg\varphi(C, x) & \varphi(\exists R.C, x) = \exists y(\varphi(R, x, y) \wedge \varphi(C, y)) \\ \varphi(C_1 \sqcap C_2, x) = \varphi(C_1, x) \wedge \varphi(C_2, x) & \\ \varphi(P, x, y) = P(x, y) & \varphi(P^-, x, y) = P(y, x) \end{array}$$

Inclusion assertions  $Cl \sqsubseteq Cr$  and  $R_1 \sqsubseteq R_2$  of the TBox correspond then, respectively, to the universally quantified FOL sentences:

$$\forall x(\varphi(Cl, x) \rightarrow \varphi(Cr, x)) \qquad \forall x\forall y(\varphi(R_1, x, y) \rightarrow \varphi(R_2, x, y))$$

In  $DL-Lite_{R,\sqcap}$ , an ABox is constituted by a set of assertions on *individuals*, of the form  $A(c)$  or  $P(a, b)$ , where  $A$  and  $P$  denote respectively an atomic concept and an atomic role, and  $a$ , and  $b$  denote constants. As in FOL, each constant is interpreted as an element of the interpretation domain, and we assume that distinct constants are interpreted as distinct individuals, i.e., we adopt the *unique name assumption* (UNA). However, in  $DL-Lite_{R,\sqcap}$ , we may drop such an assumption without affecting the complexity of reasoning, as established below. The above ABox assertions correspond to the analogous FOL facts, or, by resorting to the above mapping, to  $\varphi(A, x)(c)$  and  $\varphi(R, x, y)(a, b)$ , respectively.

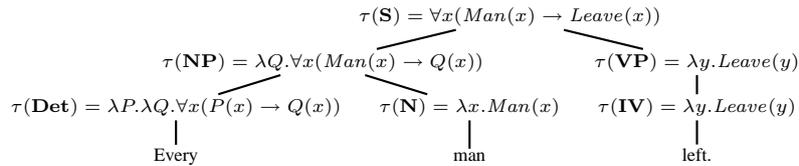
The reasoning services of interest for  $DL-Lite_{R,\sqcap}$  knowledge bases are the standard ones, namely *knowledge base satisfiability*, and concept and role *satisfiability*, and *subsumption*. It has been shown in [4] that in  $DL-Lite_{R,\sqcap}$  all such reasoning services are polynomial in the size of the knowledge base, and LOGSPACE in the size of the ABox only, i.e., in *data complexity*. Moreover, answering conjunctive queries whose atoms have as predicates atomic concepts and roles of a knowledge base, is also polynomial in the size of the knowledge base and in LOGSPACE in data complexity [3, 4].

### 3 Fragments of English

In this section we give a brief overview of Third and Pratt’s controlled fragments of English (cf. [8]). They are subsets of standard English meant to capture some simple, albeit for our purpose important, structure of English. Their interest, as we said in the introduction, lies in the fact that we would like to know which subset of English we can use to express only those data constraints required by ontology-driven data access. It is thus crucial to know which natural language constructs express these constraints and, more specifically, those suitable for a  $DL-Lite$  ontology.

The key feature of these fragments of English is that they compositionally translate, modulo the standard semantic mapping foreseen by montagovian natural language formal semantics (cf. [6, 7]) into several fragments of FOL. Roughly: (i) A parse tree is

computed. (ii) A FOL formula enriched with lambda operators from the lambda calculus is assigned to the words, i.e., the terminal nodes of the tree, representing their set-theoretical meaning. (iii) The logical formula representing the meaning of the parsed sentence is computed bottom-up by means of function application and beta-reduction at each internal node or component of the tree. This yields, ultimately, a FOL closed formula for the whole utterance called its *meaning representation* (MR). An example is given in the parse tree below, where  $\tau$  returns the current value of the translation at each node.



For instance, by applying the translation procedure described above we get the following MRs from their corresponding English utterances:

- |  |                    |  |
|--|--------------------|--|
| 1. Some people are weak.                                   | $\rightsquigarrow$ | $\exists x(People(x) \wedge Weak(x)).$   |
| 2. Every husband has a wife.                               | $\rightsquigarrow$ | $\forall x(Hasband(x) \rightarrow \exists y(Wife(y) \wedge Has(x, y))).$   |
| 3. Every salesman sells some merchandise to some customer. | $\rightsquigarrow$ | $\forall x(Salesman(x) \rightarrow \exists y(Customer(y) \wedge \exists z(Merchandise(z) \wedge Sells(x, z, y))).$ |

Note that in 2. and 3. above, other translations might be possible due to NL ambiguity. However, these are discarded by the grammar studied by Third and Pratt that generates MRs following exclusively the surface order of components.

Schematically, the sentences above have the shape “Det N VP”, where the verb phrase (VP) is the constituent built out of a verb and its complements. We come back to this schema later to summarize the kind of constructs corresponding to *DL-Lite*<sub>R,Γ</sub>.

The fragments of English themselves are built step by step, by starting with copula, nouns, negation, and the universal and existential quantifiers and by extending coverage to larger portions of English – covering relative constructions, ditransitive verbs, anaphora, as summarized by the table below.

This analysis is important for our purposes because each NL construct has a meaning representation built out of some constant or some logical operation in FOL: The MRs of relatives (e.g., “who”) are built by conjunction ( $\wedge$ ); negations (e.g., “no”, “not”) introduce logical negation ( $\neg$ ); intransitive verbs (e.g., “runs”) and nouns (e.g., “man”) correspond to unary predicates; transitive verbs (e.g., “loves”) correspond to binary predicates, and ditransitive verbs (e.g., “sells to”) to ternary predicates; universal quantifiers (“every”, “all”, “everyone”) to  $\forall$ , and existential (“some”, “someone”) to  $\exists$ .

By building a family of fragments, Pratt and Third [8] have studied the impact on expressive power and computational complexity these constructs have (see Figure 1). As the reader can see, this process leads ultimately to an undecidable fragment of English. As a matter of fact, only the first two fragments, COP and COP+TV+DTV are tractable. Notice that as soon as we add rules dealing with the relative clause, we lose tractability. COP+Rel (i.e., COP with relative clauses), is already NP-Complete. COP+TV+DTV+Rel is NEXPTIME-Complete. This is because, as we said, relatives express conjunctions which, together with negation, generate logics (i.e., fragments of FOL) that contain the propositional calculus. But covering relatives to a certain extent is crucial: as we mentioned before, they occur quite frequently in NL utterances.

Fragment	Coverage	Sat. decision class
COP	Copula, common and proper nouns, negation, universal and existential quantifiers	P
COP+TV+DTV	COP + Transitive verbs (e.g. "reads") + Ditransitive verbs (e.g., "sells")	P
COP+Rel	COP + Relative pronoun (i.e., "who", "that", "which", etc.)	NP-Complete
COP+Rel+TV	COP + Transtive verbs + Relative pronoun	EXPTIME-Complete
COP+Rel+TV+DTV	COP+TV+DTV + Relative pronouns	NEXPTIME-Complete
COP+Rel+TV+RA	COP+Rel+TV + Restricted anaphora	NEXPTIME-Complete
COP+Rel+TV+GA	COP+Rel+TV + Generalized anaphora	undecidable

Fig. 1. Fragments of English studied by Pratt and Third [8].

Some means to cover them without yielding an exponential blowup should be found. As shown in [1], this is possible if we choose as MR logic  $DL-Lite_{R,\sqcap}$ , which allows relatives ( $\wedge$ ) to occur both in subject and in predicate position of sentences with an universal quantified subject, i.e., in the left and right concepts, respectively, of inclusion assertions.

COP and COP+TV+DTV generate, through this process of compositional translation described above, the following FOL fragments:

COP	COP+TV+DTV
$\pm A_1(c)$	$\psi$
$\exists x_1(A_1(x_1) \wedge \pm A_2(x_1))$	$Q_1 x_1(A_1(x_1) \sqcap \psi(x_1))$
$\forall x_1(A_1(x_1) \rightarrow \pm A_2(x_1))$	$Q_1 x_1(A_1(x_1) \sqcap \pm Q_2 x_2(A_2(x_2) \sqcap \psi(x_1, x_2)))$
	$Q_1 x_1(A_1(x_1) \sqcap \pm Q_2 x_2(A_2(x_2) \sqcap \pm Q_3 x_3(A_3(x_3) \sqcap \psi(x_1, x_2, x_3))))$

In the table above,  $Q_i \in \{\forall, \exists\}$ , for  $1 \leq i \leq 3$ ,  $\sqcap \in \{\wedge, \rightarrow\}$ ,  $c$  is an individual constant, the  $A_i$ 's, for  $1 \leq i \leq 3$ , are unary predicates, and  $\psi$  is an  $n$ -place *literal* (postive or negative) over the variables  $\{x_1, \dots, x_n\}$  containing possibly constants (thus  $\psi$  is a grounded literal). A quick glance at these logic fragments tells us that they can express IS-A constraints, as well as ABoxes almost directly. Moreover, we can express unary and binary (and even ternary) predicates, together with quantification.

So the questions now are: exactly to what extent these two tractable fragments or, equivalently, the FOL fragments thereof generated can express  $DL-Lite_{R,\sqcap}$ ? How much of  $DL-Lite_{R,\sqcap}$  can not be expressed by these fragments? Answering these questions will shed light on the issue of which NL constructs can ultimately express  $DL-Lite_{R,\sqcap}$  ontologies in a subset of English.

## 4 Comparing Expressive Power

In this section, we compare the expressive power of  $DL-Lite_{R,\sqcap}$  with that of the two tractable fragments of English COP and COP+TV+DTV. We show that, under certain conditions, COP is contained in  $DL-Lite_{R,\sqcap}$ , as it should be expected, but that COP+TV+DTV only overlaps with  $DL-Lite_{R,\sqcap}$ . This is interesting, since, as shown in [1], Lite English, the controlled language that compositionally translates into  $DL-Lite_{R,\sqcap}$ , covers relative pronouns (mirrored by the qualified existential  $\exists R.C$ ) without yielding an exponential blowup, as is the case with Pratt's fragments.

We begin by recalling some basic notions of FOL (without function symbols) model theory. An *interpretation structure* over a FOL signature (without function symbols)  $\mathcal{L}$  is a tuple  $\mathfrak{M} = \langle M; \{R_i^{\mathfrak{M}}\}_{i \in I}; \{c_j^{\mathfrak{M}}\}_{j \in J} \rangle$  where the  $R_i^{\mathfrak{M}}$  are  $n$ -ary relations over  $M$  and the  $c_j^{\mathfrak{M}}$  distinguished elements of  $M$ , for  $i \in I, j \in J$ . A structure  $\mathfrak{M}'$  is said to be an *extension* of  $\mathfrak{M}$  whenever the relations of  $\mathfrak{M}$  are contained in those of  $\mathfrak{M}'$  and they coincide on the distinguished elements. A structure is said to be a *model* of a sentence or formula  $\phi$  whenever  $\mathfrak{M} \models \phi$ . The sentence  $\phi$  *characterizes* the classes of its models (i.e., the class  $\{\mathfrak{M} \mid \mathfrak{M} \models \phi\}$ ). These classes are called *properties*. The *expressive power* of a fragment of FOL is then formally given by the model theoretic properties its sentences can characterize. Finally, a FOL fragment  $\Lambda'$  is said to be *as expressive as* a fragment  $\Lambda$  when, and only when,  $\Lambda'$  can express all properties of  $\Lambda$  [9]. The idea of the proofs is to individuate properties expressible in one logic and not in the other – that is, classes of structures that  $DL-Lite_{R,\sqcap}$  expresses but that COP+TV+DTV and COP may or may not express.

**Theorem 1.**  *$DL-Lite_{R,\sqcap}$  is as expressive as COP, assuming the unique name assumption does not hold.*

*Proof.* Some COP sentences cannot be *a priori* expressed in  $DL-Lite_{R,\sqcap}$ . In particular, as we have seen,  $DL-Lite_{R,\sqcap}$  as it is, cannot express negative facts: ABox assertions (i.e., ground atoms) cannot be negative following the standard definition of  $DL-Lite_{R,\sqcap}$ . However, we can easily express a negative fact  $\neg A(c)$ , by extending our signature with a new concept name  $A'$ , and introducing the disjointness assertion  $A' \sqsubseteq \neg A$  and the membership assertion  $A'(c)$ . To deal with COP formulas of the form  $\exists x(P(x) \wedge Q(x))$  we proceed as follows: (i) we skolemise and extend our signature by adding a new constant  $c$  (expanding models  $\mathfrak{M}$  to their skolem expansion  $(\mathfrak{M}, c^{\mathfrak{M}})$ ) and (ii) we drop the unique name assumption (UNA) regarding constants when it comes to these new constants produced by skolemisation. We can then express these statements as ABox assertions  $P(c)$  and  $Q(c)$ .  $\square$

**Theorem 2.**  *$DL-Lite_{R,\sqcap}$  is not as expressive as COP+TV+DTV.*

*Proof.* To prove this result, we exhibit a closure property of  $DL-Lite_{R,\sqcap}$  that is not preserved by COP+TV (and *a fortiori* by COP+TV+DTV). The formulas in  $DL-Lite_{R,\sqcap}$  are all FOL  $\forall\exists$  formulas, *modulo* the standard translation. A  $\forall\exists$  formula or sentence is a formula  $\phi := \forall x_1 \cdots \forall x_n \exists x_1 \cdots \exists x_m \psi$ , for  $n, m \geq 0$ , where  $\psi$  is quantifier-free. Now,  $\forall\exists$  formulas are closed under the *union of chains* property [5], defined as follows. We say that a formula  $\phi$  is *closed under union of chains* iff for every partial order  $\langle T, \prec_T \rangle$ , for every model  $\mathfrak{M}$  of  $\phi$ , and every family  $\{\mathfrak{M}_t\}_{t \in T}$  of extensions of  $\mathfrak{M}$ , s.t.  $i \prec_T j$  implies that  $\mathfrak{M}_j$  is an extension of  $\mathfrak{M}_i$ , for  $i, j \in T$ , then the structure  $\mathfrak{M}_\omega$ , called the *union structure* and defined below, is also a model of  $\phi$ :

1.  $M_\omega = \bigcup_{t \in T} M_t$
2. Every  $R_i^{\mathfrak{M}_\omega}$  is the union of all the relations of the same arity and position among the  $\mathfrak{M}_t$ 's, for  $t \in T, i \in I$
3. Every  $c_j^{\mathfrak{M}_\omega}$  is a distinguished element among the the  $\mathfrak{M}_t$ 's, for  $t \in T, j \in J$ .

Therefore, every set of  $DL-Lite_{R,\sqcap}$  sentences (assertions) will be closed under union of chains. Now, suppose, towards a contradiction that every property that is expressible in

COP+TV+DTV is expressible also in  $DL-Lite_{R,\sqcap}$ , in particular:  $\exists x(P(x) \wedge \forall y(Q(y) \rightarrow R(x,y)))$ . That is, after prenexing:  $\exists x \forall y (P(x) \wedge (Q(y) \rightarrow R(x,y)))$ . This sentence should be closed under union of chains, following the hypothesis. But this does not hold. To show this define a model  $\mathfrak{M}$  of this sentence as follows:  $M = \mathbb{N}$ ;  $P^{\mathfrak{M}} = Q^{\mathfrak{M}} = M$ ;  $R^{\mathfrak{M}} = \leq_{\mathbb{N}}$  (i.e., the usual loose order over positive integers).

Define next a sequence  $\{\mathfrak{M}_i\}_{i \in \mathbb{N}}$  of extensions of  $\mathfrak{M}$  as follows:

- $\mathfrak{M}_0$ :  $M_0 = M \cup \{e_0\}$ ;  $P^{\mathfrak{M}_0} = Q^{\mathfrak{M}_0} = M_0$ ;  $R^{\mathfrak{M}_0} = R^{\mathfrak{M}} \cup \{(e_0, 0)\}$ .
- $\mathfrak{M}_{i+1}$ :  $M_{i+1} = M_i \cup \{e_{i+1}\}$ ;  $P^{\mathfrak{M}_{i+1}} = Q^{\mathfrak{M}_{i+1}} = M_{i+1}$ ;  $R^{\mathfrak{M}_{i+1}} = R^{\mathfrak{M}_i} \cup \{(e_{i+1}, e_i)\}$ .

Now,  $\{\mathfrak{M}_i\}_{i \in \mathbb{N}}$  constitutes a chain, since (i) a sequence is a family, (ii)  $\langle \mathbb{N}, \leq_{\mathbb{N}} \rangle$  is a partial order and (iii) whenever  $i \leq_{\mathbb{N}} j$ ,  $\mathfrak{M}_j$  extends  $\mathfrak{M}_i$ . Finally, consider the union structure  $\mathfrak{M}_\omega$  for this chain.  $\mathfrak{M}_\omega$  is not a model of  $\exists x \forall y (P(x) \wedge (Q(y) \rightarrow R(x,y)))$ , since the relation  $R^{\mathfrak{M}_\omega}$  of  $\mathfrak{M}_\omega$  has no least element.  $\square$

**Theorem 3.** *COP+TV+DTV is not as expressive as  $DL-Lite_{R,\sqcap}$ .*

*Proof.* A  $DL-Lite_{R,\sqcap}$  inclusion assertion of the form  $\exists R \sqsubseteq A$  corresponds to the FOL sentence  $\forall x \exists y (R(x,y) \rightarrow A(x))$ . Skolemizing and clausifying this sentence yields:  $\neg R(x, f(x)) \vee A(x)$ , i.e., a clause containing both a positive unary literal and a binary negative literal containing function symbols. But in [8] it is proven that this particular kind of clauses lies beyond COP+TV+DTV, whence the result.  $\square$

**Theorem 4.** *COP and COP+TV+DTV overlap in expressive power with  $DL-Lite_{R,\sqcap}$ .*

*Proof.* Consider this following typical meaning representation formula for COP:  $\forall x (P(x) \rightarrow Q(x))$ . The models of these sentences are the FOL interpretation structures  $\mathfrak{M} = \langle M; P^{\mathfrak{M}}, Q^{\mathfrak{M}} \rangle$ , where  $P^{\mathfrak{M}} \subseteq Q^{\mathfrak{M}}$ . But this property can be easily expressed in the  $DL-Lite_{R,\sqcap}$  with inclusion assertions.  $\square$

We finish by remarking that it can also be proved that COP is not as expressive as  $DL-Lite_{R,\sqcap}$  either, since it cannot express binary relations. Moreover, we can show, in a way analogous to Theorem 2, that COP+TV is not as expressive as  $DL-Lite_{R,\sqcap}$ , by exhibiting a closure property of COP+TV, namely COP+TV-*simulation* [10] that is not verified by  $DL-Lite_{R,\sqcap}$ .

An understanding of how the different expressivity of the compared logics is reflected on the corresponding natural language fragments can be reached by considering the general schema “Det N VP” mentioned previously. In these fragments “Det” can be either “every” or “some” or “no”, and all of these determiners can build the direct and/or indirect objects of the transitive and ditransitive verbs, i.e., the complements in the “VP”; the verb of the “VP” can be negated. Logic conjunction is introduced only by the meaning representation of “some”. The sentences whose meaning representation belong to  $DL-Lite_{R,\sqcap}$ , instead, do not have ditransitive verbs (ternary relations) and in the “Det” position only the determiners “every” and “no” can occur. Notice, that the “N” and “VP” correspond respectively to the  $Cl$  and  $Cr$  concepts of a  $DL-Lite_{R,\sqcap}$  inclusion assertion. As in the latter, the two parts can be complex: the “N” constituent can be a complex structure built out of a relative pronoun (e.g., “student who left”, “student who

knows something”); transitive verbs can occur only with an unqualified existential as object; the verb of the relative clause (e.g., “left” and “knows”, resp.) cannot be negated (negation does not occur in  $Cl$ ), and the relative clause cannot be iterated, i.e., it cannot be used to modify the object of a transitive verb (only unqualified existential can occur in  $Cl$ ). Similarly, the “VP” can be a complex structure: since it corresponds to the  $Cr$  concept, copula, intransitive verbs, and transitive verbs, with unqualified existential as object, can be negated. Whereas transitive verbs with qualified existential as object cannot be negated, e.g., “every student does not know something that is interesting” (notice how the relative clause modifies an existential building a qualified existential).

Finally, neither in COP and COP+TV+DTV nor in our fragment reflexive pronouns (e.g., “itself”) and possessive (e.g., “their”) are allowed: they would correspond to the introduction of role-value-maps, which is a notoriously problematic construct that may lead to undecidability.

## 5 Conclusions

We have compared the expressive power of  $DL-Lite_{R,\square}$  with that of Pratt’s and Third’s tractable fragments of English [8, 10]. Using model theoretic arguments, we have shown that the compared logics are incomparable to each other, even though a reasonable deal of the semantic structures captured by the two tractable fragments of English is shared by  $DL-Lite_{R,\square}$ . We remark that a controlled natural language covering constructs such as intransitive and transitive verbs, copula, common nouns, adjectives, restricted occurrences of universal and existential quantification as well as of negation and relative pronouns can be “reverse engineered” from  $DL-Lite_{R,\square}$ , as shown in [1].

**Acknowledgements.** Research partially supported by the FET project TONES (Thinking ONtologiES), funded by the EU under contract FP6-7603, and by the PRIN 2006 project NGS (New Generation Search), funded by MIUR.

## References

1. R. Bernardi, D. Calvanese, and C. Thorne. Lite natural language. In *Proc. of the 7th Int. Workshop on Computational Semantics (IWCS-7)*, 2007.
2. A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *Proc. of ACM SIGMOD*, pages 59–67, 1989.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, pages 602–607, 2005.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
5. R. Cori and D. Lascar. *Logique mathématique (2 vols)*. Dunod, 2003.
6. L. T. F. Gamut. *Logic, Language and Meaning (2 vols.)*. University of Chicago Press, 1991.
7. R. Montague. The proper treatment of quantification in ordinary english. In *Approaches to Natural Language: Proc. of the 1970 Stanford Workshop on Grammar and Semantics*, 1973.
8. I. Pratt and A. Third. More fragments of language. *Notre Dame Journal of Formal Logic*, 2005.
9. L. Straßburger. What is a logic, and what is a proof? In J.-Y. Beziau, editor, *Logica Universalis*, pages 135–145. Birkhauser, 2005.
10. A. Third. *Logical Analysis of Fragments of Natural Language*. PhD thesis, Faculty of Engineering and Physical Sciences, University of Manchester, 2006.

## Consequence Finding in $\mathcal{ALC}$

Meghyn Bienvenu

IRIT, Université Paul Sabatier  
Toulouse, France  
bienvenu@irit.fr

### 1 Introduction

Research on reasoning in description logics has traditionally focused on the standard inference tasks of (un)satisfiability, subsumption, and instance checking. In recent years, however, there has been a growing interest in so-called non-standard inference services (cf. [1]) which prove useful in the creation, evolution, and utilisation of description logic knowledge bases (KBs). In this paper, we propose *consequence finding* as a new non-standard reasoning task for description logics.

As its name suggests, consequence finding is concerned with the generation of a subset of the implicit consequences of a KB. This topic has been extensively studied in propositional logic (cf. [2]) where it has been shown to be relevant to a number of areas of AI, among them knowledge compilation, abduction, and non-monotonic reasoning. In the context of description logics, we view consequence finding as a tool to enable knowledge engineers and end users alike to better understand and access the contents of a description logic KB. We consider two possible applications:

**Ontology Management** The creation and maintenance of description logic KBs is a difficult and time-consuming task. Consequence finding might prove a useful aid in ontology design and evolution by allowing the knowledge engineer to have a better idea of both what information is contained in the KB and how additions may affect it. For instance, it would allow the knowledge engineer to check the relationships holding between a set of atomic concepts, or to evaluate the impact of adding a piece of new information to the KB.

**Vague Queries** An end user may not always have a specific query in mind but instead a general idea of the information that interests her. Consequence finding could help render DL KBs more accessible to users by allowing them to pose vague queries about the contents of the KB. Possible queries might be “Give me all the facts concerning penguins” or “Tell me what is known about Bob’s students”.

Consequence finding in DLs could take one of many forms depending on the nature of the input (concept, ABox, TBox, KB) and the type of consequences desired (concepts, assertions, axioms). In this paper, we investigate consequence finding for concept expressions in the standard description logic  $\mathcal{ALC}$ . This seemed a suitable starting point for our investigation as concept expressions are more basic than assertions and axioms, and  $\mathcal{ALC}$  is a well-known and reasonably expressive description logic.

Our paper is organized as follows. In Section 2, we generalize the propositional notion of prime implicates to  $\mathcal{ALC}$  concepts in order to provide a formal definition

of the notion of a relevant consequence. More refined variants of prime implicates are then introduced to allow for more directed consequence finding, and the properties of these different forms of prime implicates are studied. In Section 3, we give sound and complete algorithms for computing prime implicates of concepts. In Section 4, we provide some first results concerning the complexity of prime implicate recognition. We conclude the paper with a discussion of future work.

## 2 $\mathcal{ALC}$ Prime Implicates

One of the first questions that presents itself when we talk about consequence finding is which consequences to generate? We obviously cannot generate *all* of the consequences, because even the simplest propositional formula has infinitely many consequences. Even if we restrict ourselves to one consequence per equivalence class, we still produce a lot of clearly irrelevant or redundant consequences. Since one of the aims of consequence finding is to make the contents of the KB more accessible to users, we clearly need a way of focusing in on the relevant subset of consequences.

In propositional logic, the solution lies in considering only the strongest clausal consequences of a formula, which are known as its *prime implicates*. By focusing on clauses, we avoid redundancies, and by only considering the logically strongest consequences, we eliminate weaker, irrelevant consequences. As every formula can be rewritten as a conjunction of clauses, and every clausal consequence of a formula is entailed by some prime implicate of the formula, prime implicates provide a complete picture of a formula’s consequences.

In order to generalize the notion of prime implicates to  $\mathcal{ALC}$  concepts, we need to come up with a suitable definition of  $\mathcal{ALC}$  clausal concepts. In [3], a number of different potential definitions are compared, and two are singled out as being most suitable for the purposes of consequence finding.

The first definition is inspired by the notion of modal atom proposed in [4]. It defines literal concepts as the concepts in NNF that cannot be decomposed propositionally:

**Definition 1.** *Literal concepts, clausal concepts, and cubal concepts are defined as follows (where  $A$  is an atomic concept and  $R$  an atomic role):*

$$\begin{aligned} L &::= \top \mid \perp \mid A \mid \neg A \mid \forall R.F \mid \exists R.F \\ Cl &::= L \mid Cl \sqcup Cl \\ Cb &::= L \mid Cb \sqcap Cb \\ F &::= \top \mid \perp \mid A \mid \neg A \mid F \sqcap F \mid F \sqcup F \mid \forall R.F \mid \exists R.F \end{aligned}$$

The second definition defines literal concepts as those concepts in NNF that cannot be decomposed *modally*, resulting in a more restrictive clausal form.

**Definition 2.** *Literal, clausal, and cubal concepts are defined as follows:*

$$\begin{aligned} L &::= \top \mid \perp \mid A \mid \neg A \mid \forall R.Cl \mid \exists R.Cb \\ Cl &::= L \mid Cl \sqcup Cl \\ Cb &::= L \mid Cb \sqcap Cb \end{aligned}$$

Both definitions yield the standard notion of literals, clauses, and cubes when restricted to the propositional fragment of  $\mathcal{ALC}$ . They can also be shown to satisfy a number of properties of the propositional definition:

**Proposition 1.** *For both Definition 1 and Definition 2, we have:*

1. *Clausal and cubal concepts are simply unions and intersections of literal concepts.*
2. *The negation of a literal concept is equivalent to a literal concept. Negations of clausal (resp. cubal) concepts are equivalent to cubal (resp. clausal) concepts.*
3. *Every concept  $C$  is equivalent to a finite intersection of clausal concepts  $D = D_1 \sqcap \dots \sqcap D_n$  such that (a)  $D$  has the same depth as  $C$ , and (b) the size of  $D$  is at most singly exponential in the size of  $C$ . Likewise every concept is equivalent to a finite union of cubal concepts.*

Definition 1 has the advantage of yielding a more compact representation while Definition 2 provides a more fine-grained decomposition of concepts. To simplify the presentation, we will henceforth consider only the second definition, but all of our results hold equally well with respect to the first definition.

Now that we have selected a definition of clausal concepts, we can define prime implicates in exactly the same manner as in propositional logic:<sup>1</sup>

**Definition 3.** *A clausal concept  $Cl$  is an implicate of a concept  $C$  if and only if  $\models C \sqsubseteq Cl$ . A clausal concept  $Cl$  is a prime implicate of  $C$  if and only if:*

1.  *$Cl$  is an implicate of  $C$*
2. *If  $Cl'$  is an implicate of  $C$  such that  $\models Cl' \sqsubseteq Cl$ , then  $\models Cl \sqsubseteq Cl'$*

This definition gives the standard notion of prime implicates, in which all of the clausal consequences of a concept are considered, but for many applications, only some of the consequences are of interest. This motivates the introduction of more refined notions of prime implicates in which additional restrictions are placed on implicates. In this paper, we consider the following three refined versions of prime implicates:

**$C$ -prime implicates:** the  $C$ -prime implicates of a concept  $D$  are defined as the most specific clausal concepts which subsume  $C \sqcap D$  but do not subsume  $C$

**only- $\mathcal{L}$ -prime implicates:** the only- $\mathcal{L}$ -prime implicates of a concept  $D$  are the most specific clausal concepts which both subsume  $D$  and contain only those atomic concepts and roles in  $\mathcal{L}$

**about- $\mathcal{L}$ -prime implicates:** the about- $\mathcal{L}$ -prime implicates of a concept  $D$  are the most specific clausal concepts which subsume  $D$  and which contain non-trivially all symbols in  $\mathcal{L}$  (i.e. such that all equivalent concepts contain all symbols in  $\mathcal{L}$ )

We have chosen to study these particular variants because they seem interesting from an applications point of view. Once appropriately extended to TBox axioms,  $C$ -prime implicates could be used by the knowledge engineer to see how a new axiom will affect the ontology  $\mathcal{O}$  under construction (“What are all of the  $\mathcal{O}$ -prime implicates of *axiom*?”), whereas only- $\mathcal{L}$ -prime implicates could allow him to explore the relationships between a set  $S$  of atomic concepts (“What are all of the  $S$ -prime implicates of  $\mathcal{O}$ ?”). Finally, about- $\mathcal{L}$ -prime implicates can be used to generate all the consequences on a particular topic of interest and are thus very useful for vague querying. It is this type of prime implicate which would enable the user to find all the information concerning penguins.

<sup>1</sup> The dual notion of prime implicants can also be straightforwardly defined, but here we consider only prime implicates as they are the most relevant to purposes of the current paper.

*Example 1.* Consider the following concept expression  $Q$ :

$$A \sqcap (B \sqcup C) \sqcap \exists R. \top \sqcap \forall R. (B \sqcap (A \sqcup C)) \sqcap \forall R. (B \sqcup D)$$

The prime implicates of  $Q$  are  $A$ ,  $B \sqcup C$ ,  $\exists R. (B \sqcap A) \sqcup \exists R. (B \sqcap C)$ ,  $\forall R. B$ , and  $\forall R. (A \sqcup C)$ . There is just one only- $\{A\}$ -prime implicates of  $Q$ , the atomic concept  $A$ . There are three about- $\{A\}$ -prime implicates of  $Q$ :  $A$ ,  $\exists R. (B \sqcap A) \sqcup \exists R. (B \sqcap C)$ , and  $\forall R. (A \sqcup C)$ . The  $Q$ -prime implicates of  $\forall R. \neg C$  are  $\forall R. \neg C$ ,  $\forall R. A$ , and  $\exists R. A \sqcap B$ .

It is not hard to see that standard prime implicates can be recovered as special cases of  $C$ -, only- $\mathcal{L}$ -, and about- $\mathcal{L}$ -prime implicates. The following proposition further clarifies the relationship between standard and refined prime implicates:

**Proposition 2.**

1. Every  $C$ -prime implicate of a concept  $D$  is a prime implicate of  $C \sqcap D$ .
2. Every about- $\mathcal{L}$ -prime implicate of a concept  $D$  is a prime implicate of  $D$ .
3. An only- $\mathcal{L}$ -prime implicate of a concept  $D$  may not be a prime implicate of  $D$ .

To see why (3) holds, consider the concept  $\exists R. A$  which is an only- $\{A, R\}$ -prime implicate of  $\exists R. (A \sqcap (\forall R. B))$  but not a standard prime implicate.

We now consider some important properties of our notions of prime implicates:

**Proposition 3.** *The set of prime implicates satisfy the following properties:*

**Finiteness** *The number of standard,  $C$ -, only- $\mathcal{L}$ -, and about- $\mathcal{L}$ -prime implicates of a concept is finite modulo logical equivalence.*

**Covering** *Every standard,  $C$ -, only- $\mathcal{L}$ -, or about- $\mathcal{L}$ -implicate of a concept subsumes respectively some standard,  $C$ -, only- $\mathcal{L}$ -, or about- $\mathcal{L}$ -prime implicate of the concept.*

**Distribution** *The prime implicates (respectively  $C$ -, only- $\mathcal{L}$ -prime implicates) of a concept  $C_1 \sqcup \dots \sqcup C_n$  are equivalent to the logically strongest unions of the prime implicates (respectively  $C$ -, only- $\mathcal{L}$ -prime implicates) of the  $C_i$ .*

**Finiteness** ensures that the prime implicates of a concept can be finitely represented, which is of course essential from a computational perspective. **Covering**, as its name suggests, implies that every implicate of a concept is “covered” by one of its prime implicates. For the standard definition of prime implicates, **Covering** implies that the set of prime implicates of a concept is equivalent to the concept itself, thus guaranteeing that no information is lost in replacing a formula by its prime implicates. For  $\mathcal{L}$ -concepts, **Covering** implies that the set of  $\mathcal{L}$ -prime implicates is equivalent to the uniform interpolant<sup>2</sup> of the concept with respect to  $\mathcal{L}$ . **Distribution** is at the heart of several prime implicate generation algorithms in propositional logic, and we will exploit this property in our own algorithms presented in the next section. Notice however that **Distribution** does not hold for about- $\mathcal{L}$ -prime implicates since some disjuncts of about- $\mathcal{L}$ -concepts may not contain all (or any) of the symbols in  $\mathcal{L}$ .

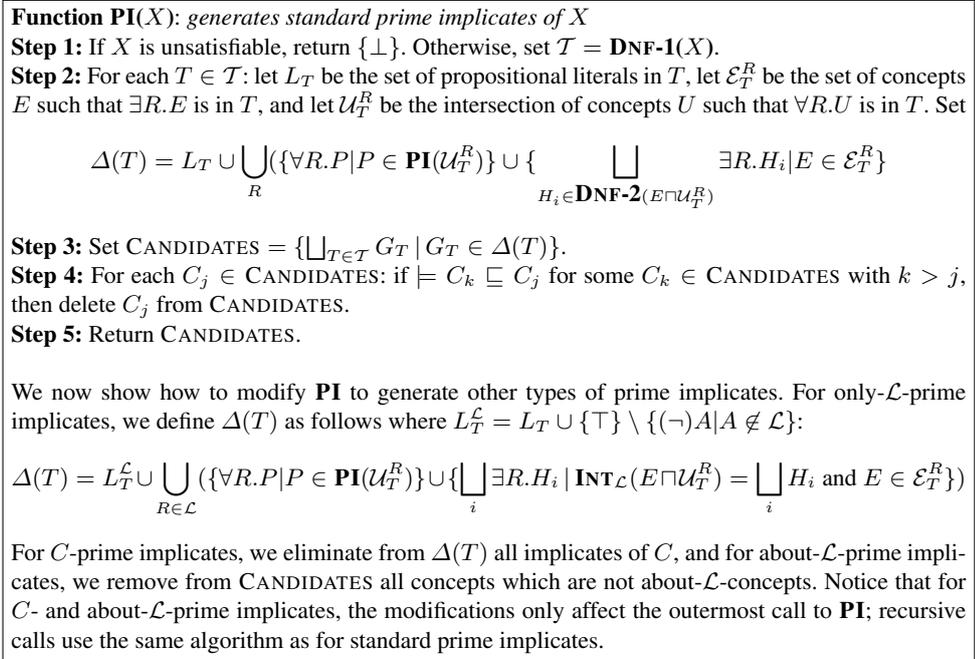
<sup>2</sup> We recall the *uniform interpolant* of a concept  $D$  with respect to a signature  $\mathcal{L}$  is the most specific concept built from  $\mathcal{L}$  which subsumes  $D$ .

### 3 Prime Implicate Generation

Figure 1 presents algorithms for computing the prime implicates of a concept for the different forms of prime implicates that we have defined. Our algorithms make use of the following auxilliary functions:

- **DNF-1** and **DNF-2** which return a set of satisfiable cubal concepts (w.r.t. Definitions 1 and 2 respectively) whose union is equivalent to the input concept
- **INT<sub>ℒ</sub>** which returns a finite union of cubal concepts (w.r.t. Definition 2) which is equivalent to the uniform interpolant of the input concept w.r.t.  $\mathcal{L}$

We remark that we can implement these functions so that the size of the output of these functions is at most singly exponential in the size of the input concept.



**Fig. 1.** Algorithms for prime implicate generation.

Our algorithms all work in a similar manner. In Step 1, we check whether the input concept  $X$  is unsatisfiable, outputting  $\perp$  if this is the case. For satisfiable  $X$ , we set  $\mathcal{T}$  equal to a set of satisfiable concepts whose union is equivalent to  $X$ . We know from the distribution property (Proposition 3) that every prime implicate of  $X$  is equivalent to some union of prime implicates of the concepts in  $\mathcal{T}$ . In Step 2, we construct a set  $\Delta(T)$  of clausal concepts for each  $T \in \mathcal{T}$  in such a way that every prime implicate of  $T$  is

equivalent to some element in  $\Delta(T)^3$ . This means that in Step 3, we are guaranteed that every prime implicate of the input concept is equivalent to some candidate prime implicate in CANDIDATES. During the comparison phase in Step 4, non-prime candidates are eliminated, and exactly one prime implicate per equivalence class is retained.

The algorithms differ in their definition of  $\Delta(T)$  in Step (2). For standard prime implicates, we set  $\Delta(T)$  equal to the propositional literals in  $T$  ( $L_T$ ) plus the strongest  $\forall$ -literal concepts implied by  $T$  ( $\bigcup_R(\{\forall R.P \mid P \in \mathbf{PI}(\mathcal{U}_T^R)\})$ ) plus the strongest  $\exists$  clausal concepts implied by  $T$  ( $\{\bigwedge_{H_i \in \mathbf{DNF-2}(E \cap \mathcal{U}_T^R)} \exists R.H_i \mid E \in \mathcal{E}_T^R\}$ , i.e. the concepts  $\exists R.(E \cap \mathcal{U}_T^R)$  put into clausal form). It can be shown that every standard prime implicate of  $T$  must be equivalent to one of the elements in  $\Delta(T)$  (note however that some elements in  $\Delta(T)$  may not be prime implicates). For only- $\mathcal{L}$ -prime implicates, we modify  $\Delta(T)$  in order to ensure that the elements of  $\Delta(T)$  contain only the symbols in  $\mathcal{L}$  (and that they include all only- $\mathcal{L}$ -prime implicates of  $T$ ). For  $C$ -prime implicates, we remove from  $\Delta(T)$  all implicates of  $C$ . Finally, for about- $\mathcal{L}$ -prime implicates, we use the same  $\Delta(T)$  as for standard prime implicates, but we eliminate from CANDIDATES all clausal concepts which are not about- $\mathcal{L}$ -concepts.

**Proposition 4.** *The prime implicate generation algorithms presented in Figure 1 are sound, complete, and always terminate.*

Our algorithms correspond to the simplest possible implementation of the distribution property, and it is well-known that naive implementations of the distribution property are computationally infeasible even for propositional logic. More efficient versions of our algorithms can be obtained using techniques developed for propositional logic, cf. [2]. For instance, instead of generating all of the candidate concepts and *then* comparing them, we can build them incrementally, comparing them as we go.

By performing induction on the depth of the input concept, it is possible to place an upper bound on the size of the prime implicates generated by our algorithms:

**Proposition 5.** *The size of the smallest representation of a standard,  $C$ -, only- $\mathcal{L}$ -, and about- $\mathcal{L}$ -prime implicate of a concept is at most singly exponential in the size of the concept.*

The following proposition shows that this bound is optimal.

**Proposition 6.** *The size of the smallest representation of a standard,  $C$ -, only- $\mathcal{L}$ -, or about- $\mathcal{L}$ -prime implicate of a concept can be exponential in the size of the concept.*

*Proof.* Consider the concept  $(\prod_{i=1}^n (\forall R.(A_{i1} \sqcup A_{i2})) \sqcap \exists R.\top$  and its prime implicate  $\bigwedge_{i_k \in \{1,2\}} \exists R.(A_{1i_1} \sqcap \dots \sqcap A_{ni_n})$ .

## 4 Prime Implicate Recognition

Prime implicate recognition consists in deciding whether a given concept is a prime implicate of another. The purpose of this section is to study the complexity of this decision problem for the different notions of prime implicates that we have introduced.

<sup>3</sup> A worked-out example of Step 2 for standard prime implicates can be found in [3].

It is not hard to see that this decision problem must be at least as difficult as unsatisfiability: a concept is unsatisfiable just in the case that it has  $\perp$  as a prime implicate (irrespective of the notion of prime implicate considered).

**Proposition 7.** *Recognition of standard,  $C$ -, only- $\mathcal{L}$ -, and about- $\mathcal{L}$ -prime implicates are all PSPACE-hard problems.*

In order to obtain an upper bound, we exploit Proposition 5 which tells us that there is some polynomial  $p$  such that for every concept  $C$  the size of its prime implicates is bounded by  $2^{p(|C|)}$ . This leads to a simple non-deterministic procedure for determining if a clausal concept  $Cl$  is a prime implicate of a concept  $C$ . We simply guess a clausal concept  $W$  of size at most  $2^{p(|C|)}$  and check whether  $W$  is an implicate of  $C$  which is subsumed by  $Cl$  and does not subsume  $Cl$ . If this is the case, then  $Cl$  is not a prime implicate (we have found a more specific implicate of  $C$ ), otherwise, there exists no stronger implicate, so  $Cl$  is indeed a prime implicate.

**Proposition 8.** *Recognition of standard,  $C$ -, only- $\mathcal{L}$ -, and about- $\mathcal{L}$ -prime implicates are all in EXSPACE<sup>4</sup>.*

The following proposition improves on the above complexity bounds.

**Proposition 9.** *We have the following:*

1. *Standard prime implicate recognition is in EXPTIME.*
2.  *$C$ -prime implicate recognition is in EXPTIME<sup>5</sup>.*
3. *about- $\mathcal{L}$ -prime implicates recognition is in NEXPTIME.*
4. *only- $\mathcal{L}$ -prime implicate recognition is CONEXPTIME-hard.*

*Proof.* To demonstrate (1), we have constructed an algorithm for deciding standard prime implicate recognition in single-exponential time. Our algorithm first checks that the clausal concept is indeed an implicate and then verifies that each of the component literals is as specific as possible. Refer to [3] for more details.

(2) follows directly from (1) since  $C$ -prime implicates are just standard prime implicates which are not implied by  $C$  (by Proposition 2).

(3): We can check whether a concept  $Cl$  is an about- $\mathcal{L}$ -prime implicate of a concept  $D$  in three steps: first, we check that the  $Cl$  contains all symbols in  $\mathcal{L}$  (linear time), next, we verify that  $Cl$  is indeed a standard prime implicate of  $D$  (exponential time by (1)), and finally, we ensure that each symbol in  $\mathcal{L}$  appears non-trivially in  $Cl$ . For this last step, it suffices to show that for each symbol  $s \in \mathcal{L}$  the uniform interpolant  $UI_s$  of  $Cl$  over  $Sig(Cl) \setminus \{s\}$  is not subsumed by  $Cl$ . This can be accomplished in non-deterministic exponential time by guessing and verifying an open branch of the tableaux of  $UI_s \sqcap \neg Cl$  for each  $s$ .

For (4), we reduce the conservative extension decision problem for  $\mathcal{K}$  (proven CONEXPTIME-complete in [5]) to the only- $\mathcal{L}$ -prime implicate recognition problem. We recall that a formula  $\phi_1 \wedge \phi_2$  is a conservative extension of  $\phi_1$  if for every formula  $\psi$

<sup>4</sup> While the proof given here is non-constructive, we do have constructive exponential-space algorithms which we were unable to include for lack of space.

<sup>5</sup> Here we assume that the concept  $C$  is considered as part of the input

with  $\text{var}(\psi) \subseteq \text{var}(\phi_1)$  we have  $\phi_1 \wedge \phi_2 \models \psi$  implies  $\phi_1 \models \psi$ . The reduction is straightforward:  $\phi_1 \wedge \phi_2$  is a cubal formula  $\phi_1$  if and only if  $\exists R.f(\phi_1)$  is a  $\text{var}(\phi_1) \cup \{R\}$ -prime implicate of  $\exists R.(f(\phi_1 \wedge \phi_2))$ , where  $f$  is the standard mapping between  $\mathcal{K}$ -formulae and  $\mathcal{ALC}$  concepts. This is sufficient to show CONEXPTIME-hardness since the conservative extension problem remains CONEXPTIME-hard even when  $\phi_1$  is restricted to be a cubal formula.

We leave the determination of the exact complexity classes of the different prime implicate recognition tasks as an interesting open problem.

## 5 Future Work

While there are several possible continuations to this work, the question that interests us most is the appropriate extension of consequence finding to ABox assertions and TBox axioms. This is a non-trivial task as while the definition of clausal forms of assertions and axioms, and thus of assertion and axiom prime implicates, is rather straightforward, we lose some of the nice properties enjoyed by concept prime implicates. In particular, prime implicate axioms and assertions do not in general satisfy the covering property since there may be infinite sequences of increasingly more specific assertions or axioms. This is a familiar problem as these infinite sequences are also responsible for the inexistence of most specific concepts of ABox individuals in many common DLs (cf. [6]) and the lack of uniform interpolation for  $\mathcal{ALC}$  TBoxes [7]. There appear to be two possible solutions to this problem. The simplest solution consists in bounding the depth of the assertions/axioms to be generated. A second more elegant possibility is to enrich the language by fixpoint constructs (cf. [8]) so that these infinite sequences of assertions/axioms might be finitely represented.

## References

1. Baader, F., Küsters, R.: Non-standard inferences in description logics: The story so far. In Gabbay, D.M., Goncharov, S.S., Zakharyashev, M., eds.: *Mathematical Problems from Applied Logic I. Logics for the XXIst Century*. Vol. 4 IMS. Springer-Verlag (2006) 1–75
2. Marquis, P.: Consequence Finding Algorithms. In: *Handbook on Defeasible Reasoning and Uncertainty Management Systems*. Volume 5. Kluwer (2000) 41–145
3. Bienvenu, M.: Prime implicates and prime implicants in modal logic. In: *Proc. AAI-07*. (2007) To appear.
4. Giunchiglia, F., Sebastiani, R.: A sat-based decision procedure for ALC. In: *Proc. KR’96*. (1996) 304–314
5. Ghilardi, S., Lutz, C., Wolter, F., Zakharyashev, M.: Conservative extensions in modal logic. In: *Proc. AiML’06*. (2006)
6. Küsters, R., Molitor, R.: Approximating most specific concepts in logics with existential restrictions. *AI Communications* (15) (2002) 47–59
7. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: *Proc. KR’06*. (2006)
8. de Giacomo, G., Lenzerini, M.: A uniform framework for concept descriptions in description logics. *Journal of Artificial Intelligence Research* 6 (1997) 87–110

## On Importing Knowledge from DL Ontologies: some intuitions and problems

Alex Borgida  
Dept. of Computer Science,  
Rutgers University, New Brunswick, NJ, USA

**Abstract.** This paper argues for the benefits of distinguishing the notions of “ontology module” and “importing terms from an ontology”, by sampling some papers on these topics in the AI and Database communities. It then proposes intuitions and a formal definition for “importing terms  $S$  from KB under rules  $\mathcal{G}$ ”, and looks at the problems of implementing this for very simple kinds of TBoxes.

### 1 Introduction

There has been considerable recent interest in the notion of “module” in ontologies, including a workshop on this topic at ISWC’06. We wish to consider modules not just as units of development, but also as sources of information used by other ontologies. In this regard, modern programming languages, such as Python provide interesting patterns of use: “`from YourModule import name1 as name2, name3 as name4, ...`”. Such an ability to selectively import ontology fragments will also be beneficial in ontology engineering. For example, the enormous medical ontology ON9.3, developed at CNR in Italy (<http://www.loa-cnr.it/medicine/>), documents each of its theories (modules) with a list of imported terms. Thus, *Anatomy*, which defines 55 classes, specifies in its documentation not just

Theories included by *Anatomy*:

Meronymy, Positions, Topo-Morphology

but also

The following constants were used from included theories:

\* 3d-Area-Of defined as a relation in theory Topo-Morphology

\* > defined as a relation in theory Kif-Numbers

... (60+ other terms)

and even more interestingly

The following constants were used from theories not included:

\* Anatomical-Abnormality defined as a class in theory Abnormalities

... (20+ other terms)

Given the decade-long experience of the scientists on the above project, one should not ignore their insight that such specifications are helpful in understanding, developing, and maintaining large ontologies.

To establish some intuitions concerning the desirable properties of “importing terms”, we survey a small sample of relevant techniques that have been proposed in the literature. (Many additional papers are omitted for lack of space.)

### 2 Previous Approaches to Knowledge Import

A variety of papers provide more subtle approaches than importing entire ontology files, as in OWL. The first two categories (a-b) below, rely on (semi-automatically) fragmenting an ontology into modules, and then importing only relevant modules. The last two (c-d) directly address importing individual terms.

**(a) Logical Specification of Modules** A *logical module* KB1 of a theory KB is required to be *locally sound* (if  $KB1 \models \psi$  then  $KB \models \psi$ ) and *locally complete* (if  $KB \models \psi$  for a formula  $\psi$  that uses only symbols from  $\text{vocab}(KB1)$ , then in fact also  $KB1 \models \psi$ ). Cuenca Grau et al [4] extend this idea, by requiring that  $\text{module}(N, KB)$  — the module of name N in KB, also be “a coherent and self-contained *subset* of KB” (which in this case is a description logic TBox). As such, it should contain N’s subsuming and subsumed concept names in KB, and ensure “self-containment”. A more recent proposal, “minimal S-modules” [6], will be reviewed later.

**(b) (Automatic) Graphical Segmentation of Modules.** Seidenberg and Rector [10] suggest that  $\text{module}(N, KB)$  start out with axioms specifying the (1) subclasses of N, (2) super-classes of N, (3) restrictions on the roles of N, and (4) super-roles of N in KB. One then repeatedly adds new identifiers and axioms according to steps 2-4 above, until a fixed point is reached. If one were to draw a graph  $G_{KB}$  with concept names as nodes connected by edges representing role restrictions or IsA relationships, then this can be described as a simple graph traversal algorithm. To reduce a large module, [10] allow limiting the depth of the traversal, resulting in “dangling” *boundary classes*.

**(c) Importing Terms by Ontology Winnowing.** A surprising number of papers argue for the development of *domain-specific* ontologies by reusing *fragments of generic, top-level ontologies* such as Cyc, WordNet, etc. In such cases, the portion of the top-level ontology KB to be imported is influenced by a set of “seed concepts” S, that are to be re-used in the domain-specific ontology. The key to each such technique are the principles which *automatically* derive the axioms and possibly additional concepts to be imported.

For example, Navigli [9] starts with WordNet, whose concepts are organized by hyponym subsumption. The elements of S are concepts corresponding to the roots of local ontology trees for domain specific terms. The algorithm first eliminates concepts not on a path from the top of the WordNet hierarchy to some element of S, in a “pruning phase”; it then eliminates, as uninteresting, concepts with only one child in the hierarchy left, in a “trimming phase”. As a result,  $\text{import}(S, KB)$  is a taxonomy where every node has at least two children, so that long chains of uninteresting subsumptions are not present.

Conesa and Olive [2] elaborate Navigli’s technique, to build database conceptual schemas by starting from OpenCyc as KB. The paper describes  $\text{import}(S, KB)$  as a minimal subset of KB whose vocabulary contains S and its superclasses, and the algorithm may eliminate concepts between the topmost classes in S and the root of the taxonomy in KB, as well as classes that only provide “redundant inheritance paths”.

Note that in all proposals in this category the imported concept names are restricted to be used in the importing KB according to the following simple grammar  $\mathcal{G}_{winnow}$  for TBox axioms

```
<TBox axiom> ::= <local axiom> | <connect up axiom>
<local axiom> ::= <local DL concept>  $\sqsubseteq$  <local DL concept>
```

`<connect up axiom> ::= <local DL concept>  $\sqsubseteq$  <Imported concept identifier>`  
`<local DL concept> ::= ...`

**(d) Importing Terms in Local-Model Semantics**

The theory of binary  $\mathcal{E}$ -connections between description logics has been used in [3] to connect DL ontologies  $KB_1$  and  $KB_2$  (which are interpreted in disjoint “local” domains) through a number of binary relations (“links”)  $p_k$  between objects in these interpretations. The result is that in  $KB_1$  one can now construct concepts by restricting  $p_k$  with terms  $C_2$  from  $KB_2$ , such as  $\forall p_k.C_2$ . (In  $KB_2$ , one can use  $p_k^-$ .) This is like importing concepts from  $KB_2$  into  $KB_1$ , but restricting their syntactic occurrence to express value restrictions on the roles  $p_k$  — something that can obviously be expressed by another grammar,  $\mathcal{G}_{\mathcal{E}}$ , for subsumption axioms involving imported concepts.

Distributed Description Logics (DDL) [1] use “bridge rules” with approximately the meaning  $1:A \sqsubseteq 2:B$  to relate concepts A and B from ontologies  $KB_1$  and  $KB_2$  respectively. Such a rule can also be viewed as importing concept B into  $KB_1$  and then highly restricting its syntactic occurrence in an axiom.

**3 Importing S: intuitions and definition**

From the Anatomy example in Section 1, we start by using syntactic expressions of the form “import S from  $KB_{expt}$ ” for  $\mathbf{import}(S,KB_{expt})$ , where S is a set of identifiers  $\{N_1, N_2, \dots\}$  contained in  $\text{vocab}(KB_{expt})$ . For simplicity, when some  $KB_{impt}$  uses axioms relating the symbols in S to its own local identifiers, we assume that  $\text{vocab}(KB_{impt}) \cap \text{vocab}(KB_{expt}) \subseteq S$ .

Based on the preceding survey, we take it that the purpose of importing some set of identifiers S and their related axioms from ontology KB, as opposed to including the entire KB as a file, is to *minimize the material  $\mathbf{import}(S,KB)$  required to understand S, in order to facilitate comprehension by humans, and possibly to help with local caching.* This philosophy is most evident above in the ontology winnowing work, but also appears in the work on automatic ontology modularization.

There is some sentiment that  $\mathbf{import}(S,KB)$  should be a subset of KB, rather than its theorems [4–6]. This means that the syntactic presentation of axioms in KB is taken to matter, presumably since it helps humans understand the problem domain. We shall modify this requirement somewhat to say that *explanations* of reasoning in  $\mathbf{import}(S,KB)$  should correspond to explanations in KB. The main reason for switching to this alternate requirement is that, as argued in [7], explanations need not always be complete logical proofs, since some obvious steps may be omitted. One example of this is simple inheritance: chaining of IsA in a classification hierarchy of primitives. For example, if KB contains  $\{\text{Dog} :< \text{Canine}, \text{Canine} :< \text{Animal}\}$  and  $S = \{\text{Dog}, \text{Animal}\}$  then it should be sufficient to import  $\{\text{Dog} :< \text{Animal}\}$ <sup>1</sup>. Note that Navigli’s proposal [9] omits exactly such kinds of trivial steps.

<sup>1</sup> Contrary to standard practice, we will use  $A:< B$  to indicate an axiom in the theory, and  $A \sqsubseteq B$  to indicate the subsumption judgment, entailed or proven in a theory.

Another implication of the need for explanations is that  $\mathbf{import}(S,KB)$  may have to contain symbols other than those in  $S^2$ . For example, if  $KB = \{\text{Married} \equiv \text{Person} \sqcap \geq 1.spouse, \text{Unmarried} \equiv \text{Person} \sqcap \leq 0.spouse\}$ , and  $S = \{\text{Married}, \text{Unmarried}\}$ , then, to explain why they are disjoint, we will want  $\mathbf{import}(S,KB)$  to contain  $\{\text{Married} :< \geq 1.spouse, \text{Unmarried} :< \leq 0.spouse\}$ . One could also make a case that the actual definitions should be included, since users of the term should appreciate that these are defined, as opposed to primitive concepts. A compromise might be to allow for definitions with ellipses:  $\{\text{Married} \equiv \dots \sqcap \geq 1.spouse, \text{Unmarried} \equiv \dots \sqcap \leq 0.spouse\}$ .

Once we admit the need for seeing additional symbols from  $\text{vocab}(KB)$ , other than those in  $S$ , the question arises whether such symbols should become part of  $S$ , allowing the importer to use them in constructing new concepts/axioms. We suggest that this should *not* be the case, since the user has specified  $S$  as the set of concepts (s)he will be using. Therefore we will keep the set  $S$  unchanged, and consider  $\text{vocab}(\mathbf{import}(S,KB)) - S$  to be boundary concepts, used only in explanations. In line with our desire to reduce the need to understand all of  $KB$ , the set of such additional concepts should however be minimized.

From a logical point of view, we will obviously want  $\mathbf{import}(S,KB)$  to be locally sound. We will not however insist on full “local completeness”. The reason for this is that we have seen in both the work on upper-ontology pruning and local-model semantics that the imported identifiers might only be used in a limited way in  $KB_{\text{impt}}$ . (The use of external symbols is also limited in [5], in order to guarantee the desired property of “conservative extension”.) For example, if imported names  $N_i$  can only appear in axioms of the form  $\alpha :< N_i$ , as per  $\mathcal{G}_{\text{window}}$ , then it might not matter whether  $\mathbf{import}(S,KB) \models \neg N_1 \sqsubseteq N_2$ , since the syntax does not allow asking such questions directly of  $K3 = KB_{\text{impt}} \cup \mathbf{import}(S,KB)$ , and knowledge of this fact might not affect inferences from  $K3$  in certain DLs. (See Sec. 4 for examples.) The same might happen if the importing ontology uses a different, weaker logical language than the exporting one. The limited use of imported concept identifiers can then be exploited to decrease the set of concepts and axioms from  $KB$  that need to be included in  $\mathbf{import}(S,KB)$ .

The  $\mathbf{import}$  syntax should therefore reflect rules about the use of symbols in  $S$  in the importing ontology. An instruction of the form “ $KB_{\text{impt}}$  imports  $S$  from  $KB_{\text{expt}}$ ” would however seem to be too specific, since the material imported might change as the local ontology evolves. For this reason, we suggest characterizing the use of imported identifiers using a *grammar*  $\mathcal{G}$ , as we have done in (c) and (d) above.

To formalize the above discussion for the case of DL TBoxes, we assume that every description logic  $\mathcal{D}$  provides, as usual, a syntax for the concepts and roles, as well as axioms allowed in a TBox, plus a semantic entailment relationship  $\models$  of the logic for various kinds of judgements  $\psi$ , such as subsumption. In addition, we also need a specification  $\mathcal{XPL}$  of acceptable explanations for judgements in the logic<sup>3</sup>. As part of  $\mathcal{XPL}$ , we assume that every DL  $\mathcal{D}$  provides

<sup>2</sup> Of course, this is an integral part of all the proposals surveyed in Section 2.

<sup>3</sup> Normally, such a specification is based on a proof theory for the logic, e.g., [7].

an operator  $expand_{\mathcal{D}}(KB)$ , which may add some redundant axioms to  $KB$  to avoid unnecessarily long explanations. For example<sup>4</sup>,  $expand()$  may collapse inheritance according to

$$inherit(KB) = \{ A :< \alpha \mid A :< B_0 :< \dots :< B_n :< \alpha \text{ in } KB \}.$$

The following definition then summarizes the above intuitions

**Definition 1.** *Given (1) an (exporting) TBox  $KB$  of description logic  $\mathcal{D}_1$ , (2) a set of concept names  $S \subseteq vocab(KB)$  to be imported, (3) a description logic  $\mathcal{D}_2$  for importing Tboxes, and (4) a grammar  $\mathcal{G}$  specifying the syntax of axioms in importing Tboxes, including the occurrence of identifiers from  $S$ : We seek a minimal set of identifiers  $\tilde{S}$  containing  $S$ , and a minimal subset  $K$  of axioms from  $expand(KB)$  involving only names from  $\tilde{S}$  such that for every  $KB_{impt}$  satisfying  $\mathcal{G}$ , and every judgement  $\psi$  with  $vocab(\psi) \subseteq vocab(KB_{impt}) \cup S$ , we have that  $KB_{impt} \cup KB \models \psi$  iff  $KB_{impt} \cup K \models \psi$ , with all explanations in the latter being valid in the former.*

Such a TBox  $K$  will be referred to as **import $_{\mathcal{G}}$** ( $S, KB$ ). □

Note that such a  $K$  is guaranteed to exist, since one can start with  $\tilde{S} = vocab(KB)$ , and  $K = KB$ , and then minimize from there. Of course,  $K$  may not be unique.

Independently, in a soon-to-appear paper [6], Cuenca Grau et al have proposed a definition for the notion of “minimal  $S$ -module”, which can be viewed as a special case of the above, where  $\mathcal{D}_1 = \mathcal{D}_2$ ,  $\mathcal{G}$  does not officially restrict the occurrence of imported names (though a sufficient syntactic test for it is proposed), explanations play no role, and the above-stated conditions must hold for every  $\mathcal{D}$  with Tarski-style set-theoretic semantics. Please note that our more general definition was motivated by actual proposals in the prior literature (categories (c) and (d)), rather than just our own intuitions. The above cited paper, as well as [5] contain discussions concerning the related notion of “conservative extension”.

## 4 Computing import in some simple cases

We propose to explore some computational consequences of the above definition. Because of the observation concerning the work in [6], their negative complexity results (e.g., undecidability for  $\mathcal{ALCO}$ ) immediately transfer to our case. So, rather than jumping to consider very expressive DLs, we propose to see how various characteristics of DLs (e.g., primitive vs. defined concepts, ability to specify inconsistent concepts, role restrictions) interact with the above definition in the absence of other sources of complexity. Hence we restrict ourselves to “weak” DLs. In fact,  $\mathcal{D}_{import}$  will be taken to be only atomic concepts; and in  $\mathcal{D}_{export}$  subsumption will be determinable by a proof theory consisting of normalization rules followed by structural subsumption rules. As in [7], explanations of  $\alpha \sqsubseteq \beta$  are provided by (i) decomposing  $\beta$  into a conjunction of “atomic descriptions”  $\beta_j$ <sup>5</sup>, (ii) computing  $normalize(\alpha)$ , and (iii) then showing how  $normalize(\alpha) \sqsubseteq \beta_j$

<sup>4</sup> Below, we will use  $A, B, C, \dots$  to denote concept identifiers, and Greek letters  $\alpha, \beta, \dots$  to denote possibly complex concept expressions.

<sup>5</sup> An atomic description  $\beta_j$  cannot be expressed as the conjunction of two or more descriptions, each of which is smaller in size.

for each  $j$ . The decomposition of  $\beta$  into atomic description is not usually explained, since it is rather trivial.

We also restrict  $\mathcal{G}$  throughout to very simple “top-level ontology” style imports, allowing  $B \in S$  to only appear in axioms  $A :< B$  for  $A \notin S$ . (But note that axioms  $\{F :< C, F :< B\}$  will give the effect of conjoining elements of  $S$  in  $KB_{impt}$ !)

Throughout, we allow  $KB_{expt}$  to have axioms of the form  $A :< \alpha$ , providing necessary conditions for primitive concepts,  $A$ . But we forbid recursion in axioms.

#### 4.1 Conjunction in Necessary Conditions

For this, we decompose axioms involving conjunction into ones without them. Thus, if  $KB$  contained  $F :< (A \sqcap G)$ , but all we needed for a proof is  $F :< A$ , we will avoid the trivial steps of going from  $F :< (A \sqcap G)$  to  $F :< A$ . For this purpose, we use an operator  $expand_{\sqcap}()$  defined as

$$\begin{aligned} expand_{\sqcap}(\alpha_1 \sqcap \dots \sqcap \alpha_n) &= \{\alpha_1, \dots, \alpha_n\} \\ expand_{\sqcap}(\alpha :< \beta) &= \{\alpha :< \gamma \mid \gamma \in expand_{\sqcap}(\beta)\} \\ expand_{\sqcap}(KB) &= \{expand_{\sqcap}(\alpha :< \beta) \mid \alpha :< \beta \in KB\} \end{aligned}$$

and call the fixed point of this operator  $expand_{\sqcap}^*$ .

Subsumption reasoning in  $KB_{impt} \cup expand_{\sqcap}^*(KB)$  now consists solely of transitive chaining of axioms, which is abbreviated by  $inherit()$ . This yields

$$\mathbf{import}(S, KB) = reduce(select(S, inherit(expand_{\sqcap}^*(KB))))$$

where  $select(V, KB) = \{\psi \in KB \mid vocab(\psi) \subseteq V\}$  and  $reduce()$  removes redundant axioms — in this case, redundancy introduced earlier by  $inherit()$ .

The complexity of this computation is clearly polynomial.

#### 4.2 Disjoint Concepts and Necessary Conditions

If the exporting DL now also has atomic negation, say, one can specify disjoint concepts  $B$  and  $C$ , which means that if  $KB_{impt}$  has axioms  $\{F :< B, F :< C\}$ , then we must be able to conclude that  $F \sqsubseteq \perp$ , in addition to  $inherit$ .

This may require considering identifiers,  $A$ , not in  $S$ , as in the case where  $S = \{B, C\}$  but  $KB$  contains  $\{B :< A, C :< \neg A\}$ .

If we define  $T_S = \{A \in vocab(KB) \mid \text{there exist } B, C \in S, KB \models B \sqsubseteq A, C \sqsubseteq \neg A\}$ , then it is sufficient to also include in  $\mathbf{import}(S, KB)$  for all such  $A, B$  and  $C$ , axioms  $\{B :< A, C :< \neg A\}$ , testifying to the presence of  $A$  in  $T_S$ .

In and of itself this is not hard. However, the two concepts  $B$  and  $C$  may be disjoint for more than one reason: e.g., they may also be subsumed by  $\hat{A}$  and  $\neg \hat{A}$  respectively. According to our definition, and its intuitions, we should minimize the set of *additional* concepts introduced; hence  $vocab(\mathbf{import}(S, KB))$  should not contain both  $A$  and  $\hat{A}$ . Unfortunately, this minimization is a combinatorial problem:

**Proposition 1.** *There are simple  $KB$  with axioms of the form  $C :< D$  and  $C' :< \neg D'$ , ( $C, C' \in S, D, D' \notin S$ ), such that the following problem is NP-hard: find the smallest set  $W \subseteq vocab(KB) - S$  with the property that for all  $B, C \in S$ :  $KB \models (B \sqcap C) \sqsubseteq \perp$  iff  $select(S \cup W, KB) \models (B \sqcap C) \sqsubseteq \perp$*

The proof is by reduction from the hitting set problem [Garey & Johnson, SP8].

Similar arguments will hold for any DL that has some way of describing inconsistent concepts, such as number restrictions; they also seem to apply to the minimal S-modules of [6].

### 4.3 Definitions with Conjunction

The novelty here is that KB can now have axioms of the form  $D \equiv A \sqcap B$ , as well as  $C :< E$ . In this case, definitions can no longer be replaced by simple subsumptions on atoms. In fact, if we have  $S = \{A', B', D, H\}$ ,  $KB_0 = \{D \equiv A \sqcap B, A' :< A, B' :< B\}$ , while  $KB_{impt}$  contains  $\{E :< A', E :< B'\}$ , then  $KB_{impt} \cup KB_0 \models E \sqsubseteq D$ . Thus **import**(S,KB<sub>0</sub>) needs to support this inference by importing all of KB<sub>0</sub>.

Now note that since the conjunction of all concepts in S not subsumed by D ( $\delta_D = \sqcap_{C \in S, KB \not\sqsubseteq C} \sqsubseteq_D C$ ) is the strongest possible condition w.r.t. KB applicable to some concept F in  $KB_{impt}$ , if this is not sufficient to entail D ( $KB \not\models \delta_D \sqsubseteq D$ ) then D's definition need not be considered, and hence can be omitted from **import**(S,DB), since it would never be needed as part of an explanation for why an F is subsumed by D. Therefore it would be sufficient to repeatedly add to **import**(S,KB) definitions for concepts  $D \in \text{vocab}(KB)$  as long as  $KB \models \delta_D \sqsubseteq D$ .

Unfortunately, while the result will include enough axioms, it may include too many. For example, if concepts A and B in KB are subsumed by  $\hat{A}$  and  $\hat{B}$  individually, as well as n other concepts  $C_1, \dots, C_n$  jointly, then importing the defined concept  $D \equiv \hat{A} \sqcap C_1 \sqcap \dots \sqcap C_n \sqcap \hat{B}$ , allows for  $2^n$  possible minimal combinations of axioms to be imported (depending on how the  $C_i$  are allotted to A and B). The presence of other concepts and axioms will then tilt in favor of some of these choices.

**Proposition 2.** *If one allows necessary conditions on definitions, the problem of finding **import**(S,KB) when KB has conjunctive definitions or axioms of the form  $A :< B$ , is NP-hard*

Proof is by reduction from the NP-hard problem of minimizing Horn proofs, or minimizing input to monotone boolean circuits. We strongly suspect that the theorem holds even if definitions cannot have necessary conditions.

### 4.4 Using $\mathcal{FL}^-$

We have seen so far the effect of allowing disjoint concepts and definitions. Let us consider now the other *sine qua non* of DLs, role restrictions.

When considering necessary conditions, restrictions of the form  $\exists p.\top$  are treated as atomic, while nested  $\forall$ -restrictions need to be separated into atomic descriptions, which do not involve conjunction. For this purpose, extend  $expand_{\sqcap}()$  as follows:

$$expand_{\sqcap}(\forall p.\beta) = \{\forall p.\gamma \mid \gamma \in expand_{\sqcap}(\beta)\}.$$

Now  $inherit(expand_{\sqcap}^*(KB))$  again contains axioms abbreviating chains of subsumption from a concept A in S to atomic descriptions  $\beta_i$  appearing on the right hand side of axioms in S. **import**(S,KB) can now be computed as in 4.1 above.

As illustrated above, the general pattern for adding new constructors for DLs with structural subsumption seems to be to extend the notion of atomic description and  $expand_{\sqcap}()$  so that  $inherit(expand_{\sqcap}(KB))$  contains the axioms needed to find the normal forms of concepts in S, and detect conjunctions that can lead to  $\perp$ . One is then faced with a minimization problem for deciding which new identifiers and axioms to include, and this is likely to be difficult to solve precisely.

## 5 Conclusions

Starting from a sample of works on ontology modularization and reuse, we have argued for a set of desirable properties for the notion of “ $KB_1$  imports terms  $S$  from  $KB_2$ ”, distinguishing this from the problem of ontology modularization by: allowing restrictions on the place where imported names can be used, and requiring both minimization of material imported and preservation of explanations — all properties motivated by prior examples of importing studied in the literature. We then investigated the difficulties encountered with implementing the corresponding formal definition in the case of TBoxes that use simple DLs, where subsumption itself is easy. Perhaps not surprisingly, attempts to *minimize* the set of axioms imported leads to combinatorial difficulties. It remains to be seen if the definition can be modified in a motivated manner (e.g., importing should provide *all* explanations in the exporting KB) and if approximate solutions to NP-hard problems would help. Forthcoming work with Fausto Giunchiglia will apply this framework to UML.

**Acknowledgements.** I am grateful to Fausto Giunchiglia for discussions starting down this path, and to the referees for pointing out unclear aspects. This work was supported in part by the U.S. DHS under ONR grant N00014-07-1-0150.

## References

1. Alex Borgida, Luciano Serafini: Distributed Description Logics — Assimilating Information from Peer Sources. *J. Data Semantics 1*: 153-184 (2003)
2. Jorge Conesa, Antoni Olive: A Method for Pruning Ontologies in the Development of Conceptual Schemas of Information Systems. *J. Data Semantics V*: 64-90 (2006)
3. Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin: Working with Multiple Ontologies on the Semantic Web. *International Semantic Web Conference 2004*: 620-634
4. Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, Aditya Kalyanpur: Modularity and Web Ontologies. *KR 2006*: 198-209
5. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov and Ulrike Sattler, A Logical Framework for Modularity of Ontologies. *IJCAI 2007*.
6. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov and Ulrike Sattler Just the Right Amount: Extracting Modules from Ontologies. *WWW 2007*.
7. Deborah L. McGuinness, Alexander Borgida: Explaining Subsumption in Description Logics. *IJCAI 1995*: 816-821
8. Carsten Lutz, Dirk Walther, and Frank Wolter: Conservative Extensions in Expressive Description Logics. *IJCAI 2007*.
9. Roberto Navigli: Extending, Pruning and Trimming General Purpose Ontologies. *2nd IEEE SMC 2002*
10. Julian Seidenberg, Alan L. Rector: Web ontology segmentation: analysis, classification and use. *WWW 2006*: 13-22

## A constructive semantics for $\mathcal{ALC}$

Loris Bozzato<sup>1</sup>, Mauro Ferrari<sup>1</sup>, Camillo Fiorentini<sup>2</sup>, Guido Fiorino<sup>3</sup>

<sup>1</sup> DICOM, Univ. degli Studi dell’Insubria, Via Mazzini 5, 21100, Varese, Italy

<sup>2</sup> DSI, Univ. degli Studi di Milano, Via Comelico, 39, 20135 Milano, Italy

<sup>3</sup> DIMEQUANT, Univ. degli Studi di Milano-Bicocca  
P.zza dell’Ateneo Nuovo 1, 20126 Milano, Italy

**Abstract.** One of the main concerns of constructive semantics is to provide a computational interpretation for the proofs of a given logic. In this paper we introduce a constructive semantics for the basic description logic  $\mathcal{ALC}$  in the spirit of the BHK interpretation. We prove that such a semantics provides an interpretation of  $\mathcal{ALC}$  formulas consistent with the classical one and we show how, according to such a semantics, proofs of a suitable natural deduction calculus for  $\mathcal{ALC}$  support a proofs-as-programs paradigm.

### 1 Introduction

In recent works, see e.g. [3, 5, 6], starting from different motivations, various constructive interpretations of description logics have been proposed. However, as far as we know, no computational interpretation for proofs has been given in this context. The aim of this paper is to propose a constructive semantics for  $\mathcal{ALC}$  formulas, we call *information-terms semantics*, that allows us to give a computational interpretation of the proofs of a natural deduction calculus for  $\mathcal{ALC}$ . In particular, we will be able to read proofs of  $\mathcal{ALC}$ -“goals” as programs to compute goal answers.

The information-terms semantics is related to the BHK constructive explanation of logical connectives (see [7, 11] for a deeper discussion) and has already been applied in several frameworks [4, 8]. An information term is a mathematical object that explicitly explains the truth of a formula in a given classical model. For instance, if we prove that an individual  $c$  belongs to the concept  $\exists R.C$ , the information term provides the witness  $d$  such that  $(c, d) \in R$  and  $d \in C$ . Differently from other approaches, such as [3, 5], information-terms semantics relies on the classical reading of logical connectives; as a consequence, we can read  $\mathcal{ALC}$  formulas in the usual way.

In this paper we introduce the information-terms semantics and we compare it with the classical one. Then, we introduce a natural deduction calculus  $\mathcal{ND}_c$  for  $\mathcal{ALC}$  and we show that it is sound with respect to information-terms semantics. As a by-product of the Soundness Theorem, we get a computational interpretation of proofs. We show, by means of an example, that this interpretation supports the proofs-as-programs paradigm.

## 2 $\mathcal{ALC}$ language and semantics

We begin introducing the language  $\mathcal{L}$  for  $\mathcal{ALC}$  [1, 10], based on the following denumerable sets: the set  $\text{NR}$  of *role names*, the set  $\text{NC}$  of *concept names*, the set  $\text{NI}$  of *individual names*. A *concept*  $H$  is a formula of the kind:

$$H ::= C \mid \neg H \mid H \sqcap H \mid H \sqcup H \mid \exists R.H \mid \forall R.H$$

where  $C \in \text{NC}$  and  $R \in \text{NR}$ . Let  $\text{Var}$  be a denumerable set of *individual variables*, our calculus works on *formulas*  $K$  of  $\mathcal{L}$  defined according to the following grammar:

$$K ::= \perp \mid (s, t) : R \mid (s, t) : \neg R \mid t : H \mid \forall H$$

where  $s, t \in \text{NI} \cup \text{Var}$ ,  $R \in \text{NR}$  and  $H$  is a concept. We remark that variables, that usually are not used in description logic formalization, are useful to put in evidence the “parameters” of natural deduction proofs. An *atomic formula* of  $\mathcal{L}$  is a formula of the kind  $\perp$ ,  $(s, t) : R$ ,  $t : C$ , with  $C$  a concept name; a *negated formula* is a formula of the kind  $(s, t) : \neg R$  or  $t : \neg H$ . A formula is *closed* if it does not contain variables. We write  $\neg((s, t) : R)$ ,  $\neg((s, t) : \neg R)$ ,  $\neg(t : H)$  as abbreviations for  $(s, t) : \neg R$ ,  $(s, t) : R$ ,  $t : \neg H$  respectively;  $A \sqsubseteq B$  stands for  $\forall(\neg A \sqcup B)$ .

A *model (interpretation)*  $\mathcal{M}$  for  $\mathcal{L}$  is a pair  $(\mathcal{D}^{\mathcal{M}}, \cdot^{\mathcal{M}})$ , where  $\mathcal{D}^{\mathcal{M}}$  is a non-empty set (the *domain* of  $\mathcal{M}$ ) and  $\cdot^{\mathcal{M}}$  is a *valuation* map such that: for every  $c \in \text{NI}$ ,  $c^{\mathcal{M}} \in \mathcal{D}^{\mathcal{M}}$ ; for every  $C \in \text{NC}$ ,  $C^{\mathcal{M}} \subseteq \mathcal{D}^{\mathcal{M}}$ ; for every  $R \in \text{NR}$ ,  $R^{\mathcal{M}} \subseteq \mathcal{D}^{\mathcal{M}} \times \mathcal{D}^{\mathcal{M}}$ . A non atomic concept  $H$  is interpreted by a subset  $H^{\mathcal{M}}$  of  $\mathcal{D}^{\mathcal{M}}$ :

$$\begin{aligned} (\neg A)^{\mathcal{M}} &= \mathcal{D}^{\mathcal{M}} \setminus A^{\mathcal{M}} & (A \sqcap B)^{\mathcal{M}} &= A^{\mathcal{M}} \cap B^{\mathcal{M}} & (A \sqcup B)^{\mathcal{M}} &= A^{\mathcal{M}} \cup B^{\mathcal{M}} \\ (\exists R.A)^{\mathcal{M}} &= \{d \in \mathcal{D}^{\mathcal{M}} \mid \text{there is } d' \in \mathcal{D}^{\mathcal{M}} \text{ s.t. } (d, d') \in R^{\mathcal{M}} \text{ and } d' \in A^{\mathcal{M}}\} \\ (\forall R.A)^{\mathcal{M}} &= \{d \in \mathcal{D}^{\mathcal{M}} \mid \text{for all } d' \in \mathcal{D}^{\mathcal{M}}, (d, d') \in R^{\mathcal{M}} \text{ implies } d' \in A^{\mathcal{M}}\} \end{aligned}$$

An *assignment* on a model  $\mathcal{M}$  is a map  $\theta : \text{Var} \rightarrow \mathcal{D}^{\mathcal{M}}$ . If  $t \in \text{NI} \cup \text{Var}$ ,  $t^{\mathcal{M}, \theta}$  is the element of  $\mathcal{D}$  denoting  $t$  in  $\mathcal{M}$  w.r.t.  $\theta$ , namely:  $t^{\mathcal{M}, \theta} = \theta(t)$  if  $t \in \text{Var}$ ;  $t^{\mathcal{M}, \theta} = t^{\mathcal{M}}$  if  $t \in \text{NI}$ . A formula  $K$  is *valid* in  $\mathcal{M}$  w.r.t.  $\theta$ , and we write  $\mathcal{M}, \theta \models K$ , if  $K \neq \perp$  and one of the following conditions holds:

$$\begin{aligned} \mathcal{M}, \theta \models (s, t) : R &\text{ iff } (s^{\mathcal{M}, \theta}, t^{\mathcal{M}, \theta}) \in R^{\mathcal{M}} & \mathcal{M}, \theta \models t : H &\text{ iff } t^{\mathcal{M}, \theta} \in H^{\mathcal{M}} \\ \mathcal{M}, \theta \models (s, t) : \neg R &\text{ iff } (s^{\mathcal{M}, \theta}, t^{\mathcal{M}, \theta}) \notin R^{\mathcal{M}} & \mathcal{M}, \theta \models \forall H &\text{ iff } H^{\mathcal{M}} = \mathcal{D}^{\mathcal{M}} \end{aligned}$$

We write  $\mathcal{M} \models K$  iff  $\mathcal{M}, \theta \models K$  for every assignment  $\theta$ . Note that  $\mathcal{M} \models \forall H$  iff  $\mathcal{M} \models x : H$ , with  $x$  any variable. If  $\Gamma$  is a set of formulas,  $\mathcal{M} \models \Gamma$  means that  $\mathcal{M} \models K$  for every  $K \in \Gamma$ . We say that  $K$  is a *logical consequence* of  $\Gamma$ , and we write  $\Gamma \models K$ , iff, for every  $\mathcal{M}$  and every  $\theta$ ,  $\mathcal{M}, \theta \models \Gamma$  implies  $\mathcal{M}, \theta \models K$ .

Now, we introduce *information terms*, that will be the base structure of our constructive semantics. Let  $\mathcal{N}$  be a finite subset of  $\text{NI}$ . By  $\mathcal{L}_{\mathcal{N}}$  we denote the set of formulas  $K$  of  $\mathcal{L}$  such that all the individual names occurring in  $K$  belong to  $\mathcal{N}$ . Given a closed formula  $K$  of  $\mathcal{L}_{\mathcal{N}}$ , we define the set of information terms

$\text{IT}_{\mathcal{N}}(K)$  by induction on  $K$  as follows.

$$\begin{aligned} \text{IT}_{\mathcal{N}}(K) &= \{\mathbf{tt}\}, \text{ if } K \text{ is an atomic or negated formula} \\ \text{IT}_{\mathcal{N}}(c : A \sqcap B) &= \{(\alpha, \beta) \mid \alpha \in \text{IT}_{\mathcal{N}}(c : A) \text{ and } \beta \in \text{IT}_{\mathcal{N}}(c : B)\} \\ \text{IT}_{\mathcal{N}}(c : A_1 \sqcup A_2) &= \{(k, \alpha) \mid k \in \{1, 2\} \text{ and } \alpha \in \text{IT}_{\mathcal{N}}(c : A_k)\} \\ \text{IT}_{\mathcal{N}}(c : \exists R.A) &= \{(d, \alpha) \mid d \in \mathcal{N} \text{ and } \alpha \in \text{IT}_{\mathcal{N}}(d : A)\} \\ \text{IT}_{\mathcal{N}}(c : \forall R.A) &= \text{IT}_{\mathcal{N}}(\forall A) = \{\phi : \mathcal{N} \rightarrow \bigcup_{d \in \mathcal{N}} \text{IT}_{\mathcal{N}}(d : A) \mid \phi(d) \in \text{IT}_{\mathcal{N}}(d : A)\} \end{aligned}$$

Let  $\mathcal{M}$  be a model for  $\mathcal{L}$ ,  $K$  a closed formula of  $\mathcal{L}_{\mathcal{N}}$  and  $\eta \in \text{IT}_{\mathcal{N}}(K)$ . We define the *realizability relation*  $\mathcal{M} \triangleright \langle \eta \rangle K$  by induction on the structure of  $K$ .

$$\begin{aligned} \mathcal{M} \triangleright \langle \mathbf{tt} \rangle K &\text{ iff } \mathcal{M} \models K, \text{ where } K \text{ is an atomic or negated formula} \\ \mathcal{M} \triangleright \langle (\alpha, \beta) \rangle c : A \sqcap B &\text{ iff } \mathcal{M} \triangleright \langle \alpha \rangle c : A \text{ and } \mathcal{M} \triangleright \langle \beta \rangle c : B \\ \mathcal{M} \triangleright \langle (k, \alpha) \rangle c : A_1 \sqcup A_2 &\text{ iff } \mathcal{M} \triangleright \langle \alpha \rangle c : A_k \\ \mathcal{M} \triangleright \langle (d, \alpha) \rangle c : \exists R.A &\text{ iff } \mathcal{M} \models (c, d) : R \text{ and } \mathcal{M} \triangleright \langle \alpha \rangle d : A \\ \mathcal{M} \triangleright \langle \phi \rangle c : \forall R.A &\text{ iff } \mathcal{M} \models c : \forall R.A \text{ and, for every } d \in \mathcal{N}, \\ &\quad \mathcal{M} \models (c, d) : R \text{ implies } \mathcal{M} \triangleright \langle \phi(d) \rangle d : A \\ \mathcal{M} \triangleright \langle \phi \rangle \forall A &\text{ iff } \mathcal{M} \models \forall A \text{ and, for every } d \in \mathcal{N}, \mathcal{M} \triangleright \langle \phi(d) \rangle d : A \end{aligned}$$

If  $\Gamma$  is a set of closed formulas  $\{K_1, \dots, K_n\}$  of  $\mathcal{L}_{\mathcal{N}}$ ,  $\text{IT}_{\mathcal{N}}(\Gamma)$  denotes the set of  $n$ -tuples  $\bar{\eta} = (\eta_1, \dots, \eta_n)$  such that, for every  $1 \leq j \leq n$ ,  $\eta_j \in \text{IT}_{\mathcal{N}}(K_j)$ ;  $\mathcal{M} \triangleright \langle \bar{\eta} \rangle \Gamma$  iff, for every  $1 \leq j \leq n$ ,  $\mathcal{M} \triangleright \langle \eta_j \rangle K_j$ .

We remark that  $\mathcal{M} \triangleright \langle \eta \rangle K$  implies  $\mathcal{M} \models K$ , hence the constructive semantics is compatible with the usual classical one. The converse in general does not hold and stronger conditions are required:

**Proposition 1.** *Let  $K$  be a closed formula of  $\mathcal{L}$  and let  $\mathcal{M}$  be a finite model for  $\mathcal{L}$ . If  $\mathcal{M} \models K$ , there exists a finite subset  $\mathcal{N}$  of NI and  $\eta \in \text{IT}_{\mathcal{N}}(K)$  such that  $\mathcal{M} \triangleright \langle \eta \rangle K$ .*

We point out that in our setting negation has a classical meaning, thus negated formulas are not constructively explained by an information term. However, how we will discuss in future works, information terms semantics can be extended to treat various kinds of constructive negation as those discussed in [6].

In the following example, we show how an information term provides all the information needed to “constructively” explain the meaning of a formula.

*Example 1.* Let us consider the knowledge base, inspired to the classical example of [2], consisting of the Tbox  $\mathcal{T}$

$$\begin{aligned} (Ax_1) : \forall(\neg\text{FOOD} \sqcup \exists\text{goesWith.COLOR}) &\equiv \text{FOOD} \sqsubseteq \exists\text{goesWith.COLOR} \\ (Ax_2) : \forall(\neg\text{COLOR} \sqcup \exists\text{isColorOf.WINE}) &\equiv \text{COLOR} \sqsubseteq \exists\text{isColorOf.WINE} \end{aligned}$$

and the Abox  $\mathcal{A}$

barolo:WINE	red:COLOR	(red, barolo) : isColorOf
chardonnay:WINE	white:COLOR	(white, chardonnay) : isColorOf
fish:FOOD		(fish, white) : goesWith
meat:FOOD		(meat, red) : goesWith

Let  $\mathbf{WNI}$  be the set of individual names occurring in  $\mathcal{A}$ . An element of  $\text{IT}_{\mathbf{WNI}}(Ax_1)$  is a function  $\phi$  mapping each  $c \in \mathbf{WNI}$  to an element  $\delta \in \text{IT}_{\mathbf{WNI}}(c : \neg\text{FOOD} \sqcup \exists \text{goesWith.COLOR})$ , where either  $\delta = (1, \mathbf{tt})$  (intuitively,  $c$  is not a food) or  $\delta = (2, (d, \mathbf{tt}))$  (intuitively,  $d$  is a wine color which goes with food  $c$ ). For instance, let us consider the following  $\gamma_1 \in \text{IT}_{\mathbf{WNI}}(Ax_1)$ , where we enclose between square brackets the pairs  $(c, \phi(c))$ :

[ (barolo, (1, tt)), (chardonnay, (1, tt)), (red, (1, tt)), (white, (1, tt))  
 (fish, (2, (white, tt))), (meat, (2, (red, tt))) ]

Let  $\mathcal{M}_{\mathbf{W}}$  be a model of  $\mathcal{A} \cup \mathcal{T}$ . One can easily check that  $\mathcal{M}_{\mathbf{W}} \triangleright \langle \gamma_1 \rangle Ax_1$ . Similarly, if  $\gamma_2 \in \text{IT}_{\mathbf{WNI}}(Ax_2)$  is the information term

[ (barolo, (1, tt)), (chardonnay, (1, tt)), (red, (2, (barolo, tt))),  
 (white, (2, (chardonnay, tt))), (fish, (1, tt)), (meat, (1, tt)) ]

then  $\mathcal{M}_{\mathbf{W}} \triangleright \langle \gamma_2 \rangle Ax_2$  as well. We conclude  $\mathcal{M}_{\mathbf{W}} \triangleright \langle \langle \gamma_1, \gamma_2 \rangle \rangle \mathcal{T}$ .

### 3 The natural calculus $\mathcal{ND}_c$

In this section we introduce a calculus  $\mathcal{ND}_c$  for  $\mathcal{ALC}$  similar to the usual natural deduction calculi for classical and intuitionistic logic (see, e.g., [9]). The rules of  $\mathcal{ND}_c$  are given in Figure 1. We remark that we have *introduction* and *elimination* rules for all the logical constants; some rules (namely,  $\sqcup E$ ,  $\exists E$  and  $\forall I$ ) allow to discharge some of the assumptions (we put them between square brackets). The rules  $\exists E$ ,  $\forall I$  and  $\forall_U I$  need a side condition on the rule parameter to guarantee correctness. We notice that the rule  $\perp E$  is intuitionistic, we will briefly discuss in the conclusions the relation with the calculus using the classical rule of *reductio ad absurdum*.

By  $\pi : \Gamma \vdash K$ , with  $\Gamma$  a set of formulas, we denote a proof of  $K$  with undischarged formulas  $\Gamma$ . We say that  $\pi : \Gamma \vdash K$  is over  $\mathcal{L}_{\mathcal{N}}$  if all the formulas occurring in the proof belong to  $\mathcal{L}_{\mathcal{N}}$ .

First of all, one can easily check that  $\mathcal{ND}_c$  preserves the validity of formulas. Indeed, let  $\pi : \Gamma \vdash K$  be a proof of  $\mathcal{ND}_c$ ; then:

(P1). For every model  $\mathcal{M}$  and assignment  $\theta$ ,  $\mathcal{M}, \theta \models \Gamma$  implies  $\mathcal{M}, \theta \models K$ .

As a consequence,  $\pi : \Gamma \vdash K$  implies  $\Gamma \models K$ . Let  $\mathcal{N}$  be a finite subset of  $\mathbf{NI}$ . An  $\mathcal{N}$ -substitution  $\sigma$  is a map  $\sigma : \mathbf{Var} \rightarrow \mathcal{N}$ . We extend  $\sigma$  to  $\mathcal{L}$  as usual: if  $c \in \mathbf{NI}$ ,  $\sigma c = c$ ; for a formula  $K$ ,  $\sigma K$  denotes the closed formula of  $\mathcal{L}_{\mathcal{N}}$  obtained by replacing every variable  $x$  occurring in  $K$  with  $\sigma(x)$ ; if  $\Gamma$  is a set of formulas,  $\sigma \Gamma$  is the set of  $\sigma K$  such that  $K \in \Gamma$ . If  $c \in \mathbf{NI}$ ,  $\sigma[c/p]$  is the  $\mathcal{N}$ -substitution  $\sigma'$  such that  $\sigma'(p) = c$  and  $\sigma'(x) = \sigma(x)$  for  $x \neq p$ .

We associate with every proof  $\pi : \Gamma \vdash K$  of  $\mathcal{ND}_c$  over  $\mathcal{L}_{\mathcal{N}}$  and every  $\mathcal{N}$ -substitution  $\sigma$  a function

$$\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma \Gamma) \rightarrow \text{IT}_{\mathcal{N}}(\sigma K)$$

that will provide the computational interpretation of  $\pi$ . To this aim  $\Phi_{\sigma, \mathcal{N}}^\pi$  will be defined, by induction on the depth of  $\pi$ , in order to fulfill the following property:

---


$$\begin{array}{c}
 \frac{\Gamma_1 \quad \Gamma_2}{\vdots \pi_1 \quad \vdots \pi_2} \frac{K \quad \neg K}{\perp} \perp I \\
 \\
 \frac{\Gamma}{\vdots \pi'} \frac{\perp}{K} \perp E \\
 \\
 \frac{\Gamma_1 \quad \Gamma_2}{\vdots \pi_1 \quad \vdots \pi_2} \frac{t : A \quad t : B}{t : A \sqcap B} \sqcap I \\
 \\
 \frac{\Gamma}{\vdots \pi'} \frac{t : A_1 \sqcap A_2}{t : A_k} \sqcap E_k \quad k \in \{1, 2\} \\
 \\
 \frac{\Gamma}{\vdots \pi'} \frac{t : A_k}{t : A_1 \sqcup A_2} \sqcup I_k \quad k \in \{1, 2\} \\
 \\
 \frac{\Gamma_1 \quad \Gamma_2, [t : A] \quad \Gamma_3, [t : B]}{\vdots \pi_1 \quad \vdots \pi_2 \quad \vdots \pi_3} \frac{t : A \sqcup B \quad K \quad K}{K} \sqcup E \\
 \\
 \frac{\Gamma_1 \quad \Gamma_2}{\vdots \pi_1 \quad \vdots \pi_2} \frac{(t, u) : R \quad u : A}{t : \exists R.A} \exists I \\
 \\
 \frac{\Gamma_1 \quad \Gamma_2, [(t, p) : R, p : A]}{\vdots \pi_1 \quad \vdots \pi_2} \frac{t : \exists R.A \quad K}{K} \exists E \quad \text{where } p \in \text{Var}, p \text{ does not occur in } \Gamma_2 \cup \{K\} \text{ and } p \neq t \\
 \\
 \frac{\Gamma, [(t, p) : R]}{\vdots \pi'} \frac{p : A}{t : \forall R.A} \forall I \quad \text{where } p \in \text{Var}, p \text{ does not occur in } \Gamma \text{ and } p \neq t \\
 \\
 \frac{\Gamma_1 \quad \Gamma_2}{\vdots \pi_1 \quad \vdots \pi_2} \frac{s : \forall R.A \quad (s, t) : R}{t : A} \forall E \\
 \\
 \frac{\Gamma}{\vdots \pi'} \frac{p : A}{\forall A} \forall UI \quad \text{where } p \in \text{Var} \text{ and } p \text{ does not occur in } \Gamma \\
 \\
 \frac{\Gamma}{\vdots \pi'} \frac{\forall A}{t : A} \forall UE
 \end{array}$$


---

**Fig. 1.** The rules of the calculus  $\mathcal{ND}_c$

(P2). For every model  $\mathcal{M}$  and  $\bar{\gamma} \in \text{IT}_{\mathcal{N}}(\sigma\Gamma)$ ,  $\mathcal{M} \triangleright \langle \bar{\gamma} \rangle \sigma\Gamma$  implies  $\mathcal{M} \triangleright \langle \Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}) \rangle \sigma K$ .

If  $\pi$  only consists of the introduction of an assumption  $K$ , then  $\Phi_{\sigma, \mathcal{N}}^\pi$  is the identity function on  $\text{IT}_{\mathcal{N}}(\sigma K)$ . Otherwise,  $\pi$  is obtained by applying a rule  $r$  of Figure 1 to some subproofs:

- (1)  $r = \perp I$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}_1, \bar{\gamma}_2) = \mathbf{tt}$ .
- (2)  $r = \perp E$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \rightarrow \text{IT}_{\mathcal{N}}(\sigma K)$  and  $\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}) = \eta^+$ , where  $\eta^+$  is any element of  $\text{IT}_{\mathcal{N}}(K)$  (for the definiteness of  $\Phi_{\sigma, \mathcal{N}}^\pi$ , one has to assume that, for every  $K \in \mathcal{L}_{\mathcal{N}}$ , an element  $\eta^+ \in \text{IT}_{\mathcal{N}}(K)$  is defined).
- (3)  $r = \sqcap I$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \rightarrow \text{IT}_{\mathcal{N}}(\sigma t : A \sqcap B)$  and

$$\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}_1, \bar{\gamma}_2) = (\Phi_{\sigma, \mathcal{N}}^{\pi_1}(\bar{\gamma}_1), \Phi_{\sigma, \mathcal{N}}^{\pi_2}(\bar{\gamma}_2))$$

- (4)  $r = \sqcap E_k$  ( $k \in \{1, 2\}$ ). Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \rightarrow \text{IT}_{\mathcal{N}}(\sigma t : A_k)$  and

$$\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}) = \text{Pro}_k(\Phi_{\sigma, \mathcal{N}}^{\pi'}(\bar{\gamma}))$$

where  $\text{Pro}_k$  is the  $k$ -projection function.

- (5)  $r = \sqcup I_k$  ( $k \in \{1, 2\}$ ). Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \rightarrow \text{IT}_{\mathcal{N}}(\sigma t : A_1 \sqcup A_2)$  and

$$\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}) = (k, \Phi_{\sigma, \mathcal{N}}^{\pi'}(\bar{\gamma}))$$

- (6)  $r = \sqcup E$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_3) \rightarrow \text{IT}_{\mathcal{N}}(\sigma K)$  and

$$\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}_1, \bar{\gamma}_2, \bar{\gamma}_3) = \begin{cases} \Phi_{\sigma, \mathcal{N}}^{\pi_2}(\bar{\gamma}_2, \alpha) & \text{if } \Phi_{\sigma, \mathcal{N}}^{\pi_1}(\bar{\gamma}_1) = (1, \alpha) \\ \Phi_{\sigma, \mathcal{N}}^{\pi_3}(\bar{\gamma}_3, \beta) & \text{if } \Phi_{\sigma, \mathcal{N}}^{\pi_1}(\bar{\gamma}_1) = (2, \beta) \end{cases}$$

- (7)  $r = \exists I$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \rightarrow \text{IT}_{\mathcal{N}}(\sigma t : \exists R.A)$  and

$$\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}_1, \bar{\gamma}_2) = (\sigma u, \Phi_{\sigma, \mathcal{N}}^{\pi_2}(\bar{\gamma}_2))$$

- (8)  $r = \exists E$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \rightarrow \text{IT}_{\mathcal{N}}(\sigma K)$  and

$$\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}_1, \bar{\gamma}_2) = \Phi_{\sigma[c/p], \mathcal{N}}^{\pi_2}(\bar{\gamma}_2, \mathbf{tt}, \alpha)$$

where  $(c, \alpha) = \Phi_{\sigma, \mathcal{N}}^{\pi_1}(\bar{\gamma}_1)^1$ .

- (9)  $r = \forall I$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma) \rightarrow \text{IT}_{\mathcal{N}}(\sigma t : \forall R.A)$  and<sup>2</sup>

$$[\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma})](c) = \Phi_{\sigma[c/p], \mathcal{N}}^{\pi'}(\bar{\gamma}, \mathbf{tt}) \quad \text{for every } c \in \mathcal{N}$$

- (10)  $r = \forall E$ . Then,  $\Phi_{\sigma, \mathcal{N}}^\pi : \text{IT}_{\mathcal{N}}(\sigma\Gamma_1) \times \text{IT}_{\mathcal{N}}(\sigma\Gamma_2) \rightarrow \text{IT}_{\mathcal{N}}(\sigma t : A)$  and

$$\Phi_{\sigma, \mathcal{N}}^\pi(\bar{\gamma}_1, \bar{\gamma}_2) = [\Phi_{\sigma, \mathcal{N}}^{\pi_1}(\bar{\gamma}_1)](\sigma t)$$

- (11)  $r = \forall_U I$ . Analogous to the case  $r = \forall I$ .

- (12)  $r = \forall_U E$ . Analogous to the case  $r = \forall E$ .

One can easily check that  $\Phi_{\sigma, \mathcal{N}}^\pi$  is a well-defined function and that (P2) holds. Let  $\Phi_{\mathcal{N}}^\pi = \Phi_{\sigma, \mathcal{N}}^\pi$ , where  $\sigma$  is any  $\mathcal{N}$ -substitution. By (P1) and (P2), we get:

**Theorem 1 (Soundness).** *Let  $\mathcal{N}$  be a finite subset of  $\text{NI}$  and let  $\pi : \Gamma \vdash K$  be a proof of  $\mathcal{ND}_c$  over  $\mathcal{L}_{\mathcal{N}}$  such that the formulas in  $\Gamma \cup \{K\}$  are closed. Then:*

- (i)  $\Gamma \models K$ .
- (ii) For every model  $\mathcal{M}$  and  $\bar{\gamma} \in \text{IT}_{\mathcal{N}}(\Gamma)$ ,  $\mathcal{M} \triangleright \langle \bar{\gamma} \rangle \Gamma$  implies  $\mathcal{M} \triangleright \langle \Phi_{\mathcal{N}}^\pi(\bar{\gamma}) \rangle K$ .

To conclude this section we give an example of the information one can extract from a proof using Theorem 1.

<sup>1</sup> We remark that, by the side condition on  $p$ ,  $(\sigma[c/p])\Gamma_2 = \sigma\Gamma_2$  and  $(\sigma[c/p])K = \sigma K$ .

<sup>2</sup> By the side condition on  $p$ ,  $(\sigma[c/p])\Gamma = \sigma\Gamma$  and  $(\sigma[c/p])t : \forall R.A = \sigma t : \forall R.A$ .

*Example 2.* Let us consider the knowledge base of Example 1. We can build a proof

$$\pi : \mathcal{T} \vdash \forall(\neg\text{FOOD} \sqcup \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}))$$

in  $\mathcal{ND}_c$ , namely a proof of  $\text{FOOD} \sqsubseteq \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}$  from  $\mathcal{T}$ . The proof  $\pi$  is

$$\frac{\frac{Ax_1}{y : \neg\text{FOOD} \sqcup \exists\text{goesWith}.\text{COLOR}} \forall\cup\text{E} \quad \frac{\frac{[y : \neg\text{FOOD}]}{K} \sqcup\text{I} \quad \frac{Ax_2 [y : \exists\text{goesWith}.\text{COLOR}]}{\vdots \pi_1} K}{K \equiv y : \neg\text{FOOD} \sqcup \exists\text{goesWith}.\text{COLOR}} \sqcup\text{E}}{\forall(\neg\text{FOOD} \sqcup \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE})} \forall\cup\text{I}}$$

where  $\pi_1$  is the proof

$$\frac{\frac{\frac{Ax_2 [z : \text{COLOR}]}{\vdots \pi_2} z : \text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}}{[(y, z) : \text{goesWith}]} \exists\text{I}}{y : \exists\text{goesWith}.\text{COLOR} \quad y : \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}} \exists\text{E}}{\frac{y : \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}}{y : \neg\text{FOOD} \sqcup \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}} \sqcup\text{I}}$$

and  $\pi_2$  is the proof

$$\frac{\frac{Ax_2}{z : \neg\text{COLOR} \sqcup \exists\text{isColorOf}.\text{WINE}} \forall\cup\text{E} \quad \frac{\perp}{H} \perp\text{E} \quad \frac{z : \text{COLOR} [z : \neg\text{COLOR}]}{z : \text{COLOR} [z : \exists\text{isColorOf}.\text{WINE}]} \perp\text{I}}{\frac{H}{H \equiv z : \text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}} \sqcup\text{E}} \sqcap\text{I}$$

Note that individual names do not occur in  $\pi$ . Let  $\mathcal{M}_W$ ,  $\gamma_1$  and  $\gamma_2$  be defined as in Example 1. Since  $\mathcal{M}_W \triangleright \langle (\gamma_1, \gamma_2) \rangle \mathcal{T}$ , by Theorem 1 we get that  $\Phi_{\text{WNI}}^\pi(\gamma_1, \gamma_2)$  is a function  $\psi$  such that, for every  $c \in \text{WNI}$ :

$$\mathcal{M}_W \triangleright \langle \psi(c) \rangle c : \neg\text{FOOD} \sqcup \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE}$$

If  $\psi(c) = (1, \text{tt})$ , then  $c^{\mathcal{M}_W} \notin \text{FOOD}^{\mathcal{M}_W}$  ( $c$  is not a food). Otherwise,  $\psi(c)$  has the form  $(2, (d, (\text{tt}, (e, \text{tt}))))$ , meaning that  $(c^{\mathcal{M}_W}, d^{\mathcal{M}_W}) \in \text{goesWith}^{\mathcal{M}_W}$  (food  $c$  goes with color  $d$ ) and  $(d^{\mathcal{M}_W}, e^{\mathcal{M}_W}) \in \text{isColorOf}^{\mathcal{M}_W}$  (wine  $e$  has color  $d$ ), hence we have found a wine  $e$  to pair with  $c$ . In our example we get

$$\begin{aligned} \psi(\text{meat}) &= (2, (\text{red}, (\text{tt}, (\text{barolo}, \text{tt})))) \\ \psi(\text{fish}) &= (2, (\text{white}, (\text{tt}, (\text{chardonnay}, \text{tt})))) \end{aligned}$$

and  $\psi(c) = (1, \text{tt})$  for all the other  $c \in \text{WNI}$ .

Note that, since in our setting negation has not a constructive meaning, the choice of axioms is crucial to extract information. As an example, if we replace  $Ax_1$  with the classically equivalent formula  $\forall(\neg(\text{FOOD} \sqcap \neg\exists\text{goesWith}.\text{COLOR}))$ , we cannot build a proof of the formula  $\forall(\neg\text{FOOD} \sqcup \exists\text{goesWith}.\text{COLOR} \sqcap \exists\text{isColorOf}.\text{WINE})$ .

To conclude this section we remark that, along the lines of the previous example, Theorem 1 allows us to interpret a proof of a “goal” as a program to solve it. We defer to a future work a deeper discussion on the notion of “solvable goal”.

## 4 Conclusions

First of all, we compare information-terms semantics with the classical one. Let  $\overline{\mathcal{ALC}}$  denote the set of formulas  $K$  such that  $\mathcal{M} \models K$ , and let  $\overline{\mathcal{ALC}}_c$  be the set of formulas  $K$  such that there exists a proof  $\pi : \vdash K$  in  $\mathcal{ND}_c$ . By Theorem 1,  $\overline{\mathcal{ALC}}_c \subseteq \overline{\mathcal{ALC}}$ . However, one can easily prove that the classically valid formula  $x : D \sqcup \neg D$  is not provable in  $\mathcal{ND}_c$ ; hence,  $\overline{\mathcal{ALC}}_c \neq \overline{\mathcal{ALC}}$ . We remark that in general a constructive explanation of  $x : D \sqcup \neg D$  cannot be given. If we replace the rule  $\perp E$  of  $\mathcal{ND}_c$  with the classical rule of *reductio ad absurdum*, the set of provable formulas of the resulting calculus coincides with  $\overline{\mathcal{ALC}}$ ; obviously, the computational interpretation of proofs provided by Theorem 1 cannot be extended to such a rule. Finally, we remark that our constructive semantics and  $\mathcal{ND}_c$  can be exploited to handle intuitionistic implication and stronger negation (as discussed in [6]). As for future works, we are developing an extension of  $\mathcal{ND}_c$  sound and complete with respect to the information-terms semantics for  $\mathcal{ALC}$ . Moreover, we plan to extend our framework to treat other description logics.

## References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language, 1991.
3. V. de Paiva. Constructive description logics: what, why and how. Technical report, Xerox Parc, 2005.
4. M. Ferrari, C. Fiorentini, and M. Ornaghi. Extracting exact time bounds from logical proofs. In A. Pettorossi, editor, *LOPSTR 2001*, volume 2372 of *LNCS*, pages 245–265. Springer-Verlag, 2002.
5. M. Hofmann. Proof-theoretic approach to description-logic. In *LICS*, pages 229–237. IEEE Computer Society, 2005.
6. K. Kaneiwa. Negations in description logic - contraries, contradictories, and sub-contraries. In *ICCS'05*, pages 66–79. Kassel University Press, 2005.
7. P. Miglioli, U. Moscato, M. Ornaghi, and G. Usberti. A constructivism based on classical truth. *Notre Dame Journal of Formal Logic*, 30(1):67–90, 1989.
8. M. Ornaghi, M. Benini, M. Ferrari, C. Fiorentini, and A. Momigliano. A Constructive Modeling Language for Object Oriented Information Systems. In *CLASE'05*, volume 153 of *ENTCS*, pages 55–75, 2006.
9. D. Prawitz. *Natural Deduction*. Almqvist and Winksell, 1965.
10. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
11. A. S. Troelstra. From constructivism to computer science. *TCS*, 211(1-2):233–252, 1999.

## MASTRO-I: Efficient integration of relational data through DL ontologies

Diego Calvanese<sup>1</sup>, Giuseppe De Giacomo<sup>2</sup>, Domenico Lembo<sup>2</sup>,  
Maurizio Lenzerini<sup>2</sup>, Antonella Poggi<sup>2</sup>, Riccardo Rosati<sup>2</sup>

<sup>1</sup> Faculty of Computer Science  
Free University of Bozen-Bolzano  
calvanese@inf.unibz.it

<sup>2</sup> Dip. di Informatica e Sistemistica  
SAPIENZA Università di Roma  
lastname@dis.uniroma1.it

### 1 Introduction

The goal of data integration is to provide a uniform access to a set of heterogeneous data sources, freeing the user from the knowledge about where the data are, how they are stored, and how they can be accessed. The problem of designing effective data integration solutions has been addressed by several research and development projects in the last years. One of the outcomes of this research work is a clear conceptual architecture for data integration<sup>1</sup>. According to this architecture [9], the main components of a data integration system are the global schema, the sources, and the mapping. Thus, a data integration system is seen as a triple  $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ , where:

- $\mathcal{G}$  is the *global schema*, providing both a conceptual representation of the application domain, and a reconciled, integrated, and virtual view of the underlying sources.
- $\mathcal{S}$  is the *source schema*, i.e., the schema of the sources where real data are stored.
- $\mathcal{M}$  is the *mapping* between  $\mathcal{G}$  and  $\mathcal{S}$ , constituted by a set of assertions establishing the connection between the elements of the global schema and those of the source schema. Two basic approaches have been proposed in the literature. The first approach, called *global-as-view* (or simply GAV), focuses on the elements of the global schema, and associates to each of them a view (query) over the sources. On the contrary, in the second approach, called *local-as-view* (or simply LAV), the focus is on the sources, in the sense that a view (query) over the global schema is associated to each of them.

We use the term “data integration management system” to denote a software tool supporting the conceptual architecture described above. Among the various services to be provided by a data integration management system, we concentrate on query answering: Queries are posed in terms of the global schema, and are to be answered by suitably reasoning on the global schema, and exploiting the mappings to access data at the sources.

Data integration is still one of the major challenges in Information Technology. One of the reasons is that large amounts of heterogeneous data are nowadays available within an organization, but these data have been often collected and stored by different

<sup>1</sup> Here we are concerned with the so-called centralized data integration. Other architectures, e.g. [4], are outside the scope of this paper.

applications and systems. Therefore, the need of accessing data by means of flexible and unified mechanisms is becoming more and more important. On the other hand, current commercial data integration tools have several drawbacks. In particular, none of them realizes the goal of describing the global schema independently from the sources. In particular, these tools do not allow for specifying integrity constraints in the global schema, and this implies that the global schema is a sort of data structure for accommodating a reconciled view of the source data, rather than a faithful description of the application domain. It follows that current state-of-the-art data integration tools do not support the conceptual architecture mentioned above.

In this paper, we present a comprehensive approach to, and a complete management system for ontology-based data integration. The system, called MASTRO-I, is based on the following principles:

- The system fully adheres to the conceptual architecture developed by the scientific community.
- The global schema is specified in terms of an ontology, specifically in terms of a TBox expressed in a tractable Description Logics, namely  $DL-Lite_A$ . So, our approach conforms to the view that the global schema of a data integration system can be profitably represented by an ontology, so that clients can rely on a shared conceptualization when accessing the services provided by the system.
- The source schema is the schema of a relational database.
- The mapping language allows for expressing GAV *sound* mappings between the sources and the global schema. A GAV sound mapping specifies that the extension of a source view provides a subset of the tuples satisfying the corresponding element of the global schema.

Moreover, the mapping language has specific mechanisms for addressing the so-called *impedance mismatch* problem. This problem arises from the fact that, while the data sources store values, the instances of concepts in the ontology (global schema) are objects, each one denoted by an identifier (e.g., a constant in logic), not to be confused with any data item.

MASTRO-I is based on the system QUONTO [1], a reasoner for  $DL-Lite_A$ , and is coupled with DB2 Information Integrator, the IBM tool for data federation <sup>2</sup>.

We point out that our proposal is not the first one advocating the use of ontologies in data integration (see, for example, [7, 2]). However, to the best of our knowledge, MASTRO-I is the first data integration management system addressing simultaneously the following aspects:

- providing a solution to the impedance mismatch problem;
- answering unions of conjunctive queries posed to the global schema according to a method which is sound and complete with respect to the semantics of the ontology;
- careful design of the various languages used in the system, resulting in a very efficient technique (LOGSPACE with respect to data complexity) which reduces query answering to standard SQL query evaluation over the sources.

One might wonder whether the expressive power of the data integration framework underlying MASTRO-I can be improved. We answer this question by showing

<sup>2</sup> <http://www-128.ibm.com/developerworks/db2/zones/db2ii/>

that even very slight extensions of the expressive abilities of MASTRO-I in modeling the three components of a data integration system lead beyond the LOGSPACE complexity bound.

We end this section by pointing out that MASTRO-I addresses the problem of data integration, and not the one of schema or ontology integration. In other words, MASTRO-I is not concerned with the task of building the ontology representing the global schema starting from the source schema, or from other ontologies. This task, which is strongly related to other important problems, such as database schema integration [3], ontology alignment, matching, merging, or integration, are outside the scope of MASTRO-I.

## 2 MASTRO-I: The data integration framework

In this section we instantiate the conceptual architecture for data integration systems introduced in Section 1, by describing the form of the global schema, the source schema, and the mapping for data integration systems managed by MASTRO-I.

**The global schema.** Global schemas managed by MASTRO-I are given in terms of TBoxes expressed in  $DL-Lite_{\mathcal{A}}$  [5], a DL of the  $DL-Lite$  family. Besides the use of concepts and roles, denoting sets of objects and binary relations between objects, respectively,  $DL-Lite_{\mathcal{A}}$  allows one to use value-domains, a.k.a. concrete domains, denoting unbounded sets of (data) values, and concept attributes, denoting binary relations between objects and values<sup>3</sup>. In particular, the value-domains that we consider here are those corresponding to unbounded (i.e., value-domains with an unbounded size) RDF data types, such as integers, real, strings, etc.

To describe  $DL-Lite_{\mathcal{A}}$ , we first introduce the DL  $DL-Lite_{\mathcal{FR}}$ , which combines the main features of two DLs presented in [6], called  $DL-Lite_{\mathcal{F}}$  and  $DL-Lite_{\mathcal{R}}$ , respectively. We use the following notation:  $A$  denotes an *atomic concept*,  $B$  a *basic concept*,  $C$  a *general concept*, and  $\top_C$  the *universal concept*;  $E$  denotes a basic value-domain, i.e., the range of an attribute,  $T_1, \dots, T_n$  denote the  $n$  pairwise disjoint unbounded RDF data types used in our logic, and  $F$  denotes a *general value-domain*, which can be either an unbounded RDF data type  $T_i$  or the *universal value-domain*  $\top_D$ ;  $P$  denotes an *atomic role*,  $Q$  a *basic role*, and  $R$  a *general role*;  $U_C$  denotes an *atomic attribute*, and  $V_C$  a *general attribute*. Given an attribute  $U_C$ , we call *domain* of  $U_C$ , denoted by  $\delta(U_C)$ , the set of objects that  $U_C$  relates to values, and we call *range* of  $U_C$ , denoted by  $\rho(U_C)$ , the set of values related to objects by  $U_C$ .

We are now ready to define  $DL-Lite_{\mathcal{FR}}$  expressions as follows.

- Basic and general concept expressions:

$$B ::= A \mid \exists Q \mid \delta(U_C) \quad C ::= \top_C \mid B \mid \neg B \mid \exists Q.C$$

- Basic and general value-domain expressions:

$$E ::= \rho(U_C) \quad F ::= \top_D \mid T_1 \mid \dots \mid T_n$$

<sup>3</sup> The logic discussed in [5] is actually more expressive than  $DL-Lite_{\mathcal{A}}$ , since it includes role attributes, user-defined domains, as well as inclusion assertions over such domains.

- General attribute expressions:

$$V_C ::= U_C \mid \neg U_C$$

- Basic and general role expressions:

$$Q ::= P \mid P^- \quad R ::= Q \mid \neg Q$$

A  $DL\text{-Lite}_{\mathcal{FR}}$  TBox allows one to represent intensional knowledge by means of assertions of the following forms:

- *Inclusion assertions*:  $B \sqsubseteq C$  (concept inclusion assertion);  $Q \sqsubseteq R$  (role inclusion assertion);  $E \sqsubseteq F$  (value-domain inclusion assertion);  $U_C \sqsubseteq V_C$  (attribute inclusion assertion). A concept inclusion assertion expresses that a (basic) concept  $B$  is subsumed by a (general) concept  $C$ . Analogously for the other types of inclusion assertions.
- *Functionality assertions* on atomic attributes or basic roles: (funct  $I$ ), where  $I$  denotes either an atomic attribute or a basic role.

$DL\text{-Lite}_A$  TBoxes are  $DL\text{-Lite}_{\mathcal{FR}}$  TBoxes with suitable limitations in the combination of  $DL\text{-Lite}_{\mathcal{FR}}$  TBox assertions. To describe such limitations we first introduce some preliminary notions. An atomic attribute  $U_C$  (resp. a basic role  $Q$ ) is called an *identifying property in a TBox  $\mathcal{T}$* , if  $\mathcal{T}$  contains a functionality assertion (funct  $U_C$ ) (resp. (funct  $Q$ ) or (funct  $Q^-$ )). Then, an atomic attribute or a basic role is called *primitive in  $\mathcal{T}$* , if it does not appear positively in the right-hand side of an inclusion assertion of  $\mathcal{T}$ , and it does not appear in an expression of the form  $\exists Q.C$  in  $\mathcal{T}$ .

Then, a  $DL\text{-Lite}_A$  TBox is a  $DL\text{-Lite}_{\mathcal{FR}}$  TBox  $\mathcal{T}$  satisfying the condition that every identifying property in  $\mathcal{T}$  is primitive in  $\mathcal{T}$ .

Roughly speaking, in our logic, *identifying properties cannot be specialized*, i.e., they cannot be used positively in the right-hand side of inclusion assertions. As shown in [5], reasoning over a  $DL\text{-Lite}_A$  knowledge base (constituted by a TBox and an ABox) is tractable. More precisely, TBox reasoning is in PTIME and query answering is in LOGSPACE w.r.t. data complexity, i.e., the complexity measured in the size of the ABox only (whereas query answering for  $DL\text{-Lite}_{\mathcal{FR}}$  is PTIME-hard). Thus,  $DL\text{-Lite}_A$  presents the same computational behavior of all DLs of the  $DL\text{-Lite}$  family, and therefore is particularly suited for integration of large amounts of data.

**The source schema.** The source schema in MASTRO-I is a flat relational database schema, representing the schemas of all the data sources. Since MASTRO-I integrates data sources that are distributed, possibly heterogeneous, and not necessarily in relational format, the source schema may in fact be obtained by wrapping a set of physical sources. Indeed, MASTRO-I is coupled with the IBM DB2 Information Integrator, and relies on both the wrapping facilities provided by this data federation tool, and on its ability to answer queries posed to a set of distributed physical sources.

**The mapping.** The mapping in MASTRO-I establishes the relationship between the source schema and the global schema, thus specifying how data stored at the sources are linked to the instances of the concepts and the roles in the global schema. To this aim, the mapping specification takes suitably into account the impedance mismatch problem, i.e., the mismatch between the way in which data is (and can be) represented in a data

source, and the way in which the corresponding information is rendered through the global schema.

The MASTRO-I mapping assertions keep data value constants separate from object identifiers, and construct identifiers as (logic) terms over data values. More precisely, object identifiers in MASTRO-I are *terms* of the form  $f(d_1, \dots, d_n)$ , where  $f$  is a function symbol of arity  $n > 0$ , and  $d_1, \dots, d_n$  are data values stored at the sources. Note that this idea traces back to the work done in deductive object-oriented databases [8].

We detail below the above ideas. The mapping in MASTRO-I is a set of assertions of the following forms:

- *Typing mapping assertions*:  $\Phi(v) \rightsquigarrow T_i(v)$ , where  $\Phi$  is a query over the source schema  $\mathcal{S}$  denoting the projection of one relation over one of its attributes,  $T_i$  is one of the  $DL-Lite_{\mathcal{A}}$  data types, and  $v$  is a variable,
- *Data-to-ontology mapping assertions*:  $\Phi(v) \rightsquigarrow P(t, v')$ , where  $\Phi$  is a first-order logic (FOL) query over the source schema  $\mathcal{S}$ ,  $P$  is an atom in the global schema  $\mathcal{G}$ ,  $v, v'$  are variables such that  $v' \subseteq v$  and  $t$  are *variable object terms*, i.e., terms having the form  $f(v'')$ , where  $f$  is a function symbol, and  $v''$  are variables such that  $v'' \subseteq v$ .

Typing mapping assertions are used to assign appropriate  $DL-Lite_{\mathcal{A}}$  types to values occurring in the tuples at the sources. Basically, these assertions are used for interpreting the values stored at the sources in terms of the types used in the global schema. Data-to-ontology, on the other hand, are used to map source relations (and the tuples they store), to global concepts, roles, and attributes (and the objects and the values that constitute their instances).

### 3 MASTRO-I: Semantics

We now illustrate the semantics of a data integration system managed by MASTRO-I.

Let  $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  be a data integration system. The general idea is to start with a database  $D$  for the source schema  $\mathcal{S}$ , i.e., the extensions of the data sources, and define the semantics of  $\mathcal{J}$  as the set of interpretations for  $\mathcal{G}$  that both satisfy the TBox assertions of  $\mathcal{G}$ , and satisfy the mapping assertions in  $\mathcal{M}$  with respect to  $D$ .

The above informal definition makes use of different notions that we detail below.

- First, the notion of interpretation for  $\mathcal{G}$  is the usual one in DL. An *interpretation*  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  for  $\mathcal{G}$  consists of an interpretation domain  $\Delta^{\mathcal{I}}$  and an *interpretation function*  $\cdot^{\mathcal{I}}$ .  $\Delta^{\mathcal{I}}$  is the disjoint union of the domain of objects  $\Delta_O^{\mathcal{I}}$ , and the domain of values  $\Delta_V^{\mathcal{I}}$ , while the interpretation function  $\cdot^{\mathcal{I}}$  assigns the standard formal meaning to all expressions and assertions of the logic  $DL-Lite_{\mathcal{A}}$  (see [5]). The only aspect which is not standard here is the need of dealing with objects denoted by terms (see previous section). To this end, we now introduce two disjoint alphabets, called  $\Gamma_O$  and  $\Gamma_V$ , respectively. Symbols in  $\Gamma_O$  are called object terms, and are used to denote objects, while symbols in  $\Gamma_V$ , called value constants, are used to denote data values. More precisely,  $\Gamma_O$  is built starting from  $\Gamma_V$  and a set  $\Lambda$  of function symbols of any arity (possibly 0), as follows: If  $f \in \Lambda$ , the arity of  $f$  is  $n$ , and  $d_1, \dots, d_n \in \Gamma_V$ , then  $f(d_1, \dots, d_n)$  is a term in  $\Gamma_O$ , called *object term*. In other words, object terms are either functional terms of arity 0, called object constants, or terms constituted

by a function symbol applied to data value constants. We are ready to state how the interpretation function  $\cdot^{\mathcal{I}}$  treats  $\Gamma_V$  and  $\Gamma_O$ :  $\cdot^{\mathcal{I}}$  simply assigns a different value in  $\Delta_V^{\mathcal{I}}$  to each symbol in  $\Gamma_V$ , and a different element of  $\Delta_O^{\mathcal{I}}$  to every object term (not only object constant) in  $\Gamma_O$ . In other words, *DL-Lite<sub>A</sub>* enforces the unique name assumption on both value constants and object terms.

- To the aim of describing the semantics of mapping assertions with respect to a database  $D$  for the source schema  $\mathcal{S}$ , we first assume that all data values stored in the database  $D$  belong to  $\Gamma_V^4$ . Then, if  $q$  is a query over the source schema  $\mathcal{S}$ , we denote by  $ans(q, D)$  the set of tuples obtained by evaluating the query  $q$  over the database  $D$  (if  $q$  has not distinguished variables, then  $ans(q, D)$  is a boolean). Finally, we introduce the notion of ground instance of a formula. Let  $\gamma$  be a formula with free variables  $\mathbf{x} = (x_1, \dots, x_n)$ , and let  $\mathbf{s} = (s_1, \dots, s_n)$  be a tuple of elements in  $\Gamma_V \cup \Gamma_O$ . A ground instance  $\gamma[\mathbf{x}/\mathbf{s}]$  of  $\gamma$  is obtained from  $\gamma$  by substituting every occurrence of  $x_i$  with  $s_i$ .

We are now ready to specify the semantics of mapping assertions. We say that an interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  satisfies the mapping assertion  $\varphi \rightsquigarrow \psi$  with respect to  $D$ , if for every ground instance  $\varphi[\mathbf{x}/\mathbf{s}] \rightsquigarrow \psi[\mathbf{x}/\mathbf{s}]$  of  $\varphi \rightsquigarrow \psi$ , we have that  $ans(\varphi[\mathbf{x}/\mathbf{s}], D) = true$  implies  $\psi[\mathbf{x}/\mathbf{s}]^{\mathcal{I}} = true$  (where, for a ground atom  $p(\mathbf{t})$ , with  $\mathbf{t} = (t_1, \dots, t_n)$  a tuple of object terms, we have that  $p(\mathbf{t})^{\mathcal{I}} = true$  if  $(t_1^{\mathcal{I}}, \dots, t_n^{\mathcal{I}}) \in p^{\mathcal{I}}$ ). Note that the above definition formalizes the notion of sound mapping, as it treats each mapping assertion as an implication.

- With the above notion in place, we can simply define the semantics of  $\mathcal{J}$  with respect to  $D$  as follows:

$$sem_D(\mathcal{J}) = \{ \mathcal{I} \mid \mathcal{I} \text{ is a model of } \mathcal{G}, \text{ and } \mathcal{I} \text{ satisfies all assertions in } \mathcal{M} \text{ wrt } D \}$$

As we said in the introduction, in this paper we are mainly interested in the problem of answering unions of conjunctive queries (UCQs) posed to the global schema. The semantics of query answering is given in terms of certain answers to the query, defined as follows. Given a data integration system  $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ , and a database  $D$  for  $\mathcal{S}$ , the set of *certain answers* to the query  $q(\mathbf{x})$  over  $\mathcal{G}$  is the set (denoted by  $ans(q, \mathcal{J}, D)$ ) of all tuples  $\mathbf{t}$  of elements of  $\Gamma_V \cup \Gamma_O$  such that  $\mathcal{I} \models_{FOL} q[\mathbf{x}/\mathbf{t}]$  for every  $\mathcal{I} \in sem_D(\mathcal{J})$  (notice that  $q[\mathbf{x}/\mathbf{t}]$  is a boolean UCQ, i.e., a FOL sentence).

## 4 Query answering

In this section, we sketch our query answering technique (more details can be found in [10]). Consider a data integration system  $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  and a database  $D$  for  $\mathcal{S}$ .

We start with the following observation. Suppose we evaluate (over  $D$ ) the queries in the left-hand sides of the mapping assertions, and we materialize accordingly the corresponding assertions in the right-hand sides. This would lead to a set of ground assertions, that can be considered as a *DL-Lite* ABox, denoted by  $\mathcal{A}^{\mathcal{M}, D}$ . It can be shown that query answering over  $\mathcal{J}$  can be reduced to query answering over the *DL-Lite<sub>A</sub>* knowledge base constituted by the TBox  $\mathcal{G}$  and the ABox  $\mathcal{A}^{\mathcal{M}, D}$ . However, due to the materialization of  $\mathcal{A}^{\mathcal{M}, D}$ , the query answering algorithm resulting from this approach

<sup>4</sup> We could also introduce suitable conversion functions in order to translate values stored at the sources into value constants in  $\Gamma_V$ , but we do not deal with this issue here.

would be polynomial in the size of  $D$ . On the contrary, our idea is to avoid any ABox materialization, but rather answer  $Q$  by reformulating it into a new query that can be afterwards evaluated directly over the database  $D$ . This can be achieved by following three steps, called *rewriting*, *unfolding* and *evaluation*.

**Query rewriting.** Given a UCQ  $Q$  over a data integration system  $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ , and a database  $D$  for  $\mathcal{S}$ , the rewriting step computes a UCQ  $Q'$  over  $\mathcal{J}$ , where the assertions of  $\mathcal{G}$  are compiled in. It can be shown [10] that  $Q'$  is such that  $ans(Q', \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle, D) = ans(Q, \mathcal{J}, D)$ , i.e. rewriting allows to get rid of  $\mathcal{G}$ . Moreover, the rewriting procedure does not depend on  $D$ , runs in polynomial time in the size of  $\mathcal{G}$ , and returns a query  $Q'$  whose size is at most exponential in the size of  $Q$ .

**Unfolding.** Given a UCQ  $Q'$  over  $\mathcal{J}$ , this step computes, by using logic programming technology, an SQL query  $Q''$  over the source schema  $\mathcal{S}$ , that possibly returns object terms. It can be shown [5, 10] that  $Q''$  is such that  $ans(Q'', D) = ans(Q', \langle \emptyset, \mathcal{S}, \mathcal{M} \rangle, D)$ , i.e. unfolding allows to get rid of  $\mathcal{M}$ . Moreover, the unfolding procedure does not depend on  $D$ , runs in polynomial time in the size of  $\mathcal{M}$ , and returns a query  $Q''$ , whose size is at most exponential in the size of  $Q'$ .

**Evaluation.** The evaluation step consists in simply delegating the evaluation of  $Q''$  to the data federation tool managing the data sources. Formally, such a tool returns the set  $ans(Q'', D)$ , i.e. the set of tuples obtained from the evaluation of  $Q''$  over  $D$ .

From the above discussion, we have the following:

**Theorem 1.** *Let  $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  be a MASTRO-I data integration system, and  $D$  a database for  $\mathcal{S}$ . Answering a UCQ over  $\mathcal{J}$  with respect to  $D$  can be reduced to the evaluation of an SQL query over  $D$ , and is LOGSPACE in the size of  $D$ .*

Finally, we remark that we are implicitly assuming that the database  $D$  for  $\mathcal{S}$  is consistent with the data integration system  $\mathcal{J}$ , i.e.,  $sem_D(\mathcal{J})$  is non-empty. Notably, checking consistency can also be reduced to sending appropriate SQL queries to the sources [5, 10].

## 5 Extending the data integration framework

In this section we study whether the data integration setting presented above can be extended while keeping the same complexity of query answering. In particular, we investigate possible extensions for all the three components  $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  of the system.

**Extensions to  $DL-Lite_{\mathcal{A}}$ .** With regard to the logic used to express the global schema  $\mathcal{G}$ , the results in [6] already imply that it is not possible to go beyond  $DL-Lite_{\mathcal{A}}$  (at least by means of the usual DL constructs) and at the same time keep the data complexity of query answering within LOGSPACE. Here we consider the possibility of removing the unique name assumption (UNA), i.e., the assumption that, in every interpretation of a data integration system  $\mathcal{J}$ , both two distinct value constants, and two distinct object terms denote two different domain elements. Unfortunately, this leads query answering out of LOGSPACE. This result can be proved by a reduction from Graph Reachability to instance checking in  $DL-Lite_{\mathcal{F}}$  [6], i.e., query answering for a boolean query whose body is a single instantiated atom, over a DL that is a subset of  $DL-Lite_{\mathcal{A}}$ .

**Theorem 2.** *Let  $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  be a MASTRO-I data integration system extended by removing the UNA, and  $D$  a database for  $\mathcal{S}$ . Answering a UCQ over  $\mathcal{J}$  with respect to  $D$  is NLOGSPACE-hard in the size of  $D$ .*

**Different source schemas.** Although MASTRO-I is currently able to deal with relational sources only, it is not hard to see that all the results presented in this paper apply also if we consider data at the sources structured according to a different data model (e.g. XML). Obviously, depending on the specific data model, we have to resort to a suitable query language for expressing the source queries appearing in the mapping assertions. To adhere to our framework, the only constraint on this language is that it is able to extract tuples of values from the sources, a constraint that is trivially satisfied by virtually all query languages used in practice.

**Extensions to the mapping language.** As for the language used to express the mapping  $\mathcal{M}$ , we investigate the extension of the mapping language to allow for GLAV assertions, i.e., assertions that relate CQs over the sources to CQs over the global schema. Such assertions are therefore an extension of both GAV and LAV mappings. The result we present is that, even with LAV mappings only, instance checking and query answering are no more in LOGSPACE wrt data complexity.

**Theorem 3.** *Let  $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  be a MASTRO-I data integration system extended with LAV mapping assertions, and  $D$  a database for  $\mathcal{S}$ . Answering a UCQ over  $\mathcal{J}$  with respect to  $D$  is NLOGSPACE-hard in the size of  $D$ .*

The above result can be proved again by a reduction from Graph Reachability to instance checking in *DL-Lite<sub>F</sub>*.

**Acknowledgments.** This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU under contract number FP6-7603, by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant, and by MIUR FIRB 2005 project “Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet” (TOCAI.IT).

## References

1. A. Acciari, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: QUerying ONTOlogies. In *Proc. of AAAI 2005*, pages 1670–1671.
2. B. Amann, C. Beerli, I. Fundulaki, and M. Scholl. Ontology-based integration of XML web resources. In *Proc. of ISWC 2002*, pages 117–131.
3. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
4. P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zahravey. Data management for peer-to-peer computing: A vision. In *Proc. of WebDB 2002*.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic DL-Lite<sub>A</sub>. In *Proc. of OWLED 2006*.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270.
7. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.
8. R. Hull. A survey of theoretical research on typed complex database objects. In J. Paredaens, editor, *Databases*, pages 193–256. Academic Press, 1988.
9. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246.
10. A. Poggi. *Structured and Semi-Structured Data Integration*. PhD thesis, Dip. di Inf. e Sist., Univ. di Roma “La Sapienza”, 2006.

## A Proof Theory for DL-Lite\*

Diego Calvanese, Evgeny Kharlamov, Werner Nutt

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy  
{calvanese, kharlamov, nutt}@inf.unibz.it

**Abstract.** In this work we propose an alternative approach to inference in DL-Lite, based on a reduction to reasoning in an extension of function-free Horn Logic (EHL). We develop a calculus for EHL and prove its soundness and completeness. We also show how to achieve decidability by means of a specific strategy, and how alternative strategies can lead to improved results in specific cases. On the one hand, we propose a strategy that mimics the query-answering technique based on first computing a query rewriting and then evaluating it. On the other hand, we propose strategies that allow one to anticipate the grounding of atoms, and that might lead to better performance in the case where the size of the TBox is not dominated by the size of the data.

### 1 Introduction

The description logics (DLs) of the *DL-Lite* family [1, 2] have been proposed recently as DLs providing a good compromise between expressive power and computational complexity of inference. Indeed, *DL-Lite* and its variants are able to capture the fundamental features of conceptual modeling formalisms, while still keeping the basic reasoning polynomial in the size of the whole DL knowledge base (KB), and LOGSPACE in the size of the data. Notably, such reasoning services include answering conjunctive queries (CQs) formulated over a KB. Moreover, techniques have been developed to perform query answering by leveraging database technology: the ABox is actually stored in a relational database (DB), and (after suitable pre-processing) the query is answered by exploiting the relational DB engine. This approach ensures scalability of query answering over DL KBs to billions of data items. More precisely, the approach for query answering proposed in [1] is actually divided in three phases: (1) *Consistency* of the knowledge base w.r.t. functionality and (pre-processed) disjointness assertions in the TBox is verified by posing appropriate queries to the DB (i.e., the ABox) only (and independently on the actual query); (2) The user query is *rewritten* into a new query using the inclusion assertions in the TBox; (3) The rewritten query is shipped to the DB, and the returned tuples constitute the answer returned to the user.

In this work, we still rely on Phase (1), but take a closer look at Phases (2) and (3), and at the underlying formal properties of *DL-Lite*. Specifically, we exploit the similarity of TBox inclusion assertions and of ABox membership assertions to clauses in an extension of function-free Horn Logic (which we call *EHL*), in which existentially

\* Research supported by the EU FET project TONES (Thinking ONtologiES, contract FP6-7603), and by the PRIN 2006 project NGS (New Generation Search), funded by MIUR.

quantified variables may appear in the clauses. We develop a sound and complete calculus for EHL that bears similarity to resolution [3], but is equipped with a specific rule to handle existentially quantified variables. In three cases, we show how to obtain complete algorithms for query answering by imposing control strategies on the calculus. The general algorithm ensures termination by using loop detection, exploiting the fact that the number of non-isomorphic clauses that can be generated from a query goal using the knowledge base is bounded. A second control strategy mimics the perfect reformulation algorithm in [4], by strictly separating the derivation steps that correspond to operations in Phases (2) and (3). Finally, the third algorithm prunes the search space in a way that is analogous to SLD-resolution in Logic Programming. Moreover, the third algorithm prunes the space by detecting failure derivations in advance.

We obtained the results for  $DL-Lite_{\mathcal{F}}$  only. However, similar results can be obtained for other DLs in the  $DL-Lite$  family, such as  $DL-Lite_{\mathcal{R}}$  [2].

## 2 $DL-Lite_{\mathcal{F}}$

**Syntax, Semantics of  $DL-Lite_{\mathcal{F}}$  and Queries.** Let  $AC = \{A_1, \dots, A_{|AC|}\}$  be a set of *atomic concepts*,  $AR = \{R_1, \dots, R_{|AR|}\}$  a set of *atomic roles*, and  $Const$  a countable set of *constants*. We inductively define  $DL-Lite_{\mathcal{F}}$  concepts in the following way: *basic concept*  $B \longrightarrow A \mid \exists R \mid \exists R^-$ , (general) *concept*  $C \longrightarrow B \mid \neg B \mid C_1 \sqcap C_2$ , where  $A \in AC$ ,  $R \in AR$ . With  $R^-$  we denote the *inverse* of the role  $R$ . In the following,  $A$  denotes an atomic concept,  $B$  a basic concept,  $C$  a concept, and  $R$  an atomic role.

A  $DL-Lite_{\mathcal{F}}$  knowledge base (KB)  $\mathcal{K}$  is constituted by a TBox (denoted as  $\mathcal{T}$ ) and an ABox (denoted as  $\mathcal{A}$ ). Each  $DL-Lite$  TBox consists of *inclusion assertions* of the form  $B \sqsubseteq C$  and *functionality assertions* of the form (funct  $R$ ) or (funct  $R^-$ ). An ABox consists of *membership assertions* of the form  $A(a)$ ,  $R(a, b)$ , where  $a, b$  are constants. Note that negation can occur only on the right side of inclusion assertions, and that an inclusion assertion  $B \sqsubseteq C_1 \sqcap C_2$  can be rewritten as a pair of inclusion assertions  $B \sqsubseteq C_1$  and  $B \sqsubseteq C_2$ . Therefore, in the following, we will assume w.l.o.g., that conjunction does not occur in the TBox, and we denote with  $Pos(\mathcal{K})$  the set of all inclusion assertions in  $\mathcal{K}$  without negation on the right hand side.

The semantics of  $DL-Lite_{\mathcal{F}}$  is defined in the usual way for DLs, by resorting to interpretations  $I = (\Delta, \cdot^I)$  over a fixed infinite countable domain  $\Delta$ . We just remark that (funct  $R$ ) is interpreted as functionality of role  $R$ . We assume that there is a bijection between  $Const$  and  $\Delta$  (i.e., we have *standard names*). Hence, we do not distinguish between the alphabet of constants  $Const$  and the domain  $\Delta$ . We define models for assertions and KBs in the usual way and say that a KB is *satisfiable* if it has a model.

We use the following rule based notation for defining *conjunctive queries* (CQs):

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} \text{ body}(\mathbf{x}; \mathbf{y}),$$

where  $\exists \mathbf{y} \text{ body}(\mathbf{x}; \mathbf{y})$  (also denoted as  $\text{body}(q)$ ) is a formula of the form  $\exists \mathbf{y} R_1(t_1, t'_1) \wedge \dots \wedge R_n(t_n, t'_n) \wedge A_1(t''_1) \wedge \dots \wedge A_k(t''_k)$ , where all  $R_i$  are binary and  $A_i$  unary predicate symbols, all  $t_i$  are either variables or constants and each variable that occurs in the conjunction is from  $\mathbf{x}$  or  $\mathbf{y}$ . The vectors  $\mathbf{x}$  and  $\mathbf{y}$  are called the *distinguished* and *non-distinguished* variables of  $q$ , respectively. If  $\mathbf{x}$  is empty, we call the query *boolean*. We denote as True the boolean query with no atoms in its body.

We denote the set of all constants that occur in  $\mathcal{K}$  as  $adom(\mathcal{K})$ . We say that  $q$  is a query over a KB  $\mathcal{K}$  if all the predicate symbols occurring in  $body(q)$  also occur in  $\mathcal{K}$  and the constants are from  $adom(\mathcal{K})$ . For a given KB  $\mathcal{K}$ , model  $I$  of  $\mathcal{K}$  and query  $q(\mathbf{x}) \leftarrow \exists \mathbf{y} body(\mathbf{x}; \mathbf{y})$  over  $\mathcal{K}$ , the set  $q(I)$  of answers of  $q$  over  $I$  is defined as:  $q(I) = \{\gamma \mathbf{x} \mid I \models \exists \mathbf{y} body(\gamma \mathbf{x}; \gamma \mathbf{y}), \gamma : Var \rightarrow \Delta\}$ .

The definition above says how to answer a query over a given model of a KB. Now we define how to answer a query over a KB itself. For this purpose we use the so-called *certain answers semantics*, i.e., for a given query  $q$  and a KB  $\mathcal{K}$ , the set  $q(\mathcal{K})$  of *certain answers* (or the *answer set*) of  $q$  over  $\mathcal{K}$  is defined as

$$q(\mathcal{K}) = \{\mathbf{c} \mid \mathbf{c} \in adom(\mathcal{K})^{|\mathbf{c}|} \text{ and } \mathbf{c} \in q(I), \text{ for every model } I \text{ of } \mathcal{K}\}.$$

**Reasoning.** Here we define query answering and discuss ways to perform it.

*Conjunctive query (CQ) answering* for a CQ  $q$  and a KB  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  is the task of finding  $q(\mathcal{K})$ . One can easily see that CQ answering is equivalent to finding all tuples  $\mathbf{c}$  of constants from  $adom(\mathcal{K})$  such that the entailment  $\mathcal{K} \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$  holds. It turns out (see the separation theorem in [4]) that  $\mathcal{K} \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$  holds if and only if  $\mathcal{K}$  is satisfiable and  $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$  holds. In order to check satisfiability of  $\mathcal{K}$  it is enough to verify that the minimal model of  $\mathcal{A}$  (the intersection of all models of  $\mathcal{A}$ ) satisfies all functional assertions of  $\mathcal{K}$  and all negative inclusion assertions entailed by  $\mathcal{K}$  [4].

In order to decide  $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$ , one needs to check that all models of the premises satisfy the conclusion. It turns out there may be infinitely many (possibly infinite) “different” models of  $\mathcal{K}$  [5]. Hence, at a first glance, it is not clear at all whether query answering is decidable.

It turns out [5] that any satisfiable *DL-Lite<sub>F</sub>* KB meets the so-called *universal model property*. That is, there exists a model  $UI$  (called a *universal model*) of  $\mathcal{K}$  that can be homomorphically embedded in any another model of  $\mathcal{K}$ . Due to this fact, the entailment checking for  $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$  is equivalent to model checking for  $UI \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$ . For instance, a *chase* [6] of the minimal model of  $\mathcal{A}$  with  $Pos(\mathcal{K})$  “produces” a universal model of  $\mathcal{K}$  (denoted as  $chase(\mathcal{K})$ ). The constructive nature of  $chase(\mathcal{K})$  allows one to model check  $chase(\mathcal{K}) \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$  in finite time, even if  $chase(\mathcal{K})$  is infinite [1]. In [1] an algorithm is presented (called *perfect reformulation*) to decide whether  $chase(\mathcal{K}) \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$  holds.

Perfect reformulation allows one even more, while deciding whether  $chase(\mathcal{K}) \models \exists \mathbf{x}. \mathbf{y} body(\mathbf{x}; \mathbf{y})$  holds, it returns all the vectors  $\mathbf{c}$  from the answer set  $q(\mathcal{K})$ . The algorithm works in two stages as follows: (1) it rewrites a CQ  $\exists \mathbf{y} body(\mathbf{x}; \mathbf{y})$  to a set  $S$  of CQs using assertions from  $Pos(\mathcal{K})$  and (2) it evaluates  $S$  over  $\mathcal{A}$  stored as an RDB. The evaluation returns precisely  $q(\mathcal{K})$ .

In this work we propose an orthogonal (proof theoretical) approach for deciding the entailment  $\mathcal{A} \cup Pos(\mathcal{K}) \models \exists \mathbf{y} body(\mathbf{c}; \mathbf{y})$ , based on a deductive system (calculus). Moreover, while verifying the entailment  $\mathcal{K} \models \exists \mathbf{x}. \mathbf{y} body(\mathbf{x}; \mathbf{y})$ , the calculus allows one to construct deductions that return precisely  $q(\mathcal{K})$ . We will show later that the perfect reformulation algorithm can be obtained from the calculus by putting a specific control strategy over it.

### 3 *DL-Lite<sub>F</sub>* vs Extended Horn Logic

The idea to make a deductive system for CQ answering has arisen from the observation that *DL-Lite<sub>F</sub>* in fact is a syntactic variation of a fragment of slightly extended Horn Clause Logic, such that existentially quantified variables are allowed to occur in positive literals of Horn clauses (heads of horn rules). Using this observation we adopted the resolution calculus in order to deal with the extension to Horn Logic. In this section we present the extension of Horn Logic and the calculus.

**Extended Horn Logic.** In *Extended Horn Logic* (EHL) formulas (called *e-clauses*) are of the form

$$\forall \mathbf{x} \exists \mathbf{y} (L_1(\mathbf{x}, \mathbf{y}) \vee \dots \vee L_m(\mathbf{x}, \mathbf{y})),$$

where each  $L_i$  is a literal over  $AC \cup AR$ , the vectors  $\mathbf{x}$  and  $\mathbf{y}$  contain all variables occurring in  $L_1, \dots, L_m$ , and at most one literal is positive. As usual we use the terms *goal* and *fact* to refer to e-clauses with no positive literal and no negative literal, respectively.

In [7], natural translations  $\pi_y$  and  $\pi$  respectively from *SHIQ* concepts and assertions to FOL were presented. In fact  $\pi$  maps positive inclusion and membership assertions of *DL-Lite* to EHL. We extend  $\pi$  to conjunctive queries and present both  $\pi_y$  and  $\pi$  in the following table. The variable  $y$  in  $\pi_y(B, x)$  indicates that an implicit existential variable in the the DL expression  $B$  will be explicitly denoted as  $y$  in the FOL version.

$$\begin{array}{l} \pi_y(A, x) = A(x) \qquad \pi_y(\exists R, x) = \exists y R(x, y) \\ \pi_y(\exists R^-, x) = \exists y R(y, x) \\ \hline \pi(A(a)) = A(a) \qquad \pi(B \sqsubseteq B') = \pi_y(B', x) \vee \neg \pi_z(B, x) \\ \pi(R(a, b)) = R(a, b) \qquad \pi(\exists \mathbf{y} \text{ body}(\mathbf{x}; \mathbf{y})) = \neg \text{body}(\mathbf{x}; \mathbf{y}) \end{array}$$

where  $A \in AC$ ;  $R \in AR$ ;  $a, b$  are constants;  $B, B'$  are basic concepts;

We call *CQ goal* a goal corresponding to a CQ. We say that  $\mathbf{x}$  is the *vector of distinguished variables of a CQ goal*  $\Gamma$  if  $\Gamma = \pi(\exists \mathbf{y} \text{ body}(\mathbf{x}; \mathbf{y}))$ . Note, that CQ goals do not contain existential variables and e-clauses that correspond to membership assertions are facts.

The following are examples of applying  $\pi_y$  and  $\pi$ :  $\pi(A \sqsubseteq \exists R) = \pi_y(\exists R, x) \vee \neg \pi_z(A, x) = \exists y R(x, y) \vee \neg A(x)$ , and  $\pi(\exists R^- \sqsubseteq \exists R') = \pi_y(\exists R', x) \vee \neg \pi_z(\exists R^-, x) = \exists y R'(x, y) \vee \neg \exists z R(z, x)$ , and  $\pi(\text{True}) = \perp$ .

**Calculus for Extended Horn Logic.** Our calculus consists of three rules. Rules should be read from top to bottom. We assume that the order of literals in the goals is irrelevant.

1. **Factorization Rule:**

$$fr: \frac{\Gamma \vee L_1 \vee L_2}{\Gamma \theta \vee L_1 \theta} [\theta]$$

where the literals  $L_1$  and  $L_2$  are unifiable, and  $\theta$  is an mgu that is the identity on distinguished variables. If  $L_1 = L_2$ , then  $\theta = id$ .

2.  **$\exists$ -resolution rule:**

$$erl: \frac{\Gamma \vee \neg \pi_x(B, t) \quad \pi_v(B, y) \vee \neg \pi_z(B', y)}{\Gamma \vee \neg \pi_w(B', t)} [id]$$

where  $x$  is a non-distinguished variable that occurs only once in  $\Gamma \vee \neg\pi_x(B, t)$  and  $w$  is a fresh variable.

3. **Resolution rule:**

$$rl : \frac{\Gamma \vee L \quad D}{\Gamma\theta} [\theta]$$

where  $D$  is a ground atom of the form  $A(a)$  or  $R(a, b)$ , i.e., a membership assertion, and  $\theta$  is an mgu of the literals  $L$  and  $\neg D$ .

We say that in the derivation rules  $fr$ ,  $erl$  and  $rl$ , the literals  $L_2$ ,  $\neg\pi_x(B, t)$  and  $L$  are the *leading* ones, respectively. Since each time a derivation step is performed the leading literal is either eliminated or substituted with another literal, we say the leading literal is *processed* by the derivation rule with the e-clause on the right in the premisses of the rule (which is assumed to be  $\perp$  in the case of the  $fr$  rule).

Note, that each time a derivation step is performed the leading literal is either eliminated or substituted with another literal. Because of this reason we say that the leading literal is *processed* by the derivation rule with the e-clause that occurs on the right in the premisses of the rule (in the case of the  $fr$  rule the e-clause is assumed to be  $\perp$ ).

As usual, we say that a goal  $\Gamma'$  is *directly derived* from a goal  $\Gamma$  and an e-clause  $g$  by a rule  $r$  with a substitution  $\theta$ , denoted as  $\Gamma \vdash_{g,\theta} \Gamma'$ , if  $r$  is either an  $fr$ , or an  $erl$ , or an  $rl$  rule,  $\Gamma$  and  $g$  are respectively on the left and on the right in the premisses of  $r$ ,  $\Gamma'$  is in the conclusion of  $r$ , and  $\theta$  *participates* in  $r$  (it occurs in  $r$ ). If a sequence  $\Gamma_1 \cdots \Gamma_n$  of goals is such that for each  $i < n$  the direct derivation  $\Gamma_i \vdash_{g_i,\theta_i} \Gamma_{i+1}$  holds, then we say that  $\Gamma_n$  is *derived* from  $\Gamma_1$  with the substitution  $\Theta = \theta_1 \circ \cdots \circ \theta_{n-1}$  and the set of e-clauses  $L = \{g_1, \dots, g_{n-1}\}$ , and denote it as  $\Gamma_1 \vdash_{L,\Theta}^* \Gamma_n$ .

We say that a query  $q'$  is *derived from* a query  $q$  and a KB  $\mathcal{K}$  with a substitution  $\theta$ , denoted as  $q \vdash_{\mathcal{K},\theta}^* q'$ , if there is a derivation of a CQ goal  $\pi(q')$  from  $\pi(q)$  with  $\theta$  and a set of e-clauses  $L$ , where each  $l \in L$  corresponds either to an assertion from  $Pos(\mathcal{K})$  or to a membership assertion of  $\mathcal{K}$ .

## 4 CQ Answering as Deduction in EHL

We motivated the e-calculus as a general instrument to answer conjunctive queries over satisfiable  $DL-Lite_{\mathcal{F}}$  knowledge bases. In this section we state several formal properties of the general calculus and of some control strategies for it. The proofs are contained in a forthcoming technical report [8].

As a first result, we can show that the calculus can be used to verify that a boolean CQ is entailed by a satisfiable knowledge base.

**Theorem 1 (Soundness and Completeness).** *Let  $\mathcal{K}$  be a satisfiable KB, and  $q() \leftarrow \exists \mathbf{y} \text{ body}(\mathbf{y})$  a boolean CQ over  $\mathcal{K}$ . Then  $q$  is entailed by  $\mathcal{K}$ , i.e.,  $\mathcal{K} \models \exists \mathbf{y} \text{ body}(\mathbf{y})$ , if and only if there exists a derivation from  $\pi(\text{body}(q))$  to the boolean query True.*

As a consequence of this theorem, we can use the calculus to verify that a tuple of constants  $c$  is a certain answer of a CQ  $q(\mathbf{x}) \leftarrow \exists \mathbf{y} \text{ body}(\mathbf{x}; \mathbf{y})$ , since  $c \in q(\mathcal{K})$  if and only if  $\mathcal{K} \models \exists \mathbf{y} \text{ body}(c; \mathbf{y})$ . The next theorem shows that the calculus can be used to generate all certain answers.

**Theorem 2 (Answer Completeness).** *Let  $\mathcal{K}$  be a satisfiable KB,  $q(\mathbf{x}) \leftarrow \exists \mathbf{y} \text{ body}(\mathbf{x}; \mathbf{y})$  a CQ over  $\mathcal{K}$ , and  $\mathbf{c}$  a vector of constants. Then  $\mathbf{c}$  is in the answer set of  $q$  over  $\mathcal{K}$ , that is  $\mathbf{c} \in q(\mathcal{K})$ , if and only if there is a derivation from  $\pi(\text{body}(q))$  to the boolean query True with the substitution  $\theta$  such that  $\mathbf{x}\theta = \mathbf{c}$ .*

**General Algorithm.** We are now in a position to formulate a non-deterministic algorithm to compute the answer set  $q(\mathcal{K})$ . For the algorithm, we specify the states of the computation, transitions between such states, and the subset of final states.

Let  $\mathbf{x}$  be the vector of distinguished variables of  $q$ . A *granule* is a pair  $\Gamma.\sigma$ , where  $\Gamma$  is a CQ goal and  $\sigma$  is a substitution that maps the variables in  $\mathbf{x}$  to constants or to themselves. The states of the computation are sets of granules, denoted by the letter  $\mathcal{G}$ . When computing  $q(\mathcal{K})$ , the initial state is the set  $\{\pi(\text{body}(q)).id\}$ .

Suppose a goal  $\Gamma$  is derived from  $\pi(\text{body}(q))$  by our calculus. Then, in addition to  $\mathbf{x}$ , the goal  $\Gamma$  may contain non-distinguished variables and new variables that are introduced by the *erl*-rule. We refer to both these new and non-distinguished variables as the *existential* variables of  $\Gamma$ . We say that two granules  $\Gamma.\sigma$  and  $\Gamma'.\sigma'$  are *similar* if  $\sigma = \sigma'$  and if  $\Gamma$  and  $\Gamma'$  are identical up to renaming of their existential variables.

In order to define the transition between states, we first need to slightly extend our calculus so that it operates on granules. A granule  $\Gamma'.\sigma'$  is derived from  $\Gamma.\sigma$  if  $\Gamma'$  is derived from  $\Gamma$  with  $\theta$  and  $\sigma' = \sigma \circ \theta$ . There is a transition from a state  $\mathcal{G}$  to a state  $\mathcal{G}' = \mathcal{G} \cup \{G'\}$  if  $G'$  is not similar to any granule in  $\mathcal{G}$  and there is a  $G \in \mathcal{G}$  such that  $G'$  can be derived from  $G$ . In this case we say that the transition *processes*  $G$ . A state  $\mathcal{G}$  is final if no transition from  $\mathcal{G}$  is possible.

We note that there are only finitely many different atomic concepts, atomic roles, and constants occurring in the KB. Hence, for a given maximal length of goals and set of distinguished variables, it is only possible to form finitely many non-similar granules. We also note that the rules of our calculus never increase the length of a goal. Hence, for a given granule, we can only derive finitely many non-similar granules.

The following theorem states that all certain answers of a CQ over a satisfiable KB can be obtained by computing a final state and collecting substitutions from granules with empty goals.

**Theorem 3.** *Let  $q$  be a CQ with distinguished variables  $\mathbf{x}$  over a satisfiable KB  $\mathcal{K}$ . Then every sequence of transitions starting from  $s = \{\pi(\text{body}(q)).id\}$  and using  $\mathcal{K}$  terminates. Moreover, if  $\mathcal{G}$  is a final state reached from  $s$ , then  $q(\mathcal{K}) = \{\sigma\mathbf{x} \mid (\perp.\sigma) \in \mathcal{G}\}$ .*

**Perfect Reformulation Algorithm.** In our framework, we can also show the soundness and completeness of the perfect reformulation algorithm [4]. In fact, an execution of this algorithm corresponds to a sequence of transitions where initially only such transitions are performed that employ the factorization and the  $\exists$ -resolution rules of our calculus. When no transitions of this kind are possible any more, then transitions corresponding to resolution steps are performed. It follows from Theorem 1 that such a strategy leads in fact to certain answers. To show completeness, we need an extra argument. This is provided by the following proposition, which shows that resolution steps can always be postponed until the end of a derivation.

**Proposition 1 (Resolution Commutes).** *Suppose that from  $\Gamma \vee L$  we can obtain  $\Gamma'$  by first applying the resolution rule to the leading literal  $L$  with substitution  $\theta$  and then another rule  $r$  to some leading literal  $M\theta$ . Then we can obtain  $\Gamma'$  as well by first applying  $r$  to  $\Gamma \vee L$  with the leading literal  $M$  and substitution  $\delta$ , and then to the result the resolution rule with leading literal  $L\delta$ . That is,*

$$\Gamma \vee L \vdash_{g,\theta} \Gamma\theta \vdash_{g',\theta'} \Gamma' \text{ implies } \Gamma \vee L \vdash_{g',\delta} (\Gamma''\delta \vee L\delta) \vdash_{g,\delta'} \Gamma' \text{ and } \theta \circ \theta' = \delta \circ \delta'.$$

**Live-Only Algorithm.** In our calculus, it is possible to construct several different derivations from a goal  $\Gamma$  to another goal  $\Gamma'$  and the transition-based algorithm will in fact compute all such derivations. Another problem is that the algorithm is unable to detect granules that will not lead to any answer and continues to process them. To avoid such unnecessary computations, we introduce a criterion to recognize when a granule needs no further processing.

We say a variable  $x$  is *critical* for a CQ goal  $\Gamma$ , if  $x$  occurs more than once in  $\Gamma$ . A literal  $L$  in  $\Gamma$  is *terminal* if  $L$  is unary or if it is binary and does not have a critical variable. Intuitively, if a literal  $L$  is terminal in  $\Gamma$  and no rules are applicable to  $L$ , then no rule will ever become applicable to  $L$  in any  $\Gamma'$  derived from  $\Gamma$ .

We say that a granule  $\Gamma.\sigma$  is *exhausted* in  $\mathcal{G}$ , if  $\Gamma$  contains a terminal literal  $L$  such that the following holds: if  $\Gamma'$  is obtained from  $\Gamma$  by applying a rule of the calculus with leading literal  $L$  and substitution  $\theta$ , then  $(\Gamma'.\sigma \circ \theta)$  is similar to some granule in  $\mathcal{G}$ . Intuitively, a granule is exhausted if it contains one terminal literal that has been completely processed. A granule is *live* if it is not exhausted.

The *Live-Only Algorithm* is a variant of the general algorithm. It is different in that transitions cannot process arbitrary granules, but only live granules. The Live-Only Algorithm allows for a specific control strategy, which resembles SLD-Resolution in Logic Programming. Under SLD-Resolution, an arbitrary literal in a goal is chosen and resolved in all possible ways. After that, the goal is discarded. In our case, if we choose a terminal literal and process it in all possible ways, then the granule becomes exhausted and is blocked from any further rule application. We do not discard exhausted granules from a state because their presence is needed to detect loops.

The completeness of the Live-Only Algorithm (and therefore also of the SLD-like strategy) can be shown by an induction argument using the proposition below, which states that in a derivation a rule application to a terminal literal commutes with all preceding derivation steps.

**Proposition 2.** *Let  $L$  be a terminal literal of  $\Gamma \vee L$ . Suppose there is a derivation of  $\Gamma'$  from  $\Gamma \vee L$  where an instantiation of  $L$  is processed at the last step. Then there is another derivation of  $\Gamma'$  from  $\Gamma \vee L$  where  $L$  is processed at the first step. That is,*

$$\Gamma \vee L \vdash_{G,\theta_1}^* (\Gamma_1 \vee L\theta_1) \vdash_{g,\theta_2} \Gamma' \text{ implies } \Gamma \vee L \vdash_{g,\delta_1} \Gamma\delta_1 \vdash_{G,\delta_2}^* \Gamma' \text{ and } \theta_1 \circ \theta_2 = \delta_1 \circ \delta_2$$

## 5 Related Works and Conclusions

Using resolution for query answering over DL KBs was already considered by several authors. In [7] Hustadt et al. adopted resolution for CQ answering over  $\mathcal{SHIQ}$  KBs

$\mathcal{K}_{SHIQ}$ . They presented a way to decide whether  $c \in q(\mathcal{K}_{SHIQ})$  holds, but they do not propose any procedure for computing answer sets  $q(\mathcal{K}_{SHIQ})$ . Since, our procedure for computing answer sets involves only positive inclusion and membership assertions, i.e., a fragment of  $SHIQ$ , the results of our paper extend the ones in [7] for this fragment. Another work on using resolution for query answering is [9], where a way is proposed to check whether an  $\mathcal{EL}$  KB implies a subsumption between two concepts by posing atomic queries to the KB. This work does not consider CQs and computing certain answers. To the best of our knowledge, our work is the first one that considers computing answer sets for CQs over DL KBs in the framework of resolution.

We envisage that our work will facilitate the combination of *DL-Lite* with other formalisms in data management tasks that are based on variants of Horn logic, such as mappings in data integration [10] and dependencies in data exchange [11]. It remains also to be investigated under which conditions the adoption of evaluation strategies for the calculus that are different from the one underlying the rewriting approach, may lead to improved performance. Specifically, such alternative strategies look promising for those cases where the size of the TBox is *not* negligible w.r.t. the size of the ABox, and the ABox may not be directly managed by a DBMS. In such cases, an approach based on rewriting would generate very large queries to be shipped to the database, while anticipating resolution with ground atoms may result in strong pruning.

## References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Proc. of AAAI 2005. (2005) 602–607
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR 2006. (2006) 260–270
3. Lloyd, J.W.: Foundations of Logic Programming (Second, Extended Edition). Springer, Berlin, Heidelberg (1987)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning (2007) To appear.
5. Kharlamov, E.: Model theory and calculus for the description logic DL-Lite. Master’s thesis, Faculty of Computer Science, Free University of Bozen-Bolzano (2006) Available at <http://www.inf.unibz.it/kharlamov/>.
6. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)
7. Hustadt, U., Motik, B., Sattler, U.: A decomposition rule for decision procedures by resolution-based calculi. In: Proc. of LPAR 2004. (2004) 21–35
8. Calvanese, D., Kharlamov, E., Nutt, W.: A proof theory for DL-Lite. Technical Report KRDB07-6, KRDB Research Center, Faculty of Computer Science, Free University of Bozen-Bolzano (2007)
9. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany (2006) Available at <http://www.cs.man.ac.uk/ykazakov/publications/thesis/Kazakov06Phd.pdf>.
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Linking data to ontologies: The description logic DL-Lite<sub>A</sub>. In: Proc. of OWLED 2006. (2006)
11. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: Proc. of PODS 2005. (2005) 61–75

## Dynamic Description Logic: Embracing Actions into Description Logic

Liang Chang<sup>1,2</sup>, Zhongzhi Shi<sup>1</sup>, Lirong Qiu<sup>1,2</sup>, and Fen Lin<sup>1,2</sup>

<sup>1</sup> Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China,

<sup>2</sup> Graduate University of Chinese Academy of Sciences, Beijing, China  
 {changl,shizz,qiulr,linf}@ics.ict.ac.cn

**Abstract.** We present a dynamic description logic  $\mathcal{D}\text{-}\mathcal{ALCO}$ @ for representing knowledge about dynamic application domains.  $\mathcal{D}\text{-}\mathcal{ALCO}$ @ is a combination of a typical action theory and the description logic  $\mathcal{ALCO}$ @, in such a way that actions are treated as citizens of the logic. Actions of  $\mathcal{D}\text{-}\mathcal{ALCO}$ @ are explicitly specified with the help of formulas, and are then used in the construction of concepts and formulas. Based on a regression operator introduced to deal with actions, we provide a tableau-based decision algorithm for this logic.

### 1 Introduction

Description logics are successfully used for representing knowledge about static application domains in a structured way. In order to describe knowledge about dynamic application domains, temporal extension of description logics has been extensively studied for more than ten years [2]. With a general temporal description logic, the concept *Mortal* can be described as  $Mortal = LivingBeing \sqcap (LivingBeing \mathcal{U} \sqcap \neg LivingBeing)$ , which states that a mortal is a living being that eventually will not be alive any more. A temporal evolution is embodied in this description, but actions that fulfilling the evolution are not referred.

In [9], Wolter proposed a dynamic description logic named  $\mathcal{PDLC}$ , by combining the description logic  $\mathcal{ALC}$  with propositional dynamic logic PDL. With  $\mathcal{PDLC}$ , the concept *Mortal* can be described as  $Mortal = LivingBeing \sqcap \langle die \rangle \neg LivingBeing$ , in which actions that bring the change are presented. Although atomic actions of  $\mathcal{PDLC}$  can be combined using PDL-like operators, these atomic actions are still short of descriptions. Moreover, efficient decision algorithm for this logic is still an open problem

In this paper, we propose a dynamic description logic  $\mathcal{D}\text{-}\mathcal{ALCO}$ @, by combining a typical action theory with the description logic  $\mathcal{ALCO}$ @. On the one hand, actions are generated from atomic actions with the help of many constructs, and each atomic action is specified by its preconditions  $P$  and conditional effects  $E$ , where  $P$  and  $E$  are described with formulas. On the other hand, actions could be used in the construction of concepts and formulas. Therefore, not only concepts with dynamic meaning, but also actions happened in dynamic domains, can all be described and reasoned with this formalism.

We first define the syntax and semantics of  $\mathcal{D}\text{-ALCCO@}$  in section 2, then introduce a regression operator to deal with actions in section 3, and provide a tableau based decision algorithm for a restricted form of the logic in section 4. Section 5 concludes the paper.

## 2 Syntax and Semantics of Dynamic Description Logic

The primitive symbols of  $\mathcal{D}\text{-ALCCO@}$  are a set  $N_C$  of concept names, a set  $N_R$  of role names, and a set  $N_I$  of individual names.

*Concepts*  $C, C'$  of  $\mathcal{D}\text{-ALCCO@}$  are formed with the following syntax rules:

$$C, C' \longrightarrow D|\@_p C|C \sqcup C'|\exists R.C|< \pi > C \quad (1)$$

$$D, D' \longrightarrow C_i|\{p\}|\@_p D|\neg D|D \sqcup D'|\exists R.D$$

where  $C_i \in N_C, p \in N_I, R \in N_R, \pi$  is an action. These syntax rules are designed to ensure that no concepts of the form  $\neg < \pi > C$  are constructed.

*Formulas*  $\varphi, \varphi'$  of  $\mathcal{D}\text{-ALCCO@}$  are formed with the following syntax rules:

$$\varphi, \varphi' \longrightarrow \phi|\varphi \vee \varphi'|< \pi > \varphi \quad (2)$$

$$\phi, \phi' \longrightarrow C(p)|R(p, q)|\neg \phi|\phi \vee \phi'$$

where  $C$  is a concept,  $p, q \in N_I, R \in N_R$ , and  $\pi$  is an action. These syntax rules are designed to ensure that no formulas of the form  $\neg < \pi > \varphi$  are constructed.

An *atomic action* of  $\mathcal{D}\text{-ALCCO@}$  is a pair  $(P, E)$ , where,

- $P$  is a finite set of formulas, used for describing the so-called pre-conditions,
- $E$  is a finite set of conditional effects of the form  $\psi/\phi$ , where  $\psi$  is a formula,  $\phi$  is of form  $A(p), \neg A(p), R(p, q)$ , or  $\neg R(p, q)$ , with  $A \in N_C, R \in N_R$ , and  $p, q \in N_I$ ,
- let  $P = \{\varphi_1, \dots, \varphi_n\}$  and  $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$ , then  $P$  and  $E$  subject to the constraint that  $\varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \neg \phi_k$  for all  $k$  with  $1 \leq k \leq m$ .

*Actions*  $\pi, \pi'$  of  $\mathcal{D}\text{-ALCCO@}$  are formed with the following syntax rule:

$$\pi, \pi' \longrightarrow (P, E)|\varphi?|\pi \cup \pi'|\pi; \pi' \quad (3)$$

where  $(P, E)$  is an atomic action,  $\varphi$  is a formula.

Before introducing the semantics for this logic, we will give some intuitive examples. Firstly, we describe an atomic action named *load* as  $(\{Gun(a), \neg loaded(a)\}, \{\top(a)/loaded(a)\})$ , where  $\top$  is an abbreviation of  $C \sqcup \neg C$  for any concept  $C$ . The description tells that the action could happen in the case that  $a$  is an unloaded gun, and the only change brought about by this action is that  $a$  is loaded. Similarly, an atomic action named *shoot* is described as  $(\{Gun(a), LivingBeing(b)\}, \{loaded(a)/\neg LivingBeing(b)\})$ , the conditional effect of it means that in the case  $a$  is loaded,  $b$  will be not alive after the execution of *shoot*.

These two atomic actions can then be used to form a formula  $\langle load ; shoot \rangle (\neg LivingBeing(b))$ , which asserts that  $b$  might be not alive after the sequential execution of *load* and *shoot*. Furthermore, we can also construct a concept  $\langle shoot \rangle LivingBeing$  to describe these individuals that might be alive after the execution of *shoot*.

A model of  $\mathcal{D}\text{-ALCCO@}$  is a pair  $M=(W, I)$ , where  $W$  is a set of states,  $I$  associates with each state  $w \in W$  an interpretation  $I(w) = (\Delta^I, C_0^{I(w)}, \dots, R_0^{I(w)}, \dots, p_0^{I(w)}, \dots)$ , with  $C_i^{I(w)} \subseteq \Delta^I$  for each  $C_i \in N_C$ ,  $R_i^{I(w)} \subseteq \Delta^I \times \Delta^I$  for each  $R_i \in N_R$ , and  $p_i^{I(w)} \in \Delta^I$  for each  $p_i \in N_I$ ; furthermore, for any  $p_i \in N_I$  and any  $u, v \in W$ , we have  $p_i^{I(u)} = p_i^{I(v)}$ . Based on the interpretations of all these states, each action  $\pi$  is interpreted as a binary relation  $\pi^I \subseteq W \times W$ .

Given a  $\mathcal{D}\text{-ALCCO@}$  model  $M=(W, I)$  and a state  $w \in W$ , the value  $C^{I(w)}$  of a concept  $C$ , the truth-relation  $(M, w) \models \varphi$  (or simply  $w \models \varphi$  if  $M$  is understood) for a formula  $\varphi$ , and the relation  $\pi^I$  for an action  $\pi$  are defined inductively as follows:

- (1)  $\{p\}^{I(w)} = \{p^{I(w)}\}$ ;
- (2) If  $p^{I(w)} \in C^{I(w)}$  then  $(@_p C)^{I(w)} = \Delta^I$ , else  $(@_p C)^{I(w)} = \{\}$ ;
- (3)  $(\neg C)^{I(w)} = \Delta^I - C^{I(w)}$ ;
- (4)  $(C \sqcup D)^{I(w)} = C^{I(w)} \cup D^{I(w)}$ ;
- (5)  $(\exists R.C)^{I(w)} = \{x \mid \exists y. ((x, y) \in R^{I(w)} \wedge y \in C^{I(w)})\}$ ;
- (6)  $(\langle \pi \rangle C)^{I(w)} = \{p \mid \exists w' \in W. ((w, w') \in \pi^I \wedge p \in C^{I(w')})\}$ ;
- (7)  $(M, w) \models C(p)$  iff  $p^{I(w)} \in C^{I(w)}$ ;
- (8)  $(M, w) \models R(p, q)$  iff  $(p^{I(w)}, q^{I(w)}) \in R^{I(w)}$ ;
- (9)  $(M, w) \models \neg \varphi$  iff  $(M, w) \models \varphi$  not holds;
- (10)  $(M, w) \models \varphi \vee \psi$  iff  $(M, w) \models \varphi$  or  $(M, w) \models \psi$ ;
- (11)  $(M, w) \models \langle \pi \rangle \varphi$  iff  $\exists w' \in W. ((w, w') \in \pi^I \wedge (M, w') \models \varphi)$ ;
- (12) Let  $S$  be a formula set, then,  $(M, w) \models S$  iff  $(M, w) \models \varphi_i$  for all  $\varphi_i \in S$ ;
- (13) Let  $(P, E)$  be an atomic action with  $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$ , then,  $(P, E)^I = \{(w_1, w_2) \in W \times W \mid (M, w_1) \models P, C^{I(w_2)} = C^{I(w_1)} \cup C^+ - C^- \text{ for each concept name } C \in N_C, \text{ and } R^{I(w_2)} = R^{I(w_1)} \cup R^+ - R^- \text{ for each role name } R \in N_R\}$ , where,
  - $C^+ = \{p^{I(w_1)} \mid \psi/\phi \in E_\alpha, \phi = C(p), \text{ and } (M, w_1) \models \psi\}$ ,
  - $C^- = \{p^{I(w_1)} \mid \psi/\phi \in E_\alpha, \phi = \neg C(p), \text{ and } (M, w_1) \models \psi\}$ ,
  - $R^+ = \{(p^{I(w_1)}, q^{I(w_1)}) \mid \psi/\phi \in E_\alpha, \phi = R(p, q), \text{ and } (M, w_1) \models \psi\}$ ,
  - $R^- = \{(p^{I(w_1)}, q^{I(w_1)}) \mid \psi/\phi \in E_\alpha, \phi = \neg R(p, q), \text{ and } (M, w_1) \models \psi\}$ ;
- (14)  $(\varphi?)^I = \{(w_1, w_1) \in W \times W \mid (M, w_1) \models \varphi\}$ ;
- (15)  $(\pi \cup \pi')^I = \pi^I \cup \pi'^I$ ;
- (16)  $(\pi; \pi')^I = \pi^I \circ \pi'^I$ .

The interpretation of atomic actions follows the possible models approach[8], and adopts the style introduced in[4].

A formula  $\varphi$  (or a formula set  $S$ ) is satisfiable if and only if there is a model  $M = (W, I)$  and a state  $w \in W$  with  $(M, w) \models \varphi$  (or  $(M, w) \models S$ ).

The goal of the following sections is to develop an algorithm for checking the satisfiability of  $\mathcal{D}\text{-ALCCO@}$  formulas. For simplicity, we take the unique name

assumption (UNA), i.e.,  $p_i^{I(w)} \neq p_j^{I(w)}$  for any  $p_i, p_j \in N_I$  with  $p_i \neq p_j$ . Furthermore, we don't take into account TBoxes that composed of concept definitions[3].

### 3 Regression Operator

In this section we introduce a *regression operator* to deal with actions.

**Definition 1 (Regression).** For a formula of the form  $\langle \pi \rangle \varphi$ ,  $\psi$  is the result of regressing  $\langle \pi \rangle \varphi$ , in symbols  $\text{Regress}(\langle \pi \rangle \varphi) = \psi$ , if

- (1) no actions occurred in  $\psi$ , i.e.,  $\psi$  is an  $\mathcal{ALCO}$  formula;
- (2)  $\langle \pi \rangle \varphi \models \psi$ , i.e., for any model  $M = (W, I)$  and any state  $w \in W$ : if  $(M, w) \models \langle \pi \rangle \varphi$ , then  $(M, w) \models \psi$ ;
- (3) for any model  $M = (W, I)$  and any state  $w \in W$ : if  $(M, w) \models \psi$ , then we can construct a model  $M' = (W', I')$  by introducing a world  $w'$ , with the constraint that  $W' = W \cup \{w'\}$ ,  $I'(w_i) = I(w_i)$  for each  $w_i \in W$ ,  $(w, w') \in \pi^I$ , and  $(M', w') \models \varphi$ ; therefore we have  $(M', w) \models \langle \pi \rangle \varphi$ .

Before presenting algorithms for the regression operator, we introduce another operator named *ABox updating triggered by atomic action*.

An ABox is a finite set of individual assertions of the form  $C(p)$ ,  $R(p, q)$ , and  $\neg R(p, q)$ , where  $C$  is a concept,  $R \in N_R$ , and  $p, q \in N_I$ . An ABox  $\mathcal{A}$  entails a formula  $\varphi$  (written  $\mathcal{A} \models \varphi$ ) if and only if for any model  $M = (W, I)$  and any state  $w \in W$ :  $(M, w) \models \mathcal{A}$  implies  $(M, w) \models \varphi$ . An ABox  $\mathcal{A}$  entails a formula set  $S$  (written  $\mathcal{A} \models S$ ) if and only if  $\mathcal{A} \models \varphi$  for any formula  $\varphi \in S$ .

**Definition 2 (ABox updating triggered by atomic action).** Let  $\mathcal{A}, \mathcal{A}'$  be ABoxes,  $(P, E)$  be an atomic action. Then,  $\mathcal{A}'$  is the result of updating  $\mathcal{A}$  with  $(P, E)$ , in symbols  $\mathcal{A} * (P, E) = \mathcal{A}'$ , if

- (1)  $\mathcal{A} \models P$ ;
- (2) For any model  $M = (W, I)$  and any states  $w, w' \in W$ : if  $(w, w') \in (P, E)^I$  and  $(M, w) \models \mathcal{A}$ , then  $(M, w') \models \mathcal{A}'$ ;
- (3) For any model  $M = (W, I)$  and any state  $w$  in  $W$ : if  $(M, w) \models \mathcal{A}'$ , then we can construct a model  $M' = (W', I')$  by introducing a world  $w'$ , with the constraint that  $W' = W \cup \{w'\}$ ,  $I'(w_i) = I(w_i)$  for each  $w_i \in W$ ,  $(w', w) \in (P, E)^{I'}$  and  $(M', w') \models \mathcal{A}$ .

This operator is similar to the ABox updating introduced by Liu et al[6]. Therefore, based on the ABox update algorithm by Liu et al[6], we develop the following algorithm to calculate  $\mathcal{A} * (P, E)$ , by adding step (1) to decide whether the action is executable on  $\mathcal{A}$ , adding step (2) to select these changes that will take place, and adding the seventh case in step (3) to calculate  $(\langle \pi \rangle C)^{Ea}$  for concepts of the form  $\langle \pi \rangle C$ .

**Algorithm 1 ( $\mathcal{A} * (P, E)$ )** Let  $\mathcal{A}$  be an ABox,  $(P, E)$  be an atomic action with  $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$ . Construct an ABox  $\mathcal{A}'$  with the following steps:

- (1) If  $\mathcal{A} \models P$  not holds, exit the algorithm with the result “ $(P, E)$  is not executable on  $\mathcal{A}$ ”;

- (2) Construct a set  $\text{Eff}(\mathcal{A}, E) := \{\phi \mid \psi/\phi \in E \text{ and } \mathcal{A} \models \psi\}$ ; If  $\text{Eff}(\mathcal{A}, E)$  is empty, return the ABox  $\mathcal{A}' := \mathcal{A}$ , exit the algorithm;
- (3) Let  $\text{ObjE}(\mathcal{A}, E)$  be all the individual names occurred in  $\text{Eff}(\mathcal{A}, E)$ ; Construct an ABox  $\mathcal{A}^E := \{C^E(p) \mid C(p) \in \mathcal{A}\} \cup \{R(p, q) \mid R(p, q) \in \mathcal{A} \text{ and } \neg R(p, q) \notin \text{Eff}(\mathcal{A}, E)\} \cup \{\neg R(p, q) \mid \neg R(p, q) \in \mathcal{A} \text{ and } R(p, q) \notin \text{Eff}(\mathcal{A}, E)\}$ , where  $C^E$  is constructed inductively as follows:

- For concept name  $C_i$ ,  $C_i^E := C_i \sqcup \bigsqcup_{-C_i(p) \in \text{Eff}(\mathcal{A}, E)} \{p\} \sqcap \bigsqcap_{C_i(p) \in \text{Eff}(\mathcal{A}, E)} \neg\{p\}$ ;
- $\{p\}^E := \{p\}$ ;
- $(@_p D)^E := @_p D^E$ ;
- $(\neg D)^E := \neg D^E$ ;
- $(D_1 \sqcup D_2)^E := D_1^E \sqcup D_2^E$ ;
- $(\exists R.D)^E := (\bigsqcap_{p \in \text{ObjE}(\mathcal{A}, E)} \neg\{p\} \sqcap \exists R.D^E) \sqcup (\bigsqcup_{p \in \text{ObjE}(\mathcal{A}, E)} \{p\} \sqcap \exists R.(\bigsqcap_{q \in \text{ObjE}(\mathcal{A}, E)} \neg\{q\} \sqcap D^E)) \sqcup \bigsqcup_{p, q \in \text{ObjE}(\mathcal{A}, E), R(p, q) \notin \text{Eff}(\mathcal{A}, E), \neg R(p, q) \notin \text{Eff}(\mathcal{A}, E)} (\{p\} \sqcap \exists R.(\{q\} \sqcap D^E)) \sqcup \bigsqcup_{\neg R(p, q) \in \text{Eff}(\mathcal{A}, E)} (\{p\} \sqcap @_q D^E)$ ;
- $(\langle \pi \rangle D)^E := \langle \{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\}; \pi \rangle C$ , where  $(\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})$  is an atomic action constructed according to the elements of  $E$ ;

- (4) Return the ABox  $\mathcal{A}' := \mathcal{A}^E \cup \text{Eff}(\mathcal{A}, E)$ .

In this algorithm, rules used for constructing  $C^E$  are technically designed to guarantee the following property:

*Property 1.* Let  $M = (W, I)$  be a  $\mathcal{D}\text{-ALCO@}$  model,  $w, w' \in W$ , and  $(w, w') \in (P, E)^I$ . Then, for any  $\mathcal{D}\text{-ALCO@}$  concept  $C$  and any individual name  $x$ ,  $w' \models C^E(x)$  if and only if  $w \models C(x)$ .

Since  $x^{I(w')} = x^{I(w)}$ , we only need to demonstrate  $(C^E)^{I(w')} = C^{I(w)}$ , by structural induction on  $C$ . In these cases that  $C$  is  $C_i$ ,  $\{p\}$ ,  $@_p D$ ,  $\neg D$ ,  $D_1 \sqcup D_2$ , and  $\exists R.D$ , the proof is similar to those given in[5]. In the case that  $C$  is  $\langle \pi \rangle D$ , we have  $((\langle \pi \rangle D)^E)^{I(w')} = \langle \{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\}; \pi \rangle D^{I(w')} = \{p \mid \exists w_1 \in W. ((w', w_1) \in ((\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\}); \pi)^I \wedge p \in D^{I(w_1)})\} = \{p \mid \exists w_1 \in W. \exists w_2 \in W. ((w', w_2) \in (\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})^I \wedge (w_2, w_1) \in \pi^I \wedge p \in D^{I(w_1)})\}$ . According to the semantics of  $\mathcal{D}\text{-ALCO@}$  actions,  $I(w_2)$  and  $I(w)$  are equivalent and  $(w', w) \in (\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})^I$ , therefore we can continue these equations as  $((\langle \pi \rangle D)^E)^{I(w')} = \{p \mid \exists w_1 \in W. ((w, w_1) \in \pi^I \wedge p \in D^{I(w_1)})\} = \langle \pi \rangle D^{I(w)}$ .

The following property is an easy consequence:

*Property 2.* Algorithm 1 is terminable, the returned  $\mathcal{A}'$  satisfies  $\mathcal{A} * \alpha = \mathcal{A}'$ .

Utilizing the algorithm of  $\mathcal{A} * (P, E)$ , we develop the following algorithm for the regression operator:

**Algorithm 2** (*Regress*( $\langle \pi \rangle \varphi$ )) *Let  $\pi$  be an action,  $\varphi$  be a formula. Calculate *Regress*( $\langle \pi \rangle \varphi$ ) recursively with the following steps:*

- (1) *If  $\varphi$  contains a subformula of the form  $\langle \pi_i \rangle \varphi_i$ , then replace all the occurrence of  $\langle \pi_i \rangle \varphi_i$  in  $\varphi$  with *Regress*( $\langle \pi_i \rangle \varphi_i$ ); Repeat this step, until no such subformulas contained in  $\varphi$ ;*
- (2) *If  $\pi$  is an atomic action  $(P, E)$  with  $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$ , then,*
  - (i) *Construct an atomic action  $\alpha' = (\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})$ ;*
  - (ii) *Translate  $\varphi$  into a disjunction normal form  $\varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_k$ , where each  $\varphi_i$  is a conjunction of individual assertions;*
  - (iii) *For each  $\varphi_i$ , let it be  $\varphi_{i1} \wedge \varphi_{i2} \wedge \dots \wedge \varphi_{im}$ , construct an ABox  $\mathcal{A}_i := \{\varphi_{i1}, \varphi_{i2}, \dots, \varphi_{im}\}$ , and construct the ABox  $\mathcal{A}'_i$  that satisfying  $\mathcal{A}_i * \alpha' = \mathcal{A}'_i$ ;*
  - (iv) *Return the formula  $(\text{Set2F}(\mathcal{A}'_1) \vee \dots \vee \text{Set2F}(\mathcal{A}'_k)) \wedge \text{Set2F}(P)$ , where  $\text{Set2F}(S)$  represents the conjunction of all the elements of  $S$ , e.g., if  $S = \{\varphi_1, \dots, \varphi_n\}$ , then  $\text{Set2F}(S) := \varphi_1 \wedge \dots \wedge \varphi_n$ .*
- (3) *If  $\pi$  is  $\phi?$ , then return the formula  $\phi \wedge \varphi$ ;*
- (4) *If  $\pi$  is  $\pi_1 \cup \pi_2$ , then return  $\text{Regress}(\langle \pi_1 \rangle \varphi) \vee \text{Regress}(\langle \pi_2 \rangle \varphi)$ ;*
- (5) *If  $\pi$  is  $\pi_1 ; \pi_2$ , then return  $\text{Regress}(\langle \pi_1 \rangle \text{Regress}(\langle \pi_2 \rangle \varphi))$ .*

*Property 3.* Algorithm 2 is terminable, the returned formula  $\psi$  satisfies  $\text{Regress}(\langle \pi \rangle \varphi) = \psi$ .

This property can be proved with three steps. Firstly, in the case that  $\pi$  is an atomic action and  $\varphi$  is a formula containing no subformulas of the form  $\langle \pi_i \rangle \varphi_i$ , it is obvious that the algorithm will terminate, it is also easy to demonstrate that the returned formula  $\psi$  satisfies  $\text{Regress}(\langle \pi \rangle \varphi) = \psi$ . Secondly, we relax the  $\pi$  to be any actions, and demonstrate the same results by structural induction on  $\pi$ . Finally, we relax the  $\varphi$  to be any formulas and demonstrate the property. Due to space limitation, we omit the details here.

## 4 Tableau Algorithm

Based on the regression operator, we can develop a tableau-based procedure for deciding the satisfiability of  $\mathcal{D}\text{-ALCO@}$  formulas.

**Algorithm 3** (*Deciding the satisfiability of a  $\mathcal{D}\text{-ALCO@}$  formula*) *For a  $\mathcal{D}\text{-ALCO@}$  formula  $\varphi$ , decide its satisfiability with the following steps:*

- (1) *Construct a formula set  $S' := \{\varphi\}$ . If  $S'$  contains clash, exit the algorithm with the result “ $\varphi$  is unsatisfiable”.*
- (2) *Construct a set  $SS := \{S'\}$ ;*
- (3) *Take out an element  $S$  from  $SS$ , apply one of the rules in table 1 to  $S$ ; For every new generated formula set, if it contains no clash, then add it into  $SS$ ;*
- (4) *Repeat step (3), until  $SS$  is empty or no rules can be applied to the formula set  $S$  that taken out from  $SS$ , in the former case return the result “ $\varphi$  is unsatisfiable”, in the latter case return the result “ $\varphi$  is satisfiable”.*

A *clash* in a formula set  $S$  is one of the following cases: (1)  $\varphi \in S$  and  $\neg\varphi \in S$  for a formula  $\varphi$ ; (2)  $C(p) \in S$  and  $(\neg C)(p) \in S$  for a concept  $C$  and an

individual name  $p$ ; (3)  $\{q\}(p) \in S$  for two different individual names  $p$  and  $q$ ;  
 (4)  $(\neg\{p\})(p) \in S$  for an individual name  $p$ .

**Table 1.** Tableau rules for  $\mathcal{D}\text{-}\mathcal{ALCC}\mathcal{O}$

**Rules on concepts:**

- $\mathbf{R}_{\textcircled{q}}$ : If  $(\textcircled{q}C)(x) \in S$  and  $C(q) \notin S$ , then  $S_1 := \{C(q)\} \cup S$ ;
- $\mathbf{R}_{\neg\textcircled{q}}$ : If  $(\neg\textcircled{q}C)(x) \in S$  and  $(\neg C)(q) \notin S$ , then  $S_1 := \{(\neg C)(q)\} \cup S$ ;
- $\mathbf{R}_{\sqcup}$ : If  $(C_1 \sqcup C_2)(x) \in S$ ,  $C_1(x) \notin S$  and  $C_2(x) \notin S$ ,  
 then  $S_1 := C_1(x) \cup S$ ,  $S_2 := C_2(x) \cup S$ ;
- $\mathbf{R}_{\neg\sqcup}$ : If  $(\neg(C_1 \sqcup C_2))(x) \in S$ , and  $(\neg C_1)(x) \notin S$  or  $(\neg C_2)(x) \notin S$ ,  
 then  $S_1 := \{(\neg C_1)(x), (\neg C_2)(x)\} \cup S$ ;
- $\mathbf{R}_{\exists}$ : If  $(\exists R.C)(x) \in S$ , there is no  $y$  such that  $R(x, y) \in S$  and  $C(y) \in S$ ,  
 then  $S_1 := \{C(z), R(x, z)\} \cup S$ , where  $z$  is a new individual name;
- $\mathbf{R}_{\neg\exists}$ : If  $(\neg(\exists R.C))(x) \in S$ , then  $S_1 := \{(\neg C)(y) \mid R(x, y) \in S, (\neg C)(y) \notin S\}$ ;
- $\mathbf{R}_{<\pi>c}$ : If  $(\langle \pi \rangle C)(x) \in S$ , and  $\text{Regress}(\langle \pi \rangle C(x)) \notin S$ ,  
 then  $S_1 := \{\text{Regress}(\langle \pi \rangle C(x))\} \cup S$ ;
- $\mathbf{R}_{\neg c}$ : If  $(\neg C)(x) \in S$ , and  $C(x) \notin S$ , then  $S_1 := \{C(x)\} \cup S$ ;

**Rules on formulas:**

- $\mathbf{R}_{\vee}$ : If  $\varphi \vee \psi \in S$ ,  $\varphi \notin S$ , and  $\psi \notin S$ , then  $S_1 := \{\varphi\} \cup S$ ,  $S_2 := \{\psi\} \cup S$
- $\mathbf{R}_{\neg\vee}$ : If  $\neg(\varphi \vee \psi) \in S$ , and  $\neg\varphi \notin S$  or  $\neg\psi \notin S$ , then  $S_1 := \{\neg\varphi, \neg\psi\} \cup S$ ;
- $\mathbf{R}_{<\pi>f}$ : If  $\langle \pi \rangle \varphi \in S$  and  $\text{Regress}(\langle \pi \rangle \varphi) \notin S$ , then  $S_1 := \{\text{Regress}(\langle \pi \rangle \varphi)\} \cup S$ ;
- $\mathbf{R}_{\neg f}$ : If  $\neg\varphi \in S$  and  $\varphi \notin S$ , then  $S_1 := \{\varphi\} \cup S$ .

It is easy to demonstrate that this algorithm holds the following properties:

- (*Termination*) The algorithm is terminable;
- (*Soundness*) For the set  $SS$ , let  $SS'$  be the result of executing step (3), then, there is a satisfiable formula set  $S \in SS$  if and only if there is a satisfiable formula set  $S' \in SS'$ .
- (*Completeness*) For any finite formula set  $S \in SS$ , if no rules can be applied to  $S$  and  $S$  contains no clash, then  $S$  is satisfiable.

## 5 Discussion

TBoxes composed of concept definitions are not taken into account in previous sections. In fact, if a TBox is referred, the set  $N_C$  could be divided into two disjoint sets  $N_{CD}$  and  $N_{CP}$ , where  $N_{CD}$  is the set of defined concept names,  $N_{CP}$  is the set of primitive concept names [3]. In this case, adopting the idea of [4], we will add a constraint to the description of atomic actions: for every conditional effect  $\psi/\phi$ ,  $\phi$  must be of form  $A(p)$ ,  $\neg A(p)$ ,  $R(p, q)$ , or  $\neg R(p, q)$ , with  $A \in N_{CP}$ . Then, if the TBox is acyclic and contains no general concept inclusion

axioms(GCIs), our algorithms are still effective, by adding processes to replace each occurrence of defined concept names with their corresponding definitions.

As actions are treated as citizens in  $\mathcal{D}\text{-}\mathcal{ALCO}$ , reasoning problems about actions could be introduced into this formalism, such as the executability, projection, and subsumption problems[1][4][7]. We think that the regression operator is useful for these reasoning tasks. For example, in order to decide that whether a formula  $\varphi$  is a consequence of applying a sequence of actions  $\pi_1, \dots, \pi_k$  in an ABox  $\mathcal{A}$ , we can calculate the formula  $\psi := \neg \text{Regress}(\langle \pi_1; \dots; \pi_k \rangle \neg \varphi)$  and check  $\mathcal{A} \models \psi$ .

Another future work is to alleviate these syntactic restrictions posed on this logic, so that concepts of the form  $\neg \langle \pi \rangle C$ , formulas of the form  $\neg \langle \pi \rangle \varphi$ , and actions of the form  $\pi^*$  can be constructed.

**Acknowledgments.** This work was partially supported by the National Science Foundation of China (No. 90604017, 60435010), and National Basic Research Priorities Programme(No. 2003CB317004). We would like to thank these anonymous reviewers for their valuable suggestions.

## References

1. Artale, A., Franconi, E.: A Temporal Description Logic for Reasoning about Actions and Plans. *Journal of Artificial Intelligence Research*, 1998, 9: 463-506.
2. Artale, A. Franconi, E.: A Survey of Temporal Extensions of Description Logics. *Annals of Mathematics and Artificial Intelligence*, 2000, 30(1-4):171-210.
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. F., eds.: *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
4. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating Description Logics and Action Formalisms: First Results. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'05)*, Pittsburgh, PA, USA, 2005.
5. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating Description Logic ABoxes. *LTCS-Report 05-10*, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005.
6. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'2006)*, 2006.
7. Lutz, C., Sattler, U.: A Proposal for Describing Services with DLs. In *Proceedings of the 2002 International Workshop on Description Logics (DL'2002)*, 2002.
8. Winslett, M.: Reasoning about action using a possible models approach. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'88)*, 1988.
9. Wolter, F., Zakharyashev, M.: Dynamic description logic. In *Advances in Modal Logic*, Vol 2. Stanford: CSLI Publications, 2000: 449-463

## Partitioning ABoxes Based on Converting DL to Plain Datalog

Jianfeng Du<sup>1,2</sup> and Yi-Dong Shen<sup>1</sup>

<sup>1</sup> State Key Laboratory of Computer Science,  
Institute of Software, Chinese Academy of Sciences

<sup>2</sup> Graduate University of the Chinese Academy of Sciences  
Beijing 100080, China  
jfdu,ydshen@ios.ac.cn

**Abstract.** To make ABox reasoning scalable for large ABoxes in description logic (DL) knowledge bases, we develop a method for partitioning the ABox so that specific kinds of reasoning can be performed separately on each partition and the results trivially combined can achieve complete answers. Our method applies to  $\mathcal{SHIQ}(\mathbf{D})$  knowledge bases. It first converts a DL knowledge base  $KB$  to a plain datalog program  $H(KB)$ , and then computes the least fixpoint of the definite part of  $H(KB)$  while generating ABox partitions. Its time data complexity is polynomial in the ABox size, under some general assumption on concrete domains. Experimental results further demonstrate the advantages of our method.

### 1 Introduction

ABox reasoning (e.g., query answering) in DL knowledge bases is a great challenge, due to high complexity of reasoning in expressive DL languages [2] and the resource limitations (e.g., physical memory) for handling large ABoxes. As [3] pointed out, there are two approaches towards scalable ABox reasoning. One approach is to partition the ABox so that some kinds of reasoning can be performed separately on each partition [3, 5]. Another approach is to convert DL to disjunctive datalog and use deductive databases to reason over the ABox [7]. Based on [7], Motik *et al.* [8] propose a resolution based algorithm which evaluates non-ground queries in one pass so that the efficiency of the query answering is further improved. However, the algorithm requires exponential space in the worst case, thus it is particularly important to partition the ABox to cut down memory consumption.

In this paper, we present a new method for partitioning the ABox. For a  $\mathcal{SHIQ}(\mathbf{D})$  knowledge base  $KB$ , we first reduce  $KB$  to an equisatisfiable disjunctive datalog program  $DD(KB)$ , using the methods in [7]. Then, we convert  $DD(KB)$  to a plain datalog program  $H(KB)$  by replacing disjunctions with definite implications, and compute the least fixpoint of the definite part of  $H(KB)$ . During the fixpoint computation, we put sets of assertions that (indirectly) trigger rules in  $H(KB)$  to the same part and keep track of the triggering information

on each assertion. At last, we adjust parts to partitions according to the tracking data. The time data complexity of our method is polynomial in the ABox size, assuming a polynomial oracle for reasoning with concrete domains and a constant bound on the arity of the concrete domain predicates. Our method always produces a disjoint and independent partitioning, such that each assertion over atomic concepts or simple roles is entailed by the whole knowledge base if and only if it is independently entailed by some partition.

## 2 Reducing $\mathcal{SHIQ}(\mathbf{D})$ to Disjunctive Datalog

A  $\mathcal{SHIQ}(\mathbf{D})$  [7] knowledge base  $KB = (KB_{\mathcal{T}}, KB_{\mathcal{R}}, KB_{\mathcal{A}})$  consists of a TBox  $KB_{\mathcal{T}}$ , an RBox  $KB_{\mathcal{R}}$  and an ABox  $KB_{\mathcal{A}}$ .  $KB_{\mathcal{T}}$  is a finite set of *concept inclusion axioms*.  $KB_{\mathcal{R}}$  is a finite set of *transitivity axioms* and *role inclusion axioms*.  $KB_{\mathcal{A}}$  is a set of *concept and role membership assertions*  $(\neg)A(a), R(a, b), T(a, c)$ , and *(in)equality assertions*  $a = b, a \neq b$ , where  $A$  is an atomic concept,  $R$  an abstract role,  $T$  a concrete role,  $c$  a concrete individual,  $a$  and  $b$  abstract individuals.

In [7], a resolution framework is proposed to reduce a  $\mathcal{SHIQ}(\mathbf{D})$  knowledge base  $KB$  to a disjunctive datalog program  $DD(KB) = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A}} \cup \Delta_{KB}$ .  $\Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}})$  is a positive disjunctive datalog program computed regardless of  $KB_{\mathcal{A}}$  and consists of rules of the form

$$A_1 \vee \dots \vee A_m \leftarrow B_1, \dots, B_n \quad (m \geq 0, n > 0) .$$

$A_1, \dots, A_m, B_1, \dots, B_n$  are all positive atoms which can be over the equality predicate  $=$ , (possibly inverse) roles, original atomic concepts or new atomic concepts introduced during the structural transformation. In addition, a body atom can also be a concrete domain atom, an atom of the form  $HU(x)$  which makes the rule safe, or an atom of the form  $S_f(x, x_f)$  which is introduced for eliminating the function symbol  $f$ .  $\Delta_{KB}$  is made up of  $HU(a)$ ,  $HU(a_f)$  and  $S_f(a, a_f)$ , instantiated for each individual  $a$  and each function symbol  $f$ .

To enable equality reasoning in disjunctive datalog, the equality predicate  $=$  is interpreted as a congruence relation and treated as an ordinary predicate, with required properties axiomatized explicitly [7, 8]. A disjunctive datalog program  $P$  with equality is thus transformed into a disjunctive datalog program  $P_=_$  without equality, by stating that  $=$  is reflexive, symmetric, and transitive, and by appending replacement rules of the form “ $R(x_1, \dots, y_i, \dots, x_n) \leftarrow R(x_1, \dots, x_i, \dots, x_n), x_i = y_i$ ”, instantiated for each distinct predicate  $R$  and each position  $i$ . In what follows, we assume that  $=$  has been treated as an ordinary predicate in  $DD(KB)$ , and with  $\models_c$  we denote the cautious entailment in positive disjunctive datalog programs.

**Theorem 1 ([7]).** *For  $KB$  a  $\mathcal{SHIQ}(\mathbf{D})$  knowledge base, (1)  $KB$  is unsatisfiable if and only if  $DD(KB)$  is unsatisfiable; (2)  $KB \models_c \alpha$  if and only if  $DD(KB) \models_c \alpha$  for each assertion  $\alpha$  of the form  $A(a)$  or  $S(a, b)$ , where  $A$  is an atomic concept and  $S$  a simple role.*

### 3 Partitioning the ABox

With the reduction,  $DD(KB)$ , of a  $\mathcal{SHIQ}(\mathbf{D})$  knowledge base  $KB$ , we can reduce the reasoning on  $KB$  to the reasoning on  $DD(KB)$ . In what follows, a ground atom is also called an *assertion*; an atom (or assertion) is called *basal* if it is over concrete domain predicates or the predicates in  $\Delta_{KB}$ ; an atom (or assertion) is called *normal* if it is not basal. A set  $S$  of assertions is said to *trigger* a rule  $R$  if  $S = \text{Body}(R\sigma)$  for some ground substitution  $\sigma$ . A set  $S$  of assertions is said to *indirectly trigger* a rule  $R$  in logic program  $\Pi$  if there is a set  $S'$  of assertions such that  $S \cup \Pi \models S'$  and  $S'$  triggers  $R$ . An assertion  $a$  is said to *participate in* the (indirect) triggering of  $R$  (in  $\Pi$ ) if there is a set  $S$  of assertions such that  $a \in S$  and  $S$  (indirectly) triggers  $R$  (in  $\Pi$ ). If  $\Pi$  is clear from the context, it is omitted. Consider  $\Pi_0$  in Example 1. We say  $\{b, c, d\}$  triggers  $r_1$ ,  $\{e, c, d\}$  indirectly triggers  $r_1$ , and  $e$  participates in the indirect triggering of  $r_1$ . If the logic program contains rules with disjunctions, such as  $r_3$  in  $\Pi_0$ , the indirect triggering is nondeterministic. We say a set  $S$  of assertions *might indirectly trigger*  $R$  in  $\Pi$  if  $S$  indirectly triggers  $R$  in a Horn logic program  $\Pi'$  which is converted from  $\Pi$  by replacing disjunctions with definite implications. Continue with Example 1, we say  $\{b, g, d\}$  might indirectly trigger  $r_1$  in  $\Pi_0$ .

*Example 1.* Let  $\Pi_0$  be a logic program consisting of the following ground rules.

$$\begin{aligned} r_1 : a \leftarrow b, c, d . \quad r_2 : b \leftarrow e . \quad r_3 : c \vee f \leftarrow g . \\ r_4 : c . \quad r_5 : d . \quad r_6 : e . \quad r_7 : g . \end{aligned}$$

We intend to partition  $KB_{\mathcal{A}}$  so that the subsequent reasoning on  $DD(KB)$  can be performed separately on each of its partitions. So we should avoid communication between partitions during reasoning on  $DD(KB)$ . Consider Example 1.  $\{c, d, e\}$  should be placed in the same partition, otherwise  $a$  cannot be independently entailed over any partition of  $\{c, d, e, g\}$ . This shows that sets of assertions that (might) indirectly trigger rules should be placed in the same partition. However, this intuition is too rough. Consider Example 1 again.  $\{d, e, g\}$  might indirectly triggers  $r_1$ , but we need not put  $g$  to the same partition where  $\{d, e\}$  locates, since  $\{c, d, e\}$  need be placed in the same partition and then  $a$  can be independently entailed over  $\{c, d, e\}$ . The intuition behind such case is that for two sets of assertions  $S_1$  and  $S_2$ , when  $S_1 \cup S_2$  (might) indirectly trigger  $R$  in logic program  $\Pi$ , i.e., there exists  $S'$  such that  $S_1 \cup S_2 \cup \Pi \models S'$  and  $S'$  triggers  $R$ ,  $S_1$  need not be placed in the same partition where  $S_2$  locates, if  $S_2 \cup \Pi \models S'$ .

To exploit above intuitions, we first convert  $DD(KB)$  to a plain datalog program  $H(KB)$  by replacing disjunctions with conjunctions and adding constraints for negative atoms. That is, a rule of the form “ $A_1 \vee \dots \vee A_m \leftarrow B_1, \dots, B_n (m > 0, n > 0)$ ” in  $DD(KB)$  is converted to “ $A_1 \wedge \dots \wedge A_m \leftarrow B_1, \dots, B_n$ ” and other rules in  $DD(KB)$  remain. In addition, we treat negative atoms in  $H(KB)$  as positive ones by adding constraints of the form “ $\leftarrow a, \neg a$ ” to  $H(KB)$  if  $\neg a \in H(KB)$ . We then separate  $H(KB)$  into the definite part  $H_1(KB)$ , which consists of rules with heads, and the constraint part  $H_0(KB)$ , which consists of rules without heads, for different treatments.

Our partitioning algorithm is shown in Figure 1. Some tracking data are used. For each normal assertion  $a$ , we use  $marked(a)$  to store whether  $a$  is marked. We *mark*  $a$ , i.e., set  $marked(a)$  to true, if and only if  $a$  participates in triggering rules in  $H(KB)$ . Secondly, we use  $parts(a)$  to store the set of identifiers of the parts where  $a$  locates, i.e.,  $parts(a) = \{id(p) | a \in p\}$ . Besides we put  $a$  to the merged part when  $a$  participates in triggering rules, we might put  $a$  to a part  $p$  if there is some  $b \in p$  *supporting*  $a$  through rule  $r \in ground(H(KB))$ , i.e.,  $a \in Head(r)$  and  $b \in Body(r)$ , according to the intuition that a set of assertions indirectly triggering rules should be placed in the same partition. However, as another intuition shows, when  $a$  is entailed over one of its parts,  $a$  need not be placed in the parts where its supporters locate. So  $parts(a)$  is unchanged in such case. We can see that the set of *marked* assertions in  $\bigcup_{id(p) \in parts(a)} p$  approximates the *support closure* of  $a$ . Thirdly, we use  $entailed(a)$  to store whether  $a$  is entailed by  $DD(KB)$ , i.e.,  $DD(KB) \models_c a$ . For efficiency, we use a simple recursive evaluation of  $entailed(a)$ , which is sound but incomplete. That is, if there exists a definite rule  $r \in ground(DD(KB))$  such that  $Head(r) = \{a\}$  and  $entailed(b) = true$  for all  $b \in Body(r)$ , we set  $entailed(a)$  to true.

There are some remarks on the merging procedure MergeParts. First, we merge parts instead of assertions for efficiency. Second, only when  $a$  participates in triggering rules can  $marked(a)$  be set to true. After  $marked(a)$  is set to true,  $parts(a)$  remains a single set. Third, for all  $a \in S_H$  with  $entailed(a) = false$ ,  $parts(a)$  is updated for enlarging the support closure of  $a$ : the support closures of all  $b \in S_B$  (approximated with  $id(p')$ ) are appended to the support closure of  $a$ , by adding  $id(p')$  to  $parts(a)$ , and  $a$  to  $p'$  correspondingly.

Our proposed method will always produce a disjoint and independent partitioning of the ABox (Theorem 2), which ensures that a query over atomic concepts or simple roles can be performed separately on each generated partition and the results trivially combined yield complete answers. Another benefit of our method is the ability to filter unmarked assertions to the unique *unmarked partition*  $p_U$  (other partitions are called *marked partitions* correspondingly). Unmarked assertions do not participate in triggering rules in  $DD(KB)$ , and thus the reasoning over  $p_U$  can be performed on a fragment of  $DD(KB)$  consisting of a kind of rules whose body has no normal atoms. This implies that reasoning over  $p_U$  is trivial.

**Lemma 1.** *Let  $KB$  be a  $SHIQ(\mathbf{D})$  knowledge base such that  $(KB_{\mathcal{T}}, KB_{\mathcal{R}}, \emptyset)$  is consistent,  $KB_{A,1}, \dots, KB_{A,n}$  the parts returned by  $PartitionABox(KB)$ . Then (1)  $\{KB_{A,1}, \dots, KB_{A,n}\}$  is a disjoint partitioning of  $KB_A$ ; (2)  $\bigcup_{i=1}^n M_i$  is a model of  $DD(KB) = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_A \cup \Delta_{KB}$  if for all  $i = 1, \dots, n$ ,  $M_i$  is a minimal model of  $DD(KB)_i = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{A,i} \cup \Delta_{KB}$ .*

*Proof Sketch.* (1) For all assertions  $a \in KB_A$ , if  $a$  is marked in PartitionABox, it is placed in a unique part, otherwise it is moved to the unmarked partition. So  $KB_{A,1}, \dots, KB_{A,n}$  is a disjoint partitioning of  $KB_A$ .

(2) Let  $\mathcal{C}$  be the set of all satisfiable basal assertions,  $\mathcal{D} = \{a \in lfp(H_1(KB)) | entailed(a) = true\}$  the set of all entailed normal assertions in the least fix-

**MergeParts**( $S_H, S_B, \mathcal{P}$ )

1. **for** each  $h \in S_H$  **with**  $parts(h)$  undefined **do**
2.    $parts(h) := \emptyset$ ;  $marked(h) := false$ ;  $entailed(h) := false$ ;
3. **if**  $S_H = \{h\}$  **then**  $entailed(h) := \bigwedge_{b \in S_B} entailed(b)$ ;
4.  $merge := \bigcup_{b \in S_B} parts(b) \cup \bigcup_{h \in S_H, marked(h)=true} parts(h)$ ;
5.  $p' := \bigcup_{id(p) \in merge} p$ ;  $\mathcal{P} := \mathcal{P} \cup \{id(p')\} - merge$ ;
6. **for** each  $b \in S_B$  **do**  $marked(b) := true$ ;
7. **for** each  $h \in S_H$  **with**  $marked(h) = false$  **do**
8.    $parts(h) := parts(h) \cup \{id(p')\} - merge$ ;  $p' := p' \cup \{h\}$ ;
9. **for** each  $a \in p'$  **do**
10.   **if**  $marked(a) = true$  **then**  $parts(a) := \{id(p')\}$ ;
11.   **else**  $parts(a) := parts(a) \cup \{id(p')\} - merge$ ;

**PartitionABox**( $KB$ )

**Input:** a  $\mathcal{SHIQ}(\mathcal{D})$  knowledge base  $KB$  such that  $(KB_{\mathcal{T}}, KB_{\mathcal{R}}, \emptyset)$  is consistent.

**Output:** the set of partitions of  $KB_{\mathcal{A}}$ .

1. **for** each  $a \in KB_{\mathcal{A}}$  **do**
2.    $p_a := \{a\}$ ;  $parts(a) := \{id(p_a)\}$ ;  $marked(a) := false$ ;  $entailed(a) := true$ ;
3.  $\mathcal{P} := \bigcup_{a \in KB_{\mathcal{A}}} parts(a)$ ; Compute the least fixpoint  $M_A$  of  $H_1(KB)$ ;
4. **Meanwhile for** each rule  $r \in ground(H_1(KB))$  such that all concrete domain atoms of  $Body(r)$  are satisfiable and all abstract domain atoms of  $Body(r)$  are in  $M_A$  **do**
5.   MergeParts( $\{h \in Head(r) | entailed(h) = false\}, \{b \in Body(r) | parts(b) \text{ is defined}\}, \mathcal{P}$ );
6. **for** each rule  $r \in ground(H_0(KB))$  such that all concrete domain atoms of  $Body(r)$  are satisfiable and all abstract domain atoms of  $Body(r)$  are in  $M_A$  **do**
7.   MergeParts( $\emptyset, \{b \in Body(r) | parts(b) \text{ is defined}\}, \mathcal{P}$ );
8.  $p_U := KB_{\mathcal{A}} \cap \bigcup_{id(p) \in \mathcal{P}} \{a \in p | marked(a) = false\}$ ;
9. **for** each part  $p$  such that  $id(p) \in \mathcal{P}$  **do**  $p := KB_{\mathcal{A}} \cap \{a \in p | marked(a) = true\}$ ;
10. **return**  $\{p \neq \emptyset | id(p) \in \mathcal{P}\} \cup p_U$ ;

**Fig. 1.** An algorithm for partitioning the ABox

point of  $H_1(KB)$ ,  $p_1, \dots, p_N$  all parts generated before line 8 in PartitionABox. W.l.o.g., we assume that  $KB_{\mathcal{A},n}$  is the unmarked partition, and  $KB_{\mathcal{A},i} = KB_{\mathcal{A}} \cap \{a \in p_i | marked(a) = true\}$  for all  $i < n$ . For all  $i < n$ , we show that  $M_i$  can be divided into layers  $L_1, \dots, L_m$  such that  $L_1 = KB_{\mathcal{A},i} \cup (M_i \cap \mathcal{C})$  and each assertion in  $L_{k+1}$  is supported by a set of assertions in  $\bigcup_{j=1}^k L_j$ . Then, we show that  $M_i \subseteq p_i \cup \mathcal{C} \cup \mathcal{D}$  by using induction on the layers of  $M_i$ , according to a fact that if there exists a rule  $r \in ground(DD(KB))$  such that  $a \in Head(r)$  and  $Body(r) \subseteq p_i \cup \mathcal{C} \cup \mathcal{D}$ , either  $a \in \mathcal{D}$  or  $a$  is put to  $p_i$  in MergeParts. Now, suppose  $M = \bigcup_{i=1}^n M_i$  is not a model of  $DD(KB)$ . There must be a rule  $r \in ground(DD(KB))$  such that  $Body(r) \subseteq M$  and  $Head(r) \cap M = \emptyset$ . Let  $b_1, \dots, b_m$  be all the normal assertions in  $Body(r)$ . Since  $M_n$  is a subset of the least fixpoint of the fragment of  $DD(KB)$  consisting of a kind of rules whose body has no normal atoms, any  $b_i$  cannot be in  $M_n$  and thus is marked in PartitionABox. On the other hand, since different parts do not together participate

in triggering rules,  $b_1, \dots, b_m$  are all in the same part, say  $p_k$ . Then, each  $b_i$  must be in  $M_k \cup \mathcal{D}$ , otherwise there is  $M_j$  ( $j \neq k$ ) such that  $b_i \in M_j - \mathcal{C} - \mathcal{D} \subseteq p_j$ , contradicting that  $b_i$  is marked. In case  $b_i \in \mathcal{D}$ ,  $b_i$  is entailed over  $p_k$  and thus  $b_i$  is in every model of  $DD(KB)_k$ . This implies that each  $b_i$  is in  $M_k$  and thus  $Hear(r) \cap M_k \neq \emptyset$ , contradicting that  $Head(r) \cap M = \emptyset$ .  $\square$

**Theorem 2 (independent partitioning).** *Let  $KB$  be a  $SHIQ(\mathcal{D})$  knowledge base such that  $(KB_{\mathcal{T}}, KB_{\mathcal{R}}, \emptyset)$  is consistent,  $\{KB_{\mathcal{A},1}, \dots, KB_{\mathcal{A},n}\}$  the disjoint partitioning of  $KB_{\mathcal{A}}$  returned by  $PartitionABox(KB)$ , and  $KB_i = (KB_{\mathcal{T}}, KB_{\mathcal{R}}, KB_{\mathcal{A},i})$  for all  $i = 1, \dots, n$ . Then (1)  $KB$  is consistent if and only if  $KB_i$  is consistent for all  $i = 1, \dots, n$ ; (2)  $KB \models \alpha$  if and only if there exists  $KB_i$  such that  $KB_i \models \alpha$  for each assertion  $\alpha$  of the form  $A(a)$  or  $S(a, b)$ , where  $A$  is an atomic concept and  $S$  a simple role.*

*Proof.* (1) The  $(\Rightarrow)$  direction is trivial. For the  $(\Leftarrow)$  direction, by Theorem 1,  $DD(KB)_i = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A},i} \cup \Delta_{KB}$  is satisfiable for all  $i = 1, \dots, n$ .  $DD(KB)_i$  is positive and thus has minimal models. Let  $M_i$  be a minimal model of  $DD(KB)_i$ . By Lemma 1,  $\bigcup_{i=1}^n M_i$  will be a model of  $DD(KB)$ . So  $KB$  is consistent by Theorem 1. (2) The  $(\Leftarrow)$  direction is trivial. For the  $(\Rightarrow)$  direction, we have  $DD(KB) \models_c \alpha$  by Theorem 1. Suppose there is no  $KB_i$  such that  $KB_i \models \alpha$ . Let  $DD(KB)_i = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A},i} \cup \Delta_{KB}$ . By Theorem 1, there exist minimal models  $M_1, \dots, M_n$  of  $DD(KB)_1, \dots, DD(KB)_n$  respectively such that  $\alpha \notin M_i$  for all  $i = 1, \dots, n$ . By Lemma 1,  $M = \bigcup_{i=1}^n M_i$  is a model of  $DD(KB)$ . That  $\alpha \notin M$  contradicts that  $DD(KB) \models_c \alpha$ .  $\square$

Regarding the complexity, we consider the *data complexity*, which is measured in  $|KB_{\mathcal{A}}|$  only, under the assumption that  $|KB_{\mathcal{T}\mathcal{R}}| = |KB_{\mathcal{T}}| + |KB_{\mathcal{R}}|$  is bounded by a constant. In addition, we assume that there is a polynomial oracle for reasoning with concrete domains and a constant bound on the arity of the concrete domain predicates. Since the number of rules and the number of atoms in each rule in  $\Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}})$ , and the computation time of  $\Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}})$  are all bounded by exponential of  $|KB_{\mathcal{T}\mathcal{R}}|$  [7], the number of different variables in each rule in  $H(KB)$  is bounded by a constant and the number of rules in  $ground(H(KB))$  is polynomial in  $|KB_{\mathcal{A}}|$ . Hence,  $PartitionABox$  runs in polynomial time in  $|KB_{\mathcal{A}}|$ .

## 4 Experimental Evaluation

We tested our partitioning method on top of the ontologies available on the KAON2 Web Site<sup>3</sup> and the Lehigh University Benchmark (LUBM) [6]. The implementation of the proposed method is based on secondary storage so as to handle large ABox data. Specifically, we used MySQL as the back-end DBMS. The input ABox data and the tracking data in run time are maintained in the database. We implemented the partitioning method in GNU C++, used the KAON2 system for the ontology reduction and performed testing on a 3.2GHz Pentium 4 CPU 2GB RAM machine running Windows XP.

<sup>3</sup> [http://kaon2.semanticweb.org/download/test\\_ontologies.zip](http://kaon2.semanticweb.org/download/test_ontologies.zip)

**Table 1.** Test results on the partition time and granularity

Test Set	#assertions	Partition Time (hh:mm:ss)	#filtered (i.e., unmarked) assertions	#marked partitions	Avg. marked partition size (#assertions)	Max. marked partition size (#assertions)
Wine-0	496	00:02:29	78	43	9.72	336
Vicodi-0	53,653	00:05:58	0	53,653	1.00	1
Semintec-0	65,240	00:07:06	4,552	48,166	1.26	2
LUBM-1	100,543	00:03:02	22,418	45,931	1.70	2,190
LUBM-10	1,273 K	02:00:24	285,844	575,703	1.71	2,362

Table 1 shows the test results. In the table, all ontologies except Wine-0 are in a Horn fragment, i.e., their reductions are plain datalog programs. The partition granularity on these Horn-fragment ontologies is fine: the average size of marked partitions is very small, and the maximum size of marked partitions is so small that all partitions can be easily manipulated in physical memory. The ontology Wine-0 is rather complex, whose reduction is a disjunctive datalog program with more than 500 rules. The partition granularity on Wine-0 is not so fine as others, but still acceptable, since the submaximum size of marked partitions is 21 (without shown in Table 1) and the average size of marked partitions is small.

## 5 Related Work and Conclusion

Guo and Hefin [5] develop a set of tableau rules for partitioning the ABox in *SHIF* knowledge bases. Their partitioning method uses an intuition that assertions in the antecedent of an inference rule should be placed in the same partition. To estimate implicit inference in polynomial time, [5] uses some approximate tableau rules, such as  $C_1 \sqsubseteq C_2$  for all concepts  $C_1$  and  $C_2$ . To reduce partition size, [5] generates overlapped parts instead of partitions which need be disjoint. Though the overlapped parts preserve the independent partitioning property as ours, the performance of the subsequent reasoning may be impaired due to introducing many duplicated assertions. For example, with 1,311K input LUBM data, [5] generates 396,197 parts with average size 21.2 and maximum size 1,141. The number of duplicated assertions are about 7,000K. As a comparison, with 1,273K input data (LUBM-10), our method generates smaller partitions in average (see in Table 1). Though the largest marked partition generated by our method is near two times larger than the largest one in [5], no partitions generated by our method overlap.

Fokoue *et al.* [3] develop some role filtering techniques for partitioning the (summary) ABox in *SHIN* knowledge bases. Their partitioning method filters role assertions whose absence will not affect the outcome of a consistency check, and then places assertions sharing individual names in the same partition. Due to different focuses, the partitioning method in [3] is rather restricted, it filters role assertions only, while our proposed method filters all kinds of assertions. Moreover, the filtering techniques proposed in [3] do not guarantee a subsequent inde-

pendent partitioning. As an example, consider a DL knowledge base  $KB$ , where  $KB_{\mathcal{T}} = \{A \sqsubseteq_{\leq 1} R\}$ ,  $KB_{\mathcal{R}} = \emptyset$  and  $KB_{\mathcal{A}} = \{A(a), R(a, b), R(a, c), S(a, b)\}$ .  $S(a, b)$  will be filtered using the method in [3]. Then, any generated partition cannot entail  $S(a, c)$  independently, while  $S(a, c)$  can be entailed by  $KB$ .

Grau *et al.* [4] propose  $\mathcal{E}$ -Connections as a formalism for representing combinations of OWL knowledge bases and an algorithm for decomposing an OWL knowledge base into connected components. Amir and McIlraith [1] present a greedy algorithm for decomposing a first-order logic theory into partitions and message passing algorithms for reasoning with the partitioned theory. Due to different goals, the partitioning produced by either [4] or [1] may not have the independent partitioning property. This is because both [4] and [1] generate partitions with potential links and the subsequent reasoning may require communication between partitions through the links.

We have presented a method for partitioning the ABox of a  $SHIQ(\mathbf{D})$  knowledge base, based on a conversion from  $SHIQ(\mathbf{D})$  to plain datalog. The primary advantage of our method is that it always produces a disjoint and independent partitioning, while all existing methods in the related work may not. For future work, we will continue to improve the performance of our partitioning method, conduct more investigations on handling nominals and complex roles, and extend the method to an incremental one for dealing with knowledge base updates.

## Acknowledgements

This work is supported in part by NSFC grants 60673103, 60421001 and 60373052.

## References

1. E. Amir and S. A. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artif. Intell.*, 162(1-2):49–88, 2005.
2. F. M. Donini. Complexity of reasoning. In *Description Logic Handbook*, pages 96–136. Cambridge University Press, 2003.
3. A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas. The summary abox: Cutting ontologies down to size. In *ISWC-06*, pages 343–356, 2006.
4. B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Automatic partitioning of owl ontologies using  $\mathcal{E}$ -connections. In *Description Logics*, 2005.
5. Y. Guo and J. Heflin. A scalable approach for partitioning owl knowledge bases. In *SSWS-06*, pages 47–60, 2006.
6. Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for owl knowledge base systems. *Journal of Web Semantics*, 3(2-3):158–182, 2005.
7. U. Hustadt, B. Motik, and U. Sattler. Reducing  $SHIQ^-$  description logic to disjunctive datalog programs. In *KR-04*, pages 152–162, 2004. (Extended version: Reasoning for Description Logics around  $SHIQ$  in a Resolution Framework. *FZI Technical Report 3-8-04/04*).
8. B. Motik, U. Sattler, and R. Studer. Query answering for owl-dl with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.

## Exploiting Conjunctive Queries in Description Logic Programs<sup>\*</sup>

Thomas Eiter<sup>1</sup>, Giovambattista Ianni<sup>1,2</sup>, Thomas Krennwallner<sup>1</sup>, and Roman Schindlauer<sup>1,2</sup>

<sup>1</sup> Institut für Informationssysteme 184/3, Technische Universität Wien  
Favoritenstraße 9-11, A-1040 Vienna, Austria

<sup>2</sup> Dipartimento di Matematica, Università della Calabria,  
I-87036 Rende (CS), Italy.

{eiter,ianni,tkren,roman}@kr.tuwien.ac.at

**Abstract.** We present cq-programs, which enhance nonmonotonic description logics (dl-) programs by conjunctive queries (CQ) and union of conjunctive queries (UCQ) over Description Logics knowledge bases, as well as disjunctive rules. dl-programs had been proposed as a powerful formalism for integrating nonmonotonic logic programming and DL-engines on a clear semantic basis. The new cq-programs have two advantages: First, they offer increased expressivity by allowing general (U)CQs in the body. And second, this combination of rules and ontologies gives rise to strategies for optimizing calls to the DL-reasoner, by exploiting (U)CQ facilities of the DL-reasoner. To this end, we discuss some equivalences which can be exploited for program rewriting. Experimental results for a cq-program prototype show that this can lead to significant performance improvements.

### 1 Introduction

Rule formalisms that combine logic programming with other sources of knowledge, especially terminological knowledge expressed in Description Logics (DLs), have gained increasing interest in the past years. This process was mainly fostered by current efforts in the Semantic Web development of designing a suitable rules layer on top of the existing ontology layer. Such combinations of DLs and logic programming can be categorized in systems with (i) strict semantic integration and (ii) strict semantic separation, which amounts to coupling heterogeneous systems [1–4]. In this paper, we will concentrate on the latter, considering ontologies as external information with semantics treated independently from the logic program. Under this category falls [5, 2], which extends the answer-set semantics to so-called *dl-programs*  $(L, P)$ , which consist of a DL part  $L$  and a rule part  $P$  that may query  $L$ . Such queries are facilitated by a special type of atoms, which also permit to enlarge  $L$  with facts imported from the logic program  $P$ , thus allowing for a bidirectional flow of information.

Since the semantics of logic programs is usually defined over a domain of explicit individuals, this approach may fail to derive certain consequences, which are implicitly contained in  $L$ . Consider a simplified version of an example from [6]:

<sup>\*</sup> This work has been partially supported by the EC NoE REVERSE (IST 506779) and the Austrian Science Fund (FWF) project P17212-N04.

$$L = \{father \sqsubseteq parent, \exists father. \exists father^- . \{Remus\}(Romulus), father(Cain, Adam), \\ father(Abel, Adam), hates(Cain, Abel), hates(Romulus, Remus)\},$$

$$P = \{BadChild(X) \leftarrow DL[parent](X, Z), DL[parent](Y, Z), DL[hates](X, Y)\}.$$

Apart from the explicit facts,  $L$  states that each *father* is also a *parent* and that Romulus and Remus have a common father. The single rule in  $P$  specifies that an individual hating a sibling is a *BadChild*. From this dl-program,  $BadChild(Cain)$  can be concluded, but not  $BadChild(Romulus)$ , though it is implicitly stated that *Romulus* and *Remus* have the same father.

The reason is that, in a dl-program, variables must be instantiated over their Herbrand base (containing the individuals in  $L$  and  $P$ ), and thus unnamed individuals like the father of Romulus and Remus, are not considered. In essence, dl-atoms only allow for building CQs that are *DL-safe* [6], which ensure that all variables in the query can be instantiated to named individuals. While this was mainly motivated by retaining decidability of the formalisms, unsafe CQs are admissible under certain conditions [1]. We thus pursue the following.

- We extend dl-programs by (U)CQs to  $L$  as first-class citizens in the language. In our example, to obtain the desired conclusion  $BadChild(Romulus)$ , we may use  $P' = \{BadChild(X) \leftarrow DL[parent(X, Z), parent(Y, Z), hates(X, Y)](X, Y)\}$ , where the body of the rule is a CQ  $\{parent(X, Z), parent(Y, Z), hates(X, Y)\}$  to  $L$  with distinguished variables  $X$  and  $Y$ .

*Example 1.* Both  $r = BadParent(Y) \leftarrow DL[parent](X, Y), DL[hates](Y, X)$  and  $r' = BadParent(Y) \leftarrow DL[parent(X, Y), hates(Y, X)](X, Y)$  equivalently pick (some of) the bad parents. Here, in  $r$  the join between *parent* and *hates* is performed in the logic program, while in  $r'$  it is performed on the DL-side.

Since DL-reasoners including RACER, KAON2, and Pellet increasingly support answering CQs, this can be exploited to push joins between the rule part and the DL-reasoner, eliminating an inherent bottleneck in evaluating cq-programs.

- We present equivalence-preserving transformation rules, by which rule bodies and rules involving cq- or ucq-atoms can be rewritten.
- We report on some experiments with a prototype implementation of cq-programs using dlvhx and RACER. They show the effectiveness of the rewriting techniques, and that significant performance increases can be gained. These results are interesting in their own right, since they shed light on combining conjunctive query results from a DL-reasoner.

## 2 dl-Atoms with Conjunctive Queries

We assume familiarity with Description Logics (cf. [7]), in particular  $\mathcal{SHIF}(\mathbf{D})$  and  $\mathcal{SHOIN}(\mathbf{D})$ .<sup>3</sup> A DL-KB  $L$  is a finite set of axioms in the respective DL. We denote logical consequence of an axiom  $\alpha$  from  $L$  by  $L \models \alpha$ .

As in [5, 2], we assume a function-free first-order vocabulary  $\Phi$  of nonempty finite sets  $\mathcal{C}$  and  $\mathcal{P}$  of constant resp. predicate symbols, and a set  $\mathcal{X}$  of variables. As usual, a *classical literal* (or *literal*),  $l$ , is an atom  $a$  or a negated atom  $\neg a$ .

<sup>3</sup> We focus on these DLs because they underly OWL-Lite and OWL-DL. Conceptually, cq-programs can be defined for other DLs as well.

**Syntax** A *conjunctive query* (CQ)  $q(\vec{X})$  is an expression  $\{ \vec{X} \mid Q_1(\vec{X}_1), Q_2(\vec{X}_2), \dots, Q_n(\vec{X}_n) \}$ , where each  $Q_i$  is a concept or role expression and each  $\vec{X}_i$  is a singleton or pair of variables and individuals, and where  $\vec{X} \subseteq \bigcup_{i=1}^n \text{vars}(\vec{X}_i)$  are its *distinguished* (or *output*) variables. A *union of conjunctive queries* (UCQ)  $q(\vec{X})$  is a disjunction  $\bigvee_{i=1}^m q_i(\vec{X})$  of CQs  $q_i(\vec{X})$ . Where it is clear from the context, we omit  $\vec{X}$  from (U)CQs.

*Example 2.* In our opening example,  $cq_1(X, Y) = \{ \text{parent}(X, Z), \text{parent}(Y, Z), \text{hates}(X, Y) \}$  and  $cq_2(X, Y) = \{ \text{father}(X, Y), \text{father}(Y, Z) \}$  are CQs with distinguished variables  $X, Y$ , and  $ucq(X, Y) = cq_1(X, Y) \vee cq_2(X, Y)$  is a UCQ.

We now define dl-atoms  $\alpha$  of form  $\text{DL}[\lambda; q](\vec{X})$ , where  $\lambda = S_1 \text{ op}_1 p_1, \dots, S_m \text{ op}_m p_m$ ,  $m \geq 0$ , is a list of expressions  $S_i \text{ op}_i p_i$ , where each  $S_i$  is either a concept or a role,  $\text{op}_i \in \{ \uplus, \cup, \cap \}$ , and  $p_i$  is a predicate symbol matching  $S_i$ 's arity, and where  $q$  is a (U)CQ with output variables  $\vec{X}$  (in this case,  $\alpha$  is called a *(u)cq-atom*), or  $q(\vec{X})$  is a dl-query. Each  $p_i$  is an *input predicate symbol*; intuitively,  $\text{op}_i = \uplus$  increases  $S_i$  by the extension of  $p_i$ , while  $\text{op}_i = \cup$  increases  $\neg S_i$ ;  $\text{op}_i = \cap$  constrains  $S_i$  to  $p_i$ .

*Example 3.* The cq-atom  $\text{DL}[\text{parent } \uplus p; \text{parent}(X, Y), \text{parent}(Y, Z)](X, Z)$  with output  $X, Z$  extends  $L$  by adding the extension of  $p$  to the property *parent*, and then joins *parent* with itself.

A *cq-rule*  $r$  is of the form  $a_1 \vee \dots \vee a_k \leftarrow b_1, \dots, b_m, \text{not } b_{m+1}, \dots, \text{not } b_n$ , where every  $a_i$  is a literal and every  $b_j$  is either a literal or a dl-atom. If  $n = 0$  and  $k > 0$ , then  $r$  is a *fact*. A *cq-program*  $KB = (L, P)$  consists of a DL-KB  $L$  and a finite set of cq-rules  $P$ .

**Semantics** For any CQ  $q(\vec{X}) = \{ Q_1(\vec{X}_1), Q_2(\vec{X}_2), \dots, Q_n(\vec{X}_n) \}$ , let  $\phi_q(\vec{X}) = \exists \vec{Y} \bigwedge_{i=1}^n Q_i(\vec{X}_i)$ , where  $\vec{Y}$  are the variables not in  $\vec{X}$ , and for any UCQ  $q(\vec{X}) = \bigvee_{i=1}^m q_i(\vec{X})$ , let  $\phi_q(\vec{X}) = \bigvee_{i=1}^m \phi_{q_i}(\vec{X})$ . Then, for (U)CQ  $q(\vec{X})$ , the set of *answers* of  $q(\vec{X})$  on  $L$  is the set of tuples  $\text{ans}(q(\vec{X}), L) = \{ \vec{c} \in \mathcal{C}^{|\vec{X}|} \mid L \models \phi_q(\vec{c}) \}$ .

Let  $KB = (L, P)$  be a cq-program. Given the semantics of (U)CQs on  $L$ , defining the semantics of cq- and ucq-atoms w.r.t. a Herbrand interpretation  $I$  of the predicates in  $P$  (using constants from  $\mathcal{C}$ ) in the same way as for dl-atoms is straightforward. We recall that a ground dl-atom  $a = \text{DL}[\lambda; Q](\vec{c})$  is satisfied w.r.t.  $I$ , denoted  $I \models_L a$ , if  $L \cup \lambda(I) \models Q(\vec{c})$ , where  $\lambda(I) = \bigcup_{i=1}^m A_i$  and

- $A_i(I) = \{ S_i(\vec{e}) \mid p_i(\vec{e}) \in I \}$ , for  $\text{op}_i = \uplus$ ;
- $A_i(I) = \{ \neg S_i(\vec{e}) \mid p_i(\vec{e}) \in I \}$ , for  $\text{op}_i = \cup$ ;
- $A_i(I) = \{ \neg S_i(\vec{e}) \mid p_i(\vec{e}) \in I \text{ does not hold} \}$ , for  $\text{op}_i = \cap$ .

Now, given a ground instance  $a(\vec{c})$  of a (u)cq-atom  $a(\vec{X}) = \text{DL}[\lambda; q](\vec{X})$  (i.e., all variables in  $q(\vec{X})$  are replaced by constants),  $I$  satisfies  $a(\vec{c})$ , denoted  $I \models_L a(\vec{c})$ , if  $\vec{c} \in \text{ans}(q(\vec{X}), L \cup \lambda(I))$ . The notion of model and (strong) answer set of  $KB$  is then defined as usual (cf. [5, 2]).

*Example 4.* Let  $KB = (L, P)$ , where  $L$  is the well-known wine ontology<sup>4</sup> and  $P$  is as follows ( $P$  uses only atomic queries and may launch our rewritings):

<sup>4</sup> <http://www.w3.org/TR/owl-guide/wine.rdf>

$$\begin{aligned}
 v(L) \vee \neg v(L) &\leftarrow \text{DL}[\text{WhiteWine}](W), \text{DL}[\text{RedWine}](R), \text{DL}[\text{locatedIn}](W, L), \\
 &\quad \text{DL}[\text{locatedIn}](R, L), \text{not DL}[\text{locatedIn}(L, L')](L). \\
 &\leftarrow v(X), v(Y), X \neq Y. \quad c \leftarrow v(X). \quad \leftarrow \text{not } c. \\
 \text{del}(W) &\leftarrow \text{DL}[\text{hasFlavor}](W, \text{wine:Delicate}). \\
 \text{del}_r(W) &\leftarrow v(L), \text{del}(W), \text{DL}[\text{locatedIn}](W, L).
 \end{aligned}$$

Informally, the first rule picks a largest region in which both red and white wine grow, and the next three rules make sure that exactly one such region is picked. The last rules choose the delicate wines in the region selected for visit.

$KB$  has the following 3 strong answer sets (only positive facts from predicates  $\text{del}_r$  and  $v$  are listed):  $\{\text{del}_r(\text{MountadamRiesling}), v(\text{AustralianRegion}), \dots\}$ ,  $\{\text{del}_r(\text{LaneTannerPinotNoir}), \text{del}_r(\text{WhitehallLanePrimavera}), v(\text{USRegion}), \dots\}$ , and  $\{\text{del}_r(\text{StonleighSauvignonBlanc}), v(\text{NewZealandRegion}), \dots\}$ .

The example in the introduction shows that cq-programs are more expressive than dl-programs in [5, 2]. Furthermore, answer set existence for  $KB$  and reasoning from the answer sets of  $KB$  is decidable if (U)CQ-answering on  $L$  is decidable, which is feasible for quite expressive DLs including  $\mathcal{SHIQ}$  and fragments of  $\mathcal{SHOIN}$ , cf. [8–10]. Rosati’s well-known  $\mathcal{DL}+log$  formalism [11, 1], and the more expressive hybrid MKNF knowledge bases [12, 13] are closest in spirit to dl- and cq-programs, since they support nonmonotonic negation and use constructions from nonmonotonic logics. However, their expressiveness seems to be different from dl- and cq-programs. It is reported in [12] that dl-programs (and hence also cq-programs) can not be captured using MKNF rules. In turn, the semantics of  $\mathcal{DL}+log$  inherently involves deciding containment of CQs in UCQs, which seems to be inexpressible in cq-programs.

### 3 Rewriting Rules for cq- and ucq-Atoms

As shown in Ex. 1, in cq-programs we might have different choices for defining the same query. Indeed, the rules  $r$  and  $r'$  are equivalent over any DL-KB  $L$ . However,  $r'$  performs the join on the DL side in a single call to the DL-reasoner, while  $r$  performs the join on the logic program side, over the results of two calls to the DL-reasoner. In general, making more calls is more costly, and thus  $r'$  may be computationally preferable. Furthermore, the result transferred by the single call in  $r'$  is smaller than the results of the two calls.

Towards exploiting such rewriting, we present some transformation rules for replacing a rule or a set of rules in a cq-program with another rule or set of rules, while preserving the semantics of the program (see Table 1). By (repeated) application of these rules, the program can be transformed into another, equivalent program. Note that ordinary dl-atoms  $\text{DL}[\lambda; Q](\vec{t})$ , may be replaced by equivalent cq-atoms  $\text{DL}[\lambda; Q(\vec{t})](\vec{X})$ , where  $\vec{X} = \text{vars}(\vec{t})$ , to facilitate rewriting.

**Query Pushing (A)** By this rule, cq-atoms  $\text{DL}[\lambda; cq_1](\vec{Y}_1)$  and  $\text{DL}[\lambda; cq_2](\vec{Y}_2)$  in the body of a rule (A1) can be merged. In rule (A2),  $cq'_1$  and  $cq'_2$  are constructed

Query Pushing

$$r : a_1 \vee \dots \vee a_k \leftarrow \text{DL}[\lambda; cq_1](\vec{Y}_1), \text{DL}[\lambda; cq_2](\vec{Y}_2), B. \quad (\text{A1})$$

$$r' : a_1 \vee \dots \vee a_k \leftarrow \text{DL}[\lambda; cq'_1 \cup cq'_2](\vec{Y}_1 \cup \vec{Y}_2), B. \quad (\text{A2})$$

where  $B = b_1, \dots, b_m$ , not  $b_{m+1}, \dots$ , not  $b_n$ .

(In)equality Pushing

$$r : a_1 \vee \dots \vee a_h \leftarrow \text{DL}[\lambda; cq](\vec{Y}), Y_{i_1} \neq Y_{i_2}, \dots, Y_{i_{2k-1}} \neq Y_{i_{2k}}, \quad (\text{B1})$$

$$Y_{i_{2k+1}} = Y_{i_{2k+2}}, \dots, Y_{i_{2l-1}} = Y_{i_{2l}}, B.$$

$$r' : a_1 \vee \dots \vee a_h \leftarrow \text{DL}[\lambda; cq' \cup \{Y_{i_1} \neq Y_{i_2}, \dots, Y_{i_{2k-1}} \neq Y_{i_{2k}}\}](\vec{Y}), B. \quad (\text{B2})$$

where each  $Y_{i_j} \in \vec{Y}$  for  $1 \leq j \leq 2l$ , and  $B = b_1, \dots, b_m$ , not  $b_{m+1}, \dots$ , not  $b_n$ .

Fact Pushing

$$\bar{P} = \left\{ \begin{array}{l} f(\vec{c}_1), f(\vec{c}_2), \dots, f(\vec{c}_l), \\ a_1 \vee \dots \vee a_k \leftarrow \text{DL}[\lambda; \bigvee_{i=1}^r cq_i](\vec{Y}), f(\vec{Y}'), B. \end{array} \right\} \quad (\text{C1})$$

$$\bar{P}' = \left\{ \begin{array}{l} f(\vec{c}_1), f(\vec{c}_2), \dots, f(\vec{c}_l), \\ a_1 \vee \dots \vee a_k \leftarrow \text{DL}[\lambda; \bigvee_{i=1}^r (\bigvee_{j=1}^l cq_i \cup \{Y^j = \vec{c}_j\})](\vec{Y}), B. \end{array} \right\} \quad (\text{C2})$$

$\vec{c}_1, \dots, \vec{c}_l$  are ground tuples,  $\vec{Y}' \subseteq \vec{Y}$ , and  $B = b_1, \dots, b_m$ , not  $b_{m+1}, \dots$ , not  $b_n$ .

Unfolding

$$\bar{P} = \left\{ \begin{array}{l} r_1 : a_1 \vee \dots \vee a_i \leftarrow a'(\vec{Y}), B_1. \\ r_2 : H' \vee a'(\vec{Y}') \leftarrow B_2. \end{array} \right\} \quad (\text{D1})$$

$$\bar{P}' = \bar{P} \cup \{ r'_1 : H' \theta \vee a_1 \theta \vee \dots \vee a_i \theta \leftarrow B_2 \theta, B_1 \theta. \} \quad (\text{D2})$$

$H' = a'_1 \vee \dots \vee a'_j$ , and  $\theta$  is the mgu of  $a'(\vec{Y})$  and  $a'(\vec{Y}')$  (thus  $a'(\vec{Y}\theta) = a'(\vec{Y}'\theta)$ );

Where  $a'(\vec{Y})$  is not unifiable with  $a'(\vec{Z}) \in H(r_1) \cup H'$ , alternatively  $\bar{P}' = \{r'_1, r_2\}$ .

**Table 1.** Equivalences

by renaming variables in  $cq_1$  and  $cq_2$  as follows. Let  $\vec{Z}_1$  and  $\vec{Z}_2$  be the non-distinguished (i.e., existential) variables of  $cq_1$  resp.  $cq_2$ . Rename each  $X \in \vec{Z}_1$  occurring in  $cq_2$  and each  $X \in \vec{Z}_2$  occurring in  $cq_1$  to a fresh variable.

Query Pushing can be similarly done when one or both of  $cq_1, cq_2$  is a UCQ  $ucq_1$  resp.  $ucq_2$ ; here, we simply distribute the subqueries and form a single UCQ.

**Pushing of (In)equalities (B)** If the DL-engine is used under the unique name assumption and supports (in)equalities in the query language, we can easily rewrite rules with equality (=) or inequality ( $\neq$ ) in the body by pushing it to the cq-query. A rule of form (B1) can be replaced by (B2), where the CQ  $cq'$  results from  $cq$  by collapsing variables according to the equalities  $Y_{i_{2k+1}} = Y_{i_{2k+2}}, \dots, Y_{i_{2l-1}} = Y_{i_{2l}}$ .

*Example 5.* Consider rule  $r = \text{bigwinery}(M) \leftarrow \text{DL}[\text{Wine}](W_1), \text{DL}[\text{Wine}](W_2), W_1 \neq W_2, \text{DL}[\text{hasMaker}](W_1, M), \text{DL}[\text{hasMaker}](W_2, M)$ . Here, we want to know all wineries producing at least two different wines. We can rewrite  $r$ , by Query and Inequality Pushing, to the rule  $r'$

$$r' : \text{bigwinery}(M) \leftarrow \text{DL} \left[ \begin{array}{l} \text{Wine}(W_1), \text{Wine}(W_2), W_1 \neq W_2 \\ \text{hasMaker}(W_1, M), \text{hasMaker}(W_2, M) \end{array} \right] (M, W_1, W_2).$$

A similar rule works for a ucq-atom  $\text{DL}[\lambda; ucq](\vec{Y})$  in place of  $\text{DL}[\lambda; cq](\vec{Y})$ .

**Fact Pushing (C)** Suppose we have a program with “selection predicates”, i.e., facts which serve to select a specific property in a rule. We can push such facts into a ucq-atom and remove the selection atom from the rule body.

*Example 6.* Consider the program  $P$ , where we only want to know the children of *joe* and *jill*:  $P = \{f(joe), f(jill), fchild(Y) \leftarrow DL[isFatherOf](X, Y), f(X)\}$ . We may rewrite the program to a more compact one with the help of ucq-atoms:

$$P' = \left\{ fchild(Y) \leftarrow DL \left[ \begin{array}{l} \{isFatherOf(X, Y), X = joe\} \vee \\ \{isFatherOf(X, Y), X = jill\} \end{array} \right] (X, Y). \right\}$$

Such a rewriting makes sense in situations where *isFatherOf* has many tuples and thus would lead to transfer all known father child relationships.

**Unfolding (D)** Unfolding rules is a standard-method for partial evaluation of ordinary logic programs under answer set semantics. It can be also applied in the context of cq-programs, with no special adaptation. After folding rules with (u)cq-atoms in the body into other rules, subsequent Query Pushing might be applied. In this way, inference propagation can be shortcut.

The following results state that the rewritings preserve equivalence. Let  $P \equiv_L Q$  denote that  $(L, P)$  and  $(L, Q)$  have the same answer sets.

**Theorem 1.** For an  $X \in \{A, B\}$  let  $r$  and  $r'$  be rules of form (X1) and (X2), respectively. Let  $(L, P)$  be a cq-program with  $r \in P$ . Then,  $P \equiv_L (P \setminus \{r\}) \cup \{r'\}$ .

**Theorem 2.** Let  $\bar{P}$  be a set of cq-rules of form (C1) (resp. (D1)) and  $\bar{P}'$  be a set of cq-rules of form (C2) (resp. (D2)). Then,  $\bar{P} \equiv_L \bar{P}'$ . For any set of cq-rules  $P$  such that  $\bar{P} \subseteq P$ ,  $P \equiv_L (P \setminus \bar{P}) \cup \bar{P}'$ , where for (D2)  $\bar{P}' = \{r_1, r_2, r'_1\}$ .

Based on these rules, we have developed an optimization algorithm, described in the extended version of this paper.<sup>5</sup> Further, more general rewriting rules (e.g., incorporating cost models) can be conceived, which we omit for space reasons.

## 4 Experimental Results

We have tested the rule transformations using the prototype implementation of the DL-plugin for *dlvhex*,<sup>6</sup> a logic programming engine featuring higher-order syntax and external atoms (see [14]), which uses RACER 1.9 as DL-reasoner (cf. [15]). To our knowledge, this is currently the only implemented system for such a coupling of nonmonotonic logic programs and Description Logics.

The tests were done on a P4 3GHz PC with 1GB RAM under Linux 2.6. As an ontology benchmark, we used the testsuite described in [16]. The experiments covered particular query rewritings (see Table 2) and version of the region program (Ex. 4) with the optimizations applied. We report only part of the results, shown in Fig. 1. Missing entries mean memory exhaustion during the evaluation.

In most of the tested programs, the performance boost using the aforementioned optimization techniques was substantial. Due to the size of the respective

<sup>5</sup> <http://www.kr.tuwien.ac.at/staff/roman/papers/dlopt-ext.pdf>

<sup>6</sup> <http://www.kr.tuwien.ac.at/research/dlvhex/>

VICODI program: (Fact Pushing)

$$P_v = \{c(\text{vicodi:Economics}), c(\text{vicodi:Social}), v(X) \leftarrow \text{DL}[\text{hasCategory}](X, Y), c(Y).\}$$

SEMINTEC query: (Query Pushing)

$$P_{s_2} = \left\{ s_2(X, Y, Z) \leftarrow \begin{array}{l} \text{DL}[\text{Man}](X), \text{DL}[\text{isCreditCard}](Y, X), \text{DL}[\text{Gold}](Y), \\ \text{DL}[\text{livesIn}](X, Z), \text{DL}[\text{Region}](Z) \end{array} \right\}$$

SEMINTEC costs: (Query Pushing, Functional Property)

$$P_l = \{l(X, Y) \leftarrow \text{DL}[\text{hasLoan}](X, Y), \text{DL}[\text{Finished}](Y).\}$$

*hasLoan* is an inverse functional property and  $|\text{hasLoan}| = 682(n + 1)$ ,  $|\text{Finished}| = 234(n + 1)$ , where  $n$  is obtained from the ontology instance SEMINTEC $_n$ .

**Table 2.** Some test queries

ontologies, in some cases the DL-engine failed to evaluate the original dl-queries, while the optimized programs did terminate with the correct result.

In detail, for the region program (upper left graph), we used the ontologies wine $_0$  through wine $_9$ . There is a significant speedup, and in case of wine $_9$  only the optimized program could be evaluated. Most of the computation time was spent by RACER. We note that the result of the join in the body of the first rule had a size merely linear in the number of top regions  $L$ ; a higher performance gain may be expected for ontologies with larger joins.

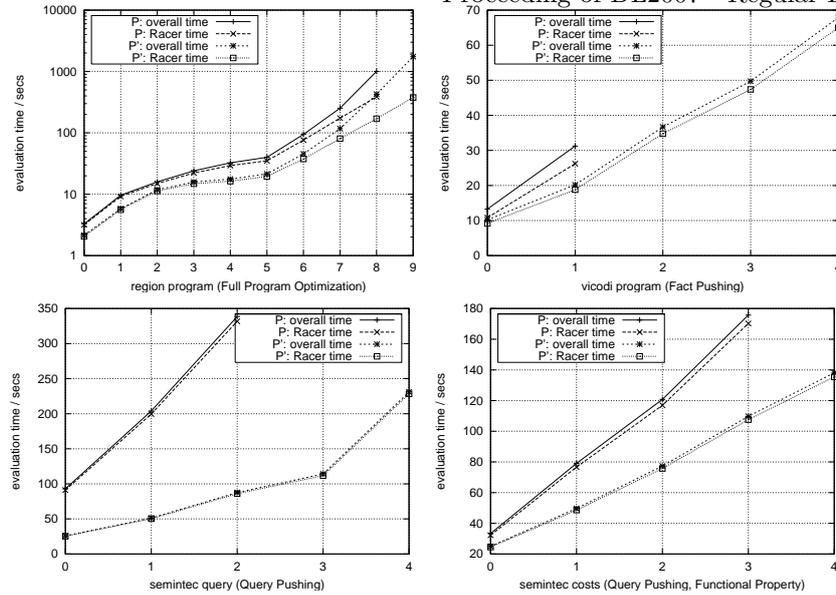
The VICODI test series revealed the power of Fact Pushing (see the upper right graph). While the unoptimized  $P_v$  could be evaluated only with VICODI $_0$  and  $_1$ , all ontologies VICODI $_i$ ,  $0 \leq i \leq 4$ , could be handled with the optimized program.

The SEMINTEC tests dealt with Query Pushing and show a significant evaluation speedup (see lower row of Fig. 1).  $P_{s_2}$  is from one of the benchmark queries in [16], while  $P_l$  tests the performance increase when pushing a query to a functional property. In both cases, we used the ontologies SEMINTEC $_i$ ,  $0 \leq i \leq 4$ , but could only complete the tests of the optimized programs on all SEMINTEC $_n$ . The performance gain for  $P_l$  is in line with the constant join selectivity.

Future work will be to compare to realizations of cq-programs based on other DL-engines which host CQs, such as Pellet and KAON2, and to enlarge and refine the rewriting techniques.

## References

1. Rosati, R.: Integrating Ontologies and Rules: Semantic and Computational Issues. In: Reasoning Web. LNCS 4126, Springer (2006) 128–151
2. Eiter, T., Ianni, G., Polleres, A., Schindlauer, R., Tompits, H.: Reasoning with Rules and Ontologies. In: Reasoning Web. LNCS 4126, Springer (2006) 93–127
3. Antoniou, G., Damásio, C.V., Grosz, B., Horrocks, I., Kifer, M., Maluszynski, J., Patel-Schneider, P.F.: Combining Rules and Ontologies: A survey. Tech. Rep. IST506779/Linköping/I3-D3/D/PU/a1, Linköping University (2005)
4. Pan, J.Z., Franconi, E., Tessaris, S., Stamou, G., Tzouvaras, V., Serafini, L., Horrocks, I., Glimm, B.: Specification of Coordination of Rule and Ontology Languages. Project Deliverable D2.5.1, KnowledgeWeb NoE (2004)



**Fig. 1.** Evaluation time for Ex. 4, VICODI, and SEMINTEC tests.

5. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with Description Logics for the Semantic Web. In Dubois, D., et al. eds.: Proceedings KR 2004, Morgan Kaufmann (2004) 141–151
6. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with Rules. *Journal of Web Semantics* **3** (2005) 41–60
7. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press (2003)
8. Ortiz de la Fuente, M., Calvanese, D., Eiter, T.: Data Complexity of Answering Unions of Conjunctive Queries in SHIQ. In: Proceedings DL 2006 (2006) 62–73
9. Ortiz de la Fuente, M., Calvanese, D., Eiter, T., Franconi, E.: Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics. In: Proceedings AAAI-2006, AAAI Press (2006)
10. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive Query Answering for the Description Logic *SHIQ*. In: Proc. IJCAI 2007, AAAI Press (2007) 399–404
11. Rosati, R.: *DL+log*: Tight Integration of Description Logics and Disjunctive Datalog. In: Proceedings KR 2006, AAAI Press (2006) 68–78
12. Motik, B., Horrocks, I., Rosati, R., Sattler, U.: Can OWL and Logic Programming live together happily ever after? In: Proceedings ISWC-2006. LNCS 4273, Springer (2006) 501–514
13. Motik, B., Rosati, R.: A faithful Integration of Description Logics with Logic Programming. In: Proceedings IJCAI 2007, AAAI Press/IJCAI (2007) 477–482
14. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming. In: Proceedings IJCAI-2005, Professional Book Center (2005) 90–96
15. Haarslev, V., Möller, R.: RACER System Description. In: Proceedings IJCAR-2001. LNCS 2083, Springer (2001) 701–705
16. Motik, B., Sattler, U.: A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In Hermann, M., Voronkov, A., eds.: LPAR. LNCS 4246, Springer (2006) 227–241

## OntoVQL: A Graphical Query Language for OWL Ontologies

Amineh Fadhil, Volker Haarslev

Concordia University, Department of Computer Science & Software Engineering,  
1455 de Maisonneuve Blvd. W., Montreal, Quebec, Canada  
{f\_amineh,haarslev}@cs.concordia.ca

**Abstract.** The database usability experience has shown that visual query languages tend to be superior to textual languages in many aspects. By applying this principle in the context of ontologies, we present OntoVQL, a graphical query language for OWL-DL ontologies. OntoVQL maps diagrammatic queries to DL based query languages such as nRQL, which is offered by the OWL-DL reasoner Racer. OntoVQL hides the complexity of the DL query language from users and allows them to query OWL ontologies with less difficulty. A visual query system equipped with this language has been implemented and is now available. This tool enables users to formulate queries incrementally by having more than one query simultaneously available for getting combined or broken down into new queries. Giving instant feedback in the form of result cardinality is another important feature of the tool that helps guiding users into building meaningful queries.

**Keywords:** ontology, owl, graphical query language.

### 1 Introduction

Ontologies occupy an increasingly important role in the domain of knowledge management and information systems. Although critical work in ontology editing and visualization has been done to assist the ontology engineer, not much has been accomplished for the domain expert or naïve user, i.e. someone who is expert in his domain of study but naïve in the sense of lacking the necessary logical background or technical skills for querying an ontology. Note that ontology querying in this article consists of ABox retrieval. Intuitively, this type of user would like to easily design meaningful queries. However, the actual state of the art translates into submitting a DL-query, written in a query language with a Lisp based syntax, to a DL-reasoner. OntoIQ [7] represents an attempt of solving this issue by providing a query interface offering the basic functionalities of nRQL [6] through some query patterns. Although the latter aids in the querying process, its predefined queries limits the querying power of the user.

The importance of an effective and easy to use query system has been recognized in the database area. In fact, “the database area has proved to be particularly fruitful for applying visual techniques specifically in accessing stored data.” [8]. Since ontologies can be considered as an extended and more powerful way of representing

knowledge than databases, it naturally follows that whatever usability experience was gained from database visual querying would be relevant in the context of ontologies. There has been an extensive work done in that field and many visual query systems (VQS) employing various visual representations and interaction strategies have been proposed in the literature. A VQS is composed of a visual query language (VQL) for pictorially expressing queries and of some functionality for facilitating the human-computer interaction [8]. A VQL is a subclass of Visual Languages that serves for data extraction from databases. Historically, database interfaces have moved from being textual (ex. SQL) to form-based, diagrammatic (ex. Gql [12]) and iconic. Many VQLs took advantage of the graph representation of databases to express queries since queries are graph patterns that are searched in the database graph [13]. There exists some work that has been done towards visually querying ontologies but these attempts are not designed for querying OWL ontologies and they are not readily available for use. However, there exists some work done for ontology based information seeking. For instance, some visual query interfaces (ex. SEWASIE [14]) were developed to allow information retrieval from heterogeneous data sources through an integrated ontology. This paper presents OntoVQL, a formal graphical query language for OWL ontologies that is an effort to apply the principles of a formal VQL. This work is an extension to our previous work, GLOO [2] which mainly consisted of a design study but due to some practical implementation/time constraint issues along with a few design imperfections, some of its properties were dropped and others modified and thus OntoVQL emerged as a new better version of GLOO.

## 2 Presenting the VQL

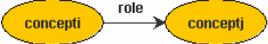
### 2.1 Visual Notations

Since VQLs are mainly based on the idea of directly manipulating database visualization as a graph by selecting the parts that are to be included in the query, then by analogy, given that the TBox of an ontology can also be visually represented by a graph, it is logical to consider that in order to formulate a query for ABox retrieval, one has to select “parts” of the TBox, i.e. concepts and roles, as components of the query. The query is thus composed of graph components and therefore can be formulated as a graph on its own. This would be one way of understanding the logistics behind viewing an ABox query as a graph. The other way is based on the nature of ABox querying itself. For instance, the starting point for ABox retrieval is the extraction of all instances of a concept. Then, one step further from getting a set of individuals based on a concept name is to learn how these individuals are related with other individuals by the mean of edges. In that sense, it is natural to construct query constraints in the form of a graph.

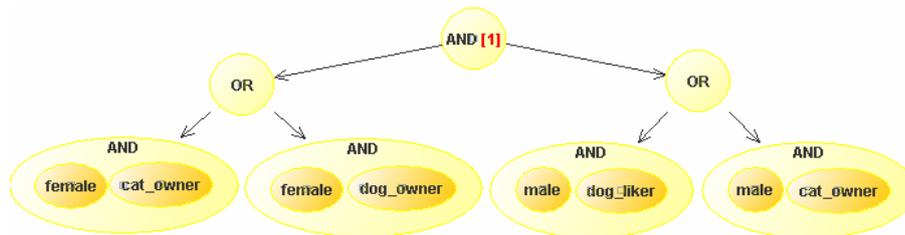
The visual notations of the basic elements of OntoVQL illustrated in Table 1 partially follow the conventions described in [1]. Concepts are represented by their name inside of a filled oval whereas individuals are represented by their name inside of a filled rectangle. The oval with ‘?’ is called an “unknown concept” and is equivalent

to owl:Thing? from the OWL language since it represents the instances that are not restricted to belong to a specific concept and can be of any type. Roles are most conveniently represented by the role name and an arrow going from one entity to another. Note that what is at stake in this representation is only the shape whereas the color is of no importance.

**Table 1.** Visualization of the basic elements of OntoVQL.

<concept>	
<unknown concept>	
<individual>	
<role>	

The operators used for assembling building blocks into a query are intersection and union. There are two kinds of intersection and union operators: AND/OR groups and AND/OR nodes. A group is represented by a circle labeled with the AND/OR keyword and encloses the intersected or unified concepts. In Figure 1, the union operator represented by the OR node is connected with directed edges to AND groups in order to symbolize the union of intersected concepts. Similarly, intersection can be applied on OR groups by connecting them to an AND node. The alternation between union and intersection operators takes the form of a tree where the root is a AND/OR node and the leaves are AND/OR groups.



**Fig. 1.** Alternation of AND/OR operators.

## 2.2 Visual Syntax

In linguistics and computer science, a generative grammar is a formal grammar that provides a precise description of a formal language. The grammar is composed by a set of rules dictating the syntax of the language, i.e., expressing how strings in a language are generated. We have adapted this principle in the context of generating a visual language instead of a textual one in order to generate OntoVQL by a formal grammar or more precisely by an adjusted version of a formal grammar. As a result,

since OntoVQL can be generated by a formal grammar and is also semantically and syntactically unambiguous, it can be viewed as a formal visual query language.

What is of primary importance that has a direct impact on the semantics of OntoVQL resides in its “connectivity syntax”, i.e., the way its components are linked together. This visual syntax can be expressed in terms of rules adopting the same form as a production rule in a formal grammar. Thus, similar to what a typical formal grammar is composed of, our grammar has a finite set of non-terminal symbols: {<Query>, <ROLE>, <AND GROUP>, <OR GROUP>, <AND NODE>, <OR NODE>, <entity>}, a finite set of terminal symbols: {<concept>, <individual>, <unknown concept>, <role>}, each of which has its visual equivalence as shown in Table 1, and a finite set of production rules that are listed below.

The rules dictate how the visual entities are allowed to be connected together by directed connection edges and roles. The first rule in the grammar below indicates that a query can simply be a concept or consists of more complex components. Among these components, the role is described in the second rule as a binary component where the domain and range are entities that can be expanded to any of the elements listed in the third rule. It is important to mention for the second rule that if a query has the shape of tree as in Figure 1, no role can link any of the query elements under a distinct “branch” of the tree. For example, none of the AND groups in Figure 1 can be linked by a role. The fourth and fifth rules illustrate the AND/OR group containing one or more concepts that are either intersected or unified. Note that an XML like syntax is used for representing the notion of a group by having a start and end tag enclosing one or more concept element. From rule 6, an AND node can be connected to an OR group and/or to an OR node and/or to an unknown concept that is linked to a role. A directed connection edge whose source is the AND/OR node and target is the AND/OR group or the unknown concept links the domain and range entities together.

Grammar generating OntoVQL:

- (1) <Query> -> <concept> | <ROLE> | <AND GROUP> | <AND NODE> | <OR GROUP> | <OR NODE>
- (2) <ROLE> -> <entity> <role> <entity>
- (3) <entity> -> <concept> | <unknown concept> | <individual> | <AND GROUP> | <OR GROUP> | <AND NODE> | <OR NODE>
- (4) <AND GROUP> -> <and group> <concept>\* </and group>
- (5) <OR GROUP> -> <or group> <concept>\* </or group>
- (6) <AND NODE> -> <and node>(<OR GROUP>\* | <OR NODE>\* | (<unknown concept><role><entity>)\*)
- (7) <OR NODE> -> <or node>(<AND GROUP>\* | <AND NODE>\* | (<unknown concept><role><entity>)\*)

As an illustrative example of how the above grammar describes the visual syntax of OntoVQL, consider the query in Figure 2. The query is composed of an OR node

that links an AND group with an unknown concept related to another unknown concept by a role R1. Note that the query's visual syntax corresponds to the 7<sup>th</sup> rule.

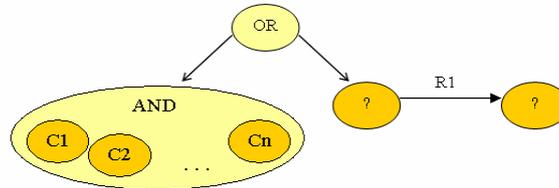


Fig. 2. An example query generated by the 7<sup>th</sup> rule of the grammar.

### 2.3 Mapping the VQL to the nRQL Language

The expressive power of OntoVQL can be informally described as being able to formulate queries on DL Abox elements (concepts and role assertions) and make use of conventional operators (union, intersection) for building up more complex, refined queries. Our proposed VQL is designed independently of any OWL query language and offers basic functionalities for querying OWL-DL ontologies. This means that there is not a one-to-one mapping between the visual components of OntoVQL and the elements of an OWL query language. Note that a number of these elements have no visual counterpart and therefore, OntoVQL does not match the full expressive power of OWL query languages. For instance, datatypes and negation are features that are not available in OntoVQL but are provided by the textual query language we are mapping to, nRQL. Even though OntoVQL is less expressive, for the simple queries mostly asked by the naïve user, the OntoVQL version of the query exhibits a lower complexity by mainly getting rid of the textual syntax and hiding the query variables of the OWL query. In that sense, we claim that OntoVQL is less complex than an OWL query language.

Table 2. Example of a graphical query and its equivalence in nRQL.

<p>Query 100 (2)</p>
<pre>(retrieve ( \$?x2 \$?x3 \$?x1 )   (and ( \$?x2  http://cohse.semanticweb.org/ontologies/people#animal      ( \$?x1  http://cohse.semanticweb.org/ontologies/people#animal        (or ( \$?x3  http://cohse.semanticweb.org/ontologies/people#woman          ( \$?x3  http://cohse.semanticweb.org/ontologies/people#man )         ( \$?x3 \$?x1  http://cohse.semanticweb.org/ontologies/people#has_pet )         ( \$?x3 \$?x2  http://cohse.semanticweb.org/ontologies/people#likes )))</pre>

It is possible to translate the visually expressed query semantics into a written version using the query language of your choice. In our case, the translation is done using nRQL. nRQL, its syntax is implemented by an optimized OWL-DL query processor known to be highly effective and efficient. A nRQL query is composed of a query head and of a query body. The query body consists of the query expression. The query head corresponds to the projected variables, that is, variables that are mentioned in the body and will be bound to the result. A tag number in brackets visually represents this variable projection. For instance, from the above query, there are three variables in the head that correspond to the three tag numbers. This “tagging” is not inherently part of OntoVQL as such but is provided for the purpose of integrating the projection feature into the VQS. Note that the “tagging” process is not up to the user and is automatically taken care of by the system.

Although no AND operator is present in the visual query, all the elements in the corresponding nRQL query are intersected. This is an implicit conjunction in contrast to the explicit one that is visualized by an AND node/group. Therefore, outgoing roles from an entity as well as domain and range specification for a role requires the use of conjunction in nRQL.

In order to be able to combine or break down queries, it must be possible to visualize more than one query simultaneously, each query having its own scope of variables. When mapping to nRQL, each concept must be mapped to a unique variable. In the case of AND/OR groups, the concepts inside the group must be mapped to the same variable. Therefore, in the nRQL translation, the concepts inside the OR group are both mapped to x3 whereas the animal concepts are mapped to different variables x1 and x2.

The reason why concepts involved in an AND/OR operation must be mapped to the same variable is because this has a direct impact on the result. For example if concepts C3 and C2 were intersected, then the nRQL query body will include (and (x1 C3) (x1 C2)) which results into binding x1 to those individuals who are C3 and C2. If distinct variables were used instead, then the answer would be those individuals who are C3, plus those individuals who are C2 but not already mentioned for C3. Hence, the semantics changes into adding up C2 and C3 individuals instead of intersecting them. This is why variable mapping is crucial when translating the visual query into nRQL.

## 2.5 Presenting the visual query system

Having an actual implementation of OntoVQL helps to evaluate its usefulness. Figure 2 shows a screenshot of the system’s main window that is split into two parts, the information tabs and the query tabs. The information tabs allow viewing the list of concepts, roles and individuals whereas the query tabs provide the query formulation pane, the query translation as well as the results.

Before starting to use the system for querying, the user must load some OWL ontology. Once the OWL file is loaded, the user can drag and drop concepts, roles or individuals into the query pane. Each dropped element becomes a distinct query and these simple queries can then be combined by using roles, connection edges, AND/OR groups and nodes. Only permitting those connections that follow the

grammar rules enforces the connectivity syntax. Thus, the user can only formulate legal queries. Deleting connection edges and roles allows breaking down complex queries into simple ones. The undo/redo pattern is implemented to facilitate query formulation. It is also possible to save queries into an XML file and load them later for further use.

Each query in the system is identified by a query number that uniquely distinguishes it from the others. One important aspect of the tool is the query preview that indicates the number of tuples returned in the result. The number next to the query identifier indicates the result cardinality. Whenever a new query is created, it is instantly evaluated and its preview set to the obtained number of tuples. Giving instant feedback plays a significant role in guiding the user into formulating meaningful queries. For instance, it would be pointless to combine concepts in an AND group when their preview is zero.

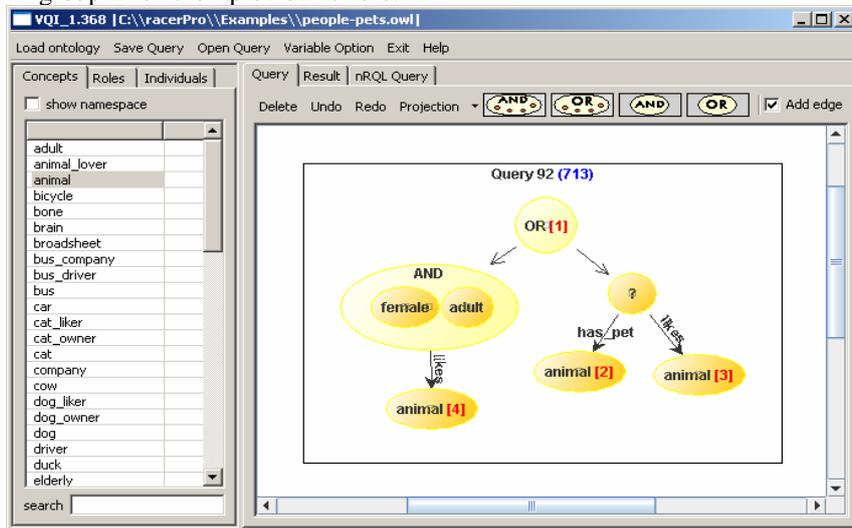


Fig. 2. Screenshot of the actual implementation of OntoVQL.

### 3 Conclusion

In conclusion, we proposed OntoVQL as a visual query language for OWL-DL ontologies and have implemented a tool for its practical use. OntoVQL is meant to be a better alternative over the traditional OWL query languages for the naïve user. In fact, it provides numerous advantages over a textual query language such as nRQL by eliminating its textual syntax with visual simplification, not allowing syntactic errors through the user interface (UI); simplifying the breaking down and merging of queries; and assisting users in the querying process through the system features such as providing immediate feedback with result cardinalities. The implementation of the UI has been completed and is available for download

([http://users.encs.concordia.ca/~f\\_amineh/](http://users.encs.concordia.ca/~f_amineh/)). A usability test for evaluating the tool's efficiency is underway and its results should be posted as soon as they are ready.

**Acknowledgments.** This work was supported in part by Genome Quebec, Natural Sciences and Engineering Research Council (NSERC) of Canada and by ENCS, Concordia University.

## References

- 1 Gaines, R.B.: An Interactive Visual Language for Term Subsumption Languages. In Proc. Of the 12<sup>th</sup> Int. Joint Conf. of Artificial Intelligence (IJCAI'91), pages 817-823, 1991.
- 2 A. Fadhil and V. Haarslev, GLOO: A Graphical Query Language for OWL ontologies. OWL: Experience and Directions 2006, Athens, 2006. Catarci, T., Costabile, M. F., Levialdi, S., Batini, C.: Visual Query Systems for Databases: A Survey. *Journal of Visual Languages and Computing*, 8(2), 215-260, 1997.
- 3 Larkin I., Simon, H.: Why a diagram is (sometimes) worth then thousand words. *Cognitive Science*, Vol. II 1987, pp:65-99.
- 4 Catarci, T., Dongilli, P., Mascio, T.D., Franconi, E., Santucci, G., Tessaris, S. : An Ontology Based Visual Tool for Query Formulation Support. In ECAI, pages 308-312, 2004.
- 5 Haarslev, V., Moeller, R., Wessel, M.: Querying the Semantic Web with Racer + nRQL. *CEUR Workshop Proceedings of KI-2004 Workshop on Applications of Description Logics (ADL 04)*, Ulm, Germany, Sep 24 2004.
- 6 Christopher, J. O. B., Xiao, S., Greg, B., Volker, H.: Ontoligent Interactive Query Tool. *Proceedings of the Canadian Semantic Web Working Symposium*, June 6, 2006, Quebec City, Quebec, Canada, Series: Semantic Web and Beyond: Computing for Human Experience, Vol. 2, Springer Verlag, 2006, pp. 155-169.
- 7 Catarci, T., Guisepppe, S., Michele, A.: Fundamental Graphical Primitives for Visual Query Languages. *Information Systems*, Vol. 18 n.2, p.75-98, March 1993.
- 8 Ni, W., Ling, T. W.: Glass: A Graphical Query Language for Semi-Structured Data. In Proc. Of Eighth International Conference on Database Systems for Advanced Applications (DASFAA '03), March 2003.
- 9 Chamberlin, D., Florescu, D., Robie, J., Simeon, J., Stefanescu, M.: Xquery: A Query Language for XML. Working draft, World Wide Web Consortium, February 2001. See <http://www.w3.org/TR/xquery/>.
- 10 Baker, P.G., Brass, A., Bechhofer, S., Goble, C., Paton, N., Stevens, R.: TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. 6<sup>th</sup> Int. Conf. on Intelligent Systems for Molecular Biology.
- 11 Suh, B., Bederson, B.: OZONE: A Zoomable Interface for Navigating Ontology. HCIL Tech Report #2001-04, University of Maryland, College Park, MD 20742.
- 12 Papantonakis, A., King, P.J.H.: Gql, a Declarative Graphical Query Language Based on the Functional Data Model. Proc. Of the Workshop on Advanced Visual Interfaces, Bari, Italy, 1994, pp.113-122.
- 13 Consens, M.P., Mendelzon, A.O.: GraphLog: A Visual Formalism for Real Life Recursion. In Proceedings of 9<sup>th</sup> ACM SIGA CT-SIGMOID Symposium on Principles of Database Systems, Pages 404-416, 1990.
- 14 Catarci, T., Dongilli, P., Mascio, T.D., Franconi, E., Santucci, G., Tessaris, S. : An Ontology Based Visual Tool for Query Formulation Support. In ECAI, pages 308-312, 2004.

## Induction of Optimal Semi-distances for Individuals based on Feature Sets

Nicola Fanizzi, Claudia d’Amato, Floriana Esposito

LACAM – Dipartimento di Informatica – Università degli Studi di Bari  
Campus Universitario, Via Orabona 4 – 70125 Bari, Italy  
{fanizzi|claudia.damato|esposito}@di.uniba.it

**Abstract.** Many activities related to semantically annotated resources can be enabled by a notion of similarity among them. We propose a method for defining a family of semi-distances over the set of individuals in a knowledge base which can be used in these activities. In the line of works on distance-induction on clausal spaces, the family is parameterized on a committee of concepts. Hence, we also present a method based on the idea of simulated annealing to be used to optimize the choice of the best concept committee.

### 1 Introduction

Recently, a growing interest is being committed to alternative inductive procedures extending the scope of the methods that can be applied to concept representations. Some are based on a notion of similarity such as *case-based reasoning* [6], *retrieval* [5, 7], *inductive generalization* [10] and *conceptual clustering* [8] or *ontology matching* [16].

As pointed out in a seminal paper [2] concerning similarity in Description Logics (DL), most of the existing measures focus on the similarity of atomic concepts within simple hierarchies. Besides, alternative approaches are based on related notions of *feature* similarity or *information content* (see also [14]). All these approaches have been specifically aimed at assessing concept similarity. In the perspective of crafting similarity-based inductive methods for DL, the need for a definition of a semantic similarity measure for *individuals* arises, that is a problem that so far received less attention in the literature.

Recently, some dissimilarity measures for individuals in specific DL representations have been proposed which turned out to be practically effective for the targeted inductive tasks (e.g. the nearest-neighbor approach applied to retrieval [5]). Although these measures ultimately rely on the semantics of primitive concepts as elicited from the ABox, still they are partly based on structural criteria (a notion of normal form coupled with a *most specific concept* operator [1]) which determine also their main weakness: they are hardly scalable to deal with standard languages used in the current knowledge management frameworks. For example, in [5] the most specific concepts w.r.t. the ABox of individuals are first computed (or their approximations in a normal form) as expressed in  $\mathcal{ALC}$ , then a structural measure assesses the similarity of the resulting AND-OR trees, where, ultimately, the computation is based on the extensions of the primitive concepts in the leaves.

Therefore, we have devised a new family of dissimilarity measures for semantically annotated resources, which can overcome the aforementioned limitations. Our measures are mainly based on Minkowski’s measures for Euclidean spaces induced by means of a proper method developed in the context of *multi-relational learning* [15]. Another source of inspiration was *rough sets* theory [13] which aims at the formal definition of vague sets (concepts) by means of their approximations determined by an *indiscernibility* relationship.

Namely, the measures are based on the degree of discernibility of the input individuals with respect to a committee of features, which are represented by concept descriptions expressed in DL. One of the advantages of these measures is that they do not rely on a particular language for semantic annotations. As such, these new measures are not absolute, since they depend on both the choice (and cardinality) of the features committee and the knowledge base they are applied to. Rather, they rely on statistics on individuals that are likely to be maintained by knowledge base management systems [9, 4], which can determine a potential speed-up in the measure computation during knowledge-intensive tasks. Furthermore, we also propose a way to extend the presented measures to the case of assessing concept similarity by means of the notion of *medoid* [11], i.e., in the DL context, the most centrally located individual in a concept extension w.r.t. a given metric.

Experimentally, it may be shown that the measures induced by large committees (e.g. including all primitive and defined concepts) can be sufficiently accurate (i.e. properly discriminating) when employed for classification tasks even though the committee of features employed were not the optimal one or if the concepts therein were partially redundant. Nevertheless, this has led us to investigate on a method to optimize the committee of features that serve as dimensions for the computation of the measure. To this purpose, the employment of genetic programming and randomized search procedures was considered. Finally we opted for an optimization procedure based on *simulated annealing* [3], a randomized approach that can overcome the problem of the local minima, i.e. finding a good solution w.r.t. the fitness function that is not globally optimal.

The remainder of the paper is organized as follows. The definition of the family of measures is proposed in Sect. 2, where we prove them to be semi-distances. In Sect. 3, we illustrate and discuss the method for optimizing the choice of concepts for the committee of features which induces the measures. Possible developments are finally examined in Sect. 4.

## 2 A Family of Semi-distances for Individuals

In the following, we assume that resources, concepts and their relationship may be defined in terms of a generic DL language endowed with the standard descriptive semantics (see the handbook [1] for a thorough reference).

For the measure definition, we simply consider a *knowledge base*  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  containing a *TBox*  $\mathcal{T}$  and an *ABox*  $\mathcal{A}$ . The set of the individuals occurring in  $\mathcal{A}$  will be denoted with  $\text{Ind}(\mathcal{A})$ .

As regards the inference services, our measures require (non)membership queries performing ABox lookups or *instance-checking* [1]. The complexity depends on the

DL of choice, however much of the computational effort can be saved by means of pre-computation (see projection functions below).

## 2.1 A Family of Measures for Individuals

We focus on the problem of assessing the semantic similarity (or dissimilarity) of individuals in the context of a knowledge base expressed in DL. To the best of our knowledge, only few measures tackle this problem so far [5]. Following some ideas borrowed from machine learning [15], a family of totally semantic distance measures for individuals can be defined in the context of a knowledge base.

It can be observed that individuals lack a syntactic structure that may be exploited for a comparison. However, on a semantic level, similar individuals should *behave* similarly with respect to the same concepts, i.e. similar assertions should be shared by them. Therefore, we introduce novel dissimilarity measures for individuals, whose rationale is the comparison of their semantics w.r.t. a fixed number of dimensions represented by DL concept descriptions. Namely, individuals are compared on the grounds of their behavior w.r.t. a reduced (yet not necessarily disjoint) committee of features, represented by a collection of concept descriptions, say  $F = \{F_1, F_2, \dots, F_m\}$ , which stands as a group of discriminating *features* expressed in the language taken into account.

In its simple formulation, a family of semi-distance functions for individuals, inspired by Minkowski’s metrics, can be defined as follows:

**Definition 1 (family of measures).** Let  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  be a knowledge base. Given a set of concept descriptions  $F = \{F_1, F_2, \dots, F_m\}$ , a family  $\{d_p^F\}_{p \in \mathbb{N}}$  of functions  $d_p^F : \text{Ind}(\mathcal{A}) \times \text{Ind}(\mathcal{A}) \mapsto [0, 1]$  is defined as follows:

$$\forall a, b \in \text{Ind}(\mathcal{A}) \quad d_p^F(a, b) := \frac{1}{m} \left[ \sum_{i=1}^m |\pi_i(a) - \pi_i(b)|^p \right]^{1/p}$$

where  $\forall i \in \{1, \dots, m\}$  the  $i$ -th projection function  $\pi_i$  is defined by:

$$\forall a \in \text{Ind}(\mathcal{A}) \quad \pi_i(a) = \begin{cases} 1 & \mathcal{K} \models F_i(a) \\ 0 & \mathcal{K} \models \neg F_i(a) \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

The superscript  $F$  will be omitted when the set of features is fixed.

As an alternative, especially when a good number of assertions are available in the ABox, the measures can be approximated by defining the projection functions based on a simple ABox look-up:

$$\forall a \in \text{Ind}(\mathcal{A}) \quad \pi_i(a) = \begin{cases} 1 & F_i(a) \in \mathcal{A} \\ 0 & \neg F_i(a) \in \mathcal{A} \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

## 2.2 Discussion

It is easy to prove that these functions have the standard properties for semi-distances:

**Proposition 1 (semi-distance).** *For a fixed feature set and  $p > 0$ , function  $d_p$  is a semi-distance.*

*Proof.* In order to prove the thesis, given any three individuals  $a, b, c \in \text{Ind}(\mathcal{A})$  it must hold that:

1.  $d_p(a, b) \geq 0$  (positivity)
2.  $d_p(a, b) = d_p(b, a)$  (symmetry)
3.  $d_p(a, c) \leq d_p(a, b) + d_p(b, c)$  (triangular inequality)

Now, we observe that:

1. trivial, by definition
2. trivial, for the commutativity of the operators involved
3. the property follows for the properties of the power function:

$$\begin{aligned}
 d_p(a, c) &= \frac{1}{m} \left[ \sum_{i=1}^m |\pi_i(a) - \pi_i(c)|^p \right]^{1/p} \\
 &= \frac{1}{m} \left[ \sum_{i=1}^m |\pi_i(a) - \pi_i(b) + \pi_i(b) - \pi_i(c)|^p \right]^{1/p} \\
 &\leq \frac{1}{m} \left[ \sum_{i=1}^m |\pi_i(a) - \pi_i(b)|^p + \sum_{i=1}^m |\pi_i(b) - \pi_i(c)|^p \right]^{1/p} \\
 &= \frac{1}{m} \left[ \sum_{i=1}^m |\pi_i(a) - \pi_i(b)|^p + \sum_{i=1}^m |\pi_i(b) - \pi_i(c)|^p \right]^{1/p} \\
 &\leq \frac{1}{m} \left[ \sum_{i=1}^m |\pi_i(a) - \pi_i(b)|^p \right]^{1/p} + \frac{1}{m} \left[ \sum_{i=1}^m |\pi_i(b) - \pi_i(c)|^p \right]^{1/p} \\
 &= d_p(a, b) + d_p(b, c)
 \end{aligned}$$

As such, these are only a semi-distances. Namely, it cannot be proved<sup>1</sup> that  $d_p(a, b) = 0$  iff  $a = b$ . This is the case of *indiscernible* individuals with respect to the given set of features F.

The underlying idea for the measure is that similar individuals should exhibit the same behavior w.r.t. the concepts in F. Here, we make the assumption that the feature-set F may represent a sufficient number of (possibly redundant) features that are able to discriminate really different individuals.

It could be criticized that the subsumption hierarchy has not been explicitly involved. However, this may be actually yielded as a side-effect of the possible partial redundancy of the various concepts, which has an impact on their extensions and thus

<sup>1</sup> In case the *unique names assumption* were made, a further projection function can be introduced  $\pi_0$ , such that  $|\pi_0(a) - \pi_0(b)| = 1$  iff  $a \neq b$ .

on the related projection function. A tradeoff is to be made between the number of features employed and the computational effort required for computing the related projection functions.

Compared to other distance (or dissimilarity) measures [2, 5], the presented functions do not depend on a specific language. Note that the computation of projection functions  $\pi_i$  ( $i = 1, \dots, m$ ) on the individuals can be performed in advance (with the support of KBMS [9, 4]) thus determining a speed-up in the actual computation of the measure. This is very important for the measure integration in algorithms which massively use this distance, such as case-based reasoning and all other instance-based methods including clustering algorithms.

Following the rationale of the average link criterion used in agglomerative clustering [11], the measures can be extended to the case of concepts, by recurring to the notion of medoids. The *medoid* of a group of individuals is the individual that has the highest similarity w.r.t. the others. Formally, given a group  $G = \{a_1, a_2, \dots, a_n\}$ , the medoid is defined:

$$\text{medoid}(G) = \underset{a \in G}{\operatorname{argmin}} \sum_{j=1}^n d(a, a_j)$$

Now, given two concepts  $C_1, C_2$ , we can consider the two corresponding groups of individuals obtained by retrieval  $R_i = \{a \in \text{Ind}(\mathcal{A}) \mid \mathcal{K} \models C_i(a)\}$ , and their resp. medoids  $m_i = \text{medoid}(R_i)$  for  $i = 1, 2$  w.r.t. a given measure  $d_p^F$  (for some  $p > 0$  and committee  $F$ ). Then we can define the function for concepts as follows:

$$d_p^F(C_1, C_2) := d_p^F(m_1, m_2)$$

### 3 Feature Set Optimization

Experimentally, we obtained satisfactory results<sup>2</sup> by testing the measure on distance-based classification. Nevertheless, various optimizations of the measures can be foreseen as concerns their parametric definition. Specifically, the choice of the concepts to be included in the committee – *feature selection* – will be examined. Among the possible committees, those that are able to better discriminate the individuals in the ABox ought to be preferred:

**Definition 2 (good feature set).** Let  $F = \{F_1, F_2, \dots, F_m\}$  be a set of concept descriptions. We call  $F$  a good feature set for the knowledge base  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  iff  $\forall a, b \in \text{Ind}(\mathcal{A}), a \neq b : \exists i \in \{1, \dots, m\} : \pi_i(a) \neq \pi_i(b)$ .

Note that, when the function defined above adopts a good feature set, it has the properties of a metric on the related instance-space.

Since the function strongly depends on the choice of concepts included in the committee of features  $F$ , two immediate heuristics can be derived:

<sup>2</sup> Results omitted for lack of space. They are available in technical reports and papers to appear. See <http://lacam.di.uniba.it:8000/people/nicola.html>.

1. controlling the number of concepts of the committee (which has an impact also on efficiency), including especially those that are endowed with a real discriminating power;
2. finding optimal sets of discriminating features of a given cardinality, by allowing also their composition employing the specific constructors made available by the DL language of choice.

Both these heuristics can be enforced by means of suitable machine learning techniques especially when knowledge bases with large sets of individuals are available. Namely, part of the entire data can be drawn in order to induce optimal F sets, in advance with respect to the application of the measure for all purposes. The adoption of genetic programming has been considered for constructing optimal sets of features. Yet these algorithms are known to suffer from being possibly caught in local minima. An alternative may consist in employing a different probabilistic search procedure which aims at a global optimization. Thus a method based on simulated annealing [3] has been devised, whose algorithm is reported in Fig. 1.

Essentially the algorithm searches the space of all feature sets starting from an initial guess (determined by MAKEINITIALFS( $\mathcal{K}$ )) based on the concepts (both primitive and defined) currently referenced in the knowledge base. The loop controlling the search is repeated for a number of times that depends on the temperature which gradually decays to 0, when the current feature committee can be returned. Meanwhile, this set is iteratively refined calling a suitable procedure RANDOMSUCCESSOR(). Then the fitness of the new feature set is compared to that of the current one determining the increment of energy  $\Delta E$ . If this is positive then the candidate committee replaces the current one. Otherwise it will (less likely) be replaced with a probability that depends on  $\Delta E$ .

As regards the heuristic FITNESSVALUE(F), it can be computed as the average *discernibility factor* [13] of the individuals w.r.t. the feature set. For example, given a set of individuals  $IS = \{a_1, \dots, a_n\} \subseteq \text{Ind}(\mathcal{A})$  (the whole or just a sample of  $\text{Ind}(\mathcal{A})$  used to induce an optimal measure) the fitness function may be defined:

$$\text{FITNESSVALUE}(F) = k \cdot \sum_{1 \leq i < j \leq n} \sum_{k=1}^{|F|} |\pi_k(a_i) - \pi_k(a_j)|$$

where  $k$  is a normalization factor which may be set to:  $(1/m)(n \cdot (n - 1)/4 - n)$ , which depends on the number of couples of different individuals that really determine the fitness measure.

As concerns finding candidates to replace the current committee, the function RANDOMSUCCESSOR() can be implemented by recurring to simple transformations of the feature set:

- adding (resp. removing) a concept  $C$ :  $\text{nextFS} \leftarrow \text{currentFS} \cup \{C\}$   
(resp.  $\text{nextFS} \leftarrow \text{currentFS} \setminus \{C\}$ )
- randomly choosing one of the current concepts from currentFS, say  $C$ ;  
replacing it with one of its refinements  $C' \in \text{REF}(C)$

Refining concept descriptions is language-dependent. For the case of  $\mathcal{ALC}$  logic, refinement operators have been proposed in [12, 10]. Complete operators are to be preferred to ensure exploring the whole search-space

```

FeatureSet OPTIMIZEFEATURESET( $\mathcal{K}$ ,  $\Delta T$ )
input  $\mathcal{K}$ : Knowledge base
         $\Delta T$ : function controlling the decrease of temperature
output FeatureSet
static currentFS: current Feature Set
        nextFS: next Feature Set
        Temperature: controlling the probability of downward steps
begin
currentFS  $\leftarrow$  MAKEINITIALFS( $\mathcal{K}$ )
for  $t \leftarrow 1$  to  $\infty$  do
    Temperature  $\leftarrow$  Temperature  $- \Delta T(t)$ 
    if (Temperature = 0)
        return currentFS
    nextFS  $\leftarrow$  RANDOMSUCCESSOR(currentFS, $\mathcal{K}$ )
     $\Delta E \leftarrow$  FITNESSVALUE(nextFS)  $-$  FITNESSVALUE(currentFS)
    if ( $\Delta E > 0$ )
        currentFS  $\leftarrow$  nextFS
    else // replace FS with given probability
        REPLACE(currentFS, nextFS,  $e^{\Delta E}$ )
end

```

**Fig. 1.** Feature Set optimization based on a Simulated Annealing procedure.

Given a suitable cooling schedule, the algorithm is known to find an optimal solution. To control the complexity of the process alternate schedules may be preferred that guarantee the construction of suboptimal solutions in polynomial time [3].

## 4 Conclusion and Extensions

We have proposed the definition of a family of semi-distances over the individuals in a DL knowledge base. The measures are not language-dependent yet they are parameterized on a committee of concepts. Therefore, we have also presented a randomized search method to find optimal committees. One of the advantages of the measures is that they are not language-dependent differently from previous proposals [5]. As previously mentioned, the subsumption relationships among concepts in the committee is not explicitly exploited in the measure for making the relative distances more accurate. The extension to the case of concept distance may also be ameliorated.

The measure may have a wide range of application of distance-based methods to knowledge bases. They have been integrated in an instance-based learning system implementing a nearest-neighbor learning algorithm: an experimentation on performing semantic-based retrieval proved the effectiveness of the new measures, compared to the outcomes obtained adopting other measures [5]. The next step concerns exploiting the measures in a conceptual clustering algorithm where clusters will be formed by grouping instances on the grounds of their similarity, possibly triggering the induction of new emerging concepts, as in [8].

## References

- [1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
- [2] A. Borgida, T.J. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Working Notes of the 2005 International Description Logics Workshop, DL2005*, volume 147 of *CEUR Workshop Proceedings*. CEUR, 2005.
- [3] E.K. Burke and G. Kendall. *Search Methodologies*. Springer, 2005.
- [4] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics, DL2005*, volume 147 of *CEUR Workshop Proceedings*. CEUR, 2005.
- [5] C. d’Amato, N. Fanizzi, and F. Esposito. Reasoning by analogy in description logics through instance-based learning. In G. Tummarello, P. Bouquet, and O. Signore, editors, *Proceedings of Semantic Web Applications and Perspectives, 3rd Italian Semantic Web Workshop, SWAP2006*, volume 201 of *CEUR Workshop Proceedings*, Pisa, Italy, 2006.
- [6] M. d’Aquin, J. Lieber, and A. Napoli. Decentralized case-based reasoning for the Semantic Web. In Y. Gil, V. Motta, E. Benjamins, and M. A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference, ISWC2005*, number 3279 in LNCS, pages 142–155. Springer, 2005.
- [7] N. Fanizzi, C. d’Amato, and F. Esposito. Instance-based retrieval by analogy. In *Proceedings of the 22nd Annual ACM Symposium of Applied Computing, SAC2007*, volume 2, pages 1398–1402, Seoul, South Korea, 2007. ACM.
- [8] N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro. Concept formation in expressive description logics. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning, ECML2004*, volume 3201 of *LNAI*, pages 99–113. Springer, 2004.
- [9] I. R. Horrocks, L. Li, D. Turi, and S. K. Bechhofer. The instance store: DL reasoning with large numbers of individuals. In V. Haarslev and R. Möller, editors, *Proceedings of the 2004 Description Logic Workshop, DL 2004*, volume 104 of *CEUR Workshop Proceedings*, pages 31–40. CEUR, 2004.
- [10] L. Iannone, I. Palmisano, and N. Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.
- [11] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
- [12] J. Lehmann. Concept learning in description logics. Master’s thesis, Dresden University of Technology, 2006.
- [13] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, 1991.
- [14] A. Rodriguez. *Assessing semantic similarity between spatial entity classes*. PhD thesis, University of Maine, 1997.
- [15] M. Sebag. Distance induction in first order logic. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming, ILP97*, volume 1297 of *LNAI*, pages 264–272. Springer, 1997.
- [16] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, IV:146–171, 2005.

## On relating heterogeneous elements from different ontologies

Chiara Ghidini<sup>1</sup>, Luciano Serafini<sup>1</sup>, and Sergio Tessaris<sup>2</sup>

<sup>1</sup> FBK-irst. Via Sommarive 18 Povo, 38050, Trento, Italy

<sup>2</sup> Free University of Bozen - Bolzano. Piazza Domenicani 3. 39100 Bolzano, Italy  
ghidini@itc.it    serafini@itc.it    tessaris@inf.unibz.it

### 1 Introduction

The majority of formalisms for distributed ontology integration based on the p2p architecture provide *mapping languages* able to express semantic relations between concepts of different ontologies. These formalisms can express that a concept  $C$  in Ontology 1 is equivalent (less/more general than) a concept  $D$  in Ontology 2 (see [13] for a survey). Few mapping languages allow also to express semantic relations between properties [8, 6], and thus state that a relation  $R$  in Ontology 1 is equivalent (less/more general than) a relation  $S$  in Ontology 2. These mappings, hereafter called *homogeneous mappings*, are able to cope with a large, but not the totality of heterogeneities between ontologies.

Assume, for instance, that a knowledge engineer builds an ontology of family unions containing the binary relations `marriedWith` and `partnerOf` between two persons. Suppose also that a second ontology engineer, asked to design a ontology for the same purpose, defines a concept `Marriage`, whose instances are the actual civil or religious marriages, and the concept `civilUnion`, whose instances are all the civil unions. We can easily see that while the first ontology prefers to model unions as relations, the second represents them as concepts. Despite this difference of style in modelling, the concept `Marriage` and the relation `marriedWith` represent the same (or a very similar) real world aspect, and similarly with `partnerOf` and `civilUnion`. To reconcile heterogeneous representations of this sort (which are instances of so-called schematic differences. See [2]) we need a mapping language that allows to map concepts of one ontology into relations of another ontology.

Motivated by these observations, we have illustrated in [9] the need of rich mapping languages that incorporate homogeneous and *heterogeneous mappings*, such as mappings between concepts and relations. [9] contains a preliminary investigation on how to define such rich mapping language in Distributed Description Logics (DDL) [12], and [8] presents a logic and an algorithm for the representation and reasoning with homogeneous mappings in DDL.

Here we address the task of representing and reasoning with both homogeneous and heterogeneous mappings. In particular, we extend the semantics of DDL to deal with heterogeneous mappings. The idea behind this is the ability to relate triples of the form  $\langle object\_1, relation\_name, object\_2 \rangle$  in one ontology with objects in the domain of another ontology. We provide a sound and complete axiomatization of the effects of all mappings from a source ontology to a target ontology. This is the crucial step towards the axiomatization for an arbitrary network of ontologies as shown in [12].

## 2 A rich language for mappings

Distributed Description Logic (DDL) [12] is a *natural* generalisation of the Description Logic (DL) framework designed to formalise multiple ontologies *pairwise* linked by semantic mappings. In DDL, ontologies correspond to description logic theories (T-boxes), while semantic mappings correspond to collections of *bridge rules* ( $\mathfrak{B}$ ).

Given a non empty set  $I$  of indexes, used to identify ontologies, let  $\{DL_i\}_{i \in I}$  be a collection of description logics<sup>3</sup>. For each  $i \in I$  let us denote a T-box of  $DL_i$  as  $\mathcal{T}_i$ . In this paper, we assume that each  $DL_i$  is description logic weaker or at most equivalent to  $\mathcal{ALCQI}_b$ , which corresponds to  $\mathcal{ALCQI}$  with role union, conjunction and difference (see [15]). Because of lack of space, we omit the precise description of  $\mathcal{ALCQI}_b$ , and we assume that the reader is familiar with DDL as described in [12].

We indicate with  $\{\mathcal{T}_i\}_{i \in I}$  a family of T-Boxes indexed by  $I$ . Intuitively,  $\mathcal{T}_i$  is the DL formalisation of the  $i$ -th ontology. To make every description distinct, we will prefix it with the index of ontology it belongs to. For instance, the concept  $C$  that occurs in the  $i$ -th ontology is denoted as  $i : C$ . Similarly,  $i : C \sqsubseteq D$  indicates that the axiom  $C \sqsubseteq D$  is being considered in the  $i$ -th ontology.

Semantic mappings between different ontologies are expressed via collections of *bridge rules*. In the following we use  $A, B, C$  and  $D$  as place-holders for concepts and  $R, S, P$  and  $Q$  as place-holders for roles. We instead use  $X$  and  $Y$  to denote both concepts and roles.

An *homogeneous bridge rule* from  $i$  to  $j$  is an expression defined as follows:

$$i : X \xrightarrow{\sqsubseteq} j : Y \quad (\text{into bridge rule}) \quad (1)$$

$$i : X \xrightarrow{\supseteq} j : Y \quad (\text{onto bridge rule}) \quad (2)$$

where  $X$  and  $Y$  are either concepts of  $DL_i$  and  $DL_j$  respectively, or roles of  $DL_i$  and  $DL_j$  respectively. An *heterogeneous bridge rule* from  $i$  to  $j$  is as follows:

$$i : R \xrightarrow{\sqsubseteq} j : C \quad (\text{role-into-concept bridge rule}) \quad (3)$$

$$i : R \xrightarrow{\supseteq} j : C \quad (\text{role-onto-concept bridge rule}) \quad (4)$$

$$i : C \xrightarrow{\sqsubseteq} j : R \quad (\text{concept-into-role bridge rule}) \quad (5)$$

$$i : C \xrightarrow{\supseteq} j : R \quad (\text{concept-onto-role bridge rule}) \quad (6)$$

where  $R$  is a role and  $C$  is a concept. A *distributed T-box* (DTB)  $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \mathfrak{B} \rangle$  consists of a collection  $\{\mathcal{T}_i\}_{i \in I}$  of T-boxes, and a collection  $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i \neq j \in I}$  of bridge rules between them.

Bridge rules (3) and (4) state that, from the  $j$ -th point of view the role  $R$  in  $i$  is less general, resp. more general, than its local concept  $C$ . Similarly, bridge rules (5) and (6) state that, from the  $j$ -th point of view the concept  $C$  in  $i$  is less general, resp. more general, than its local role  $R$ . Thus, the bridge rule

$$i : \text{marriedInChurchWith} \xrightarrow{\sqsubseteq} j : \text{Marriage}$$

<sup>3</sup> We assume familiarity with Description Logic and related reasoning systems, described in [1].

expresses the fact that, according to ontology  $j$ , the relation `marriedInChurchWith` in ontology  $i$  is less general than its local concept `Marriage`, while

$$i : \text{civilUnion} \xrightarrow{\sqsubseteq} j : \text{partnerOf} \quad i : \text{civilUnion} \xrightarrow{\supseteq} j : \text{partnerOf}$$

say that, according to ontology  $j$ , the concept `civilUnion` in ontology  $j$  is equivalent to its local relation `partnerOf`.

In this paper we require that for every (into or onto) bridge rule between roles  $i : P \rightarrow j : R$  in  $\mathfrak{B}_{ij}$ , also  $i : \text{inv}(P) \rightarrow j : \text{inv}(R)$  is in  $\mathfrak{B}_{ij}$  (where  $\text{inv}(X)$  is the inverse of  $X$ ). This to simplify the notation of the rules defined in Section 3.

The semantic of DDL assigns to each ontology  $\mathcal{T}_i$  a *local interpretation domain*. The first component of an interpretation of a DTB is a family of interpretations  $\{\mathcal{I}_i\}_{i \in I}$ , one for each T-box  $\mathcal{T}_i$ . Each  $\mathcal{I}_i$  is called a *local interpretation* and consists of a *possibly empty domain*  $\Delta^{\mathcal{I}_i}$  and a valuation function  $\cdot^{\mathcal{I}_i}$ , which maps every concept to a subset of  $\Delta^{\mathcal{I}_i}$ , and every role to a subset of  $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$ . The interpretation on the empty domain is used to provide a semantics for distributed T-boxes in which some of the local T-boxes are inconsistent. The reader interested in this aspect of DDL can refer to [12].

The second component of the DDL semantics are families of domain relations. Domain relations define how the different T-box interact and are necessary to define the satisfiability of bridge rules.

**Definition 1.** A domain relation  $r_{ij}$  from  $i$  to  $j$  is a subset of  $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ . We use  $r_{ij}(d)$  to denote  $\{d' \in \Delta^{\mathcal{I}_j} \mid (d, d') \in r_{ij}\}$ ; for any subset  $D$  of  $\Delta^{\mathcal{I}_i}$ , we use  $r_{ij}(D)$  to denote  $\bigcup_{d \in D} r_{ij}(d)$ ; for any  $R \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$  we use  $r_{ij}(R)$  to denote  $\bigcup_{(d, d') \in R} r_{ij}(d) \times r_{ij}(d')$ .

A domain relation  $r_{ij}$  represents a possible way of mapping the elements of  $\Delta^{\mathcal{I}_i}$  into its domain  $\Delta^{\mathcal{I}_j}$ , seen from  $j$ 's perspective. The domain relation is used to interpret homogeneous bridge rules.

**Definition 2.** The domain relation  $r_{ij}$  satisfies a homogeneous bridge rule w.r.t.,  $\mathcal{I}_i$  and  $\mathcal{I}_j$ , written as  $\langle \mathcal{I}_i, r_{ij}, \mathcal{I}_j \rangle \models br$ , when

$$\langle \mathcal{I}_i, r_{ij}, \mathcal{I}_j \rangle \models i : X \xrightarrow{\sqsubseteq} j : Y \quad \text{if} \quad r_{ij}(X^{\mathcal{I}_i}) \subseteq Y^{\mathcal{I}_j} \quad (7)$$

$$\langle \mathcal{I}_i, r_{ij}, \mathcal{I}_j \rangle \models i : X \xrightarrow{\supseteq} j : Y \quad \text{if} \quad r_{ij}(X^{\mathcal{I}_i}) \supseteq Y^{\mathcal{I}_j} \quad (8)$$

where  $X$  and  $Y$  are either two concepts or two roles.

Domain relations do not provide sufficient information to interpret heterogeneous mappings. Intuitively, an heterogeneous bridge rule between a relation  $R$  and a concept  $C$  connects a pair of objects related by  $R$  with an object which is in  $C$ . This suggests that, to evaluate heterogeneous bridge rules from roles in  $i$  to concepts in  $j$  we need a relation that maps triples of the form  $\langle \text{object}_1, \text{relation\_name}, \text{object}_2 \rangle$  from ontology  $i$  into objects of  $\Delta^{\mathcal{I}_j}$ . As an example we would like to map the triple  $\langle \text{John}, \text{marriedWith}, \text{Mary} \rangle$  of the first ontology into the marriage `m123` of the second ontology, with the intuitive meaning that `m123` is the marriage which correspond to the married couple composed of `John` and `Mary`. We first formally introduce the triples  $\langle \text{object}_1, \text{relation\_name}, \text{object}_2 \rangle$  for a given ontology  $i$ .

Given a local interpretation  $\mathcal{I}_i$  we consider the set of triples “induced” by the interpretation as the set of *admissible triples*  $\Sigma^{\mathcal{I}_i}$ . Let  $\mathcal{I}_i$  be a local interpretation  $\langle \Delta^{\mathcal{I}_i}, \Sigma^{\mathcal{I}_i} \rangle$  for  $\mathcal{DL}_i$ , and  $\mathcal{R}$  be the set of all atomic relations relations of  $\mathcal{DL}_i$ . We indicate with  $\Sigma^{\mathcal{I}_i}$  the set of all triples  $\langle x_1, X, x_2 \rangle$  such that  $x_1, x_2 \in \Delta^{\mathcal{I}_i}$ ;  $X \in \mathcal{R}$ ; and  $(x_1, x_2) \in X^{\mathcal{I}_i}$ .

Intuitively,  $\langle \text{John}, \text{marriedWith}, \text{Mary} \rangle$  is an admissible triple in  $\Sigma^{\mathcal{I}_i}$  if John is married with Mary, or more formally if the pair  $(\text{John}, \text{Mary})$  belongs to the interpretation of `marriedWith` in  $\mathcal{I}_i$ .

**Definition 3.** A concept-role domain relation  $cr_{ij}$  from  $i$  to  $j$  is a subset of  $\Delta^{\mathcal{I}_i} \times \Sigma^{\mathcal{I}_j}$ . A role-concept domain relation  $rc_{ij}$  from  $i$  to  $j$  is a subset of  $\Sigma^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ .

The domain relation  $rc_{ij}$  represents a possible way of mapping pairs of  $R^{\mathcal{I}_i}$  into elements of  $C^{\mathcal{I}_j}$ , seen from  $j$ 's perspective. Concept-role and role-concept domain relations are used to interpret heterogeneous mappings.

**Definition 4.** The role-concept domain relation  $rc_{ij}$  satisfies a role-(into/onto)-concept bridge rule w.r.t.,  $\mathcal{I}_i$  and  $\mathcal{I}_j$ , written  $\langle \mathcal{I}_i, rc_{ij}, \mathcal{I}_j \rangle \models br$ , when

1.  $(\mathcal{I}_i, rc_{ij}, \mathcal{I}_j) \models i : R \xrightarrow{\sqsubseteq} j : C$  if for all  $(x_1, x_2) \in R^{\mathcal{I}_i}$  and for all pairs  $((x_1, X, x_2), x) \in rc_{ij}$  with  $X^{\mathcal{I}_i} \subseteq R^{\mathcal{I}_i}$ , we have that  $x \in C^{\mathcal{I}_j}$
2.  $(\mathcal{I}_i, rc_{ij}, \mathcal{I}_j) \models i : R \xrightarrow{\supseteq} j : C$  if for all  $x \in C^{\mathcal{I}_j}$  there is a pair  $((x_1, X, x_2), x) \in rc_{ij}$ , such that  $X^{\mathcal{I}_i} \subseteq R^{\mathcal{I}_i}$ .

The concept-role domain relation  $cr_{ij}$  satisfies a concept-(into/onto)-role bridge rule w.r.t.,  $\mathcal{I}_i$  and  $\mathcal{I}_j$ , written  $\langle \mathcal{I}_i, cr_{ij}, \mathcal{I}_j \rangle \models br$ , when

3.  $(\mathcal{I}_i, cr_{ij}, \mathcal{I}_j) \models i : C \xrightarrow{\sqsubseteq} j : R$  if for all  $x \in C^{\mathcal{I}_i}$ , and for all pairs  $(x, \langle x_1, X, x_2 \rangle) \in cr_{ij}$ , it is true that  $X^{\mathcal{I}_j} \subseteq R^{\mathcal{I}_j}$ ;
4.  $(\mathcal{I}_i, cr_{ij}, \mathcal{I}_j) \models i : C \xrightarrow{\supseteq} j : R$  if for all  $(x_1, x_2) \in R^{\mathcal{I}_j}$  there is a pair  $(x, \langle x_1, X, x_2 \rangle) \in cr_{ij}$ , such that  $X^{\mathcal{I}_j} \subseteq R^{\mathcal{I}_j}$  and  $x \in C^{\mathcal{I}_i}$ .

Satisfiability of a role-into-concept bridge rule forces the role-concept domain relation  $cr_{ij}$  to map pair of elements  $(x_1, x_2)$  which belong to  $R^{\mathcal{I}_i}$  into elements  $x$  in  $C^{\mathcal{I}_j}$ . Note that, from the definition of role-concept domain relation two arbitrary objects  $y_1$  and  $y_2$  could occur in a pair  $(\langle y_1, X, y_2 \rangle, y)$  with  $X$  different from  $R$  itself but such that  $X^{\mathcal{I}_i} \subseteq R^{\mathcal{I}_i}$ . Thus also this pair  $(y_1, y_2)$  belongs to  $R^{\mathcal{I}_i}$  and we have to force also  $y$  to be in  $C^{\mathcal{I}_j}$ . In other words, we can say that satisfiability of a role-into-concept bridge rule forces the role-concept domain relation to map pairs of elements  $(x_1, x_2)$  which belong to  $R$ , or to any of its atomic subroles  $X$ , into elements  $x$  in  $C^{\mathcal{I}_i}$ .

A *distributed interpretation*  $\mathcal{J}$  of a DTB  $\mathfrak{T}$  consists of the 4-tuple

$$\langle \{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I}, \{cr_{ij}\}_{i \neq j \in I}, \{rc_{ij}\}_{i \neq j \in I} \rangle.$$

$\mathcal{J}$  satisfies the elements of a DTB  $\mathfrak{T}$  if, for every  $i, j \in I$ :

1.  $\mathcal{J} \models i : A \sqsubseteq B$ , if  $\mathcal{I}_i \models A \sqsubseteq B$
2.  $\mathcal{J} \models \mathcal{T}_i$ , if  $\mathcal{J} \models i : A \sqsubseteq B$  for all  $A \sqsubseteq B$  in  $\mathcal{T}_i$
3.  $\mathcal{J} \models \mathfrak{B}_{ij}$ , if

- $\langle \mathcal{I}_i, r_{ij}, \mathcal{I}_j \rangle$  satisfies all the homogeneous bridge rules in  $\mathfrak{B}_{ij}$ ,
  - $\langle \mathcal{I}_i, cr_{ij}, \mathcal{I}_j \rangle$  satisfies all the concept-to-role bridge rules in  $\mathfrak{B}_{ij}$ ,
  - $\langle \mathcal{I}_i, rc_{ij}, \mathcal{I}_j \rangle$  satisfies all the role-to-concept bridge rules in  $\mathfrak{B}_{ij}$
4.  $\mathcal{J} \models \mathfrak{A}$ , if for every  $i, j \in I$ ,  $\mathcal{J} \models \mathcal{T}_i$  and  $\mathcal{J} \models \mathfrak{B}_{ij}$

Entailment and satisfiability of a single concept are defined in the usual way by means of the above satisfiability of a distributed T-Box (e.g. see [12]).

### 3 The effects of bridge rules

Bridge rules can be thought of as inter-theory axioms, which constrain the models of the theories representing the different ontologies. An important characteristic of mappings specified by DDL bridge rules is that they are directional, in the sense that they are defined from a source ontology  $O_s$  to a target ontology  $O_t$ , and they allow to transfer knowledge only from  $O_s$  to  $O_t$ , without any undesired back-flow effect. In this section we show that the semantic of mappings defined in the previous Section fulfills this requirement. Furthermore we characterize the effects of the bridge rules in terms of the knowledge they allow to propagate from  $O_s$  to  $O_t$ .

We start by characterizing the effects of mappings of a simple DTB  $\langle \mathcal{T}_i, \mathcal{T}_j, \mathfrak{B}_{ij} \rangle$ , composed of two T-boxes  $\mathcal{T}_i$  and  $\mathcal{T}_j$  and a set of bridge rules  $\mathfrak{B}_{ij}$  from  $i$  to  $j$ . The first important property we prove is *directionality*:

**Proposition 1.**  $\langle \mathcal{T}_i, \mathcal{T}_j, \mathfrak{B}_{ij} \rangle \models i : X \sqsubseteq Y$  if and only if  $\mathcal{T}_i \models X \sqsubseteq Y$

The proof can be found in [7]. According to Proposition 1, bridge rules from  $i$  to  $j$  affect only the logical consequences in  $j$ , and leave the consequences in  $i$  unchanged. In the following we characterise the knowledge propagated from  $i$  (the source) to  $j$  (the target) using a set of *propagation rules* of the form:

$$\frac{\text{axioms in } i \quad \text{bridge rules from } i \text{ to } j}{\text{axiom in } j}$$

which must be read as: if  $\mathcal{T}_i$  entails all the axioms in  $i$ , and  $\mathfrak{B}_{ij}$  contains the bridge rules from  $i$  to  $j$ , then  $\langle \mathcal{T}_i, \mathcal{T}_j, \mathfrak{B}_{ij} \rangle$  satisfies axioms in  $j$ .

*Propagation rules for homogeneous mappings.* Simple propagation rules which describe the effects of the homogeneous mappings are:

$$\begin{array}{ccc} i : A \sqsubseteq B & i : P \sqsubseteq Q, & i : \exists P. \top \sqsubseteq B \\ i : A \xrightarrow{\exists} j : C & i : P \xrightarrow{\exists} j : R & i : P \xrightarrow{\exists} j : R \\ \frac{i : B \xrightarrow{\sqsubseteq} j : D}{j : C \sqsubseteq D} & \frac{i : Q \xrightarrow{\sqsubseteq} j : S}{j : R \sqsubseteq S} & \frac{i : B \xrightarrow{\sqsubseteq} j : D}{j : \exists R. \top \sqsubseteq D} \end{array} \quad (9) \quad (10) \quad (11)$$

Rule (9) describes a simple propagation of the concept hierarchy forced by bridge rules between concepts, and is widely described in [12]. This rule says that if  $A \sqsubseteq B$  is a fact of the T-box  $\mathcal{T}_i$ , then the effect of the bridge rules  $i : A \xrightarrow{\exists} j : C$  and  $i : B \xrightarrow{\sqsubseteq} j : D$  is that  $C \sqsubseteq D$  is also a fact in  $\mathcal{T}_j$ . An analogous effect concerns the

propagation of the role hierarchy due to bridge rules between roles, and is described by rule (10) where  $P, Q, R$  and  $S$  is either a role or an inverse role.<sup>4</sup> The effect of the combination of mappings between roles and mappings between concepts is the propagation of domain and range among relations linked by role-onto-role mappings. Propagation rule (11) describes a simple effect of these mappings, where  $P, R$  are roles and  $B, D$  are concepts. Rule (11) says that if the domain of  $P$  is contained in  $B$  and the appropriate bridge rules hold, then we can infer that the domain of  $R$  is contained in  $D$ . A similar rule allows to obtain  $j : \exists R^-. \top \sqsubseteq D$  from  $i : \exists P^-. \top \sqsubseteq B$  and the same bridge rules, thus expressing the propagation of the range restriction.

The general form of propagation rules (9)–(11) is given in Figure 1. Note that rule (10) can be obtained from rule (b) in Figure 1 by setting  $l = 1, p = 0, m = 0$ , while rule (11) can be obtained by setting  $l = 0, p = 0, m = 1$ . Analogously the rule for range restriction can be obtained by setting  $l = 0, p = 1, m = 0$ .

*Propagation rules for heterogeneous mappings.* The effects of the heterogeneous bridge rules is the propagation of the role hierarchy into the concept hierarchy and vice-versa. The simplest forms of these rules are:

$$\begin{array}{c}
 i : P \sqsubseteq Q \\
 i : P \xrightarrow{\exists} j : C \\
 i : Q \xrightarrow{\sqsubseteq} j : D \\
 \hline
 j : C \sqsubseteq D
 \end{array}
 \quad (12)
 \qquad
 \begin{array}{c}
 i : A \sqsubseteq B \\
 i : A \xrightarrow{\exists} j : R \\
 i : B \xrightarrow{\sqsubseteq} j : S \\
 \hline
 j : R \sqsubseteq S
 \end{array}
 \quad (13)$$

The general form of these rules is given in Figure 1. The expression  $\bigsqcup_{k=1}^n S_k$  with  $n = 0$  in rule (d) represents the empty role  $R_{\perp}$ , which is obtained with the axiom  $\top \sqsubseteq \forall R_{\perp} \perp$ .

Given a set of bridge rules  $\mathfrak{B}_{ij}$  from  $DL_i$  to  $DL_j$ , we have defined four different rules, shown in Figure 1, which take as input a T-box  $\mathcal{T}_i$  in  $DL_i$  and produce a T-box  $\mathcal{T}_j$  in  $DL_j$ . Starting from these rules we define an operator  $\mathfrak{B}_{ij}(\cdot)$ , taking as input  $\mathcal{T}_i$  and producing a T-box  $\mathcal{T}_j$ , enriched with the conclusions of rules (a)–(d) in Figure 1.

**Theorem 1 (Soundness and Completeness of  $\mathfrak{B}_{ij}(\cdot)$ ).** *Let  $\mathfrak{T}_{ij} = \langle \mathcal{T}_i, \mathcal{T}_j, \mathfrak{B}_{ij} \rangle$  be a distributed T-box, where  $\mathcal{T}_i$  and  $\mathcal{T}_j$  are expressed in the  $ALCQL_b$  descriptive language. Then  $\mathfrak{T}_{ij} \models j : X \sqsubseteq Y \iff \mathcal{T}_j \cup \mathfrak{B}_{ij}(\mathcal{T}_i) \models X \sqsubseteq Y$ .*

The proof can be found in [7]. The generalisation of the axiomatization for an arbitrary network of ontologies can be obtained following the technique used in [12].

As a final remark we can notice that the combination of homogeneous and heterogeneous bridge rules does not generate any effect in the logic proposed in this paper. This because the domain relation and the concept-role and role-concept domain relations do not affect each other. The investigation of more complex heterogeneous bridge rules, which can lead to this sort of interaction is left for future work. An additional open point concerns the extension of our framework in order to account for transitive roles. It is well known that the unrestricted interaction between number restriction and transitivity is a source of undecidability; moreover, the bridge rules may infer additional subsumption relations among the roles. Therefore, guaranteeing appropriate restrictions to ensure decidability is no longer a matter of analysing the “static” role hierarchy (e.g., a in the case of  $\mathcal{SHIQ}$ ).

<sup>4</sup> The formula  $R \sqsubseteq S$  is a shorthand for  $\exists(R \sqcap \neg S). \top \sqsubseteq \perp$ .

$$\begin{array}{l}
 i : A \sqsubseteq \bigsqcup_{k=1}^n B_k \\
 i : A \xrightarrow{\exists} j : C \\
 \frac{i : B_k \xrightarrow{\sqsubseteq} j : D_k, \text{ for } 1 \leq k \leq n}{j : C \sqsubseteq \bigsqcup_{k=1}^n D_k}
 \end{array}
 \qquad
 \begin{array}{l}
 i : \exists(P \sqcap \neg(\bigsqcup_{h=0}^l Q_h)). (\neg \bigsqcup_{h=0}^p A_h) \sqsubseteq (\bigsqcup_{h=0}^m B_h) \\
 i : P \xrightarrow{\exists} j : R \\
 i : Q_h \xrightarrow{\sqsubseteq} j : S_h, \text{ for } 1 \leq h \leq l \\
 i : A_h \xrightarrow{\sqsubseteq} j : C_h, \text{ for } 1 \leq h \leq p \\
 \frac{i : B_h \xrightarrow{\sqsubseteq} j : D_h, \text{ for } 1 \leq h \leq m}{j : \exists(R \sqcap \neg(\bigsqcup_{h=1}^l S_h)). (\neg \bigsqcup_{h=1}^p C_h) \sqsubseteq (\bigsqcup_{k=1}^m D_k)}
 \end{array}$$

(a) Generalisation of rule (9).

(b) Generalisation of rules (10) and (11).

$$\begin{array}{l}
 i : P \sqsubseteq Q \\
 i : P \xrightarrow{\exists} j : C \quad i : P \sqsubseteq \perp_R \\
 \frac{i : Q \xrightarrow{\sqsubseteq} j : D}{j : C \sqsubseteq D} \quad \frac{i : P \xrightarrow{\exists} j : C}{j : C \sqsubseteq \perp}
 \end{array}
 \qquad
 \begin{array}{l}
 i : A \sqsubseteq \bigsqcup_{k=1}^n B_k \\
 i : A \xrightarrow{\exists} j : R \\
 \frac{i : B_k \xrightarrow{\sqsubseteq} j : S_k, \text{ for } 1 \leq k \leq n}{j : R \sqsubseteq \bigsqcup_{k=1}^n S_k}
 \end{array}$$

(c) Generalisation of rule (12).

(d) Generalisation of rule (13).

Fig. 1. General version of propagation rules.

## 4 Related Work and Concluding Remarks

Recently, several proposals go in the direction of providing semantic mapping among different ontologies (e.g. [14, 12, 3]). However, to the best of our knowledge there is no specific work on heterogeneous mappings as described in this paper. This in spite of the fact that there are several attempts at providing some sort of mappings relating non-homogeneous elements. For example in [6], it is possible to express the mapping  $\forall x. (\exists y. R(x, y) \rightarrow C(x))$ ; while, in the original version of DDL (see [12]), an analogous mappings can be established by means of the formula  $1 : \exists R. \top \xrightarrow{\sqsubseteq} 2 : C$ . Note that both cases cannot be considered heterogeneous mappings because they relates the domain of the relation  $R$  with the concept  $C$ ; which are both concepts.

The work presented in this paper is clearly connected to the well known modelling process of *reification* (aka *objectification*) adopted in UML or ORM (see [10, 11]). As described in [10], this corresponds to think of certain relationship instances as objects. In UML this is supported by means of *association classes*, while in Entity-Relationship diagram this is often mediated by means of *weak entities*. Note that these modelling paradigms do not support rich inter-schema axioms in the spirit of ontology mappings as described in [14].

There are other modelling formalisms which enable the bridging between relations and classes in the context of Description Logics. In particular, the work on  $\mathcal{DLR}$  (see [4]), specifically w.r.t. the technique for encoding n-ary relations within a standard Description Logic, and [5]. The advantage of our approach lies in the fact that the local semantics (i.e. the underlying semantics of the single ontology languages) does not need to be modified in order to consider the global semantics of the system. Specifically, there is no need to provide an explicit reification of relations since this is incorporated into the global semantics.

The language and the semantics presented in this paper constitute a genuine contribution in the direction of the integration of heterogeneous ontologies. The language proposed in this paper makes it possible to directly bind a concept with a relation in a different ontology, and vice-versa. At the semantic level we have introduced a domain relation that maps pairs of object appearing in a relation into objects and vice-versa. This also constitute a novelty in the semantics of knowledge integration. Finally we have proved soundness and completeness of the effects of the mappings and we leave the study of decidability and the definition of a reasoning algorithm for future work.

## References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schemaintegration. *ACM Comput. Surv.*, 18(4):323–364, 1986.
3. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Second International Semantic Web Conference (ISWC-03)*, volume 2870 of *LNCS*, pages 164–179. Springer Verlag, 2003.
4. D. Calvanese, D. Berardi, and G. De Giacomo. Reasoning on uml class diagrams. *Artificial Intelligence*, 1(168):70–118, 2005.
5. D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Deductive and Object-Oriented Databases, DOOD’95*, pages 229–246, 1995.
6. D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *23rd ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS 2004)*, pages 241–251, 2004.
7. C. Ghidini, L. Serafini, and S. Tessaris. On relating heterogeneous elements from different ontologies. Technical report, KRDB. Free university of Bozen-Bolzano, 2007. <http://www.inf.unibz.it/krdp/pub/index.php>.
8. C. Ghidini and L. Serafini. Mapping properties of heterogeneous ontologies. In *Proceedings of the 1st Int. Workshop on Modular Ontologies (WoMo-06)*, volume 232 of *CEUR*, 2006. <http://ceur-ws.org/Vol-232>.
9. C. Ghidini and L. Serafini. Reconciling concepts and relations in heterogeneous ontologies. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, volume 4011/2006 of *LNCS*. Springer, 2006.
10. T. Halpin. Objectification of relationships. In *Proc. of the 10th Int. IFIP WG8.1 Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD’05)*, 2005.
11. J. Rumbaugh, I. Jacobson, and G. Booch, editors. *The Unified Language Reference Manual*. Addison-Wesley Longman Ltd., 1999.
12. L. Serafini, A. Borgida, and A. Tamilin. Aspects of distributed and modular ontology reasoning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 570–575, August 2005.
13. L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping language for terminological knowledge. In *19th Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 576–581, 2005.
14. H. Stuckenschmidt and M. Uschold. Representation of semantic mappings. In *Semantic Interoperability and Integration*, number 04391 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005.
15. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.

## Contextualization of a DL Knowledge Base

Krzysztof Goczyła, Aleksander Waloszek, Wojciech Waloszek

Gdańsk University of Technology, Department of Software Engineering,  
ul. Gabriela Narutowicza 11/12, 80-952 Gdańsk, Poland  
{kris,alwal,wowal}@eti.pg.gda.pl

**Abstract.** In the paper we propose a method of structuring a knowledge base into hierarchically related contexts and present how this arrangement influences the structure of TBox and ABox. We introduce a possibility to attach to a single TBox many ABoxes describing different parts of a domain and we show how to interpret such an ontology. Practical application of the method offers very interesting possibilities, like shortening the time of inference or storing mutually contradictory pieces of information in a single knowledge base. We analyse how such a structure changes purpose and mechanisms of reasoning, and we discuss their soundness and completeness. We also describe some related work about contexts.

### 1 Introduction

Contextualizing knowledge bases is an approach to make reasoning process more effective and to avoid inconsistencies in large ontologies. There are many types of relationships between contexts. In our approach we try to distinguish a group of relationships similar to inheritance. This kind of relation can be applied to both TBox and ABox. In case of TBox we separate axioms into groups that in the top of hierarchy define more general notions while passing down the hierarchy more specific and specialized concepts. In case of ABox every subset of assertions is attached to a particular context (i.e. a particular group of axioms). Moreover, this subset may also be divided into smaller groups called *context instances*. This subdivision limits the flow of conclusions and enables us to store inconsistent statements in one knowledge base.

The limited space in this paper does not allow us to present full spectrum of possible applications of a knowledge base organized according to our proposal. Our aim is to enable such tasks like modelling space-time situations, possibilities, believes or intentions, processes or even logical metalevels, i.e. such contexts where ABox contains description of ABox and TBox of another context. We realize that described method does not fulfil all of these requirements and it needs further development. But even now with its simple rules it has big potential of practical applications.

Section 2 presents formal definition of contextualized knowledge base. Section 3 contains description of reasoning problems in contextual knowledge bases. In Section 4 we try to review shortly another works on contexts. Section 5 summarizes the paper.

## 2 Formal definition of contextualized ontology

Our main goal was to introduce a kind of arrangement into large ontologies. We strive to allow for:

- introducing a hierarchical arrangement into large ontologies in order to describe various fragments of knowledge at different level of detail,
- holding contradictory assertions if they describe the same problem from different points of view,
- making it possible to integrate information from different points of view at a desired level of generality.

We propose a way of reaching these goals by introducing a notion of context into the knowledge base. First of all, we introduce contextualized TBox (contextualized terminology) that can be composed of several parts. These parts (being standard DL TBoxes), called *contexts*, remain with each other in a relation of generalization/specialization (inheritance, see Fig. 1 for an example).

**Definition 1.** A contextualized TBox  $\mathbf{T} = (\{T_i\}_{i \in I}, \trianglelefteq)$  consists of a set of TBoxes whose elements are called contexts, and a generalization relation  $\trianglelefteq \subseteq I \times I$  which is a partial order established over the set of indexes  $I$ . The poset  $(I, \trianglelefteq)$  is a tree containing the least element  $m$ . We also introduce the following notions:

- $T_m$  called the root context of the contextualized TBox  $\mathbf{T}$ ,
- $T_i$  generalizes  $T_j$  iff  $i \trianglelefteq j$ ,
- $T_i$  specializes  $T_j$  iff  $j \trianglelefteq i$ .

The idea behind such hierarchical arrangement of contexts was to allow for constrained interactions between parts of terminology. The general rule here is that more specialized terminologies may “see” more general ones, but more general terminologies are unaware of the existence of more specialized ones.

Introduced contexts encompass only terminology. To deal with assertional part of the knowledge base we allow for creation of many ABoxes for one terminology. We call these ABoxes *context instances*.

**Definition 2.** A contextualized ABox  $\mathbf{A} = (\{A_j\}_{j \in J}, inst, \ll)$  of contextualized TBox  $\mathbf{T} = (\{T_i\}_{i \in I}, \trianglelefteq)$  is a triple consisting of:

1. A set of ABoxes  $\{A_j\}_{j \in J}$ , each of which is called an instance of context,
2. The function  $inst : J \rightarrow I$  relating each ABox from  $\{A_j\}_{j \in J}$  with TBox from  $\{T_i\}_{i \in I}$ ,
3. The aggregation relation  $\ll \subseteq J \times J$ , which is a partial order established over the set of indexes  $J$ . We require that:
  - a. The poset  $(J, \ll)$  is a tree containing the least element  $n$ ,
  - b.  $inst(n) = m$ , where  $m$  is the least element of the relation  $\trianglelefteq$ ,
  - c. For each  $j \ll k$  such that  $j \neq k$  holds  $inst(j) \trianglelefteq inst(k)$  and  $inst(j) \neq inst(k)$ .

We also say that:

- $A_n$  is called the root context instance of the contextualized ABox  $\mathbf{A}$ ,
- $A_j$  is an instance of the context  $T_i$  iff  $inst(j) = i$ ,
- $A_j$  aggregates  $A_k$  iff  $j \ll k$ ,
- $A_j$  is aggregated by  $A_k$  iff  $k \ll j$ ,
- $A_j$  is an aggregating context instance iff  $\exists k : j \ll k$ .

The idea of assigning several ABoxes to a single TBox (like in Fig. 1 where the context instances  $A_7$ ,  $A_8$ , and  $A_9$  are assigned to the context  $T_5$ ) gives us distinctive opportunities: different ABoxes may contain different (consistent locally but possibly inconsistent with other ABoxes) sets of assertions.

It is worth noting that in a contextualized ABox a context instance aggregating all other context instances appears (in Fig. 1 it is  $A_1$ ). The consequence of this fact is that all context instances have to be consistent with each other at the highest (defined in a contextual TBox) level of generality. This fact justifies calling the pair of contextualized TBox and ABox the contextualized knowledge base.

**Definition 3.** A contextualized knowledge base  $\mathbf{K} = (\mathbf{T}, \mathbf{A})$  consists of a contextualized TBox  $\mathbf{T}$  and a contextualized ABox  $\mathbf{A}$  of  $\mathbf{T}$ .

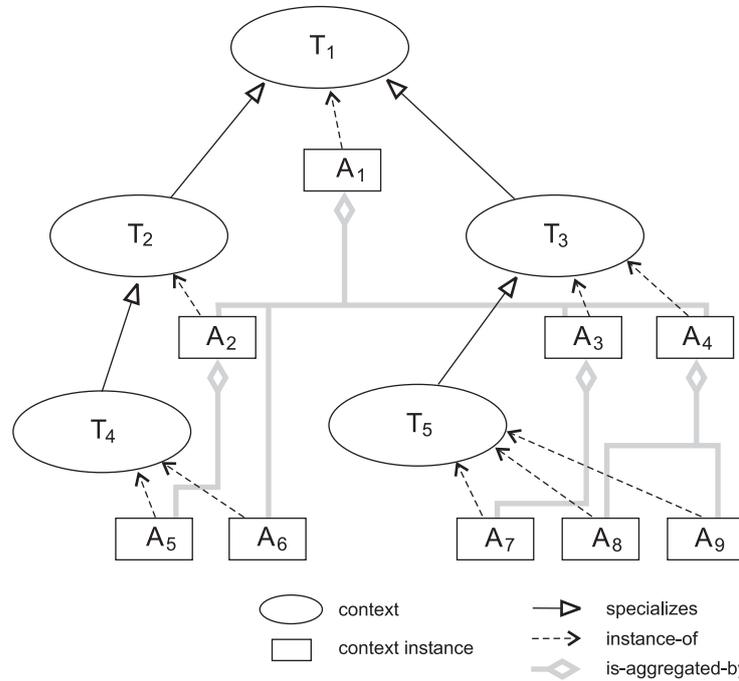
Contextualized knowledge base is given the interpretation in a specific way: each context instance is given its own interpretation. Such an approach gives some level of locality within context instances.

**Definition 4.** A contextualized interpretation  $\mathcal{I}$  of contextualized knowledge base  $\mathbf{K} = (\mathbf{T}, \mathbf{A})$  where  $\mathbf{T} = (\{T_i\}_{i \in I}, \sqsubseteq)$  and  $\mathbf{A} = (\{A_j\}_{j \in J}, inst, \ll)$ , is a set of interpretations  $\{\mathcal{I}_j\}$  where  $j \in J$ .

The next definition specifies what conditions the local interpretations have to satisfy in order to make the global interpretation a model of a knowledge base.

**Definition 5.** A contextualized interpretation  $\mathcal{I} = \{\mathcal{I}_j\}$  of a contextual knowledge base  $\mathbf{K} = (\mathbf{T}, \mathbf{A})$  where  $\mathbf{T} = (\{T_i\}_{i \in I}, \sqsubseteq)$  and  $\mathbf{A} = (\{A_j\}_{j \in J}, inst, \ll)$ , is a model of the knowledge base  $\mathbf{K}$  iff:

1. For every individual name  $a$ , there do not exist two interpretations  $\mathcal{I}_j, \mathcal{I}_k \in \mathcal{I}$  such that  $a^{\mathcal{I}_j} \neq a^{\mathcal{I}_k}$ ,
2. For every context instance  $A_j$ :
  - a.  $\mathcal{I}_j \models A_j$ ,
  - b.  $\mathcal{I}_j \models \bigcup_{i \in \{i : i \sqsubseteq inst(j)\}} T_i$ ,
  - c. for every  $k$  such that  $j \ll k$ :
    - i.  $\Delta^{\mathcal{I}_k} \subseteq \Delta^{\mathcal{I}_j}$ ,
    - ii. for every concept  $C$  from  $\bigcup_{i \in \{i : i \sqsubseteq inst(j)\}} T_i$ :  $C^{\mathcal{I}_j} \cap \Delta^{\mathcal{I}_k} = C^{\mathcal{I}_k}$ ,
    - iii. for every role  $R$  from  $\bigcup_{i \in \{i : i \sqsubseteq inst(j)\}} T_i$ :  $R^{\mathcal{I}_j} \cap (\Delta^{\mathcal{I}_k} \times \Delta^{\mathcal{I}_k}) = R^{\mathcal{I}_k}$ .



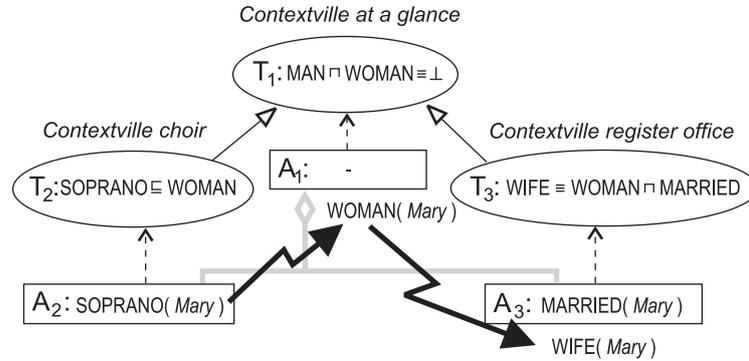
**Fig. 1.** An example of contextualized knowledge base. Relationships between context instances and between context instances and contexts are depicted in the form of graph, e.g. the instance  $A_4$  aggregates instances  $A_8$  and  $A_9$ . For the sake of clarity only the transitive reductions of generalization and aggregation relations have been depicted

The rules in the above definition may be divided into several categories. Rules 2.a and 2.b are called **local conformance** rules. They ensure that each interpretation satisfies the ABox and the TBox of the context instance it is assigned to and all TBoxes being its ancestors. An immediate corollary from this is the fact that terminologies that have any context instances assigned to cannot contradict any of their ancestors.

Other rules introduce the desired level of interaction between interpretations. Rule 1 is called **uniformity of names**. This rule was introduced to facilitate gathering pieces of information about one individual from various context instances (without necessity of defining mappings) during reasoning.

Rules 2.c (**aggregation conformance** rules) establish relations between aggregating context instance and context instances being aggregated. Rule 2.c.i introduces **aggregation conformance of domains** and states that the domain of the interpretation of the aggregating context must cover domains of interpretations of all context instances being aggregated.

Rules 2.c.ii and 2.c.iii establish **aggregation conformance of denotation**. They state that within the limited domain of the context instance  $A_k$  being aggregated by  $A_j$ , at the level of generality of the terminology  $T_i$  ( $inst(j) = i$ ),



**Fig. 2.** An example of the aggregation conformance of denotation. Here we have three contexts:  $T_1$  that describes general notions of **WOMAN** and **MAN**,  $T_2$  that specializes  $T_1$  towards description of voices in a choir, and  $T_3$  that also specializes  $T_1$  but towards description of social relations. Context instance  $A_1$  aggregates context instances  $A_2$  and  $A_3$ . Although  $A_1$  is empty, according to the rule 2.c.ii, interpretation  $\mathcal{I}_1$  in order to be a model of the knowledge base has to assign **Mary** to the concept **WOMAN** (i.e.  $\text{Mary}^{\mathcal{I}_1} \in \text{WOMAN}^{\mathcal{I}_1}$ ). As a consequence of this, the same rule enforces that in the interpretation  $\mathcal{I}_3$  **Mary** is assigned to the concept **WIFE** (i.e.  $\text{Mary}^{\mathcal{I}_3} \in \text{WIFE}^{\mathcal{I}_3}$ ), as the information about **Mary** being a woman “flows” down the aggregation relationships

all the concepts and roles must be interpreted in  $A_j$  in the same way (have the same extensions) as in  $A_k$ . These rules ensure flow of conclusions between aggregating context instance and the context instances being aggregated. The flow is bidirectional, as shown in the example from Fig 2. An interpretation of  $A_j$  must take into account all information from  $A_k$ , but due to the fact that is attached to a more general TBox this information must be reinterpreted in more general terms. This is also the way to avoid inconsistencies—to aggregate instances containing contradictory statements on the level of generality where the inconsistencies do not exist.

### 3 Reasoning in contextual knowledge bases

Reasoning in contextual knowledge bases is relevant to a single context instance.

**Definition 6.** *Entailment in contextual knowledge base  $\mathbf{K} = (\mathbf{T}, \mathbf{A})$  where  $\mathbf{T} = (\{T_i\}_{i \in I}, \sqsubseteq)$  and  $\mathbf{A} = (\{A_j\}_{j \in J}, inst, \ll)$ :*

1.  $C \sqsubseteq D$  is entailed by  $\mathbf{K}$  in the context  $T_i$  (denoted  $\mathbf{K} \models^i C \sqsubseteq D$ ) iff for every contextual interpretation  $\mathcal{I} = \{\mathcal{I}_j\}_{j \in J}$  that is a model of  $\mathbf{K}$  for every  $j$  such that  $inst(j) = i$  it is true that  $\mathcal{I}_j \models C \sqsubseteq D$ ,
2.  $C(a)$  (and analogically  $R(a, b)$ ) is entailed by  $\mathbf{K}$  in the context instance  $A_j$  (denoted  $\mathbf{K} \models^j C(a)$  and  $\mathbf{K} \models^j R(a, b)$ , respectively) iff for every contextual interpretation  $\mathcal{I} = \{\mathcal{I}_j\}_{j \in J}$  that is a model of  $\mathbf{K}$  it is true that  $\mathcal{I}_j \models C(a)$  ( $\mathcal{I}_j \models R(a, b)$ , respectively).

To show the possibility of employment of known reasoning algorithms for contextual knowledge bases we will use a method similar to the one exploited by A. Borgida and L. Serafini in [1]. For the sake of brevity we assume that all contexts and context instances use the same Description Logics  $\mathcal{DL}$ . In [1] axioms and assertions included in different Information Systems ( $\mathcal{IS}$ s) are translated to appropriate statements in a single global knowledge base. We will use similar kind of translation to transfer the contents of a contextual knowledge base  $\mathbf{K}$  to the global non-contextual knowledge base  $K$ .

We perform the translation on context instance-by-context instance basis. For each context instance  $A_j$  we have to establish a kind of separate space allowing for interpretation of concepts and roles different than in its sibling context instances. If  $A_j$  is aggregated by other context instance  $A_g$  (assume that  $g$  is a direct predecessor of  $j$ , denoted  $g = \pi(j)$ , i.e.  $g \ll j, g \neq j$  and there is no  $l$  such that  $g \ll l \ll j$ ), the space must embrace concepts and roles from  $T_i$  where  $i \sqsubseteq inst(j)$  and  $i \not\sqsubseteq inst(g)$ . We denote such a set of indexes  $\tau(j)$ , and for the least element  $n$  of  $\ll$  we will assume that  $\tau(n) = \{inst(n)\}$ . Using this technique two assertions  $\text{DOCTOR}(\text{John})$  and  $\neg\text{DOCTOR}(\text{John})$  from two context instances  $A_j$  and  $A_k$  (if e.g.  $\text{John}$  is a doctor in Poland but not in Great Britain from the legal point of view) will be translated to  $j : \text{DOCTOR}(\text{John})$  and  $k : \top \sqcap \neg k : \text{DOCTOR}(\text{John})$ , which will not generate inconsistency.

The mapping  $\#(j, E)$  translating an expression  $E$  describing a concept/role within the context instance  $A_j$  (or the context  $T_{inst(j)}$ ) in  $\mathbf{K}$  to an appropriate expression in  $K$  is defined as follows:

- $\#(j, \top) = j : \top$
- $\#(j, \perp) = \perp$
- $\#(j, A) = j : A$ , for atomic concept  $A$  introduced in  $T_i$  such that  $i \in \tau(j)$
- $\#(j, A) = j : \top \sqcap \#(\pi(j), A)$ , for any other atomic concept  $A$ .
- $\#(j, R) = j : R$ , for atomic role  $R$  introduced in  $T_i$  such that  $i \in \tau(j)$ ,
- $\#(j, R) = \#(\pi(j), R)$ , for any other atomic role  $R$ ,
- $\#(j, \rho(E_1, E_2, \dots, E_n)) = j : \top \sqcap \rho(\#(j, E_1), \#(j, E_2), \dots, \#(j, E_n))$ , for concept constructor  $\rho$  taking  $n$  arguments (*structural recursion*)

Now we can define rules of transferring axioms and assertions from  $\mathbf{K}$  to  $K$ :  
For each  $j \in J$  do the following:

1. For each  $C \sqsubseteq D$  included in  $T_i$  such that  $i \in \tau(j)$ , insert to  $K$ :  
 $\#(j, C) \sqsubseteq \#(j, D)$
2. For each atomic concept  $A$  introduced in  $T_i$  such that  $i \in \tau(j)$ , insert to  $K$ :  
 $\#(j, A) \sqsubseteq j : \top$
3. For each atomic role  $R$  introduced in  $T_i$  such that  $i \in \tau(j)$  restrict their domain and range in  $K$ :  
 $\top \sqsubseteq \forall \#(j, R).j : \top$   
 $\neg j : \top \sqsubseteq \forall \#(j, R).\perp$
4. If  $j$  is the least element of  $\ll$ , insert to  $K$ :  
 $\top \sqsubseteq j : \top$   
Otherwise, insert to  $K$ :  
 $j : \top \sqsubseteq \pi(j) : \top$

5. Copy all assertions of  $A_j$  to  $K$  in the following form:
  - $\#(j, C)(a)$  for each  $C(a)$  included in  $A_j$
  - $\#(j, R)(a, b)$  for each  $R(a, b)$  included in  $A_j$

Following a very similar line of argumentation as the one in [1] we can show that for every  $\mathcal{DL}$  in which every concept and role constructor is *local*<sup>1</sup> (e.g.  $\mathcal{SHIQ}$ ) the following holds:  $\mathbf{K} \models^i C \sqsubseteq D$  iff for every  $j$  such that  $inst(j) = i$  it is true that  $K \models \#(j, C) \sqsubseteq \#(j, D)$ . This result can be extended to ABox:  $\mathbf{K} \models^j C(a)$  iff  $K \models \#(j, C)(a)$  and  $\mathbf{K} \models^j R(a, b)$  iff  $K \models \#(j, R)(a, b)$ .

The above discussion was intended to show that reasoning from contextual knowledge base is possible with use of existing tools. However the inference algorithm derived directly from the sketched method of translation may turn out to be inefficient. This is the reason why in practice, in the inference engine KASEA [8] being implemented by our group, we use other technique of reasoning based on translating assertions from an aggregated context instance to the terms appropriate for the aggregating context instance. This task is similar to finding the most specific concept but within the constrained set of terms.

Besides reasoning problems discussed above the separation of ABoxes and TBoxes gives us a possibility of defining a class of novel inference problems, e.g.: “Find all context instances in which a given assertion is entailed by the contextualized knowledge base” or: “Given a set of context instances  $\{A_j\}$ , find the lowest level of generality (i.e. the most specific terminology  $T_i$ ) at which they are not inconsistent (i.e. that there might exist a context instance  $A_k$  aggregating all context instances  $\{A_j\}$  with  $inst(k) = i$  not making the contextualized knowledge base inconsistent)”. Such problems might be interesting for Semantic Web communities focusing on integration of knowledge. More comprehensive set of similar problems and algorithms for solving them is under preparation.

## 4 Related work

Bouquet et al. in [2] divides the theories of context into two categories. The first category, called *divide-and-conquer (d-a-c)*, contains these theories which state that contextualization is a mean of partitioning a global theory of the world. The second category, called *compose-and-conquer (c-a-c)*, contains those ones which want to perceive a context as a local independent theory which can (but not has to) be integrated with another one with particular integration rules.

Local Model Semantics/Multi-context Systems (LMS/MCS) published in [5][7] form the theoretical basis for the c-a-c approach. There are several works [1][3][9] concerning ontology decomposition in the field of Description Logic. They are based on the foundation of *bridge rules*, a notion originally introduced in [6]. Bridge rules are descriptions of mappings between two portions of information. Although [1] does not use the notion of context (they are called *information*

<sup>1</sup> in practice it means all  $\mathcal{DL}$ s that do not have role constants and role constructors other than conjunction, disjunction, inverse, composition, role hierarchies and transitive roles; for the formal definition of locality see [1]

*sources - IS*) it gives a method to describe data integration between ontologies. In [9] contexts are called *ontology modules* and bridge rules are replaced by *ontology-based queries*.

The theoretical basis for the d-a-c approach is the Propositional Logic of Context (PLC) introduced by McCarthy and formalized by Buvač and Mason [4]. A model  $\mathfrak{M}$  for PLC defines a function, called the *vocabulary*, that associates formulae that are meaningful in a given context to this context. Contexts are arranged hierarchically. *Lifting axioms* play similar role as bridge rules in LMS/MCS.

Our work could be counted among those related with the d-a-c approach but is based on and develops division of propositions between TBox and ABox introduced by DL. By formulating some rules of relating contexts and their instances we intend to eliminate the necessity of defining a significant group of mappings.

## 5 Summary

In our paper we have shown our idea of contextualization of an ontology. We also have proposed an idea of context instances. Then we have described reasoning problems in such knowledge bases. Finally we have tried to place our approach among another work on ontology contexts.

## References

1. A.Borgida, L.Serafini. *Distributed description logics: Assimilating information from peer sources*. In Journal of Data Semantics, pp. 153–184, 2003.
2. P. Bouquet, Ch. Ghidini, F. Giunchiglia, E. Blanzieri. *Theories and uses of context in knowledge representation and reasoning*, Journal of Pragmatics, 35(3), pp. 455–484, 2003.
3. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt, C-OWL: Contextualizing Ontologies, Second International Semantic Web Conference (ISWC-2003), vol. 2870 of LNCS, pp. 164–179, Springer Verlag, 2003.
4. S. Buvač, I. A. Mason. *Metamathematics of Contexts*. Fundamenta Informaticae, 23(3), 1995.
5. C. Ghidini, F. Giunchiglia. *Local Model Semantics, or Contextual Reasoning = Locality + Compatibility*. Artificial Intelligence, 127(2):221–259, 2001.
6. F. Giunchiglia. *Contextual Reasoning*. Epistemologia, special issue on I Linguaggi e le Macchine, XVI, pp. 345–364, 1993.
7. F. Giunchiglia, L. Serafini. *Multilanguage hierarchical logics or: how we can do without modal logics*. Artificial Intelligence, 65(1):29–70, 1994.
8. K. Goczyła, T. Grabowska, W. Waloszek., M. Zawadzki: *The Knowledge Cartography A New Approach to Reasoning over Description Logics Ontologies*. SOFSEM 2006: Theory and Practice of Computer Science, vol. 3831 of LNCS, pp. 293–302, Springer, 2006.
9. H. Stuckenschmidt, M. Klein. *Integrity and Change in Modular Ontology*. In Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'03, pp. 900–905, Acapulco, 2003.

## EXPTIME Tableaux for $\mathcal{ALC}$ Using Sound Global Caching

Rajeev Goré<sup>1</sup> and Linh Anh Nguyen<sup>2</sup>

<sup>1</sup> The Australian National University  
Canberra ACT 0200, Australia

<sup>2</sup> Institute of Informatics, University of Warsaw  
ul. Banacha 2, 02-097 Warsaw, Poland

Rajeev.Gore@anu.edu.au      nguyenv@mimuw.edu.pl

**Abstract.** We show that global caching can be used with propagation of both satisfiability and unsatisfiability in a sound manner to give an EXPTIME algorithm for checking satisfiability w.r.t. a TBox in the basic description logic  $\mathcal{ALC}$ . Our algorithm is based on a simple traditional tableau calculus which builds an and-or graph where no two nodes of the graph contain the same formula set. When a duplicate node is about to be created, we use the pre-existing node as a proxy, even if the proxy is from a different branch of the tableau, thereby building global caching into the algorithm from the start. Doing so is important since it allows us to reason explicitly about the correctness of global caching. We then show that propagating both satisfiability and unsatisfiability via the and-or structure of the graph remains sound. In the longer paper, by combining global caching, propagation and cutoffs, our framework reduces the search space more significantly than the framework of [1]. Also, the freedom to use arbitrary search heuristics significantly increases its application potential.

A longer version with all optimisations is currently under review for a journal. An extension for SHI will appear in TABLEAUX 2007.

**Keywords:** sound caching, decision procedures, optimal complexity.

### 1 Motivation, Notation and Semantics of $\mathcal{ALC}$

We show that there is a simple way to use global caching and propagation to achieve an EXPTIME decision procedure for  $\mathcal{ALC}$ . Our algorithm is based on a simple traditional tableau calculus. It builds an and-or graph, where an or-node reflects the application of an “or” branching rule as in a tableau, while an and-node reflects the choice of a tableau rule and possibly many different applications of that rule to a given node of a tableau. We build caching into the construction of the and-or graph by ensuring that no two nodes of the graph have the same content. The status of a non-end-node is computed from the status of its successors using its kind (and-node/or-node) and treating satisfiability w.r.t. the TBox (i.e. **sat**) as true and unsatisfiability w.r.t. the TBox (i.e. **unsat**) as

false. When a node gets status **sat** or **unsat**, the status is propagated to its predecessors in a way appropriate to the graph’s and-or structure. With global caching and the assumption that  $\text{EXPTIME} \neq \text{PSPACE}$ , depth-first search has no advantages over other search strategies for our framework. That is, the naive version of our  $\text{EXPTIME}$  algorithm can accept any systematic search strategy.

By combining global caching, propagation and cutoffs, our framework significantly reduces the search space when compared with the framework of Donini and Massacci [1]. Furthermore, the freedom to use arbitrary search heuristics significantly increases the application potential of our framework.

We use  $A$  for atomic concepts, use  $C$  and  $D$  for arbitrary concepts, and use  $R$  for a role name. Concepts in  $\mathcal{ALC}$  are formed using the following BNF grammar:

$$C, D ::= \top \mid \perp \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid C \sqsubseteq D \mid C \doteq D \mid \forall R.C \mid \exists R.C$$

An *interpretation*  $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$  consists of a non-empty set  $\Delta^{\mathcal{I}}$ , the *domain* of  $\mathcal{I}$ , and a function  $\cdot^{\mathcal{I}}$ , the *interpretation function* of  $\mathcal{I}$ , that maps every atomic concept to a subset of  $\Delta^{\mathcal{I}}$  and every role name to a subset of  $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ . The interpretation function is extended to complex concepts as usual.

An interpretation  $\mathcal{I}$  *satisfies* a concept  $C$  if  $C^{\mathcal{I}} \neq \emptyset$ , and *validates* a concept  $C$  if  $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ . Clearly,  $\mathcal{I}$  *validates* a concept  $C$  iff it does not *satisfy*  $\neg C$ .

A TBox (of global axioms/assumptions)  $\Gamma$  is a finite set of concepts: traditionally, a TBox is defined to be a finite set of terminological axioms of the form  $C \doteq D$ , where  $C$  and  $D$  are concepts, but the two definitions are equivalent. An interpretation  $\mathcal{I}$  is a *model* of  $\Gamma$  if  $\mathcal{I}$  validates all concepts in  $\Gamma$ . We also use  $X, Y$  to denote finite sets of concepts. We say that  $\mathcal{I}$  *satisfies*  $X$  if there exists  $d \in \Delta^{\mathcal{I}}$  such that  $d \in C^{\mathcal{I}}$  for all  $C \in X$ . Note: satisfaction is defined “locally”, and  $\mathcal{I}$  satisfies  $X$  does not mean that  $\mathcal{I}$  is a model of  $X$ .

We say that  $\Gamma$  *entails*  $C$ , and write  $\Gamma \models C$ , if every model of  $\Gamma$  validates  $C$ . We say that  $C$  is satisfiable w.r.t.  $\Gamma$  if some model of  $\Gamma$  satisfies  $\{C\}$ . Similarly,  $X$  is satisfiable w.r.t. (a TBox of global axioms/assumptions)  $\Gamma$  if there exists a model of  $\Gamma$  that satisfies  $X$ . Observe that  $\Gamma \models C$  iff  $\neg C$  is unsatisfiable w.r.t.  $\Gamma$ .

**Note:** We now assume that concepts are in *negation normal form*, where  $\doteq$  and  $\sqsubseteq$  are translated away and  $\neg$  occurs only directly before atomic concepts.

## 2 A Tableau Calculus for $\mathcal{ALC}$

We consider tableaux with a fixed TBox of global axioms/assumptions  $\Gamma$ . The numerator of each tableau rule contains one or more distinguished concepts called the *principal concepts*. We write  $X; Y$  for  $X \cup Y$ , and  $X; C$  for  $X \cup \{C\}$ . The calculus  $\text{CALC}$  for  $\mathcal{ALC}$  consists of the tableau rules below:

$$\begin{aligned} (\perp) \frac{X; A; \neg A}{\perp} \quad & (\sqcap) \frac{X; C \sqcap D}{X; C; D} \quad & (\sqcup) \frac{X; C \sqcup D}{X; C \mid X; D} \\ (\exists R) \Gamma : \frac{X; \exists R.C}{\{D : \forall R.D \in X\}; C; \Gamma} \end{aligned}$$

The rules ( $\perp$ ), ( $\sqcap$ ), and ( $\sqcup$ ) are *static* rules, while ( $\exists R$ ) is a *transitional rule*.

A *CALC*-tableau (tableau, for short) w.r.t. a TBox  $\Gamma$  for a finite set  $X$  of concepts is a tree with root  $(\Gamma; X)$  whose nodes carry finite sets of concepts obtained from their parent nodes by instantiating a *CALC*-tableau rule with the proviso that: if a child  $s$  carries a set  $Y$  and no rule is applicable to  $Y$  or  $Y$  has already appeared on the branch from the root to  $s$  then  $s$  is an *end node*.

A branch in a tableau is *closed* if its end node carries only  $\perp$ . A tableau is *closed* if every one of its branches is closed. A tableau is *open* if it is not closed.

A finite set  $X$  of concepts is *consistent* w.r.t. a TBox  $\Gamma$  if every tableau w.r.t.  $\Gamma$  for  $X$  is open. If some tableau w.r.t.  $\Gamma$  for  $X$  is closed then  $X$  is *inconsistent* w.r.t.  $\Gamma$ . Calculus *CALC* is *sound* if for all finite sets  $\Gamma$  and  $X$  of concepts,  $X$  is satisfiable w.r.t.  $\Gamma$  implies  $X$  is consistent w.r.t.  $\Gamma$ . It is *complete* if for all finite sets  $\Gamma$  and  $X$  of concepts,  $X$  is consistent w.r.t.  $\Gamma$  implies  $X$  is satisfiable w.r.t.  $\Gamma$ . A tableau rule is *sound* if, whenever the numerator is *ALC*-satisfiable w.r.t. the TBox then one of the denominators is *ALC*-satisfiable w.r.t. the TBox.

**Lemma 1.** *The calculus CALC is sound because all rules of CALC are sound.*

Observe that every concept appearing in a tableau w.r.t.  $\Gamma$  for  $X$  is a subformula of  $\Gamma \cup X \cup \{\perp\}$ . Thus *CALC* thus has the *analytic subformula property*.

### 3 Completeness

A model graph is a tuple  $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ , where:  $\Delta$  is a finite set;  $\tau$  is a distinguished element of  $\Delta$ ;  $\mathcal{C}$  is a function that maps each element of  $\Delta$  to a set of concepts; and  $\mathcal{E}$  is a function that maps each role name to a binary relation on  $\Delta$ .

A model graph  $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$  is *saturated* if every  $x \in \Delta$  satisfies:

1. if  $C \sqcap D \in \mathcal{C}(x)$  then  $\{C, D\} \subseteq \mathcal{C}(x)$
2. if  $C \sqcup D \in \mathcal{C}(x)$  then  $C \in \mathcal{C}(x)$  or  $D \in \mathcal{C}(x)$
3. if  $\forall R. C \in \mathcal{C}(x)$  and  $\mathcal{E}(R)(x, y)$  holds then  $C \in \mathcal{C}(y)$
4. if  $\exists R. C \in \mathcal{C}(x)$  then there exists  $y \in \Delta$  with  $\mathcal{E}(R)(x, y)$  and  $C \in \mathcal{C}(y)$ .

A saturated model graph  $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$  is *consistent* if no  $x \in \Delta$  has a  $\mathcal{C}(x)$  containing  $\perp$  or containing a pair  $A, \neg A$  for some atomic concept  $A$ .

Given a model graph  $M = \langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ , the *interpretation corresponding to  $M$*  is the interpretation  $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$  where  $A^{\mathcal{I}} = \{x \in \Delta \mid A \in \mathcal{C}(x)\}$  for every atomic concept  $A$  and  $R^{\mathcal{I}} = \mathcal{E}(R)$  for every role name  $R$ .

**Lemma 2.** *By induction on the structure of  $\mathcal{C}$  we can show that if  $\mathcal{I}$  is the interpretation corresponding to a consistent saturated model graph  $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ , then for every  $x \in \Delta$  and  $C \in \mathcal{C}(x)$  we have  $x \in C^{\mathcal{I}}$ .*

Given finite sets  $X$  and  $\Gamma$  of concepts, where  $X$  is consistent w.r.t.  $\Gamma$ , we construct a model of  $\Gamma$  that satisfies  $X$  by constructing a consistent saturated model graph  $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$  with  $X \subseteq \mathcal{C}(\tau)$  and  $\Gamma \subseteq \mathcal{C}(x)$  for every  $x \in \Delta$ .

**Algorithm 1**

Input: a TBox  $\Gamma$  and a finite set  $X$  of concepts, where  $X$  is consistent w.r.t.  $\Gamma$ .

Output: a model graph  $M = \langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ .

1. For an arbitrary node name  $\tau$ , let  $\Delta := \{\tau\}$ , and  $\mathcal{E}(R) := \emptyset$  for every role name  $R$ . Let  $\mathcal{C}(\tau)$  be a saturation of  $\Gamma \cup X$  and mark  $\tau$  as unexpanded.
2. While  $\Delta$  contains unexpanded elements, take one, say  $x$ , and do:
  - (a) For every concept  $\exists R.C \in \mathcal{C}(x)$ :
    - i. Let  $Y = \{D \mid \forall R.D \in \mathcal{C}(x)\} \cup \{C\} \cup \Gamma$  be the result of applying rule  $(\exists R)$  to  $\mathcal{C}(x)$ , and let  $Z$  be a saturation of  $Y$ .
    - ii. If there exists a (proxy)  $y \in \Delta$  with  $\mathcal{C}(y) = Z$  then add pair  $(x, y)$  to  $\mathcal{E}(R)$ ;
    - iii. Else add a new element  $y$  with  $\mathcal{C}(y) := Z$  to  $\Delta$ , mark  $y$  as unexpanded, and add the pair  $(x, y)$  to  $\mathcal{E}(R)$ .
  - (b) Mark  $x$  as expanded.

**Fig. 1.** Constructing a Model Graph

**Saturation** The rules  $(\sqcap)$  and  $(\sqcup)$  do not carry their principal concept into their denominators. For these rules, let  $(\rho')$  be the version that carries the principal concept into each of its denominators. Each new rule is clearly sound for  $\mathcal{ALC}$ .

For a finite set  $X$  of concepts that is consistent w.r.t. a TBox  $\Gamma$ , a set  $Y$  of concepts is called a *saturation of  $X$*  w.r.t.  $\Gamma$  if  $Y$  is a maximal set consistent w.r.t.  $\Gamma$  that is obtainable from  $X$  (as a leaf node in a tableau) by applications of the rules  $(\sqcap')$  and  $(\sqcup')$ . A set  $X$  is *closed* w.r.t. a tableau rule if applying that rule to  $X$  gives back  $X$  as one of the denominators.

**Lemma 3.** *Let  $X$  be a finite set of concepts consistent w.r.t. a TBox  $\Gamma$ , and  $Y$  a saturation of  $X$  w.r.t.  $\Gamma$ . Then  $X \subseteq Y \subseteq Sf(\Gamma \cup X)$  and  $Y$  is closed w.r.t. the rules  $(\sqcap')$  and  $(\sqcup')$ . Furthermore, there is an effective procedure that constructs such a set  $Y$  from  $\Gamma$  and  $X$ .*

**Constructing Model Graphs** Figure 1 contains an algorithm for constructing a model graph. Algorithm 1 assumes that  $X$  is consistent w.r.t.  $\Gamma$  and constructs a model of  $\Gamma$  that satisfies  $X$ . Algorithm 1 terminates because each  $x \in \Delta$  has a unique finite set  $\mathcal{C}(x) \subseteq Sf(\Gamma \cup X)$ , so eventually Step 2(a)ii always finds a proxy. Note that Step 2(a)ii builds caching into the algorithm.

**Lemma 4.** *Let  $\Gamma$  be a TBox,  $X$  be a finite set of concepts consistent w.r.t.  $\Gamma$ ,  $M = \langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$  be the model graph constructed by Algorithm 1 for  $\Gamma$  and  $X$ , and  $\mathcal{I}$  be the interpretation corresponding to  $M$ . Then  $\mathcal{I}$  validates  $\Gamma$  and satisfies  $X$ .*

**Theorem 1.** *The calculus  $\mathcal{CALC}$  is sound and complete.*

#### 4 A Simple EXPTIME Decision Procedure for $\mathcal{ALC}$

In Figure 2 we present an EXPTIME decision procedure for  $\mathcal{ALC}$  which directly uses the tableau rules of  $\mathcal{CALC}$  to create an and-or graph as follows.

**Algorithm 2**

Input: two finite sets of concepts  $\Gamma$  and  $X$

Output: an and-or graph  $G = \langle V, E \rangle$  with  $\tau \in V$  as the initial node such that  $\tau.status = \mathbf{sat}$  iff  $X$  is satisfiable w.r.t.  $\Gamma$

1. create a new node  $\tau$  with  $\tau.content := \Gamma \cup X$  and  $\tau.status := \mathbf{unexpanded}$ ;  
let  $V := \{\tau\}$  and  $E := \emptyset$ ;
2. while  $\tau.status \notin \{\mathbf{sat}, \mathbf{unsat}\}$  and we can choose an unexpanded node  $v \in V$  do:
  - (a)  $\mathcal{D} := \emptyset$ ;
  - (b) if no  $\mathcal{CALC}$ -tableau rule is applicable to  $v.content$  then  $v.status := \mathbf{sat}$
  - (c) else if  $(\perp)$  is applicable to  $v.content$  then  $v.status := \mathbf{unsat}$
  - (d) else if  $(\sqcap)$  is applicable to  $v.content$  giving denominator  $Y$  then  
 $v.kind := \mathbf{and-node}$ ,  $\mathcal{D} := \{Y\}$
  - (e) else if  $(\sqcup)$  is applicable to  $v.content$  giving denominators  $Y_1$  and  $Y_2$  then  
 $v.kind := \mathbf{or-node}$ ,  $\mathcal{D} := \{Y_1, Y_2\}$
  - (f) else
    - i.  $v.kind := \mathbf{and-node}$ ,
    - ii. for every  $\exists R.C \in v.content$ , apply  $(\exists R)$  to  $v.content$  giving denominator  $\{D \mid \forall R.D \in v.content\} \cup \{C\} \cup \Gamma$  and add this denominator to  $\mathcal{D}$ ;
  - (g) for every denominator  $Y \in \mathcal{D}$  do
    - i. if some (proxy)  $w \in V$  has  $w.content = Y$  then add edge  $(v, w)$  to  $E$
    - ii. else let  $w$  be a new node, set  $w.content := Y$ ,  $w.status := \mathbf{unexpanded}$ , add  $w$  to  $V$ , and add edge  $(v, w)$  to  $E$ ;
  - (h) if  $(v.kind = \mathbf{or-node}$  and one of the successors of  $v$  has status  $\mathbf{sat}$ )  
or  $(v.kind = \mathbf{and-node}$  and all the successors of  $v$  have status  $\mathbf{sat}$ ) then  
 $v.status := \mathbf{sat}$ ,  $propagate(G, v)$
  - (i) else if  $(v.kind = \mathbf{and-node}$  and one of the successors of  $v$  has status  $\mathbf{unsat}$ )  
or  $(v.kind = \mathbf{or-node}$  and all the successors of  $v$  have status  $\mathbf{unsat}$ ) then  
 $v.status := \mathbf{unsat}$ ,  $propagate(G, v)$
  - (j) else  $v.status := \mathbf{expanded}$ ;
3. if  $\tau.status \notin \{\mathbf{sat}, \mathbf{unsat}\}$  then  
for every node  $v \in V$  with  $v.status \neq \mathbf{unsat}$ , set  $v.status := \mathbf{sat}$ ;

**Fig. 2.** A Simple EXPTIME Decision Procedure for  $\mathcal{ALC}$

A node in the constructed and-or graph is a record with three attributes:

*content*: the set of concepts carried by the node  
*status*:  $\{\mathbf{unexpanded}, \mathbf{expanded}, \mathbf{sat}, \mathbf{unsat}\}$   
*kind*:  $\{\mathbf{and-node}, \mathbf{or-node}\}$

To check whether a given finite set  $X$  is satisfiable w.r.t. the given TBox  $\Gamma$ , the content of the initial node  $\tau$  with status  $\mathbf{unexpanded}$  is  $\Gamma \cup X$ . The main while-loop continues processing nodes until the status of  $\tau$  is determined to be in  $\{\mathbf{sat}, \mathbf{unsat}\}$ , or until every node is expanded, whichever happens first.

Inside the main loop, Steps (2b) to (2f) try to apply one and only one of the tableau rules in the order  $(\perp)$ ,  $(\sqcap)$ ,  $(\sqcup)$ ,  $(\exists R)$  to the current node  $v$ . The set  $\mathcal{D}$

**Procedure** *propagate*( $G, v$ )

Parameters: an and-or graph  $G = \langle V, E \rangle$  and  $v \in V$  with  $v.status \in \{\mathbf{sat}, \mathbf{unsat}\}$

Returns: a modified and-or graph  $G = \langle V, E \rangle$

1.  $queue := \{v\}$ ;
2. while  $queue$  is not empty do
3. (a) extract  $x$  from  $queue$ ;
- (b) for every  $u \in V$  with  $(u, x) \in E$  and  $u.status = \mathbf{expanded}$  do
  - i. if ( $u.kind = \mathbf{or-node}$  and one of the successors of  $u$  has status  $\mathbf{sat}$ )  
or ( $u.kind = \mathbf{and-node}$  and all the successors of  $u$  have status  $\mathbf{sat}$ ) then  
 $u.status := \mathbf{sat}$ ,  $queue := queue \cup \{u\}$
  - ii. else if ( $u.kind = \mathbf{and-node}$  and one of the successors of  $u$  has status  $\mathbf{unsat}$ )  
or ( $u.kind = \mathbf{or-node}$  and all the successors of  $u$  have status  $\mathbf{unsat}$ ) then  
 $u.status := \mathbf{unsat}$ ,  $queue := queue \cup \{u\}$ ;

**Fig. 3.** Propagating Satisfiability and Unsatisfiability Through an And-Or Graph

contains the contents of the resulting denominators of  $v$ . If the applied tableau rule is  $(\sqcap)$  then  $v$  has one denominator in  $\mathcal{D}$ ; if the applied rule is  $(\sqcup)$  then  $v$  has two denominators in  $\mathcal{D}$ ; otherwise, each concept  $\exists R.C \in v.content$  contributes one appropriate denominator to  $\mathcal{D}$ . At Step (2g), for every denominator in  $\mathcal{D}$ , we create the required successor in the graph  $G$  only if it does not yet exist in the graph: this step merely mimics Algorithm 1 and therefore uses global caching.

In Algorithm 2, a node that contains both  $A$  and  $\neg A$  for some atomic concept  $A$  becomes an end-node with status  $\mathbf{unsat}$  (i.e. unsatisfiable w.r.t.  $T$ ). A node to which no tableau rule is applicable becomes an end-node with status  $\mathbf{sat}$  (i.e. satisfiable w.r.t.  $T$ ). Both conclusions are **irrevocable** because each relies only on classical propositional principles and not on modal principles. That is, we do not need to undo either of these at any stage.

On the other hand, an application of  $(\sqcup)$  to a node  $v$  causes  $v$  to be an *or-node*, while an application of  $(\sqcap)$  or  $(\exists R)$  to a node  $v$  causes  $v$  to be an *and-node*. Steps (2h) and (2i) try to compute the status of such a non-end-node  $v$  using the kind (or-node/and-node) of  $v$  and the status of the successors of  $v$ , treating  $\mathbf{unsat}$  as irrevocably **false** and  $\mathbf{sat}$  as irrevocably **true**.

If these steps cannot determine the status of  $v$  as  $\mathbf{sat}$  or  $\mathbf{unsat}$ , then its status is set to  $\mathbf{expanded}$ . But if these steps do determine the status of a node  $v$  to be  $\mathbf{sat}$  or  $\mathbf{unsat}$ , this information is itself propagated to the predecessors of  $v$  in the and-or graph  $G$  via the routine *propagate*( $G, v$ ), explained shortly.

The main loop ends when the status of the initial node  $\tau$  becomes  $\mathbf{sat}$  or  $\mathbf{unsat}$  or all nodes of the graph have been expanded. In the latter case, all nodes with status  $\neq \mathbf{unsat}$  are given status  $\mathbf{sat}$  (effectively giving the status *open* to tableau branches which loop). Again, caching is present at Step 2(g)i.

The procedure *propagate* used in the above algorithm is specified in Figure 3. As parameters, it accepts an and-or graph  $G$  and a node  $v$  with (irrevocable)

status **sat** or **unsat**. The purpose is to propagate the status of  $v$  through the and-or graph and alter  $G$  to reflect the new information.

Initially, the queue of nodes to be processed contains only  $v$ . Then while the queue is not empty: a node  $x$  is extracted; the status of  $x$  is propagated to each predecessor  $u$  of  $x$ ; and if the status of a predecessor  $u$  becomes (irrevocably) **sat** or **unsat** then  $u$  is inserted into the queue for further propagation.

This construction thus uses both caching and propagation techniques.

**Proposition 1.** *Algorithm 2 runs in EXPTIME.*

*Proof.* Let  $G = \langle V, E \rangle$  be the graph constructed by Algorithm 2 for  $\Gamma$  and  $X$  and  $n$  be the size of input, i.e. the sum of the lengths of the concepts of  $\Gamma \cup X$ .

Each  $v \in V$  has  $v.content \subseteq Sf(\Gamma \cup X)$ , hence  $v.content$  has size  $2^{O(n)}$ . For all  $v, w \in V$ , if  $v \neq w$  then  $v.content \neq w.content$ . Hence  $V$  contains  $2^{O(n)}$  nodes.

Every  $v \in V$  is expanded (by Steps (2a)–(2j)) only once and every expansion takes  $2^{O(n)}$  time units not counting the execution time of procedure *propagate* since  $v.content$  contains  $2^{O(n)}$  concepts and we still have to search for proxies amongst possibly  $2^{O(n)}$  nodes in  $V$ . When  $v.status$  becomes **sat** or **unsat**, the procedure *propagate* executes  $2^{O(n)}$  basic steps directly involved with  $v$ , so the total time of the executions of *propagate* is of rank  $2^{2 \cdot O(n)}$ . Hence Algorithm 2 runs in exponential time.

**Lemma 5.** *It is an invariant of Algorithm 2 that for every  $v \in V$ :*

1. if  $v.status = \mathbf{unsat}$  then
  - $v.content$  contains both  $A$  and  $\neg A$  for some atomic concept  $A$ ,
  - or  $v.kind = \mathbf{and-node}$  and there exists  $(v, w) \in E$  such that  $w \neq v$  and  $w.status = \mathbf{unsat}$ ,
  - or  $v.kind = \mathbf{or-node}$  and for every  $(v, w) \in E$ ,  $w.status = \mathbf{unsat}$ ;
2. if  $v.status = \mathbf{sat}$  then
  - no *CALC*-tableau rule is applicable to  $v.content$ ,
  - or  $v.kind = \mathbf{or-node}$  and there exists  $(v, w) \in E$  with  $w.status = \mathbf{sat}$ ,
  - or  $v.kind = \mathbf{and-node}$  and for every  $(v, w) \in E$ ,  $w.status = \mathbf{sat}$ .

(If  $v.kind = \mathbf{or-node}$  and  $(v, w) \in E$  then  $w \neq v$  since  $w.content \neq v.content$ .)

**Lemma 6.** *Let  $G = \langle V, E \rangle$  be the graph constructed by Algorithm 2 for  $\Gamma$  and  $X$ . For every  $v \in V$ , if  $v.status = \mathbf{unsat}$  then  $v.content$  is inconsistent w.r.t.  $\Gamma$ .*

*Proof.* Using Lemma 5, we can construct a closed tableau w.r.t.  $\Gamma$  for  $v.content$  by induction on the way  $v$  depends on its successors and by copying nodes to ensure that the resulting structure is a (tree) tableau rather than a graph.

Let  $G = \langle V, E \rangle$  be the graph constructed by Algorithm 2 for  $\Gamma$  and  $X$ . For  $v \in V$  with  $v.status = \mathbf{sat}$ , we say that  $v_0 = v, v_1, \dots, v_k$  with  $k \geq 0$  is a *saturation path* of  $v$  in  $G$  if for each  $1 \leq i \leq k$ , we have  $v_i.status = \mathbf{sat}$ , the edge  $E(v_{i-1}, v_i)$  was created by an application of  $(\sqcap)$  or  $(\sqcup)$ , and  $v_k.content$  contains no concepts of the form  $C \sqcap D$  nor  $C \sqcup D$ . By Lemma 5, if  $v.status = \mathbf{sat}$  then there exists a saturation path of  $v$  in  $G$ .

**Lemma 7.** *Let  $G = \langle V, E \rangle$  be the graph constructed by Algorithm 2 for  $\Gamma$  and  $X$ . For all  $v \in V$ , if  $v.status = \mathbf{sat}$  then every tableau w.r.t.  $\Gamma$  for  $v.content$  is open.*

*Proof.* Choose any  $v \in V$  with  $v.status = \mathbf{sat}$  and let  $T$  be an arbitrary tableau (tree) w.r.t.  $\Gamma$  for  $v.content$ .

We maintain a *current node*  $cn$  of  $T$  that will follow edges of  $T$  to pin-point an open branch of  $T$ . Initially we set  $cn := v$ . We also keep a (finite) saturation path  $\sigma$  of the form  $\sigma_0, \dots, \sigma_k$  for some  $\sigma_0 \in V$  and call  $\sigma$  the *current saturation path in  $G$* . At the beginning, set  $\sigma_0 := v$ , so  $v$  is a node of both  $T$  and  $G$  and let  $\sigma$  be a saturation path for  $\sigma_0$  in  $G$ : we know  $\sigma$  exists since  $v.status = \mathbf{sat}$ .

We maintain the following invariant where  $cn.content$  is the set carried by  $cn$ :

Invariant:  $\forall C \in cn.content. \exists i. 0 \leq i \leq k, C \in \sigma_i.content$ .

*Remark 1.* Observe that if  $C \in \sigma_i.content$  for some  $0 \leq i \leq k$  and  $C$  is of the form  $A, \neg A, \exists R.D$ , or  $\forall R.D$  then  $C \in \sigma_k.content$  since the saturation process does not affect concepts of these forms. By the definition of saturation path, we know that  $\sigma_k.status = \mathbf{sat}$ , hence the  $(\perp)$ -rule is not applicable to  $\sigma_k.content$ . Hence, the invariant implies that  $cn.content$  does not contain a pair  $A, \neg A$  for any atomic concept  $A$ , and thus the rule  $(\perp)$  is not applicable to  $cn$ . Also, note that the universal quantification over  $C$  encompasses the existential quantification over  $i$ , so each  $C$  can have a different  $\sigma_i$  in the invariant.

Clearly, the invariant holds at the beginning with  $i = 0$  since  $\sigma_0 = v = cn$  is in  $\sigma$ . Depending upon the rule applied to  $cn$  in the tableau  $T$ , we maintain the invariant by changing the value of the current node  $cn$  of  $T$  and possibly also the current saturation path  $\sigma$  in  $G$ . By Remark 1, the branch formed by the instances of  $cn$  is an open branch of  $T$ .

**Theorem 2.** *Let  $G = \langle V, E \rangle$  be constructed by Algorithm 2 for  $\Gamma$  and  $X$ , with  $\tau \in V$  as the initial node. Then  $X$  is satisfiable w.r.t.  $\Gamma$  iff  $\tau.status = \mathbf{sat}$ .*

**Corollary 1.** *Algorithm 2 is an EXPTIME decision procedure for  $\mathcal{ALC}$ .*

We have extended our method to SHI and also to regular RBoxes. It can also be extended for checking consistency of an ABox w.r.t. a TBox in  $\mathcal{ALC}$ .

## References

1. F. Donini and F. Massacci. EXPTIME tableaux for  $\mathcal{ALC}$ . *Artificial Intelligence*, 124:87–138, 2000.

# Optimizing Tableau Reasoning in $\mathcal{ALC}$ Extended with Uncertainty

Volker Haarslev, Hsueh-Ieng Pai, and Nematollaah Shiri

Concordia University  
Dept. of Computer Science & Software Engineering  
Montreal, Quebec, Canada  
{haarslev, hsueh\_pa, shiri}@cse.concordia.ca

**Abstract.** There has been an increased interest in recent years to incorporate uncertainty in Description Logics (DLs), and a number of proposals have been put forward for modeling uncertainty in DL frameworks. While much progress has been made on syntax, semantics, and query processing issues, optimizing queries in this context has received little attention. In this paper, we study query processing for a tableau-based DL framework with uncertainty and focus on optimization of resolution of certainty inequality constraints, obtained from a translation in query processing phase. We develop a running prototype which evaluates DL knowledge bases with ABoxes and TBoxes annotated with uncertainty parameters and computes the corresponding semantics encoded as a set of constraints in the form of linear and/or nonlinear inequations. We also explore various existing and new opportunities for optimizing the reasoning procedure in this context. Our experimental evaluation indicates that the optimization techniques we considered result in improved efficiency significantly.

## 1 Introduction

*Uncertainty* is a form of imperfection commonly found in the real-world information, and refers to situations where the truth of such information is not established definitely. Despite of recent advances on extending Description Logics (DLs) with various forms of uncertainty (such as vagueness or probability), there is generally a lack of effort in studying optimization aspects of uncertainty reasoning. This paper is the first step in this direction.

This work is a continuation of our previous theoretical work on extending the DL fragment  $\mathcal{ALC}$  with various forms of uncertainty [4–6] in which we abstract away the notion of uncertainty in the description language, the knowledge base, and the reasoning services, and we encode their corresponding semantics as a set of constraints in the form of linear and/or nonlinear inequations. In this paper, we explore various opportunities for optimizing the tableau-based reasoning procedure for the prototype of our generic framework called GURDL – a Generic Uncertainty Reasoner for the DL  $\mathcal{ALC}$ .

The rest of this paper is organized as follows. Section 2 provides a brief overview of our generic framework for DL with uncertainty. We also review the

existing tools that are available in this area. In Section 3, we present some optimization techniques that are implemented in GURDL, while Section 4 reports our performance evaluation results. Finally, we conclude in Section 5 with some directions for future work.

## 2 Related Work

In this section, we first give a brief overview of our generic framework for DL with uncertainty. We then survey the existing tools that are available for reasoning with DL and uncertainty.

### 2.1 Generic Framework for DL with Uncertainty

As mentioned in [6], existing extensions of DLs with uncertainty can be classified into one of the three approaches according to the underlying mathematical foundation and the type of uncertainty they model: (1) the fuzzy approach (such as [9, 10]), based on fuzzy set theory, essentially deals with the vagueness in the knowledge; (2) the probabilistic approach (such as [1, 3, 8]), based on the classical probability theory, deals with the uncertainty due to lack of knowledge; (3) the possibilistic approach [7], based on possibility theory, allows necessity and possibility measures to be handled in the same formalism.

In order to support the various forms of uncertainty within the same framework, we abstracted away the notion of uncertainty (fuzzy logic, probability, possibilistic logic) and proposed a generic framework for DL with uncertainty [5]. In particular, our generic framework consists of three components:

1. *Description Language with Uncertainty*: In our framework, we keep the syntax of the description language identical to that of the classical  $\mathcal{ALC}$ , while extending the corresponding semantics with uncertainty. In order to flexibly represent various forms of uncertainty, we assume that certainty values form a complete lattice  $\mathcal{L} = \langle \mathcal{V}, \preceq \rangle$ , where  $\mathcal{V}$  is the certainty domain, and  $\preceq$  is the partial order defined on  $\mathcal{V}$ . We also use  $b$  to denote the least element in  $\mathcal{V}$ ,  $t$  for the greatest element in  $\mathcal{V}$ ,  $\oplus$  for the join operator in  $\mathcal{L}$ ,  $\otimes$  for its meet operator, and  $\sim$  for the negation operator.

The semantics of the description language is based on the notion of an interpretation, where an interpretation  $\mathcal{I}$  is defined as a pair  $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where  $\Delta^{\mathcal{I}}$  is the domain and  $\cdot^{\mathcal{I}}$  is an interpretation function. For example, if individual  $John \in \Delta^{\mathcal{I}}$ , then  $Obese^{\mathcal{I}}(John)$  gives the certainty that  $John$  belongs to concept  $Obese$ . The syntax and the semantics of the description language supported in our framework are summarized in Table 1. Note that  $f_c$  and  $f_d$  in the table denote conjunction and disjunction functions. They are used to specify how one should interpret a given description language. For example, in the fuzzy approach, we would have the *min* function as  $f_c$  and *max* function as  $f_d$ , whereas in a probabilistic approach, we might have algebraic product ( $prod(\alpha, \beta) = \alpha\beta$ ) as  $f_c$ , and the independent function ( $ind(\alpha, \beta) = \alpha + \beta - \alpha\beta$ ) as  $f_d$ .

Name	Syntax	Semantics ( $a \in \Delta^{\mathcal{I}}$ )
Atomic Concept	$A$	$A^{\mathcal{I}}(a) = CF_C$ , with $CF_C : \Delta^{\mathcal{I}} \rightarrow \mathcal{V}$
Atomic Role	$R$	$R^{\mathcal{I}}(a, b) = CF_R$ , with $CF_R : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow \mathcal{V}$
Top Concept	$\top$	$\top^{\mathcal{I}}(a) = t$
Bottom Concept	$\perp$	$\perp^{\mathcal{I}}(a) = b$
Concept Negation	$\neg C$	$(\neg C)^{\mathcal{I}}(a) = \sim C^{\mathcal{I}}(a)$
Concept Conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}}(a) = f_c(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$
Concept Disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}}(a) = f_d(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$
Role Exists Restriction	$\exists R.C$	$(\exists R.C)^{\mathcal{I}}(a) = \oplus_{b \in \Delta^{\mathcal{I}}} \{f_c(R^{\mathcal{I}}(a, b), C^{\mathcal{I}}(b))\}$
Role Value Restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}}(a) = \otimes_{b \in \Delta^{\mathcal{I}}} \{f_d(\sim R^{\mathcal{I}}(a, b), C^{\mathcal{I}}(b))\}$

**Table 1.** Syntax/Semantics of the Description Language Supported

2. *Knowledge Bases with Uncertainty:* As usual, the knowledge base ( $\Sigma$ ) consists of both the TBox and the ABox. However, unlike the classical case, each axiom and assertion is associated with a certainty value, as well as the conjunction/disjunction functions used to interpret the concept descriptions. More specifically, the TBox includes a set of terminological axioms that could be concept subsumptions  $\langle C \sqsubseteq D, \alpha \rangle \langle f_c, f_d \rangle$  and/or concept definitions  $\langle C \equiv D, \alpha \rangle \langle f_c, f_d \rangle$ , where  $C$  and  $D$  are concept descriptions,  $\alpha \in \mathcal{V}$  is the certainty that the axiom holds, and  $f_c$  and  $f_d$  are the conjunction and disjunction functions. As an example, the certainty of the axiom  $\langle Rich \sqsubseteq ((\exists owns. ExpensiveCar \sqcup \exists owns. Airplane) \sqcap Golfer), [0.8, 1] \rangle \langle min, max \rangle$  is at least 0.8, with all the concept conjunctions interpreted using  $min$ , and all the concept disjunctions interpreted using  $max$ . Note that, although our framework supports simple probabilities such as independent or mutually exclusive events, we are investigating ways to model knowledge base with more general probability theory such as conditional independence, since reasoning with it requires extra information about the events and facts in the world ( $\Sigma$ ).

The ABox in our framework consists of a set of concept assertions of the form  $\langle a : C, \alpha \rangle \langle f_c, f_d \rangle$  or role assertions  $\langle (a, b) : R, \alpha \rangle \langle -, - \rangle$ , where  $a, b$  are individuals,  $C$  is a concept,  $R$  is a role,  $\alpha \in \mathcal{V}$ ,  $f_c$  is the conjunction function,  $f_d$  is the disjunction function, and  $-$  denotes that the corresponding function is not applicable. For instance, the assertion “Mary is tall and thin with a degree between 0.6 and 0.8” can be expressed as  $\langle Mary : Tall \sqcap Thin, [0.6, 0.8] \rangle \langle min, - \rangle$ . Here,  $min$  is used as the conjunction function, and the disjunction function is not necessary since there is no concept disjunction here.

3. *Reasoning with Uncertainty:* The inference problems supported by our framework include the satisfiability problem and the entailment problem, where the former checks if an admissible knowledge base is satisfiable and the later determines the degree with which an assertion is true given the knowledge base. Similar to the classical DL reasoning, pre-processing steps are first applied to abstract the TBox. Then, completion rules are applied to simplify the ABox, and blocking is introduced to ensure termination [4]. However, unlike the classical case, each rule application generates a set of derived assertions and a set of constraints in the form of linear/nonlinear inequations

which encode the semantics of the assertion. For example, given the assertion  $\langle Mary : Tall \sqcap Thin, [0.6, 0.8] \rangle \langle min, - \rangle$ , the conjunction rule can be applied, which yields  $\langle Mary : Tall, x_{Mary:Tall} \rangle \langle -, - \rangle$  and  $\langle Mary : Thin, x_{Mary:Thin} \rangle \langle -, - \rangle$ , and the constraint  $(min(x_{Mary:Tall}, x_{Mary:Thin}) = [0.6, 0.8])$ , where  $x_{Mary:Tall}$  (resp.,  $x_{Mary:Thin}$ ) is the variable representing the certainty that *Mary* is *Tall* (resp., *Thin*). The completion rules we introduced in [4] are applied in arbitrary order until either the extended ABox contains a clash or no further rule could be applied. If a clash is encountered (such as an assertion has two conflicting certainty values), the knowledge base is unsatisfiable. Otherwise, a constraint solver is called to solve/optimize the system of inequations to check satisfiability of the knowledge base or the degree with which an assertion is true. Details and the proof for the correctness of the reasoning procedure can be found in [4].

## 2.2 Existing Tools for DL with Uncertainty

To the best of our knowledge, the only DL/uncertainty reasoner that is publicly available is fuzzyDL [2], which inspired our preliminary prototype. As the name suggests, fuzzyDL supports only fuzzy knowledge (i.e., it cannot handle other uncertainty formalisms such as probabilities). Although fuzzyDL supports two types of fuzzy knowledge – those with Zadeh semantics and those with Lukasiewicz logic, it uses two sets of completion rules instead of using a generic set of inference rules to deal with different semantics. Nevertheless, fuzzyDL has some interesting features. For example, it supports concept modifiers and a more expressive fragment of DL *SHIF*.

## 3 Optimization Techniques Employed in GURDL

GURDL is the prototype of our generic framework for DL with uncertainty. A number of optimization techniques have been incorporated in GURDL. Due to the limited space, we discuss only some of them here. The idea is to investigate whether some existing optimization techniques used in classical DL systems could be applied to the uncertainty case (including lexical normalization, concept simplification, partition based on connectivity as Individual Groups, and caching), while exploring new optimization technique that is specific to deal with uncertainty (partition based on connectivity as Assertion Groups).

### 3.1 Lexical Normalization

Lexical normalization is a common optimization technique used in classical DL systems, where concepts are transformed into a canonical form. For example, concepts like  $(C \sqcap (B \sqcap A))$ ,  $(B \sqcap (C \sqcap A))$ , and  $((B \sqcap A) \sqcap C)$  can all be transformed into the canonical form  $(A \sqcap (B \sqcap C))$ . In GURDL, lexical normalization is realized by sorting the sub-concepts in the concept description. The major advantage of lexical normalization is that it allows obvious clashes be detected early. For example, given the assertions  $\langle Mary : Tall \sqcap Thin, [0.8, 1] \rangle$  and

$\langle Mary : Thin \sqcap Tall, [0, 0.4] \rangle$ , the second assertion becomes  $\langle Mary : Tall \sqcap Thin, [0, 0.4] \rangle$  after the normalization. This allows us to easily notice the inconsistency between the two assertions, which would be hard to detect otherwise. Another advantage of lexical normalization is that it facilitates concept simplification.

### 3.2 Concept Simplification

Concept simplification is another optimization technique that is commonly employed in classical DL systems, done by removing redundant sub-concepts in a given concept. In GURDL, the following simplifications are applied:

$$\begin{array}{lll} \top \sqcap C \rightsquigarrow C & \top \sqcup \dots \rightsquigarrow \top & \forall R. \top \rightsquigarrow \top \\ \perp \sqcap \dots \rightsquigarrow \perp & \perp \sqcup C \rightsquigarrow C & \exists R. \perp \rightsquigarrow \perp \end{array}$$

The above simplifications are valid due to the boundary-condition properties of the combination functions [5]. Note that simplification must be applied with care when uncertainty is introduced. For example, it is a common practice in classical DL systems to remove duplicated sub-concepts in a concept conjunction or disjunction, such as simplifying  $(A \sqcap A)$  to  $A$ . However, such simplification is not valid once uncertainty is introduced. For example, assume that the interpretation of concept  $A$  is 0.4. If the conjunction function is *min*, then  $(A \sqcap A)$  is  $A$  since  $\min(0.4, 0.4) = 0.4$ . However, if the conjunction function is the algebraic product ( $\times$ ), then  $(A \sqcap A)$  is not the same as  $A$ , since  $\times(0.4, 0.4) = 0.16 \neq 0.4$ . Therefore, such simplification is invalid, hence cannot be applied.

The major advantage of the simplification method is that it could potentially reduce the number of sub-concepts in a concept description, hence reducing the number of completion rule applications. In some extreme case, a complicated concept description can be simplified to only  $\top$  or  $\perp$ , hence eliminating the need to apply the completion rule.

### 3.3 Partition Based on Connectivity

In GURDL, the ABox is partitioned into Individual Groups (IGs) and Assertions Groups (AGs) based on the notion of connectivity.

**Individual Groups (IGs)** Similar to the classical DL systems, the individuals in the ABox are divided into one or more partitions called Individual Groups. Each group consists of individuals that are “related” to each other through role assertions. By partitioning the ABox this way, inferences can be performed independently for each IG. Once no more completion rule can be applied to a given IG, we could pass the derived assertions and their corresponding constraints to the constraint solver to build the model, and we can be sure that the model built will not be changed even if we perform inference on other IGs in the ABox. This allows the consistency of the ABox be checked incrementally and hence reduces the reasoning complexity when the knowledge base includes many individuals which could be partitioned as described. This also allows us to check the consistency of the ABox related to one particular individual without checking the consistency of the complete ABox.

**Assertion Groups (AGs)** As mentioned in Section 2.1, the reasoning procedure for our uncertainty framework differs from the classical one because, in addition to derive assertions, a set of constraints in the form of linear/nonlinear inequations is also generated, which is later on feed into the constraint solver to check for its consistency. Since the number of constraints generated is usually large, it is important to optimize the constraint solving process.

In GURDL, each IG is partitioned into one or more independent subsets called Assertion Groups. In general, two assertions  $A_1$  and  $A_2$  are in the same AG if  $A_1$  is directly or indirectly inferred from  $A_2$  (through the application of completion rules), or  $A_1$  and  $A_2$  differ only in terms of their certainty values and/or conjunction and disjunction functions. The interesting property about this partition is that, when we union all the constraints (resp., variables associated with the constraints) in the AGs that belong to a particular IG, we obtain all the constraints (resp., variables associated with the constraints) in that IG. On the other hand, if we take the intersection, we obtain an empty set. This implies that constraints in each AG can be solved independently, while assuring that the model built will not be changed when we solve constraints in other AGs.

This has several advantages. First, the consistency of the IG can be checked incrementally. At any given time, the constraints in one single AG are fed into the constraint solver. If any AG is found to be inconsistent, this implies that the whole IG is inconsistent. A related advantage is that, in case an IG is inconsistent, the reasoner will be able to more precisely identify the assertions that cause the inconsistency. Another advantage is that we are now able to determine the degree to which a particular assertion (say,  $X$ ) is true by simply solving the constraints in the AG that  $X$  belongs. Finally, since the number of constraints (and the variables used in the constraints) in one single AG is, in general, no more than those of the whole IG, the speed of solving a few small constraint sets would be faster than solving one large constraint set. The performance evaluation of AG-partitioning is studied in Section 4.

### 3.4 Caching

To save the reasoner from doing redundant/repeated work, each assertion and constraint is stored only once. A flag is set to indicate whether completion rules have been applied to a given assertion (resp., IG). In addition, after the constraints in an AG are solved, the result is cached for later use.

## 4 Performance Evaluation

In this section, we study the performance of GURDL. All the experiments were conducted under Windows XP on a Pentium 2.40 GHz computer with 3.25 GB of RAM. Due to the limited space, we present only highlights of our results here.

Table 2 lists a few test cases, the number of concept assertions (C) in each test case, the number of role assertions (R), the number of axioms with necessary condition (N), the number of axioms with concept definitions (D), the functions

Test Case	C	R	N	D	F	$\mathcal{V}$	IG	AG	W	H	L	I	S	O	Total
1. Classical	15	2	5	0	<i>min/max</i>	{0, 1}	3	84	25	6	0.014	0.10	2.19	0.22	2.52
2. Min-Max	15	2	5	0	<i>min/max</i>	[0, 1]	3	84	25	6	0.015	0.10	2.47	0.19	2.78
3. Mixed	15	2	5	0	<i>mixed</i>	[0, 1]	3	159	44	6	0.016	0.17	12.92	0.32	13.43
4. Min-Max/Def.	15	2	0	5	<i>min/max</i>	[0, 1]	3	13	58	6	0.016	0.54	21.36	0.29	22.20
5. University	1	0	47	6	<i>min/max</i>	{0, 1}	1	231	45	5	0.020	1.25	17.15	0.75	19.17

**Table 2.** Performance of test cases (in seconds)

used to interpret the concept description (F), the certainty domain ( $\mathcal{V}$ ), the number of IGs (IG), the number of AGs (AG), the width of the ABox (W), the height of the ABox (H), the time to load the knowledge base (L), the time to apply the inference rules (I), the time to solve constraints (S), other time (mostly I/O) (O), and the total time for ABox consistency checking (L + I + S + O). All the time measures are in seconds.

As shown in the table, the time spent on solving constraints (S) dominates the overall reasoning time (Total) for all the test cases. Note also that test cases 1 and 2 differ by the certainty domain, but this has limited effect on the performance. Test cases 2 and 3 differ by the functions used to interpret the description language (F). We can see that it takes longer to solve constraints that include a mix of nonlinear functions (*prod*, *ind*) and simple ones (*min*, *max*). Test cases 4 illustrates that it takes longer when we have axioms with concept definitions (D) instead of those with necessary conditions (N). Test case 5 shows the case where an IG is partitioned into many AGs.

Note that our prototype runs slower than the classical reasoners for standard knowledge bases, where we use {0, 1} for the certainty domain, and *min* and *max* for conjunction and disjunction functions (one or two seconds vs. many seconds). This was expected, partly because standard reasoners implement many more optimization techniques, some of which we could not use in our context. Also, unlike in our context, they do not need to rely on constraint solvers as part of their reasoning process. Note also that we have not compared the performance with fuzzyDL here, because fuzzyDL uses a different constraint solver than GURDL, and the effect of such factor is not negligible.

Table 3 compares the total time for solving constraints when we partition the ABox into AGs, IGs, or no partition at all (ALL). Note that when we partition the ABox into AGs, the performance is the best. Note also that for the test case University, when the ABox is not partitioned into AGs, the constraint set is simply too large for the constraint solver to handle (we get stack overflow error). This shows the importance of keeping the constraint set as small as possible by partitioning the ABox.

## 5 Conclusion and Future Work

In this paper, we have explored various existing and new optimizing techniques for the reasoning procedure of our generic framework for DL with uncertainty, for which we have incorporated in our prototype. Due to the space limit, we present

Test Case	AG	IG	ALL	Gain1	Gain2	Gain3
1. Classical	2.52	3.42	4.84	26.31%	29.37%	47.95%
2. Min-Max	2.78	4.15	6.17	32.99%	32.68%	54.88%
3. Mixed	13.43	25.54	28.99	47.43%	11.90%	53.69%
4. Min-Max/Def.	22.20	37.24	100.58	40.37%	62.98%	77.93%
5. University	19.17	N/A	N/A	N/A	N/A	N/A

**Table 3.** Performance evaluation for partition based on connectivity (in seconds) (Gain1: AG vs. IG, Gain2: IG vs. ALL, Gain3: AG vs. ALL)

the partial performance evaluation result, which shows that the optimization techniques we employed are effective. As future research, we plan to extend the generic framework to a more expressive portion of DL. We also plan to optimize the reasoning procedure further. For example, since constraint-solving is the phase that takes the longest time, in case we have multiple AGs, we could solve them concurrently by running multiple threads on different computers. Another optimization would be to reduce the number of constraints or the number of variables in the constraints generated during the reasoning procedure. These methods are expected to greatly enhance the performance.

**Acknowledgement.** This work was supported in part by Natural Sciences and Engineering Research Council (NSERC) of Canada, and by ENCS Concordia University. We also thank anonymous reviewers for their helpful comments.

## References

1. Ding, Z., Peng, Y., and Pan, R. A Bayesian approach to uncertainty modeling in OWL ontology. In *Proc. of AISTA*, Luxembourg, 2004.
2. fuzzyDL. <http://gaia.isti.cnr.it/straccia/software/fuzzyDL/fuzzyDL.html>.
3. Giugno, R. and Lukasiewicz, T. P-*SHOQ(D)*: A probabilistic extension of *SHOQ(D)* for probabilistic ontologies in the Semantic Web. In *Proc. of JELIA*, pages 86–97, London, 2002. Springer-Verlag.
4. Haarslev, V., Pai, H.I., and Shiri, N. A formal framework for description logics with uncertainty. *Submitted to International Journal of Automate Reasoning*.
5. Haarslev, V., Pai, H.I., and Shiri, N. Completion rules for uncertainty reasoning with the description logic *ALC*. In *Proc. of CSWWS*, pages 205–225, Quebec City, 2006. Springer Verlag.
6. Haarslev, V., Pai, H.I., and Shiri, N. Uncertainty reasoning in description logics: A generic approach. In *Proc. of FLAIRS*, pages 818–823, Florida, 2006. AAAI Press.
7. Hollunder, B. An alternative proof method for possibilistic logic and its application to terminological logics. In *Proc. of UAI*, pages 327–335. Morgan Kaufmann, 1994.
8. Koller, D., Levy, A. Y., and Pfeffer, A. P-CLASSIC: A tractable probabilistic description logic. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 390–397, Providence, Rhode Island, July 1997. AAAI Press.
9. Straccia, U. Description logics with fuzzy concrete domains. In *Proc. of UAI*, 2005.
10. Tresp, C. and Molitor, R. A description logic for vague knowledge. In *Proc. of ECAI*, pages 361–365, Brighton, UK, 1998. John Wiley and Sons.

## Distributed Description Logics Revisited

Martin Homola

Comenius University, Bratislava, Slovakia,  
 Faculty of Mathematics, Physics and Informatics,  
 homola@fmph.uniba.sk

**Abstract.** Distributed Description Logics (DDLs) is a KR formalism that enables reasoning with multiple ontologies interconnected by directional semantic mapping (bridge rules). DDLs capture the idea of importing and reusing concepts between ontologies and thus combine well with intuitions behind Semantic Web.

We modify the original semantics of DDLs in order to cope with a modeling discrepancy that has been pointed out in the literature. We do so by introducing a new kind of bridge rules, which we call conjunctive. Using conjunctive bridge-rules instead of the normal ones solves the problem. All the basic properties that have been established for DDLs hold also for the adjusted framework. We also provide a transformational semantics for conjunctive bridge rules, and thus, at least theoretically, a decision procedure for the new semantics.

### 1 Introduction

Distributed description logic (DDL) is a KR formalism introduced by Borgida, Serafini and Tamarin in [1,2,3], intended especially to enable reasoning between multiple ontologies connected by directional semantic mapping (bridge rules), built upon the formal, logical and well established framework of Description Logics (DLs). DDLs capture the idea of importing and reusing concepts between several ontologies. This idea combines well with the basic assumption of Semantic Web that no central ontology but rather many ontologies with redundant knowledge will exist [4].

It has been noted in [5] that DDLs and the derived framework of C-OWL [6] suffer from several drawbacks. Among these is the unintuitive behaviour in a modeling scenario outlined therein. We analyse this problem and cope with it by introducing a new kind of bridge rules with modified semantics. We then evaluate the new semantics with respect to the desiderata that have been postulated for DDLs. We also provide a transformational semantics for conjunctive bridge rules, and thus, at least theoretically, a decision procedure for the new semantics.

### 2 Distributed Description Logics

As introduced in [1,2,3], a DDL knowledge base consists of a distributed TBox  $\mathfrak{T}$  – a set of local TBoxes  $\{\mathcal{T}_i\}_{i \in I}$ , and a set of bridge rules  $\mathfrak{B} = \bigcup_{i,j \in I, i \neq j} \mathfrak{B}_{ij}$

between these local TBoxes, for some non-empty index-set  $I$ . Each of the local TBoxes  $\mathcal{T}_i$  is a collection of axioms called general concept inclusions (GCIs) in its own local DL  $\mathcal{L}_i$  of the form:  $i : C \sqsubseteq D$ . It is assumed that each  $\mathcal{L}_i$  is a sub-language of  $\mathcal{SHIQ}$  [7]. Each  $\mathfrak{B}_{ij}$  is a set of directed bridge rules from  $\mathcal{T}_i$  to  $\mathcal{T}_j$ . Intuitively, these are meant to “import” information from  $\mathcal{T}_i$  to  $\mathcal{T}_j$ , and therefore  $\mathfrak{B}_{ij}$  and  $\mathfrak{B}_{ji}$  are possibly, and expectedly, distinct. Bridge rules of  $\mathfrak{B}_{ij}$  are of two forms, *into*-bridge rules and *onto*-bridge rules (in the respective order):

$$i : A \stackrel{\sqsubseteq}{\Rightarrow} j : G \quad , \quad i : B \stackrel{\supseteq}{\Rightarrow} j : H \quad .$$

Given a TBox  $\mathcal{T}$ , a hole is an interpretation  $\mathcal{I}^\epsilon = \langle \emptyset, \cdot^\epsilon \rangle$  with empty domain. Holes are used for fighting propagation of inconsistency. We use the most recent definition for holes, introduced in [3]. A distributed interpretation  $\mathfrak{J} = \langle \{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \in I, i \neq j} \rangle$  of a distributed TBox  $\mathfrak{T}$  consists of a set of local interpretations  $\{\mathcal{I}_i\}_{i \in I}$  such that for each  $i \in I$  either  $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i} \rangle$  is an interpretation of local TBox  $\mathcal{T}_i$  or  $\mathcal{I}_i = \mathcal{I}^\epsilon$  is a hole, and a set of domain relations  $r_{ij}$  between these domains – each  $r_{ij}$  is a subset of  $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ . We denote by  $r_{ij}(d)$  the set  $\{d' \mid \langle d, d' \rangle \in r_{ij}\}$  and by  $r_{ij}(D)$  the set  $\bigcup_{d \in D} r_{ij}(d)$ .

**Definition 1.** For every  $i$  and  $j$ , a distributed interpretation  $\mathfrak{J}$  satisfies the elements of a distributed TBox  $\mathfrak{T}$  (denoted by  $\mathfrak{J} \models_\epsilon \cdot$ ) according to the following clauses:

1.  $\mathfrak{J} \models_\epsilon i : C \sqsubseteq D$  if  $\mathcal{I}_i \models C \sqsubseteq D$ .
2.  $\mathfrak{J} \models_\epsilon \mathcal{T}_i$  if  $\mathfrak{J} \models_\epsilon i : C \sqsubseteq D$  for each  $C \sqsubseteq D \in \mathcal{T}_i$ .
3.  $\mathfrak{J} \models_\epsilon i : C \stackrel{\sqsubseteq}{\Rightarrow} j : G$  if  $r_{ij}(C^{\mathcal{I}_i}) \subseteq G^{\mathcal{I}_j}$ .
4.  $\mathfrak{J} \models_\epsilon i : C \stackrel{\supseteq}{\Rightarrow} j : G$  if  $r_{ij}(C^{\mathcal{I}_i}) \supseteq G^{\mathcal{I}_j}$ .
5.  $\mathfrak{J} \models_\epsilon \mathfrak{B}$  if  $\mathfrak{J}$  satisfies all bridge rules in  $\mathfrak{B}$ .
6.  $\mathfrak{J} \models_\epsilon \mathfrak{T}$  if  $\mathfrak{J} \models_\epsilon \mathfrak{B}$  and  $\mathfrak{J} \models_\epsilon \mathcal{T}_i$  for each  $i$ .

If  $\mathfrak{J} \models_\epsilon \mathfrak{T}$  then we say that  $\mathfrak{J}$  is a (distributed) model of  $\mathfrak{T}$ . Finally, given  $C$  and  $D$  of some local TBox  $\mathcal{T}_i$  of  $\mathfrak{T}$ ,  $C$  is subsumed by  $D$  in  $\mathfrak{T}$  (denoted by  $\mathfrak{T} \models_\epsilon i : C \sqsubseteq D$ ) whenever, for every distributed interpretation  $\mathfrak{J}$ ,  $\mathfrak{J} \models_\epsilon \mathfrak{T}$  implies  $\mathfrak{J} \models_\epsilon i : C \sqsubseteq D$ .

### 3 The Problem

In [5] it is pointed out that certain properties of subsumption relations are not modeled properly by DDL. This problem is demonstrated by the following example that we borrow from [5].

*Example 1 ([5]).* Consider the ontology  $\mathcal{O}$ :

$$\begin{aligned} \text{NonFlying} &\equiv \neg \text{Flying} \quad , & \text{Penguin} &\sqsubseteq \text{Bird} \quad , \\ \text{Bird} &\sqsubseteq \text{Flying} \quad , & \text{Penguin} &\sqsubseteq \text{NonFlying} \quad . \end{aligned}$$

And the distributed counterpart of  $\mathcal{O}$ , divided into two ontologies  $\mathcal{O}_1$  (on the left) and  $\mathcal{O}_2$  (on the right):

$$\begin{aligned} \text{NonFlying}_1 &\equiv \neg\text{Flying}_1 , & 1 : \text{Bird}_1 &\stackrel{\exists}{=} 2 : \text{Penguin}_2 , \\ \text{Bird}_1 &\sqsubseteq \text{Flying}_1 . & 1 : \text{NonFlying}_1 &\stackrel{\exists}{=} 2 : \text{Penguin}_2 . \end{aligned}$$

As it is argued in [5], while the concept Penguin of  $\mathcal{O}$  is not satisfiable, the corresponding concept Penguin<sub>2</sub> of  $\mathcal{O}_2$  is. The problem is that, in a perfectly sane interpretation, each instance  $x \in \text{Penguin}_2^{\mathcal{I}_2}$ , is assigned to two distinct elements of  $\Delta^{\mathcal{I}_1}$ , say  $y_1$  and  $y_2$ , by  $r$ , one instance of Bird<sub>1</sub> and the other one of NonFlying<sub>1</sub>. Note that this is possible even if Bird<sub>1</sub> <sup>$\mathcal{I}_1$</sup>  and NonFlying<sub>1</sub> <sup>$\mathcal{I}_1$</sup>  are disjoint as required by ontology  $\mathcal{O}_1$ . We agree with [5] that it is intuitive to expect that bridge rules retain certain properties that GCIs have. So, we would expect Penguin<sub>2</sub> to be unsatisfiable, as we made it a “subconcept of two imported concepts” Bird<sub>1</sub> and NonFlying<sub>1</sub> which in their original ontology  $\mathcal{O}_1$  are disjoint.

Let us generalize the problem illustrated by Example 1 a bit further. We have two local TBoxes in  $\mathfrak{T}$ , say  $\mathcal{T}_i$  and  $\mathcal{T}_j$ , and we have two onto-bridge rules from  $i$  to  $j$ ,  $i : C \stackrel{\exists}{=} j : G \in \mathfrak{B}$  and  $i : D \stackrel{\exists}{=} j : H \in \mathfrak{B}$ . The problem is that the inclusion  $(G \sqcap H)^{\mathcal{I}_j} \subseteq r_{ij}((C \sqcap D)^{\mathcal{I}_i})$  does not necessarily hold in every model of  $\mathfrak{T}$ , as we would have expected. The source of our intuition here is indeed the fact that the respective inclusion  $(G \sqcap H)^{\mathcal{I}} \subseteq (C \sqcap D)^{\mathcal{I}}$  holds in every model  $\mathcal{I}$  in the case when  $C, D, G$  and  $H$  are all local concepts of some  $\mathcal{T}$  and instead of the bridge rules we have two GCIs  $G \sqsubseteq C \in \mathcal{T}$  and  $H \sqsubseteq D \in \mathcal{T}$ . We push our generalization even further and expect the respective to hold in case if  $n > 0$  onto-bridge rules are involved. Please note that this issue does not arise in case of into-bridge rules (see Theorem 3 below).

To justify this generalization, we offer Example 2 in which two distinct pairs of concepts are bridged by two onto-bridge rules.

*Example 2.* Consider two ontologies  $\mathcal{O}_1$  and  $\mathcal{O}_2$  with the following GCIs (and also possibly some other):

$$1 : \text{Tokaji}_1 \sqcap \text{Selection}_1 \sqsubseteq \text{DessertWine}_1 , \quad 2 : \text{SixPuttony}_2 \sqsubseteq \text{Tokaji}_2 \sqcap \text{Selection}_2 .$$

In order to import knowledge from  $\mathcal{O}_1$  to  $\mathcal{O}_2$  we add the following bridge rules:

$$\begin{aligned} 1 : \text{Tokaji}_1 &\stackrel{\exists}{=} 2 : \text{Tokaji}_2 , & 1 : \text{DessertWine}_1 &\stackrel{\sqsubseteq}{=} 2 : \text{SweetWine}_2 , \\ 1 : \text{Selection}_1 &\stackrel{\exists}{=} 2 : \text{Selection}_2 . \end{aligned}$$

We argue, that intuitively SixPuttony  $\sqsubseteq$  SweetWine should hold in  $\mathcal{O}_2$ . This is not the case however, since  $(\text{Tokaji}_2 \sqcap \text{Selection}_2)^{\mathcal{I}_2} \subseteq r_{12}((\text{Tokaji}_1 \sqcap \text{Selection}_1)^{\mathcal{I}_1})$  does not necessarily hold in every distributed model, as discussed above.

In the following, we introduce an alternative kind of onto-bridge rules with slightly modified semantics. We then show that this semantics follows the intuitions outlined above (Theorem 2 below).

## 4 Conjunctive Bridge Rules

We address the problem outlined above by introducing new form of onto-bridge rules. We call these new bridge rules *conjunctive* and the original form *normal*. We introduce the following syntax for them:

$$i : D \overset{\sqsupseteq}{\rightarrow} j : H .$$

In the following, we use  $i : D \overset{\rightsquigarrow}{\rightarrow} j : H$  to denote onto-bridge rules that are possibly of both kinds, either conjunctive or normal.

**Definition 2.** *The semantics of conjunctive onto-bridge rules is established by adding the following clause to Definition 1:*

$$7. \mathfrak{I} \models_{\epsilon} i : C \overset{\sqsupseteq}{\rightarrow} j : G \text{ if for each } i : D \overset{\sqsupseteq}{\rightarrow} j : H \in \mathfrak{B}, r_{ij}(C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}) \supseteq G^{\mathcal{I}_j} \cap H^{\mathcal{I}_j}.$$

Our choice of adding new kind of onto-bridge rules instead of simply replacing the old semantics is to underline the fact that both kinds can co-exist and be used according to the modeling scenario and the intentions of the ontology editor. Also, it is not yet clear, how the usage of conjunctive bridge rules affects the computational complexity of the framework. It surely introduces a significant number of additional conditions to verify. Hence it might be desirable to be allowed to choose the exact form of a bridge rule according to the modeling scenario.

## 5 Properties of Conjunctive Bridge Rules

We first show that conjunctive bridge rules are somewhat strictly stronger, in a sense, than normal bridge rules. That is, all the semantic implications caused by normal bridge rules are also in effect if conjunctive bridge rules are used instead. With conjunctive bridge rules, we have some more implications in addition.

**Theorem 1.** *Given a distributed TBox  $\mathfrak{T}$  with a set of bridge rules  $\mathfrak{B}$  and some local TBoxes  $\mathcal{T}_i$  and  $\mathcal{T}_j$  such that  $i \neq j$  and  $i : C \overset{\sqsupseteq}{\rightarrow} j : G \in \mathfrak{B}$ , for each distributed interpretation  $\mathfrak{I}$  such that  $\mathfrak{I} \models_{\epsilon} \mathfrak{T}$  it holds that  $r_{ij}(C^{\mathcal{I}_i}) \supseteq G^{\mathcal{I}_j}$ .*

The next theorem provides a characterization of conjunctive bridge rules. It says, that if we bridge between several pairs of concepts with conjunctive onto-bridge rules, say  $i : C_1 \overset{\sqsupseteq}{\rightarrow} j : G_1, \dots, i : C_n \overset{\sqsupseteq}{\rightarrow} j : G_n$ , then the implications caused to the pairs of concepts pair-wise, do propagate to intersections  $C_1 \sqcap \dots \sqcap C_n$  and  $G_1 \sqcap \dots \sqcap G_n$  of these concepts. This does not hold for normal bridge rules however, as demonstrated by Examples 1 and 2. It follows that indeed the choice of conjunctive bridge rules does solve the problem outlined by the examples.

**Theorem 2.** *Given a distributed TBox  $\mathfrak{T}$  with a set of bridge rules  $\mathfrak{B}$  and some local TBoxes  $\mathcal{T}_i$  and  $\mathcal{T}_j$  such that  $i \neq j$ , if for some  $n > 0$  the bridge rules  $i : C_1 \xrightarrow{\exists} j : G_1, \dots, i : C_n \xrightarrow{\exists} j : G_n$  are all part of  $\mathfrak{B}$  then for every distributed interpretation  $\mathfrak{I}$  such that  $\mathfrak{I} \models_{\epsilon} \mathfrak{T}$  it holds that*

$$r_{ij} \left( (C_1 \sqcap \dots \sqcap C_n)^{\mathcal{T}_i} \right) \supseteq (G_1 \sqcap \dots \sqcap G_n)^{\mathcal{T}_j} .$$

The next theorem shows that the corresponding characterization indeed holds for normal into-bridge rules, hence there is no need to introduce conjunctive into-bridge rules.<sup>1</sup>

**Theorem 3.** *Given a distributed TBox  $\mathfrak{T}$  with a set of bridge rules  $\mathfrak{B}$  and some local TBoxes  $\mathcal{T}_i$  and  $\mathcal{T}_j$  such that  $i \neq j$ , if for some  $n > 0$  the bridge rules  $i : C_1 \xrightarrow{\sqsubseteq} j : G_1, \dots, i : C_n \xrightarrow{\sqsubseteq} j : G_n$  are all part of  $\mathfrak{B}$  then for every distributed interpretation  $\mathfrak{I}$  such that  $\mathfrak{I} \models_{\epsilon} \mathfrak{T}$  it holds that*

$$r_{ij} \left( (C_1 \sqcap \dots \sqcap C_n)^{\mathcal{T}_i} \right) \subseteq (G_1 \sqcap \dots \sqcap G_n)^{\mathcal{T}_j} .$$

## 6 Transformational Semantics

It follows that the problem of deciding subsumption with respect to a distributed knowledge base that allows conjunctive bridge rules is reducible to the case with normal bridge rules only (Theorem 4 below). As a tableaux decision procedure is known for the latter case (see [2,3]), this result provides us with reasoning support for DDLs with conjunctive bridge-rules. However, the transformation leads to quadratic blowup in the number of bridge rules in the worst case, and so the computational properties of the overall procedure may not be satisfiable. This suggests further investigation of reasoning in presence of conjunctive bridge rules.

**Theorem 4.** *Given a distributed TBox  $\mathfrak{T}$  with a set of bridge rules  $\mathfrak{B}$  that contains conjunctive bridge rules, let  $\mathfrak{T}'$  and  $\mathfrak{B}'$  be obtained in two steps:*

1. *adding  $i : C \sqcap D \xrightarrow{\exists} j : G \sqcap H$  to  $\mathfrak{B}$  for each pair of  $i : C \xrightarrow{\exists} j : G \in \mathfrak{B}$  and  $i : D \xrightarrow{\exists} j : H \in \mathfrak{B}$ ,*
2. *removing all conjunctive bridge rules from  $\mathfrak{B}$ .*

*Then for every  $i \in I$  and for every two concepts, say  $C$  and  $D$ , of  $\mathcal{T}_i$  it holds that  $\mathfrak{T} \models_{\epsilon} i : C \sqsubseteq D$  if and only if  $\mathfrak{T}' \models_{\epsilon} i : C \sqsubseteq D$ .*

Given the reduction, it is now clear that the expressive power of the framework is not enhanced by addition of conjunctive bridge rules. We argue, however, that conjunctive bridge rules still are an interesting update since using them instead of normal onto-bridge rules guarantees intuitive behaviour of the semantics, as demonstrated in Examples 1 and 2 and formally established by Theorem 2.

<sup>1</sup> We are indebted to one of the anonymous referees for pointing this out.

## 7 Evaluation and Comparison

In [1,2,3] various intuitions on how the semantics of a distributed DL environment, such as DDL, should behave are presented. The original semantics of DDLs has been evaluated with respect to these desiderata throughout [1,2,3]. We proceed with evaluating the new framework with respect to these desiderata.

First of all, monotonicity is a desired property, that is, the requirement that bridge rules do not delete local subsumptions as postulated in [1,2].

**Theorem 5 (Monotonicity).** *In every distributed TBox  $\mathfrak{T}$  that also allows conjunctive bridge-rules it holds that  $\mathcal{T}_i \models A \sqsubseteq B \implies \mathfrak{T} \models_{\epsilon} i : A \sqsubseteq B$ .*

Another desired property is that there is no backflow of information against the direction of bridge rules. This property (we use the version of [3]) also holds in the presence of conjunctive bridge-rules.

**Theorem 6 (Directionality).** *Given a distributed TBox  $\mathfrak{T}$  that allows conjunctive bridge rules in its set of bridge rules  $\mathfrak{B}$ , if there is no directed path of bridge rules from  $\mathcal{T}_i$  to  $\mathcal{T}_j$  in  $\mathfrak{T}$ , then  $\mathfrak{T} \models_{\epsilon} j : C \sqsubseteq D$  if and only if  $\mathfrak{T}' \models_{\epsilon} j : C \sqsubseteq D$ , where  $\mathfrak{T}'$  is obtained by removing  $\mathcal{T}_i$  from  $\mathfrak{T}$  as well as removing all bridge-rules involving  $\mathcal{T}_i$  from  $\mathfrak{B}$ .*

Another interesting desideratum that has been postulated in [2] is that one should be only able to add new knowledge by combination of into- and onto-bridge rules.

**Desideratum 1 (Strong directionality)** *If either for all  $k \neq i$ ,  $\mathfrak{B}_{ki}$  contains no into-bridge rules or for all  $k \neq i$ ,  $\mathfrak{B}_{ki}$  contains no onto-bridge rules, then  $\mathfrak{T} \models_{\epsilon} i : A \sqsubseteq B$  implies  $\mathcal{T}_i \models A \sqsubseteq B$ .*

Unfortunately, this does not hold for DDLs, with or without conjunctive bridge rules. As a counterexample consider the distributed TBox of Example 1 and replace all bridge-rules therein by conjunctive ones. This setting counters the desideratum. Using the reduction of Theorem 4 one obtains an equivalent knowledge base with no conjunctive bridge rules that still counters the desideratum.

Yet another interesting desideratum for DDLs is that local inconsistency that occurs in some of the local TBoxes does not spread and pollute the whole system. In [3] a precise characterization of how inconsistent local TBoxes affect a DDL knowledge base is given. We confirm this property also in presence of conjunctive bridge rules.

**Theorem 7 (Local inconsistency).** *Given a distributed TBox  $\mathfrak{T}$  that also allows conjunctive bridge rules,  $\mathfrak{T} \models_{\epsilon} i : C \sqsubseteq D$  if and only if for any  $J \subseteq I$  not containing  $i$ ,  $\mathfrak{T}(\epsilon_J) \models_{\text{d}} i : C \sqsubseteq D$ , where  $\models_{\text{d}}$  is a kind of entailment that does not allow holes, and  $\mathfrak{T}(\epsilon_J)$  is obtained from  $\mathfrak{T}$  by removing each  $\mathcal{T}_j$ ,  $j \in J$ , and adding  $\{D \sqsubseteq \perp \mid j : C \overset{\exists}{\rightsquigarrow} i : D \in \mathfrak{B} \wedge j \in J\}$  to each  $\mathcal{T}_i$ ,  $i \in I \setminus J$ .*

Two desiderata of [1,2,3] show how subsumption is propagated along bridge rules. Since only one onto-bridge rule is involved here, it follows immediately that these desiderata also hold when a conjunctive onto-bridge rule is used.

**Theorem 8 (Simple subsumption propagation).** *If  $i : C \overset{\exists}{\rightsquigarrow} j : G \in \mathfrak{B}$  and  $i : D \overset{\exists}{\rightsquigarrow} j : H \in \mathfrak{B}$  then  $\mathfrak{T} \models_{\epsilon} i : C \sqsubseteq D \implies \mathfrak{T} \models_{\epsilon} j : G \sqsubseteq H$ .*

**Theorem 9 (Generalized subsumption propagation).** *If  $i : C \overset{\exists}{\rightsquigarrow} j : G \in \mathfrak{B}$  and  $i : D_k \overset{\exists}{\rightsquigarrow} j : H_k \in \mathfrak{B}$ , for  $1 \leq k \leq n$  then  $\mathfrak{T} \models_{\epsilon} i : C \sqsubseteq \bigsqcup_{k=1}^n D_k$  implies  $\mathfrak{T} \models_{\epsilon} j : G \sqsubseteq \bigsqcup_{k=1}^n H_k$ .*

So far we have evaluated the adjusted DDLs framework, with respect to the desiderata postulated for DDLs in [1,2,3]. We have showed that all the desiderata that are satisfied for the original framework also hold when conjunctive bridge rules are present. Moreover, we introduce a variant of the Generalized subsumption propagation desideratum, in which concept intersection is involved instead of concept union. We consider this a desired property and are pleased to report that it also holds for DDLs (with or without conjunctive bridge rules allowed).

**Theorem 10 (Subsumption propagation over concept intersection).** *If  $i : C \overset{\exists}{\rightsquigarrow} j : G \in \mathfrak{B}$  and  $i : D_k \overset{\exists}{\rightsquigarrow} j : H_k \in \mathfrak{B}$ , for  $1 \leq k \leq n$  then  $\mathfrak{T} \models_{\epsilon} i : C \sqsubseteq \prod_{k=1}^n D_k$  implies  $\mathfrak{T} \models_{\epsilon} j : G \sqsubseteq \prod_{k=1}^n H_k$ .*

## 8 Related Work

Besides of DDLs of Borgida, Serafini and Taminin [1,2,3], another major contribution to distributed and modular ontologies is the approach of Cuenca Grau et al. [8,5] where a combination of several ontologies using  $\mathcal{E}$ -connections [9] is proposed. In this framework, link relations – inter-ontology roles between local ontologies – are favored instead of bridge rules. While both are related [9,10], each maintains its own primary intuitions – in DDLs inter-ontology subsumption is modeled directly with bridge rules, while the preference of links in the latter framework has lead to such results as automated ontology decomposition [11].

An extension of DDLs called C-OWL has been introduced by Bouquet et al. in [6]. Several improvements were suggested, including a richer family of bridge rules, allowing bridging between roles, etc. Also, Ghidini and Serafini in [12,13] enrich DDLs with heterogenous mappings, that is mappings between concepts and roles.

## 9 Conclusion and Future Work

We have proposed an adjustment/extension of DDLs of [1,2,3] in order to address an issue noted in [5]. We have introduced so called conjunctive onto-bridge rules with modified semantics; there is no need for conjunctive into-bridge rules. Even

if the expressive power of the framework does not grow when conjunctive onto-bridge rules are added, using them instead of normal onto-bridge rules guarantees that the unintuitive behaviour of the semantics does not occur any more. All desired properties that hold for DDLs, as established in [1,2,3], also hold when conjunctive bridge rules are added. We have postulated one additional property which holds with and without conjunctive bridge rules as well. We have also provided a transformational semantics for conjunctive bridge rules and so, at least theoretically, a decision procedure, given the known results for DDLs [2,3].

Other interesting issues regarding distributed ontologies that we would like to address include evaluation of the adjusted DDL framework; and further investigation of reasoning algorithms and computational properties.

**Acknowledgement.** Extended version including proofs is available via author’s homepage (<http://ii.fmph.uniba.sk/~homola/>). We would like to thank to anonymous referees for numerous helpful comments. Support from Slovak Agency for Promotion of Research and Development projects, contract no. APVV-20-P04805 and APVV-99-PO5005; and project VEGA of Slovak Ministry of Education and Slovak Academy of Sciences no.1/0173/03 is acknowledged.

## References

1. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics* **1** (2003) 153–184
2. Serafini, L., Tamin, A.: Local tableaux for reasoning in distributed description logics. In: *Procs. of DL’04. CEUR-WS* (2004)
3. Serafini, L., Borgida, A., Tamin, A.: Aspects of distributed and modular ontology reasoning. In: *Procs. of IJCAI’05.* (2005) 570–575
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. *Scientific American* **284**(5) (2001) 34–43
5. Cuenca Grau, B., Parsia, B., Sirin, E.: Combining OWL ontologies using  $\mathcal{E}$ -connections. *Journal of Web Semantics* **4**(1) (2006) 40–59
6. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: C-OWL: Contextualizing ontologies. In: *Procs. of ISWC2003.* (2003) 164–179
7. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: *Procs. of LPAR’99. Number 1705 in LNAI, Springer* (1999) 161–180
8. Cuenca Grau, B., Parsia, B., Sirin, E.: Tableaux algorithms for  $\mathcal{E}$ -connections of description logics. Technical report, University of Maryland Institute for Advanced Computer Studies (2004)
9. Kutz, O., Lutz, C., Wolter, F., Zakharyashev, M.:  $\mathcal{E}$ -connections of abstract description systems. *Artificial Intelligence* **156**(1) (2004) 1–73
10. Serafini, L., Tamin, A.: Distributed reasoning services for multiple ontologies. Technical Report DIT-04-029, University of Trento (2004)
11. Cuenca Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Automatic partitioning of OWL ontologies using  $\mathcal{E}$ -connections. In: *Procs. of DL’05. CEUR-WS* (2005)
12. Ghidini, C., Serafini, L.: Reconciling concepts and relations in heterogeneous ontologies. In: *Procs. ESWC 2006. Volume 4011/2006 of LNCS., Springer* (2006)
13. Ghidini, C., Serafini, L.: Mapping properties of heterogeneous ontologies. In: *Procs. of WoMo-06. CEUR WS* (2006)

## Multimedia Interpretation as Abduction

S. Espinosa Peraldi, A. Kaya, S. Melzer, R. Möller, M. Wessel

Hamburg University of Technology, Germany

**Abstract.** In this work we present an approach to interpret information extracted from multimedia documents through Abox abduction, which we consider as a new type of non-standard retrieval inference service in Description Logics (DLs). We discuss how abduction can be adopted to interpret multimedia content through explanations. In particular, we present a framework to generate explanations, and introduce a preference measure for selecting ‘preferred’ explanations.<sup>1</sup>

### 1 Introduction

Automated extraction of information from different types of multimedia documents such as image, text, video, and audio becomes more and more relevant for intelligent retrieval systems. An intelligent retrieval system is a system with a knowledge base and capabilities that can be used to establish connections between a request and a set of data based on the high-level semantics of the data (which can also be documents). Typically, nowadays, automated semantics extraction from multimedia occurs by using low-level features and is often limited to the recognition of isolated items if even. Examples are single objects in an image, or single words (or maybe phrases) in a text. However, multimedia documents such as images usually present more than objects detectable in a bottom-up fashion. For instance, an image may illustrate an abstract concept such as an event. An event in a still image can hardly be perceived without additional high-level knowledge.

We see multimedia interpretation as abduction (reasoning from effects to causes) in that we reason from observations (effects) to explanations (causes). The aim of this work is to present a novel approach for multimedia interpretation through Abox abduction, which we consider as a new type of non-standard retrieval inference service in DLs. In particular, we focus on the use of DL-safe-like rules for finding explanations and introduce a preference measure for selecting ‘preferred’ explanations.

### 2 Related Work in Media Interpretation and Abduction

The idea of formalizing interpretation as abduction is investigated in [4] in the context of text interpretation. In [8], Shanahan presents a formal theory of robot

<sup>1</sup> This work is partially supported by the EU-funded projects BOEMIE (Bootstrapping Ontology Evolution with Multimedia Information Extraction, IST-FP6-027538) and TONES (Thinking ONtologiES, FET-FP6-7603).

perception as a form of abduction. In this work, low-level sensor data is transformed into a symbolic representation of the world in first-order logic and abduction is used to derive explanations. In the context of scene interpretation, recently, in [7] the use of DLs for scene interpretation processes is described.

In this paper we present a novel approach based on the combination of the works in [4, 8] and [7], and indicate how formal representation and reasoning techniques can be used for interpretation of information extracted from multimedia documents. The approach used description logics and rules, with abduction implemented with backward-chaining applied to the rules. In contrast to approaches such as [5], which use abduction in the context of rules in logic programming, we use description-logic reasoning for proving subgoals of (non-recursive) rules. Other approaches for abduction in description logics (e.g., [1]) have dealt with concept abduction only. In [3] among other abductive reasoning tasks in DLs also Abox abduction is discussed. A solution to the Abox abduction problem is formally presented, but it is not shown how to derive solutions.

Abduction is investigated for supporting information retrieval based on high-level descriptions on media content. The approach builds on [6] and, in contrast to later related work such as [2], the approach is integrated into a mainstream description logic system and is based on high-level descriptions of media content.

### 3 Retrieval Inference Services

Before introducing abduction as a new inference service, we start with an overview of retrieval inference services that are supported by state-of-the-art DL reasoners.

The *retrieval* inference problem w.r.t. a Tbox  $\mathcal{T}$  is to find all individuals mentioned in an Abox  $\mathcal{A}$  that are instances of a certain concept  $C$ :  $\{x \text{ mentioned in } \mathcal{A} \mid (\mathcal{T}, \mathcal{A}) \models x : C\}$ . In addition to the basic retrieval inference service, expressive query languages are required in practical applications. Well-established is the class of conjunctive queries. A *conjunctive query* consists of a *head* and a *body*. The head lists variables for which the user would like to compute bindings. The body consists of query atoms (see below) in which all variables from the head must be mentioned. If the body contains additional variables, they are seen as existentially quantified. A query answer is a set of tuples representing bindings for variables mentioned in the head. A query is a structure of the form  $\{(X_1, \dots, X_n) \mid atom_1, \dots, atom_m\}$ .

Query atoms can be *concept* query atoms ( $C(X)$ ), *role* query atoms ( $R(X, Y)$ ), *same-as* query atoms ( $X = Y$ ) as well as so-called *concrete domain* query atoms. The latter are introduced to provide support for querying the concrete domain part of a knowledge base and will not be covered in detail here. Complex queries are built from query atoms using boolean constructs for conjunction (indicated with comma) or union ( $\vee$ ).

In *standard* conjunctive queries, variables (in the head and in query atoms in the body) are bound to (possibly anonymous) domain objects. A system supporting (unions of) standard conjunctive queries is QuOnto. In so-called *grounded*

conjunctive queries,  $C(X)$ ,  $R(X, Y)$  or  $X = Y$  are true if, given some bindings  $\alpha$  for mapping from variables to *individuals mentioned in the Abox*  $\mathcal{A}$ , it holds that  $(\mathcal{T}, \mathcal{A}) \models \alpha(X) : C$ ,  $(\mathcal{T}, \mathcal{A}) \models (\alpha(X), \alpha(Y)) : R$ , or  $(\mathcal{T}, \mathcal{A}) \models \alpha(X) = \alpha(Y)$ , respectively. In grounded conjunctive queries the standard semantics can be obtained for so-called tree-shaped queries by using corresponding existential restrictions in query atoms. Due to space restrictions, we cannot discuss the details here. In the following, we consider only grounded conjunctive queries, which are supported by KAON2, Pellet, and RacerPro.

In practical applications it is advantageous to name subqueries for later reuse, and practical systems, such as for instance RacerPro, support this for grounded conjunctive queries with non-recursive rules of the following form

$$P(X_1, \dots, X_{n_1}) \leftarrow A_1(Y_1), \dots, A_l(Y_l), R_1(Z_1, Z_2), \dots, R_h(Z_{2h-1}, Z_{2h}). \quad (1)$$

The predicate term to the left of  $\leftarrow$  is called the head and the rest is called the body (a set of atoms), which, informally speaking, is seen as a conjunction of predicate terms. All variables in the head have to occur in the body, and rules have to be non-recursive (with the obvious definition of non-recursivity). Since rules have to be non-recursive, the replacement of query atoms matching a rule head is possible (unfolding, with the obvious definition of matching). The rule body is inserted (with well-known variable substitutions and variable renamings). If there are multiple rules (definitions) for the same predicate  $P$ , corresponding disjunctions are generated. The unfolding process starts with the set of atoms of a query. Thus, we start with a set of atom sets.

$$\{\{atom_1, atom_2, \dots, atom_k\}\}$$

Each element of the outer set represents a disjunct. Now, wlog we assume that there are  $n$  rules matching  $atom_2$ . Then, the set  $\{atom_1, atom_2, \dots, atom_k\}$  is eliminated and replaced with the sequence of sets  $\{atom_1\} \cup \text{replace\_vars}(\text{body}(rule_1), \text{head}(rule_1), atom_2) \cup \{\dots, atom_k\}$ ,  $\dots$ ,  $\{atom_1\} \cup \text{replace\_vars}(\text{body}(rule_n), \text{head}(rule_n), atom_2) \cup \{\dots, atom_k\}$ . The unfolding process proceeds until no replacement is possible any more (no rules match). The unfold operator is used in the abduction process, which is described in the next section.

#### 4 Abduction as a Non-Standard Inference Service

In this paper, we argue that abduction can be considered as a new type of non-standard retrieval inference service. In this view, observations (or part of them) are utilized to constitute queries that have to be answered. Contrary to existing retrieval inference services, answers to a given query cannot be found by simply exploiting the knowledge base. In fact, the abductive retrieval inference service has the task of acquiring what should be added to the knowledge base in order to positively answer a query.

More formally, for a given set of Abox assertions  $\Gamma$  (in form of a query) and a knowledge base  $\Sigma = (\mathcal{T}, \mathcal{A})$ , the abductive retrieval inference service aims to

derive all sets of Abox assertions  $\Delta$  (explanations) such that  $\Sigma \cup \Delta \models \Gamma$  and the following conditions are satisfied:

- $\Sigma \cup \Delta$  is satisfiable, and
- $\Delta$  is a minimal explanation for  $\Gamma$ , i.e., there exists no other explanation  $\Delta'$  in the solution set that is not equivalent to  $\Delta$  and it holds that  $\Sigma \cup \Delta' \models \Delta$ .

In addition to minimality (simplicity), in [4] another dimension called concision is mentioned. An explanation should explain as many elements of  $\Gamma$  as possible. Both measures are contradictory.

In the next section, we will focus on the use of abductive retrieval inference services for multimedia interpretation and address two important issues, namely finding explanations that meet the conditions listed above and selecting ‘preferred’ ones.

## 5 Interpretation of Multimedia Documents

For intelligent retrieval of multimedia documents such as images, videos, audio, and texts, information extracted by media analysis techniques has to be enriched by applying high-level interpretation techniques. The interpretation of multimedia content can be defined as the recognition of abstract knowledge, in terms of concepts and relations, which are not directly extractable by low-level analysis processes, but rather require additional high-level knowledge. Furthermore, such abstract concepts are represented in the background knowledge as aggregate concepts with constraints among its parts.

In this section, we start by specifying the requirements for the abduction approach by defining its input and output. Then, we proceed with describing the framework for generating explanations, and finally introduce a scenario with a particular example involving for image interpretation where various explanations are generated and the usefulness of a preference score is demonstrated.

### 5.1 Requirements for Abduction

The abduction approach requires as input a knowledge base  $\Sigma$  consisting of a Tbox  $\mathcal{T}$  and an Abox  $\mathcal{A}$ . We assume that the information extracted from a multimedia document through low-level analysis (e.g., image analysis) is formally encoded as a set of Abox assertions ( $\Gamma$ ). For example, in the context of images for every object recognized in an image, a corresponding concept assertion is found in  $\Gamma$ . Usually, the relations that can be extracted from an image are spatial relations holding among the objects in the image. These relations are also represented as role assertions in  $\Gamma$ . In order to construct a high-level interpretation of the content in  $\Gamma$ , the abduction process will extend the Abox with new concept and role assertions describing the content of the multimedia document at a higher level.

The output of the abduction process is formally defined as a set of assertions  $\Delta$  such that  $\Sigma \cup \Delta \models \Gamma$ , where  $\Sigma = (\mathcal{T}, \mathcal{A})$  is the knowledge base (usually the

Abox  $A$  is assumed to be empty),  $\Gamma$  is a given set of low-level assertions, and  $\Delta$  is an explanation, which should be computed. The solution  $\Delta$  must satisfy certain side conditions (see Section 4). To compute the explanation  $\Delta$  in our context we modify this equation into

$$\Sigma \cup \Gamma_1 \cup \Delta \models \Gamma_2, \quad (2)$$

where the assertions in  $\Gamma$  will be split into bona fide assertions ( $\Gamma_1$ ) and assertions requiring fiat ( $\Gamma_2$ ).<sup>2</sup> Bona fide assertions are assumed to be true by default, whereas fiat assertions are aimed to be explained. The abduction process tries to find explanations ( $\Delta$ ) such that  $\Gamma_2$  is entailed. This entailment decision is implemented as (boolean) query answering. The output  $\Delta$  of the abduction process is represented as an Abox. Multiple solutions are possible.

## 5.2 The Abduction Framework

The abduction framework exploits the non-recursive rules of  $\Sigma$  to answer a given query in a backward-chaining way (see Framework 1). The function `compute_explanations` gets  $\Sigma, \Gamma_1$ , and  $\Gamma_2$  as input. We assume a function `transform_into_query` that is applied to a set of Abox assertions  $\Gamma_2$  and returns a set of corresponding query atoms. The definition is obvious and left out for brevity. Since the rules in  $\Sigma$  are non-recursive, the unfolding step (see Line 2 in Framework 1) in which each atom in the transformed  $\Gamma_2$  is replaced by the body of a corresponding rule is well-defined. The function `unfold` returns a set of atom sets (each representing a disjunct introduced by multiple matching rules, see above).

The function `explain` computes an explanation  $\Delta$  for each  $\gamma \in \Gamma'_2$ . The function `vars` (or `inds`) returns the set of all vars (or inds) mentioned in the argument structures. For each variable in  $\gamma$  a new individual is generated (see the set `new_inds` in Line 7). Besides old individuals, these new individuals are used in a non-deterministic variable substitution. The variable substitution  $\sigma_{\gamma, new\_inds}$  (line 8) is inductively extended as follows:

- $\sigma_{\gamma, new\_inds}(\{a_1, \dots, a_n\}) =_{def} \{\sigma_{\gamma, new\_inds}(a_1), \dots, \sigma_{\gamma, new\_inds}(a_n)\}$
- $\sigma_{\gamma, new\_inds}(C(x)) =_{def} C(\sigma_{\gamma, new\_inds}(x))$
- $\sigma_{\gamma, new\_inds}(R(x, y)) =_{def} R(\sigma_{\gamma, new\_inds}(x), \sigma_{\gamma, new\_inds}(y))$
- $\sigma_{\gamma, new\_inds}(x) =_{def} x$  if  $x$  is an individual

The function `transform` maps  $C(i)$  into  $i : C$  and  $R(i, j)$  into  $(i, j) : R$ , respectively. All satisfiable explanations  $\Delta$  derived by `explain` are added to the set of explanations  $\Delta_s$ . The function `compute-preferred-explanations` transforms the  $\Delta_s$  into a poset according to a preference measure and returns the maxima as a set of Aboxes. The preference score of a  $\Delta$  used for the poset order relation is:  $S_{pref}(\Delta) := S_i(\Delta) - S_h(\Delta)$  where  $S_i$  and  $S_h$  are defined as follows.

<sup>2</sup> With the obvious semantics we slightly abuse notation and allow a tuple of sets of assertions  $\Sigma$  to be unioned with a set of assertions  $\Gamma_1 \cup \Delta$ .

- $S_i(\Delta) := |\{i | i \in inds(\Delta) \text{ and } i \in inds(\Sigma \cup \Gamma_1)\}|$
- $S_h(\Delta) := |\{i | i \in inds(\Delta) \text{ and } i \in new\_inds\}|$

---

**Algorithm 1** The Abduction Framework

---

- 1: **function** compute\_explanations( $\Sigma, \Gamma_1, \Gamma_2, S$ ) : set of Aboxes
  - 2:  $\Gamma'_2 := \text{unfold}(\text{transform\_into\_query}(\Gamma_2), \Sigma) // \Gamma'_2 = \{\{atom_1, \dots, atom_m\}, \dots\}$
  - 3:  $\Delta s := \{\Delta \mid \exists \gamma \in \Gamma'_2. (\Delta = \text{explain}(\Sigma, \Gamma_1, \gamma), \Sigma \cup \Gamma_1 \cup \Delta \not\models \perp)\}$
  - 4: **return** compute\_preferred\_explanations( $\Sigma, \Gamma_1, \Delta s, S$ )
- 
- 5: **function** explain( $\Sigma, \Gamma_1, \gamma$ ) : Abox
  - 6:  $n := |\text{vars}(\gamma)|$
  - 7:  $new\_inds := \{new\_ind_i \mid i \in \{1 \dots n\}\}$ , where  $new\_inds \cap (inds(\Sigma) \cup inds(\Gamma_1)) = \emptyset$
  - 8:  $\Delta := \{\text{transform}(a) \mid \exists \sigma_{\gamma, new\_inds} : \text{vars}(\gamma) \mapsto (inds(\Sigma) \cup inds(\Gamma_1) \cup new\_inds).$
  - 9:  $(a \in \sigma_{\gamma, new\_inds}(\gamma), (\Sigma \cup \Gamma_1) \not\models a)\}$
  - 10: **return**  $\Delta$
- 
- 11: **function** compute\_preferred\_explanations( $\Sigma, \Gamma_1, \Delta s, S$ ) : set of Aboxes
  - 12: **return** maxima(poset( $\Delta s, \lambda(x, y) \bullet S(x) < S(y)$ ))
- 

Depending on the preference function given as the actual parameter for the argument  $S$ , the procedure `compute_explanations` can be considered as an approximation w.r.t. the minimality and consilience condition defined in Section 4. It adds to the explanation those query atoms that cannot be proven to hold.

For the abduction framework, only the rules are considered. The GCIs should be used for abduction as well, however. We might accomplish this by approximating the Tbox with the DLP fragment and, thereby, see the Tbox axioms from a rules perspective in order to better reflect the Tbox in the abduction process. The procedure does not add irrelevant atoms (spurious elements of an explanation), in case the rules are well-engineered and do not contain irrelevant ways to derive assertions. The procedure could be slightly modified to check for those redundancies.

### 5.3 An Example for Image Interpretation as Abduction

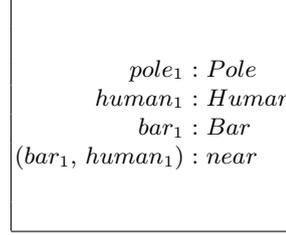
For the image shown in Figure 1, we suppose the Abox in Figure 2 is provided by low-level image analysis. Furthermore, a sample Tbox of the athletics domain and a small set of rules are assumed to be provided as background knowledge  $\Sigma$  (see Figure 3).

In order to find a ‘good’ high-level interpretation of this image, we divide the Abox  $\Gamma$  into  $\Gamma_1$  and  $\Gamma_2$  following Equation 2. In this example  $\Gamma_1$  contains  $\{pole_1 : Pole, human_1 : Human, bar_1 : Bar\}$  and  $\Gamma_2$  contains  $\{(bar_1, human_1) : near\}$ . Consequently, the abductive retrieval inference service computes the following boolean query in line 2:  $Q_1 := \{() \mid near(bar_1, human_1)\}$ . In this paper we do not elaborate on the strategy to determine which  $\Gamma_2$  to actually choose.

Obviously, both rules in  $\Sigma$  match with the ‘near’ atom in query  $Q_1$ . Therefore, the abduction framework first generates explanations by non-deterministically



**Fig. 1.** A pole vault event.



**Fig. 2.** An Abox  $\Gamma$  representing the results of low-level image analysis.

$$\begin{aligned}
 Jumper &\sqsubseteq Human \\
 Pole &\sqsubseteq Sports\_Equipment \\
 Bar &\sqsubseteq Sports\_Equipment \\
 Pole \sqcap Bar &\sqsubseteq \perp \\
 Pole \sqcap Jumper &\sqsubseteq \perp \\
 Jumper \sqcap Bar &\sqsubseteq \perp \\
 Jumping\_Event &\sqsubseteq \exists_{<1} hasParticipant.Jumper \\
 Pole\_Vault &\sqsubseteq Jumping\_Event \sqcap \exists hasPart.Pole \sqcap \exists hasPart.Bar \\
 High\_Jump &\sqsubseteq Jumping\_Event \sqcap \exists hasPart.Bar \\
 near(Y, Z) &\leftarrow Pole\_Vault(X), hasPart(X, Y), Bar(Y), \\
 &\quad hasPart(X, W), Pole(W), hasParticipant(X, Z), Jumper(Z) \\
 near(Y, Z) &\leftarrow High\_Jump(X), hasPart(X, Y), Bar(Y), \\
 &\quad hasParticipant(X, Z), Jumper(Z)
 \end{aligned}$$

**Fig. 3.** A tiny example  $\Sigma$  consisting of a Tbox and DL-safe rules.

substituting variables in the query body with different instances from  $\Gamma_1$  or with new individuals. Some intermediate  $\Delta$  results turn out to be unsatisfiable (e.g., if the bar is made into a pole by the variable substitution process). However, several explanations still remain as possible interpretations of the image. The preference score is used to identify the ‘preferred’ explanations. For example, considering the following explanations of the image

- $\Delta_1 = \{new\_ind_1 : Pole\_Vault, (new\_ind_1, bar_1) : hasPart, (new\_ind_1, new\_ind_2) : hasPart, new\_ind_2 : Pole, (new\_ind_1, human_1) : hasParticipant, human_1 : Jumper\}$
- $\Delta_2 = \{new\_ind_1 : Pole\_Vault, (new\_ind_1, bar_1) : hasPart, (new\_ind_1, pole_1) : hasPart, (new\_ind_1, human_1) : hasParticipant, human_1 : Jumper\}$
- $\Delta_3 = \{new\_ind_1 : High\_Jump, (new\_ind_1, bar_1) : hasPart, (new\_ind_1, human_1) : hasParticipant, human_1 : Jumper\}$

the preference measure of  $\Delta_1$  is calculated as follows:  $\Delta_1$  incorporates the individuals  $human_1$  and  $bar_1$  from  $\Gamma_1$  and therefore  $S_i(\Delta_1)=2$ . Furthermore, it hypothesizes two new individuals, namely  $new\_ind_1$  and  $new\_ind_2$ , such that

$S_h(\Delta_1)=2$ . The preference score of  $\Delta_1$  is  $S(\Delta_1) = S_i(\Delta_1) - S_h(\Delta_1) = 0$ . Similarly, the preference scores of the second and third explanations are  $S(\Delta_2)=2$  and  $S(\Delta_3)=1$ . After transforming the  $\Delta$ s into a poset, the algorithm computes the maxima. In our case, the resulting set of Aboxes contains only one element,  $\Delta_2$ , which represents the ‘preferred’ explanation. Indeed, the result is plausible, since this image should better be interpreted as showing a pole vault and not a high jump, due to the fact that low-level image analysis could detect a pole, which should not be ignored as in the high-jump explanation.

## 6 Summary

In this paper we presented a novel approach to interpret multimedia data using abduction with description logics that makes use of a new type of non-standard retrieval service in DLs. We showed that results from low-level media analysis can be enriched with high-level descriptions using our Abox abduction approach. In this approach, backward-chained DL-safe-like rules are exploited for generating explanations. For each explanation, a preference score is calculated in order to implement the selection of ‘preferred’ explanations. Details of the approach have been discussed with a particular example for image interpretation. An implementation of the abduction process described in this paper is available as a non-standard retrieval service integrated in RacerPro.

## References

1. S. Colucci, T. Di Noia, E. Di Sciascio, M. Mongiello, and F. M. Donini. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In *ICEC '04: Proceedings of the 6th international conference on Electronic commerce*, pages 41–50, 2004.
2. E. Di Sciascio, F.M. Donini, and M. Mongiello. A description logic for image retrieval. In *Proceedings of the 6th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, number 1792 in Lecture Notes in Computer Science, pages 13–24. Springer, 1999.
3. C. Elsenbroich, O. Kutz, and U. Sattler. A Case for Abductive Reasoning over Ontologies. In *OWL: Experiences and Directions*, 2006.
4. J. R. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.
5. A. Kakas and M. Denecker. Abduction in logic programming. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond. Part I*, number 2407 in LNAI, pages 402–436. Springer, 2002.
6. R. Möller, V. Haarslev, and B. Neumann. Semantics-based information retrieval. In *Proc. IT&KNOWS-98: International Conference on Information Technology and Knowledge Systems, 31. August- 4. September, Vienna, Budapest*, pages 49–6, 1998.
7. B. Neumann and R. Möller. On Scene Interpretation with Description Logics. In H.I. Christensen and H.-H. Nagel, editors, *Cognitive Vision Systems: Sampling the Spectrum of Approaches*, number 3948 in LNCS, pages 247–278. Springer, 2006.
8. Murray Shanahan. Perception as Abduction: Turning Sensor Data Into Meaningful Representation. *Cognitive Science*, 29(1):103–134, 2005.

## Prospects for and issues with mapping the Object-Role Modeling language into $\mathcal{DLR}_{ifd}$

C. Maria Keet

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy  
keet@inf.unibz.it

**Abstract.** Object-Role modellers miss the advantages of automated reasoning over their ORM conceptual models, which could be addressed by DL reasoners. DLs are not considered user-friendly and could benefit from the easy to use ORM diagrammatic and verbalization interfaces and modelling methodologies. Relating the two would greatly expand the scope for automated reasoning with additional scenarios to improve quality of software systems. Given that none of the extant DL languages are as expressive as ORM or its successor ORM2, the ‘best-fit’  $\mathcal{DLR}_{ifd}$  was chosen to map the formal conceptual modelling language ORM2. For the non-mappable constraints, pointers to other DL languages are provided, which could serve as impetus for research into DL language extensions or interoperability between existing DL languages.

### 1 Introduction

Description Logic (DL) languages have been shown useful for reasoning both over conceptual models like ER and UML [2, 4, 12] and ontology languages such as OWL-DL, OWL-Lite [27], its proposed successor OWL 1.1 [26] that is based on the DL language *SROIQ* [22], and *DL-Lite* [7]. In particular, we are interested in the notion of using DLs as unifying paradigm for conceptual modelling to enable automated reasoning over conceptual data models, which, be it due to legacy, preference, or applicability, are made with different conceptual modelling languages. A tool such as iCOM [15] already supports automated reasoning over UML or EER diagrams, which may have cross-conceptual model assertions. What is lacking, however, is a mapping from Object-Role Modeling (ORM) into a DL. One may wonder: why yet another mapping? There are three main reasons for this. First, ORM is a so-called “true” conceptual modelling language in the sense that it is independent of the application scenario and it has been mapped into both UML class diagrams and ER. That is, ORM and its successor ORM2 can be used in the conceptual analysis stage for database development, application software development, requirements engineering, business rules, and other areas, *e.g.*, [3, 5, 14, 17, 24]. Thus, if there is an ORM-to-DL mapping, the possible applications for automated reasoning services can be greatly expanded. Second, an important aspect of ORMing is to have great consideration for the user and therefore ORM tools such as CaseTalk and NORMA are very user-friendly, so that even domain experts unfamiliar with formalisms can start modelling after

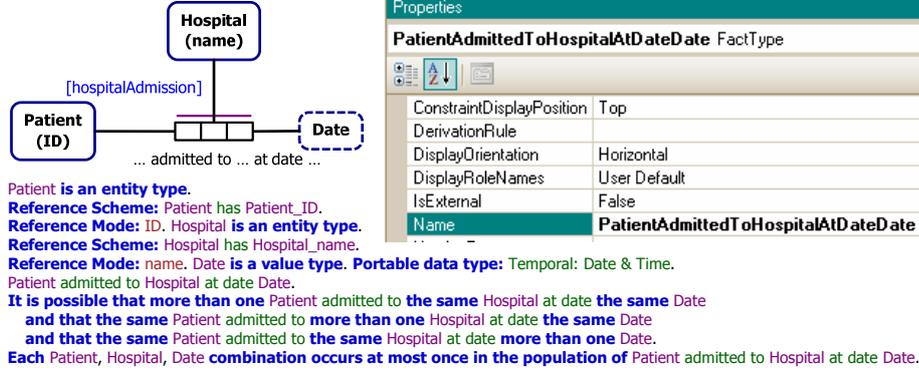
half an hour training. ORM tools have both diagrammatic and textual interfaces (the latter through so-called verbalizations, which are pseudo-natural language renderings of the axioms), thereby accommodating different user preferences toward modelling. Third, ORM is more expressive than either UML or EER and is more expressive than the extant DLs as well. Most ORM constraints are supported in one DL language or another, but none supports all ORM constraints. The proposed ORM-to- $\mathcal{DLR}_{ifd}$  mapping may provide some élan to examine DL language extensions not only based on interest and particular user requests from domain-modelling scenarios, but toward those (combinations of) extensions that are already known to be useful, or to find an implementable solution where for different (sections of) conceptual models, different languages can be used within one application interface.

The remainder of this paper is organised as follows. Subsections 1.1 and 1.2 contain brief introductions to ORM and  $\mathcal{DLR}$ , respectively. The main part is devoted to the assessment of the mapping (Section 2). Finally, some reflections and conclusions are included in Section 3.

### 1.1 Brief introduction to Object-Role Modeling (ORM)

The basic building blocks of the ORM language are object types, value types—at the conceptual level no subjective distinction has to be made between classes and attributes—relations, roles, and a plethora of constraints. A role is that what the object or value type ‘plays’ in the relation. ORM supports  $n$ -ary relations, where  $n$  is a finite integer  $\geq 1$ . An example of a fact type is shown in Fig.1, which was made with the NORMA CASE tool [25]: the diagrammatic representation of the relation –rectangle divided into three roles, one for each participating object or value type– in the ORM model has 1) the name of the relation, which is displayed in the properties box of the relation and is generated automatically by the software (called “PatientAdmittedToHospitalAtDateDate” in the example), 2) role names, such as “[hospitalAdmission]” for the the role that object type Patient plays, and 3) a label attached to the relation, “... admitted to ... at date ...”, which is used for the verbalization. ORM models can be mapped into, among others, ER and UML diagrams, IDEFX logical models, SQL table definitions, C, Visual Basic, and XML. More information on these mappings can be found in *e.g.* [17]. The ORM basics can be summarised as follows: an  $n$ -ary predicate (relation)  $R$ , with  $n \geq 1$ , is composed of  $r_1, \dots, r_n$  roles where each role has one object type, denoted with  $C_1, \dots, C_n$ , associated with it. Roles and predicates are globally unique (although the ‘surface labeling’ for the modeler may suggest otherwise).

Halpin’s first order logic formalization [16] was taken as basis for the mapping into  $\mathcal{DLR}_{ifd}$ , because it was the first formal characterisation of ORM and is relatively comprehensive in its treatment of constructors; other formalizations of ORM [13, 20, 21] do not differ significantly from Halpin’s version. [20, 21] make clearer distinctions between roles and predicates and the relation between them and the naming and labeling of ORM elements, but they cover fewer constraints.



**Fig. 1.** Top left: an ORM2 model with two object types, a value type, a ternary relation, label for the reading, and name of the first role in “[ ]”; top-right: properties box of the fact type, displaying the name of the relation; bottom-half: verbalization of the fact type, its object and value types, and uniqueness constraint (line above the rectangle).

## 1.2 DLs for conceptual modelling languages

DL languages are decidable fragments of first order logic and are used for logic-based knowledge representation. Basic ingredients of all DL languages are *concepts* and *roles* (an  $n$ -ary predicate where  $n \geq 2$ ). In addition, a DL has several constructors, giving greater or lesser expressivity and efficiency of automated reasoning over a logical theory. The Terminological Box (TBox) contains axioms at the concept-level and the ABox contains assertions about instances. A TBox corresponds to a formal conceptual data model or can be used to represent an ontology. More information and its usage can be found in [2].

For conceptual modelling, we introduce  $\mathcal{DLR}$  first [8], and subsequently the “*ifd*” extension [4, 9]. Take atomic relations ( $\mathbf{P}$ ) and atomic concepts  $A$  as the basic elements of  $\mathcal{DLR}$ . We then can construct arbitrary relations with arity  $\geq 2$  and arbitrary concepts according to the following syntax:

$$\begin{aligned} \mathbf{R} &\longrightarrow \top_n \mid \mathbf{P} \mid (\$i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2 \\ C &\longrightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid \leq k[\$i]\mathbf{R} \end{aligned}$$

$i$  denotes a component of a relation; if components are not named, then integer numbers between 1 and  $n_{max}$  are used, where  $n$  is the arity of the relation.  $k$  is a nonnegative integer for multiplicity (cardinality). Only relations of the same arity can be combined to form expressions of type  $\mathbf{R}_1 \sqcap \mathbf{R}_2$ , and  $i \leq n$ , *i.e.*, the concepts and relations must be well-typed. The semantics of  $\mathcal{DLR}$  is specified through the usual notion of interpretation, where  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , and the interpretation function  $\cdot^{\mathcal{I}}$  assigns to each concept  $C$  a subset  $C^{\mathcal{I}}$  of  $\Delta^{\mathcal{I}}$  and to each  $n$ -ary  $\mathbf{R}$  a subset  $\mathbf{R}^{\mathcal{I}}$  of  $(\Delta^{\mathcal{I}})^n$ , s.t. the conditions are satisfied following Table 1.  $\top_1$  denotes the interpretation domain,  $\top_n$  for  $n \geq 1$  denotes a subset of the  $n$ -cartesian product of the domain, which covers all introduced  $n$ -ary relations; hence “ $\neg$ ” on relations means difference rather than the complement. The  $(\$i/n : C)$  denotes all tuples in  $\top_n$  that have an instance of  $C$  as their

$\top_n^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\mathbf{P}^{\mathcal{I}} \subseteq \top_n^{\mathcal{I}}$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$(\neg \mathbf{R})^{\mathcal{I}} = \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}}$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
$(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} = \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}}$	$(\$/i/n : C)^{\mathcal{I}} = \{(d_1, \dots, d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}$
$\top_1^{\mathcal{I}} = \Delta^{\mathcal{I}}$	$(\exists \$/i \mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists (d_1, \dots, d_n) \in \mathbf{R}^{\mathcal{I}}. d_i = d\}$
	$(\leq k \$/i \mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \{(d_1, \dots, d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\} \leq k\}$

**Table 1.** Semantic rules for  $\mathcal{DLR}_{ifd}$ .

$i$ -th component.  $\mathcal{DLR}$  is a proper generalization of  $\mathcal{ALCQI}$ , where the usual DL constructs can be re-expressed in  $\mathcal{DLR}$  as:  $\exists P.C$  as  $\exists \[/1](P \sqcap (\$/2 : C))$ ,  $\exists P^{\neg}.C$  as  $\exists \[/2](P \sqcap (\$/1/2 : C))$  and so forth (see [8] for details). The following abbreviations can be used:  $C_1 \sqcup C_2$  for  $\neg(\neg C_1 \sqcap \neg C_2)$ ,  $C_1 \Rightarrow C_2$  for  $\neg C_1 \sqcup C_2$ ,  $(\geq k[i]R)$  for  $\neg(\leq k-1[i]R)$ ,  $\exists[i]R$  for  $(\geq 1[i]R)$ ,  $\forall[i]R$  for  $\neg\exists[i]\neg R$ ,  $R_1 \sqcup R_2$  for  $\neg(\neg R_1 \sqcap \neg R_2)$ , and  $(i : C)$  for  $(i/n : C)$  when  $n$  is clear from the context.

$\mathcal{DLR}_{ifd}$  supports identification assertions on a concept  $C$ , which has the form  $(\mathbf{id} C[i_1]R_1, \dots, [i_h]R_h)$ , where each  $R_j$  is a relation and each  $i_j$  denotes one component of  $R_j$ . Then, if  $a$  is an instance of  $C$  that is the  $i_j$ -th component of a tuple  $t_j$  of  $R_j$ , for  $j \in \{1, \dots, h\}$ , and  $b$  is an instance of  $C$  that is the  $i_j$ -th component of a tuple  $s_j$  of  $R_j$ , for  $j \in \{1, \dots, h\}$ , and for each  $j$ ,  $t_j$  agrees with  $s_j$  in all components different from  $i_j$ , then  $a$  and  $b$  are the same object.  $\mathcal{DLR}_{ifd}$  supports functional dependency assertions on a relation  $R$  for operations, which has the form  $(\mathbf{fd} R i_1, \dots, i_h \rightarrow j)$ , where  $h \geq 2$ , and  $i_1, \dots, i_h, j$  denote components of  $R$ .

**Other relevant DL languages** There are three other  $\mathcal{DLR}$  flavours.  $\mathcal{DLR}_{\mu}$  supports fixpoint constructs for recursive structures over single-inheritance trees of a role [10] and thereby can represent acyclicity, transitivity, asymmetry, and (ir)reflexivity.  $\mathcal{DLR}_{reg}$  adds support for regular expressions over roles (including the role composition operator and reflexive transitive closure) [11], and  $\mathcal{DLR}_{US}$  for temporal EER [1]. It has not been investigated if combining  $\mathcal{DLR}_{ifd}$ ,  $\mathcal{DLR}_{reg}$ , and  $\mathcal{DLR}_{\mu}$  remains within EXPTIME or leads to undecidability. In the other direction toward expressive DL-based ontology languages, OWL and draft OWL 1.1 [26] are based on  $\mathcal{SHOIN}$  (for OWL-DL),  $\mathcal{SHIF}$  (OWL-Lite), and  $\mathcal{SROIQ}$ , respectively.  $\mathcal{SROIQ}$  supports local (ir)reflexivity, (a)symmetry, and transitive roles [22], but does not have constructors for acyclic roles, datatypes,  $\mathbf{id}$ , and has no ‘access’ to elements of a DL-role.

## 2 Mapping issues

We now proceed to the mapping, which considers all components and constraints of ORM2, except deontic constraints (compared to ORM in [16], ORM2 also supports exclusive total covering of subtypes, role values, and deontic constraints). As basis, we used the ORM formalisation in first order logic by [16]. Graphical notation of ORM constrains and more explanation is deferred to [23] due to

space limitations; *e.g.*, the ternary relation in Fig.1 is *reified* in  $\mathcal{DLR}_{ifd}$  as:  $\text{PatientAdmittedToHospitalAtDateDate} \sqsubseteq \exists[1]r_1 \sqcap (\leq 1[1]r_1) \sqcap \forall[1](r_1 \Rightarrow (2 : \text{Patient})) \sqcap \exists[1]r_2 \sqcap (\leq 1[1]r_2) \sqcap \forall[1](r_2 \Rightarrow (2 : \text{Hospital})) \sqcap \exists[1]r_3 \sqcap (\leq 1[1]r_3) \sqcap \forall[1](r_3 \Rightarrow (2 : \text{Date}))$  and the identification of **Hospital** either as (**id Hospital [1]Hospital-HasHospital\_name**) or through a 1:1 relation (abbreviated as **HHH\_n**)  $\text{HHH}_n \sqsubseteq (1 : \text{Hospital}) \sqcap (2 : \text{Hospital\_name})$ ,  $\text{Hospital} \sqsubseteq (\leq 1 [1]\text{HHH}_n)$ , and  $\text{Hospital\_name} \sqsubseteq (\leq 1 [2]\text{HHH}_n)$  with mandatory  $\text{Hospital} \sqsubseteq \exists [1] \text{HHH}_n$ . The interesting problematic constraints are addressed in this section. The main problems concern ORM ring constraints, which are DL-role properties (Fig.2), and constraints with patterns of the type “constraint x over  $k$  ORM-roles” over an  $n$ -ary relation where  $k < n$ .

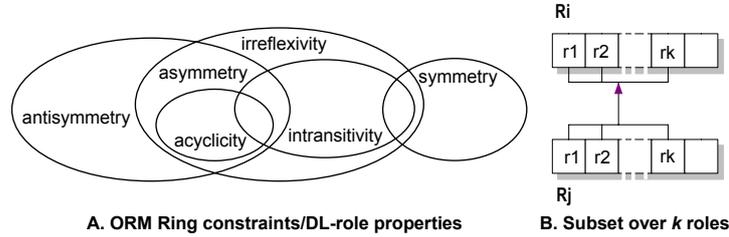


Fig. 2. A: ring constraints (after [17]); B: example of constraint over  $k$  ORM-roles.

Intransitivity over an ORM ring constraint is, obviously, supported in  $\mathcal{DLR}_{ifd}$ , but not transitivity, for which we need either  $\mathcal{DLR}_\mu$  or  $\mathcal{DLR}_{reg}$ . Antisymmetry in ORM is *reflexive antisymmetry* ( $\forall x, y (R(x, y) \wedge R(y, x) \rightarrow x = y)$ ), which no DL language supports. (Observe from Fig.2 that  $\mathcal{SROIQ}$ 's irreflexive antisymmetry is asymmetry.) The irreflexive ring constraint on a binary relation ( $\forall x \neg (R(x, x))$ ) is an open issue for  $\mathcal{DLR}_{ifd}$ , but already possible with  $\mathcal{DLR}_\mu$  thanks to least/greatest fixpoint construct and in  $\mathcal{SROIQ}$  with **Self**. The symmetric ( $\forall x, y (R(x, y) \rightarrow R(y, x))$ ) and asymmetric ( $\forall x, y (R(x, y) \rightarrow \neg R(y, x))$ ) ring constraints are not supported either, but both are supported in  $\mathcal{SROIQ}$  and the latter is supported in  $\mathcal{DLR}_\mu$  through the stronger notion of well-foundedness. The last ‘basic’ ring constraint, acyclicity (“ $R$  is acyclic iff  $\forall x \neg (x \text{ has path to } x)$ ” in [17]), probably can be added to  $\mathcal{DLR}_{ifd}$  with the repeat PDL (transitive closure of roles,  $R^+$ , *i.e.*,  $\bigcup_{n>1} (R^T)^n$ ) using the least fixpoint construct  $\mu X.C$  (*i.e.*,  $\exists R^*.C = \mu X(C \sqcup \exists R.X)$  [8, 10]). ORM also permits *combinations* of ring constraints: intersecting acyclicity and intransitivity, antisymmetry with intransitivity, intransitive symmetry, and irreflexive symmetry.

The second main problem concerns constraints over  $k$  roles in an  $n$ -ary relation, which are: *Subset over  $k$  roles* in two  $n$ -ary relations (depicted in Fig.2-B, A below with abbreviation that underlined variable stands for a sequence  $x_1, \dots, x_n$  in an  $n$ -ary relation [23]),  $k < n$ , where the corresponding roles must match in domain, *Set-equality over  $k$  roles* (B), *Role exclusion over  $k$  roles* (C) in two  $n$ -ary relations  $R_i$  and  $R_j$ , and *Multi-role frequency* spanning  $i$  roles of an  $n$ -ary

relation,  $i > 2$ , and  $i \leq n$  (TFC5 in [16]), with formalisms as follows [16].

- A.  $\forall x_1, \dots, x_n (\exists \underline{y} (R_j(\underline{y}) \wedge x_1 = y_{j_1} \wedge \dots \wedge x_n = y_{j_n}) \rightarrow \exists \underline{z} (R_i(\underline{z}) \wedge x_1 = z_{i_1} \wedge \dots \wedge x_n = z_{i_n}))$  // **Subset over  $k$  roles**
- B.  $\forall x_1, \dots, x_n (\exists \underline{y} (R_j(\underline{y}) \wedge x_1 = y_{j_1} \wedge \dots \wedge x_n = y_{j_n}) \equiv \exists \underline{z} (R_i(\underline{z}) \wedge x_1 = z_{i_1} \wedge \dots \wedge x_n = z_{i_n}))$  // **Set-equality over  $k$  roles**
- C.  $\forall x_1, \dots, x_n \neg (\exists \underline{y} (R_i(\underline{y}) \wedge x_1 = y_{i_1} \wedge \dots \wedge x_n = y_{i_n}) \wedge \exists \underline{z} (R_j(\underline{z}) \wedge x_1 = z_{j_1} \wedge \dots \wedge x_n = z_{j_n}))$  // **Role exclusion over  $k$  roles**

The problem is that these constraints lead to undecidability if those  $k$  roles do not exactly make up the primary key (spanning uniqueness), as in a relational table (A)-(C) correspond to arbitrary projections. If one does not consider an additional key constraint, a *partial* mapping of (A)-(C) can be made on a ORM role-by-role basis (DL-role element by DL-role element). This can be reduced to a minor issue for *Multi-role frequency* (D) spanning roles  $r_i, r_j$  in an  $n$ -ary relation,  $n \geq 2$ , and  $1 \leq a \leq b$  and subsequently assesses it in combination with permissible uniqueness constraints.

- D.  $\forall x, y (\exists z_1 R(x, y, z_1) \rightarrow \exists z_2, \dots, z_a (z_1 \neq z_2 \wedge \dots \wedge z_{a-1} \neq z_a \wedge R(x, y, z_2) \wedge \dots \wedge R(x, y, z_a))) \wedge \forall x, y, z_1, \dots, z_{b+1} (R(x, y, z_1) \wedge \dots \wedge R(x, y, z_{b+1}) \rightarrow z_1 = z_2 \vee z_1 = z_3 \vee \dots \vee z_b = z_{b+1})$  // **Multi-role frequency**

Given that an elementary fact type must have uniqueness over  $n-1$  roles, then either 1)  $r_i$  or  $r_j$  is part of a single role uniqueness constraint but not both, 2)  $r_i$  or  $r_j$  is part of a multi-role uniqueness constraint but not both, 3) multi-role uniqueness includes  $r_i, r_j$ , and  $\geq 1$  other role in the relation, or 4) the relation is not an elementary fact type and ought to be remodelled to be elementary. Option 1 implies that either i)  $a = 1$  or ii)  $b = 1$ , and then the constraint can be reduced to 1: $n$  and  $m$ : $n$  uniqueness, respectively; options 2-4, however, reduce to the same problem of undecidability as with (C) (see [23]).

Last, a minor mapping issue is the *Role value constraint*, which is new in ORM2: object type  $C_i$  only participates in role  $r_i$  if an instance has any of the values  $\{v_i, \dots, v_k\}$ , which is a subset of the set of values  $C_i$  can have. With a binary relation, then  $\forall x, y (x \in \{v_i, \dots, v_k\} \rightarrow (R(x, y) \rightarrow C_i(x) \wedge C_j(y)))$  holds. A ‘candidate approach’ is to try to use  $\mathcal{DLR}_{ifd}$  through breaking down the constraint by creating a new subtype  $C'_i$  for the set of values to which the role is constrained, where the value can be any of  $\{v_i, \dots, v_k\}$ , and let  $C'_i$  play role  $r_i$ , s.t.  $C'_i \sqsubseteq C_i$  and  $C'_i \sqsubseteq \forall [r_i] R$  and then use named value types for the value constraints on  $C'_i$ . Alternatively, remodelling with role values might be an option, but, at present, this is supported only in  $DL-Lite_A$  [6].

### 3 Discussion and Conclusions

As has become clear from the mapping, the ORM ring constraints/DL-role properties are most problematic for  $\mathcal{DLR}_{ifd}$ , but most of them can be met by  $\mathcal{DLR}_\mu$  or  $\mathcal{SROIQ}$ . On the other hand,  $\mathcal{DLR}_\mu$  and  $\mathcal{SROIQ}$  do not have a constructor for non-unary primary keys, and  $\mathcal{SROIQ}$  neither supports  $n$ -ary relations where  $n > 2$  nor provides a means to ‘access’ an ORM-role/DL-role element. For these reasons,  $\mathcal{DLR}_{ifd}$  was chosen. Another advantage of having taken  $\mathcal{DLR}_{ifd}$

is that the syntax of UML class diagrams have been mapped into it [4], thereby augmenting the current informal mapping in [17] and moving closer to interoperability between ORM and UML through a *formal* correspondence between the two conceptual modelling languages with  $\mathcal{DLR}$  as unifying paradigm.

Looking toward implementations, EER and UML were mapped to  $\mathcal{DLR}$  and  $\mathcal{DLR}_{ifd}$  earlier and implemented in the iCOM tool [15], where one can use tools like Racer, Pellet, and FaCT. Hence, it uses a *SHIQ*-based reasoner through an additional transformation step; note that there are differences between theoretically computationally feasible and actually implemented reasoning services. Nevertheless, reasoning over less complex ORM-models is already a considerable advantage over no automated reasoning services at all. Moreover, if iCOM adds a module for ORM support, the modeller would be able to describe already supported inter-model assertions between EER, UML, and, now, ORM models and reason over any combination. Further down the line, a software developer will benefit from a better, consistent and known to be satisfiable, conceptual model and with *e.g.* NORMA would be able take advantage of the already implemented features of automated generation of relational databases and of software code.

Concluding, most –and the most often used– ORM2 elements and constraints can be mapped into  $\mathcal{DLR}_{ifd}$ . This already could be used for a wide range of ORM-models that do not use its full expressive capabilities; *e.g.*, to carry out model checking, compute derived relations, and classification. Conversely, when the present mapping is implemented, DLs will have a sophisticated user interface enabling domain experts to take part in representing their Universe of Discourse. Several approaches are possible to narrow the gap between ORM2 and DL languages, where a “ $\mathcal{DLR}_{\mu ifd}$ ” or *SROIQ* with *n*-ary relations seem close by. But to take advantage of narrowing the gap, tools for automated reasoning services will have to expand their features list as well. Alternatively, if this leads to undecidability or intractability, one could investigate modularization where a large conceptual model can be split-up into sections (ideally, hidden from the modeller) and perform the reasoning on the separate subsections. We are currently working on formal proofs of the ‘mismatches’ between ORM and the  $\mathcal{DLR}$ s.

## References

1. Artale, A., Franconi, E., Wolter, F., Zakharyashev, M. A temporal description logic for reasoning about conceptual schemas and queries. In: *Proc. of JELIA '02*, S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Springer, 2002, LNAI 2424, 98-110.
2. Baader, F. Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds). *Description Logics Handbook*, Cambridge University Press, 2003.
3. Balsters, H., Carver, A., Halpin, T., Morgan, T. Modeling dynamic rules in ORM. *2nd International Workshop on Object-Role Modelling (ORM 2006)*. LNCS 4278. Berlin: Springer-Verlag, 2006, 1201-1210.
4. Berardi, D., Calvanese, D., De Giacomo, G. Reasoning on UML class diagrams. *Artificial Intelligence*, 2005, 168(1-2):70-118.
5. Bollen, P. Using fact-orientation for instructional design. *2nd International Workshop on Object-Role Modelling (ORM 2006)*. LNCS 4278. Berlin: Springer-Verlag, 2006, 1231-1241.

6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic DL-Lite A. In *Proc. of the 2nd Workshop on OWL: Experiences and Directions (OWLED 2006)*, 2006.
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R. DL-Lite: Tractable description logics for ontologies. In: *Proc. of AAAI 2005*, 602-607.
8. Calvanese, D., De Giacomo, G. Expressive description logics. In: *The Description Logic Handbook: Theory, Implementation and Applications*, Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (Eds). Cambridge University Press, 2003. 178-218.
9. Calvanese, D., De Giacomo, G., Lenzerini, M. Identification constraints and functional dependencies in Description Logics. In *Proc. of IJCAI 2001*, 2001, 155-160.
10. Calvanese, D., De Giacomo, G., Lenzerini, M. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In: *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, 84-89, 1999.
11. Calvanese, C., De Giacomo, G., Lenzerini, M. On the decidability of query containment under constraints. In: *Proc. of PODS'98*, 149-158, 1998.
12. Calvanese, D., Lenzerini, M., Nardi, D. (1998) Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer, Amsterdam.
13. Campbell, L.J., Halpin, T.A. and Proper, H.A. Conceptual Schemas with Abstractions: Making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering*, 1996, 20(1): 39-85.
14. Evans, K. Requirements engineering with ORM. *Intl. Workshop on Object-Role Modeling (ORM'05)*. LNCS 3762. Berlin: Springer-Verlag, 2005. 646-655.
15. Franconi, F., Ng, G. The ICOM Tool for Intelligent Conceptual Modelling. *7th Workshop on Knowledge Representation meets Databases (KRDB'00)*, Berlin, 2000.
16. Halpin, T.A. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD Thesis, University of Queensland, Australia. 1989.
17. Halpin, T. *Information Modeling and Relational Databases*. San Francisco: Morgan Kaufmann Publishers, 2001.
18. Halpin, T. Objectification of relationships. *10th Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'05)*. Porto, Portugal, 2005.
19. Halpin, T. ORM2. *International Workshop on Object-Role Modeling (ORM'05)*. LNCS 3762. Berlin: Springer-Verlag, 2005. 676-687.
20. Hofstede, A.H.M. ter, Proper, H.A., Weide, Th.P. van der. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 1993, 18(7):489-523.
21. Hofstede, A.H.M. ter, Proper, H.A.. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 1998, (40(10)): 519-540.
22. Horrocks, I., Kutz, O., Sattler, U. The Even More Irresistible *SR0IQ*. In: *Proceedings of KR-2006*, Lake District, UK, 2006.
23. Keet, C.M. *Mapping the Object-Role Modeling language ORM2 into Description Logic language  $\mathcal{DLR}_{ifc}$* . Technical Report KRDB07-2, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy. 15 February 2007. arXiv:cs.LO/0702089v1.
24. Pepels, B., Plasmeijer, R. Generating applications from Object Role Models. *International Workshop on Object-Role Modeling (ORM'05)*. LNCS 3762. Berlin: Springer-Verlag, 2005, 656-665.
25. NORMA. <http://sourceforge.net/projects/orm/>.
26. OWL 1.1 [http://owl1.1.cs.manchester.ac.uk/owl\\_specification.html](http://owl1.1.cs.manchester.ac.uk/owl_specification.html).
27. OWL. <http://www.w3.org/TR/2004/REC-owl-semantics-20040210/syntax.html>.

## Adding ABoxes to a Description Logic with Uniqueness Constraints via Path Agreements

Vitaliy L. Khizder<sup>†</sup>, David Toman<sup>‡</sup> and Grant Weddell<sup>‡</sup>

<sup>†</sup>Oracle Corporation

<sup>‡</sup>David R. Cheriton School of Computer Science  
University of Waterloo, Canada

**Abstract.** We now know that the addition of a concept constructor called a Path Functional Dependency (PFD) to the Boolean-complete description logic  $\mathcal{DLF}$  leads to undecidability of ABox consistency for  $\mathcal{DLF}$ . Consequently, we define a boundary condition for PFDs that enables the recovery of earlier  $\mathcal{DLF}$  complexity bounds for this problem. This is accomplished indirectly by adding an additional concept constructor for rooted path equality, which has the added benefit of increasing the utility of  $\mathcal{DLF}$  for reasoning about query containment problems in query optimization, in plan generation and in automated synthesis of web services.

### 1 Introduction

The introduction of web service environments with query and ontology languages such as SWRL [8] and OWL [9] make it necessary for agents to reason about query plans and about how to communicate the results of queries between agents. It is consequently imperative to allow identification constraints to be incorporated in ontologies to enable an agent to determine, e.g., that *there is at most one way of performing a currency conversion*, or that *items from a particular company are reliably identified by that company’s item code*. Indeed, this situation is anticipated by extensive experience with SQL and the relational model of data in which keys and functional dependencies have been used for reasoning about properties of query plans related, e.g., to tuple identification.

In earlier work, we have explored how a general form of uniqueness constraint called a *Path Functional Dependency* (PFD) can be added to a Boolean-complete description logic called  $\mathcal{DLF}$ , a fragment of the OWL ontology language [14]. The resulting logic is called  $\mathcal{DLFD}$  and can be used to express, e.g., that *customers who have consulted with a manager can be reliably identified by their social insurance number*. In particular, this can be captured by the following inclusion dependency in  $\mathcal{DLFD}$ .

$$\text{Customer} \sqcap \forall \text{Consults} . \text{Manager} \sqsubseteq \text{Person} : \text{SIN} \rightarrow \text{Id}$$

To paraphrase: *If a customer has consulted with a manager, then no other person will share his or her social insurance number*. Later on, we show how this constraint can help with an important reformulation of a query.

Although  $\mathcal{DLFD}$  is particularly suited to capturing such meta-data, we have recently discovered that its ABox consistency problem is undecidable [22]. This is bad news since this greatly limits how the logic can then be used for reasoning about queries and services, e.g., in query reformulation. To remedy this, we define a *boundary* syntactic condition for PFDs that enables the recovery of earlier  $\mathcal{DLF}$  complexity bounds for

reasoning about ABox consistency. This is effectively accomplished by adding an additional concept constructor for rooted path equality along the lines pioneered by the description logic CLASSIC [3]. The addition of this constructor thereby obtains the logic  $\mathcal{DLFDE}$  which serves as the primary focus of this paper.

Our main contributions are as follows.

- We establish the equivalence of the ABox consistency problem for  $\mathcal{DLFD}$  and the logical implication problem for  $\mathcal{DLFDE}$ . Thus, and this was surprising to us, the latter is also undecidable. We also show that this equivalence continues to hold in the absence of any occurrences of PFDs.
- We define the logic  $\mathcal{DLFDE}^-$  which refines  $\mathcal{DLFDE}$  by introducing a syntactic boundary condition for PFDs, and show that both its ABox consistency problem and its logical implication problem are decidable and complete for EXPTIME.

### 1.1 Background and Related Work

In addition to the web ontology language OWL, description logics have been used extensively as a formal way of understanding a large variety of languages for specifying meta-data, including ER diagrams, UML class and object diagrams, relational database schema, etc. [18].

PFDs were first introduced and studied in the context of object-oriented data models [12, 24] as a way of capturing functional constraints such as keys and functional dependencies. An FD concept constructor was subsequently proposed and incorporated in CLASSIC [4], an early DL with a PTIME reasoning procedure, without changing the complexity of its implication problem. This was particularly noteworthy since CLASSIC also included a concept constructor for rooted path (in)equalities. The generalization of the FD constructor to PFDs alone leads to EXPTIME completeness of the implication problem [14]; this complexity remains unchanged in the presence of additional concept constructors common in rich DLs [23]. More recent work has shown the need to limit where PFDs may occur in concepts to avoid undecidability, in particular outside the scope of non-monotonic concept constructors such as negation, and that ABox consistency is undecidable regardless [22].

Calvanese and others have considered a DL with functional dependencies and a general form of keys added as additional varieties of dependencies, called a *key box* [6]. They show that their dialect is undecidable for DLs with inverse roles, but becomes decidable when unary functional dependencies are disallowed. This line of investigation is continued in the context of PFDs and inverse attributes, with analogous results [21]. We therefore disallow inverse attributes in this paper to exclude an already known cause for undecidability.

A form of key dependency with left hand side feature paths has been considered for a DL coupled with various concrete domains [15, 16]. In this case, the authors explore how the complexity of satisfaction is influenced by the selection of a concrete domain together with various syntactic restrictions on the key dependencies themselves.

PFDs have been used in a number of applications: in object-oriented schema diagnosis and synthesis [1, 2], in query optimization [13] and in the selection of indexing for a database [19]. Description logics have also been used for reasoning about query containment in the presence of rich database schema [5, 10].

SYNTAX	SEMANTICS: DEFN OF “ $(\cdot)^{\mathcal{I}}$ ”
$C ::= A$	$(A)^{\mathcal{I}} \subseteq \Delta$
$C_1 \sqcap C_2$	$(C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}}$
$\neg C$	$\Delta \setminus (C)^{\mathcal{I}}$
$\forall f.C$	$\{x : (f)^{\mathcal{I}}(x) \in (C)^{\mathcal{I}}\}$
$D ::= C$	
$C : \text{Pf}_1, \dots, \text{Pf}_k \rightarrow \text{Pf}$	$\{x : \forall y \in (C)^{\mathcal{I}}.$ $\bigwedge_{i=1}^k (\text{Pf}_i)^{\mathcal{I}}(x) = (\text{Pf}_i)^{\mathcal{I}}(y) \Rightarrow (\text{Pf})^{\mathcal{I}}(x) = (\text{Pf})^{\mathcal{I}}(y)\}$
$E ::= C$	
$E_1 \sqcap E_2$	$(E_1)^{\mathcal{I}} \cap (E_2)^{\mathcal{I}}$
$\neg E$	$\Delta \setminus (E)^{\mathcal{I}}$
$\forall f.E$	$\{x : (f)^{\mathcal{I}}(x) \in (E)^{\mathcal{I}}\}$
$(\text{Pf}_1 = \text{Pf}_2)$	$\{x : (\text{Pf}_1)^{\mathcal{I}}(x) = (\text{Pf}_2)^{\mathcal{I}}(x)\}$

Fig. 1. SYNTAX AND SEMANTICS OF  $\mathcal{DLFDE}$ .

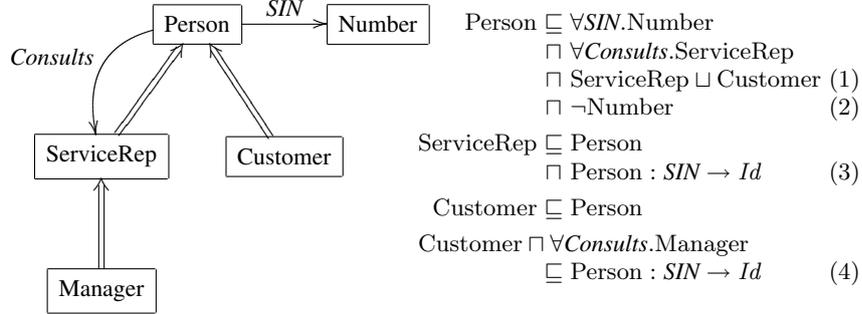
The remainder of the paper is organized as follows. The definition of  $\mathcal{DLFDE}$ , a Boolean complete DL based on attributes that includes the PFD concept constructor, immediately follows. In Section 3, we establish the first of our main results: the equivalence of the ABox consistency problem for  $\mathcal{DLFD}$  and the logical implication problem for  $\mathcal{DLFDE}$ , and that this equivalence continues to hold in the absence of any occurrences of PFDs. Section 4 introduces the logic  $\mathcal{DLFDE}^-$  which refines  $\mathcal{DLFDE}$  by introducing the above-mentioned syntactic boundary condition for PFDs, and shows that both the ABox consistency problem and the logical implication problem for  $\mathcal{DLFDE}^-$  are decidable and complete for EXPTIME. A more in depth example of using  $\mathcal{DLFDE}^-$  for reasoning about a query property follows in Section 5. Our summary comments then follow in Section 6.

## 2 Definitions

To simplify the presentation, the description logic  $\mathcal{DLFDE}$  defined below is based on *attributes* (also called *features*) instead of the more common case of roles. With regard to expressiveness, note that  $\mathcal{ALCN}$  with a suitable PFD construct can simulate our dialect. Conversely,  $\mathcal{DLFD}$  can simulate  $\mathcal{ALCQI}$  [20].

**Definition 1 (Description Logic  $\mathcal{DLFDE}$ )** Let  $F$ ,  $A$  and  $N$  be sets of (names of) attributes, primitive concepts and individuals, respectively. A path expression is defined by the grammar “ $\text{Pf} ::= f. \text{Pf} \mid \text{Id}$ ” for  $f \in F$ . We define derived concept descriptions by the grammar on the left-hand-side of Figure 1. A concept description obtained by using the fourth production of this grammar is called an attribute value restriction. A concept description obtained by using the sixth production is called a path functional dependency (PFD).

An inclusion dependency  $\mathcal{C}$  is an expression of the form  $C \sqsubseteq D$ . A posed question  $\mathcal{Q}$  is an expression of the form  $E_1 \sqsubseteq E_2$ . A terminology (TBox)  $\mathcal{T}$  consists of a finite set of inclusion dependencies.



**Fig. 2.** THE SERVICE SCHEMA AND TERMINOLOGY.

An ABox  $\mathcal{A}$  consists of a finite set of assertions of the form  $C(a)$  or  $f(a) = b$ , for  $C$  a concept description,  $f \in F$  and  $\{a, b\} \subseteq N$ .

The semantics of expressions is defined with respect to a structure  $(\Delta, \cdot^{\mathcal{I}})$ , where  $\Delta$  is a domain of “objects” and  $(\cdot)^{\mathcal{I}}$  an interpretation function that fixes the interpretation of primitive concepts  $A$  to be subsets of  $\Delta$ , primitive attributes  $f$  to be total functions  $(f)^{\mathcal{I}} : \Delta \rightarrow \Delta$  and individuals  $a$  to be elements of  $\Delta$ . The interpretation is extended to path expressions,  $(Id)^{\mathcal{I}} = \lambda x.x$ ,  $(f.Pf)^{\mathcal{I}} = (Pf)^{\mathcal{I}} \circ (f)^{\mathcal{I}}$  and derived concept descriptions  $C, D$  and  $E$  as defined on the right-hand-side of Figure 1.

An interpretation satisfies an inclusion dependency  $C \sqsubseteq D$  (resp. a posed question  $E_1 \sqsubseteq E_2$ ) if  $(C)^{\mathcal{I}} \subseteq (D)^{\mathcal{I}}$  (resp.  $(E_1)^{\mathcal{I}} \subseteq (E_2)^{\mathcal{I}}$ ). An interpretation satisfies an ABox assertion  $C(a)$  (resp.  $f(a) = b$ ) if  $(a)^{\mathcal{I}} \in (C)^{\mathcal{I}}$  (resp.  $(f)^{\mathcal{I}}((a)^{\mathcal{I}}) = (b)^{\mathcal{I}}$ ).

The logical implication problem asks if  $\mathcal{T} \models \mathcal{Q}$  holds; that is, for a posed question  $\mathcal{Q}$ , if  $\mathcal{Q}$  is satisfied by any interpretation that satisfies all inclusion dependencies in  $\mathcal{T}$ . The ABox consistency problem asks if  $\mathcal{T} \cup \mathcal{A}$  is consistent; that is, if there exists an interpretation that satisfies all inclusion dependencies in  $\mathcal{T}$  and all assertions in  $\mathcal{A}$ .

To improve readability in the remainder of the paper, we follow the simple protocol of removing “. Id” from the end of path expressions that consist of at least one attribute. We also allow the use of standard abbreviations, e.g.,  $\perp$  for  $A \sqcap \neg A$ ,  $(Pf_1 \neq Pf_2)$  for  $\neg(Pf_1 = Pf_2)$ , etc. Finally, we write  $Pf(a) = b$  as shorthand for the equivalent set of primitive ABox assertions with “single use” intermediate individuals.

**Example 2** Figure 2 illustrates an information schema for a hypothetical online customer SERVICE system where, e.g., service representatives are special cases of people who in turn have social insurance numbers and consult with service representatives, and so on. The information can be captured as a SERVICE TBox with the inclusion dependencies in the right part of Figure 2. The four marked dependencies more thoroughly exercise the capabilities of  $\mathcal{DLFDE}$ , asserting that:

1. A person is either a service representative or a customer;
2. Nothing is both a person and a number;
3. Any service representative is uniquely identified by a social insurance number; and
4. (from our introductory comments) Any customer who has consulted with a manager is also uniquely identified by a social insurance number.

### 3 Equations and ABoxes

We first explore the relationship between ABox consistency problems and allowing path agreements in posed questions. It turns out that either capacity alone is sufficient: each is able to effectively simulate the other.

Intuitively, path equations (and inequations) can *enforce* that an arbitrary finite graph (with feature-labeled edges and concept description-labeled nodes) is a part of any model that satisfies the equations. Such a graph can equivalently be enforced by an ABox. Hence we have:

**Theorem 3** *Let  $\mathcal{T}$  be a  $\mathcal{DLFD}$  terminology and  $\mathcal{A}$  an ABox. Then there is a concept  $E$  such that  $\mathcal{T} \cup \mathcal{A}$  is not consistent if and only if  $\mathcal{T} \models E \sqsubseteq \perp$ .*

Conversely, it is also possible to show that ABox reasoning can be used for reasoning about equational constraints in the posed questions. However, as the equational concepts are closed under boolean constructors, a single equational problem may need to map to several ABox consistency problems.

**Theorem 4** *Let  $\mathcal{T}$  be a  $\mathcal{DLFD}$  terminology and  $E$  an equational concept. Then there is a finite set of ABoxes  $\{\mathcal{A}_i : 0 < i \leq k\}$  such that*

$$\mathcal{T} \models E \sqsubseteq \perp \text{ iff } \mathcal{T} \cup \mathcal{A}_i \text{ is not consistent for all } 0 < i \leq k.$$

Theorems 3 and 4 hold even when the terminology  $\mathcal{T}$  is restricted to the  $\mathcal{DLF}$  fragment (i.e., does not contain any occurrences of the PFD concept constructor).

### 4 Adding PFDs

The correspondence between ABox reasoning and equational concepts in the posed questions provides us with the necessary means to understanding the impact of PFDs in the presence of an ABox or path agreements. Indeed, our  $\mathcal{DLFDE}$  grammar in Figure 1 does not explicitly allow posed questions to contain PFDs (the PFD constructor is confined by the Grammar to the TBox). However, this restriction can be easily circumvented using the following lemma:

**Lemma 5** *Let  $\mathcal{T}$  be a  $\mathcal{DLFD}$  terminology and  $E_1 \sqsubseteq E_2 : \text{Pf}_1, \dots, \text{Pf}_k \rightarrow \text{Pf}$  a posed question. Then there is an equational concept  $E$  such that  $\mathcal{T} \models E \sqsubseteq \perp$  iff*

$$\mathcal{T} \models E_1 \sqsubseteq E_2 : \text{Pf}_1, \dots, \text{Pf}_k \rightarrow \text{Pf}.$$

#### 4.1 Undecidability

While allowing general reasoning about path agreements in terminologies leads immediately to undecidability (by virtue of a straightforward reduction of the uniform word problem [17]), the following two restricted cases have decidable decision problems:

- Allowing arbitrary PFDs in terminologies; and
- Allowing path agreements in the posed question.

Unfortunately, the combination of the two cases again leads to undecidability:

**Theorem 6** *Let  $\mathcal{T}$  be a  $\mathcal{DLFD}$  terminology and  $E$  an equational concept. Then the problem  $\mathcal{T} \models E \sqsubseteq \perp$  is undecidable.*

#### 4.2 Decidability and a Boundary Condition

To regain decidability, we restrict the PFD constructor to the following two forms:

- $C : Pf_1, \dots, Pf_i, \dots, Pf_k \rightarrow Pf$ ; and
- $C : Pf_1, \dots, Pf_i, \dots, Pf_k \rightarrow Pf.f$ , for some primitive feature  $f$ .

We call the resulting fragment  $\mathcal{DLFDE}^-$ . To abstract syntax a bit for the sake of readability, the condition distinguishes, e.g., the PFDs  $f \rightarrow Id$  and  $f \rightarrow g$  from the PFD  $f \rightarrow g.f$ . Intuitively, a simple saturation procedure that “fires” PFDs on a hypothetical database is now guaranteed to terminate as a consequence.

Notice that the boundary condition still admits PFDs that express arbitrary keys or functional dependencies in the sense of the relational model, including those occurring in all our examples. Thus, we believe that restricting PFDs in this manner does not sacrifice any real world modeling utility.

**Theorem 7** *Let  $\mathcal{T}$  and  $\mathcal{T}'$  be a  $\mathcal{DLF}$  and  $\mathcal{DLFDE}^-$  terminologies, respectively, and  $E$  an equational concept. Then there is a concept  $E'$  such that*

$$\mathcal{T} \cup \mathcal{T}' \models E \sqsubseteq \perp \text{ iff } \mathcal{T} \models (E \sqcap E') \sqsubseteq \perp.$$

Moreover,  $E'$  can be constructed from  $\mathcal{T}'$  and  $E$  effectively and is polynomial in  $|\mathcal{T}'|$ .

The boundary condition on PFDs is essential for the above theorem to hold. If unrestricted PFDs are combined with either equations or an ABox, there is no limit on the length of paths participating in path agreements when measured from an initial object  $o \in E \sqcap E'$  in the associated satisfiability problem. Moreover, any minimal relaxation of this condition, i.e., allowing any PFDs of the form  $C : f \rightarrow g.h$ , already leads to undecidability [22].

**Corollary 8**  *$\mathcal{DLFDE}^-$  logical implication and ABox consistency problems are decidable and complete for EXPTIME.*

### 5 Applications to Query Optimization

Assuming set semantics for query results and the presence of a database schema, [5] and [10] have shown how conjunctive and positive query containment can be reduced to ABox consistency problems for the description logics  $\mathcal{DLR}$  [7] and  $\mathcal{SHIQ}$  [11], respectively. Analogous reductions can also be made to ABox consistency problems for  $\mathcal{DLFD}$ . In addition, more direct and transparent reductions that use path agreements are now possible with  $\mathcal{DLFDE}^-$  (c.f. the concept description  $E$  in the following example).

$\mathcal{DLFDE}^-$  can also be used in query reformulation when allowing duplicate semantics. However, in this case, PFDs serve an essential role as we now illustrate.

**Example 9** Consider the following SQL-like query on our example SERVICE schema that finds distinct social insurance numbers for all persons who have consulted with a manager.

```

select distinct N
from   Person as P, Number as N,
where  exists ( select *
                from   Manager as M
                where  P.Consults = M )
and    P.SIN = N
    
```

Clearly, there is a considerable incentive on the grounds of query performance to reason about the possibility of avoiding expensive duplication elimination operations, i.e., if the above query can be reformulated *without* the `distinct` keyword. Indeed, this is possible iff

$$\text{SERVICE} \models E \sqsubseteq E : N \rightarrow P$$

where  $E$  is the concept description

$$(\forall P.\text{Person}) \sqcap (\forall N.\text{Number}) \sqcap (\forall M.\text{Manager}) \\ \sqcap (P.\text{Consults} = M) \sqcap (P.\text{SIN} = N)$$

that encodes the above query. Note that this clear and direct formulation relies on Lemma 5.

The link between the above formulation and earlier ABox consistency approaches to query containment is explained by Theorems 4 and 3.

## 6 Conclusions

Earlier research has led to the development of a Boolean-complete description logic called  $\mathcal{DLFD}$  that incorporated a powerful concept constructor for expressing uniqueness constraints called PFDs. Unfortunately, recent negative results have shown that the ABox consistency problem for  $\mathcal{DLFD}$  is not decidable. In this paper, we have proposed a boundary condition for PFDs that re-obtains decidability and tight complexity bounds for ABox consistency and logical implication problems. We have also shown how ABox consistency checking relates to logical implication problems when path agreement is allowed in posed questions. This connection is essential to the design of the decision procedure for  $\mathcal{DLFD}$  in the presence of an ABox.

There are several directions for future research, in particular exploring alternative fragments of  $\mathcal{DLFDE}$  with decidable reasoning problems; finding further restrictions on  $\mathcal{DLFDE}$  that lead to polynomial time reasoning algorithms; and incorporating more general ordering dependencies that generalize equality based reasoning that underlies PFDs.

## References

1. Joachim Biskup and Torsten Polle. Decomposition of Database Classes under Path Functional Dependencies and Onto Constraints. In *Foundations of Information and Knowledge Systems*, pages 31–49, 2000.
2. Joachim Biskup and Torsten Polle. Adding inclusion dependencies to an object-oriented data model with uniqueness constraints. *Acta Informatica*, 39:391–449, 2003.
3. Alex Borgida and Peter F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.
4. Alexander Borgida and Grant E. Weddell. Adding Uniqueness Constraints to Description Logics (Preliminary Report). In *International Conference on Deductive and Object-Oriented Databases*, pages 85–102, 1997.
5. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the Decidability of Query Containment under Constraints. In *ACM Symposium on Principles of Database Systems*, pages 149–158, 1998.

6. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification Constraints and Functional Dependencies in Description Logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 155–160, 2001.
7. Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying Class-Based Representation Formalisms. *J. Artificial Intelligence Research*, 11:199–240, 1999.
8. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosf, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Technical report, W3C, 2004.
9. Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a Web Ontology Language. *J. Web Sem.*, 1(1):7–26, 2003.
10. Ian Horrocks, Ulrike Sattler, Sergio Tessaris, and Stephan Tobies. How to Decide Query Containment under Constraints using a Description Logic. In *LPAR Int. Conf. on Logic for Programming and Automated Reasoning*, pages 326–343. LNCS 1955, 2000.
11. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical Reasoning for Expressive Description Logics. In *Logic Programming and Automated Reasoning, LPAR'99*, pages 161–180, 1999.
12. Minoru Ito and Grant E. Weddell. Implication Problems for Functional Constraints on Databases Supporting Complex Objects. *Journal of Computer and System Sciences*, 49(3):726–768, 1994.
13. Vitaliy L. Khizder, David Toman, and Grant E. Weddell. Reasoning about Duplicate Elimination with Description Logic. In *Rules and Objects in Databases (DOOD, part of CL'00)*, pages 1017–1032, 2000.
14. Vitaliy L. Khizder, David Toman, and Grant E. Weddell. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *International Conference on Database Theory ICDT'01*, pages 54–67, 2001.
15. C. Lutz and M. Milicic. Description Logics with Concrete Domains and Functional Dependencies. In *Proc. European Conference on Artificial Intelligence (ECAI)*, pages 378–382, 2004.
16. Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, Nominals, and Concrete Domains. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 349–354, 2003.
17. Michael Machtey and Paul Young. *An Introduction to the General Theory of Algorithms*. North-Holland Amsterdam, 1978.
18. U. Sattler, D. Calvanese, and R. Molitor. Relationships with other formalisms. In *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter 4, pages 137–177. Cambridge University Press, 2003.
19. Lubomir Stanchev and Grant E. Weddell. Index Selection for Embedded Control Applications using Description Logics. In *Description Logics 2003*, pages 9–18. CEUR-WS vol.81, 2003.
20. David Toman and Grant Weddell. On Reasoning about Structural Equality in XML: A Description Logic Approach. *Theoretical Computer Science*, 336(1):181–203, 2005.
21. David Toman and Grant Weddell. On the Interaction between Inverse Features and Path-functional Dependencies in Description Logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 603–608, 2005.
22. David Toman and Grant Weddell. On Keys and Functional Dependencies as First-Class Citizens in Description Logics. In *Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR)*, pages 647–661, 2006.
23. David Toman and Grant E. Weddell. On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem. In *Description Logics 2001*, pages 76–85. CEUR-WS vol.49, 2001.
24. Grant Weddell. A Theory of Functional Dependencies for Object Oriented Data Models. In *International Conference on Deductive and Object-Oriented Databases*, pages 165–184, 1989.

## A Well-founded Semantics for Hybrid MKNF Knowledge Bases<sup>\*</sup>

Matthias Knorr<sup>1</sup>, José Júlio Alferes<sup>1</sup>, and Pascal Hitzler<sup>2</sup>

<sup>1</sup> CENTRIA, Universidade Nova de Lisboa, Portugal

<sup>2</sup> AIFB, Universität Karlsruhe, Germany

**Abstract.** In [10], hybrid MKNF knowledge bases have been proposed for combining open and closed world reasoning within the logics of minimal knowledge and negation as failure ([8]). For this powerful framework, we define a three-valued semantics and provide an alternating fixpoint construction for nondisjunctive hybrid MKNF knowledge bases. We thus provide a well-founded semantics which is a sound approximation of the cautious MKNF model semantics, and which also features improved computational properties. We also show that whenever the DL knowledge base part is empty, then the alternating fixpoint coincides with the classical well-founded model.

### 1 Introduction

One of the major open research questions in Description Logic (DL) research is how to combine the open-world semantics of DLs with the closed-world semantics featured by (nonmonotonic) logic programming (LP). Much of this research effort is being driven by the needs of the Semantic Web initiative. Indeed, the addition of rules, in LP style, on top of the DL-based ontology layer has been recognized as an important task for the success of the Semantic Web, and initiatives are being taken to define such a rule layer (cf. the Rule Interchange Format working group of the W3C). Combining LP rules and DLs indeed is a non-trivial task since these two formalisms are based on different assumptions: the former is nonmonotonic, relying on the closed world assumption, while the latter is based on first-order logic under the open world assumption.

Accordingly, several proposals have been made for dealing with knowledge bases (KB) which contain DL and LP statements (see e.g. [2–4, 7, 10, 12]). But apart from [4], they rely on the stable models semantics (SMS) of logic programs [6]. It is our stance that, especially for use in the Semantic Web, the well-founded semantics (WFS) [14], though being closely related to SMS (see e.g. [5]), is often the better choice. Indeed, in applications dealing with large amounts of information, the polynomial worst-case complexity of WFS is preferable to the NP-hard SMS. Furthermore, the WFS is defined for all programs and allows to

<sup>\*</sup> Pascal Hitzler is supported by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project (grant 01 IMD01 B), and by the Deutsche Forschungsgemeinschaft (DFG) under the ReaSem project.

answer queries by consulting only the relevant part of a program whereas SMS is neither relevant nor always defined.

While the approach in [4] is based on a loose coupling between DL and LP, others are tightly integrated. The most advanced of these approaches currently appears to be that of hybrid MKNF knowledge bases [10] which is based on the logic of Minimal Knowledge and Negation as failure (MKNF) [8]. Its advantage lies in a seamless integration of DL and LP which is nevertheless decidable due to the restriction of reasoning in the program part to known constants by means of DL-safe rules.

In this paper, we define a well-founded semantics for hybrid MKNF knowledge bases, for now restricting to nondisjunctive MKNF rules, which compares to that of [10] as the WFS does to the SMS of LP:

- our well-founded semantics is a sound approximation of the semantics of [10]
- the computational complexity is strictly lower
- the semantics retains the property of [10] of being faithful, but now wrt. the WFS, i.e. when the DL part is empty, it coincides with the WFS of LPs.

We start by recalling basic notions and then introduce models in a 3-valued setting. The paper continues with the definition of the proposed semantics and some of its properties. We end with conclusion and future work. Lack of space prevents us from presenting all proofs, which can be found in the extended report at <http://centria.di.fct.unl.pt/~mknorr/wfmknf-extd.pdf>.

## 2 Preliminaries

*MKNF notions.* We start by recalling the syntax of MKNF formulas from [10]. A *first-order atom*  $P(t_1, \dots, t_n)$  is an MKNF formula where  $P$  is a predicate and the  $t_i$  are first-order terms<sup>3</sup>. If  $\varphi$  is an MKNF formula then  $\neg\varphi$ ,  $\exists x : \varphi$ ,  $\mathbf{K}\varphi$  and  $\mathbf{not}\varphi$  are MKNF formulas and likewise  $\varphi_1 \wedge \varphi_2$  and  $\varphi_1 \subset \varphi_2$  for MKNF formulas  $\varphi_1, \varphi_2$ . We use the following symbols to represent boolean combinations of the previously introduced syntax, i.e.  $\varphi_1 \vee \varphi_2$  for  $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$ ,  $\varphi_1 \equiv \varphi_2$  for  $(\varphi_1 \subset \varphi_2) \wedge (\varphi_2 \subset \varphi_1)$ , and  $\forall x : \varphi$  for  $\neg\exists x : \neg\varphi$ . Substituting the free variables  $x_i$  in  $\varphi$  by terms  $t_i$  is denoted  $\varphi[t_1/x_1, \dots, t_n/x_n]$ . Given a (first-order) formula  $\varphi$ ,  $\mathbf{K}\varphi$  is called a *modal  $\mathbf{K}$ -atom* and  $\mathbf{not}\varphi$  a *modal  $\mathbf{not}$ -atom*. An MKNF formula  $\varphi$  without any free variables is a *sentence* and *ground* if it does not contain variables at all. It is *positive* if it does not contain the operator  $\mathbf{not}$ .

It is assumed that apart from the constants occurring in the formulas the signature contains a countably infinite supply of constants not occurring in the formulas. The Herbrand Universe of such a signature is also denoted  $\Delta$ . The signature contains the equality predicate  $\approx$  which is interpreted as congruence relation on  $\Delta$ . An *MKNF structure* is a triple  $(I, M, N)$  where  $I$  is an Herbrand first-order interpretation over  $\Delta$  and  $M$  and  $N$  are nonempty sets of Herbrand first-order interpretations over  $\Delta$ . For the 2-valued satisfiability of MKNF sentences we refer only to [10], since we will define 3-valued satisfiability in a way that, when restricted to 2-valued trivially coincides with the one of [10].

<sup>3</sup> We consider function-free first-order logic, so terms are either constants or variables.

*Hybrid MKNF Knowledge Bases.* Quoting from [10], the approach of hybrid MKNF knowledge bases is applicable to any first-order fragment  $\mathcal{DL}$  satisfying these conditions: (i) each knowledge base  $\mathcal{O} \in \mathcal{DL}$  can be translated into a formula  $\pi(\mathcal{O})$  of function-free first-order logic with equality, (ii) it supports *A-Boxes*-assertions of the form  $P(a_1, \dots, a_n)$  for  $P$  a predicate and  $a_i$  constants of  $\mathcal{DL}$  and (iii) satisfiability checking and instance checking (i.e. checking entailments of the form  $\mathcal{O} \models P(a_1, \dots, a_n)$ ) are decidable<sup>4</sup>.

We recall MKNF rules and hybrid MKNF knowledge bases from [10]. For the rationales behind these and the following notions we also refer to [9].

**Definition 2.1.** *Let  $\mathcal{O}$  be a DL knowledge base. A first-order function-free atom  $P(t_1, \dots, t_n)$  over  $\Sigma$  such that  $p$  is  $\approx$  or it occurs in  $\mathcal{O}$  is called a DL-atom; all other atoms are called non-DL-atoms. An MKNF rule  $r$  has the following form where  $H_i$ ,  $A_i$ , and  $B_i$  are first-order function free atoms:*

$$\mathbf{K} H_1 \vee \dots \vee \mathbf{K} H_l \leftarrow \mathbf{K} A_1, \dots, \mathbf{K} A_n, \mathbf{not} B_1, \dots, \mathbf{not} B_m \quad (1)$$

The sets  $\{\mathbf{K} H_i\}$ ,  $\{\mathbf{K} A_i\}$ , and  $\{\mathbf{not} B_i\}$  are called the rule head, the positive body, and the negative body, respectively. A rule is nondisjunctive if  $l = 1$ ;  $r$  is positive if  $m = 0$ ;  $r$  is a fact if  $n = m = 0$ . A program is a finite set of MKNF rules. A hybrid MKNF knowledge base  $\mathcal{K}$  is a pair  $(\mathcal{O}, \mathcal{P})$  and  $\mathcal{K}$  is nondisjunctive if all rules in  $\mathcal{P}$  are nondisjunctive.

The semantics of an MKNF knowledge base is obtained by translating it into an MKNF formula ([10]).

**Definition 2.2.** *Let  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$  be a hybrid MKNF knowledge base. We extend  $\pi$  to  $r$ ,  $\mathcal{P}$ , and  $\mathcal{K}$  as follows, where  $x$  is the vector of the free variables of  $r$ .*

$$\pi(r) = \forall x : (\mathbf{K} H_1 \vee \dots \vee \mathbf{K} H_l \subset \mathbf{K} A_1, \dots, \mathbf{K} A_n, \mathbf{not} B_1, \dots, \mathbf{not} B_m)$$

$$\pi(\mathcal{P}) = \bigwedge_{r \in \mathcal{P}} \pi(r) \quad \pi(\mathcal{K}) = \mathbf{K} \pi(\mathcal{O}) \wedge \pi(\mathcal{P})$$

An MKNF rule  $r$  is *DL-safe* if every variable in  $r$  occurs in at least one non-DL-atom  $\mathbf{K} B$  occurring in the body of  $r$ . A hybrid MKNF knowledge base  $\mathcal{K}$  is *DL-safe* if all its rules are DL-safe. Given a hybrid MKNF knowledge base  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ , the *ground instantiation* of  $\mathcal{K}$  is the KB  $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$  where  $\mathcal{P}_G$  is obtained by replacing in each rule of  $\mathcal{P}$  all variables with constants from  $\mathcal{K}$  in all possible ways. Then it was shown in [9], for a DL-safe hybrid KB  $\mathcal{K}$  and a ground MKNF formula  $\psi$ , that  $\mathcal{K} \models \psi$  if and only if  $\mathcal{K}_G \models \psi$ .

### 3 Well-founded MKNF Semantics

#### 3.1 Three-valued Models

Satisfiability as defined in [10] allows modal atoms only to be either true or false in a given MKNF structure. We extend the framework by allowing a third

<sup>4</sup> For more details on DL notation we refer to [1].

truth value  $\mathbf{u}$ , denoting undefined, to be assigned to modal atoms while first-order atoms remain two-valued due to being interpreted solely in one first-order interpretation. We therefore introduce *consistent* MKNF structures which, for all MKNF formulas  $\varphi$  over some given signature, do not allow  $\varphi$  to be true for all  $J \in M$  and false for some  $J \in N$  at the same time. Subsequently, we evaluate MKNF sentences in consistent MKNF structures with respect to the set  $\{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$  of truth values with the order  $\mathbf{f} < \mathbf{u} < \mathbf{t}$ :

$$\begin{aligned}
 - (I, M, N)(p(t_1, \dots, t_n)) &= \begin{cases} \mathbf{t} & \text{iff } p(t_1, \dots, t_n) \in I \\ \mathbf{f} & \text{iff } p(t_1, \dots, t_n) \notin I \end{cases} \\
 - (I, M, N)(\neg\varphi) &= \begin{cases} \mathbf{t} & \text{iff } (I, M, N)(\varphi) = \mathbf{f} \\ \mathbf{u} & \text{iff } (I, M, N)(\varphi) = \mathbf{u} \\ \mathbf{f} & \text{iff } (I, M, N)(\varphi) = \mathbf{t} \end{cases} \\
 - (I, M, N)(\varphi_1 \wedge \varphi_2) &= \min\{(I, M, N)(\varphi_1), (I, M, N)(\varphi_2)\} \\
 - (I, M, N)(\varphi_1 \supset \varphi_2) &= \mathbf{t} \text{ iff } (I, M, N)(\varphi_2) \geq (I, M, N)(\varphi_1) \text{ and } \mathbf{f} \text{ otherwise} \\
 - (I, M, N)(\exists x : \varphi) &= \max\{(I, M, N)(\varphi[\alpha/x]) \mid \alpha \in \Delta\} \\
 - (I, M, N)(\mathbf{K} \varphi) &= \begin{cases} \mathbf{t} & \text{iff } (J, M, N)(\varphi) = \mathbf{t} \text{ for all } J \in M \\ \mathbf{f} & \text{iff } (J, M, N)(\varphi) = \mathbf{f} \text{ for some } J \in N \\ \mathbf{u} & \text{otherwise} \end{cases} \\
 - (I, M, N)(\mathbf{not} \varphi) &= \begin{cases} \mathbf{t} & \text{iff } (J, M, N)(\varphi) = \mathbf{f} \text{ for some } J \in N \\ \mathbf{f} & \text{iff } (J, M, N)(\varphi) = \mathbf{t} \text{ for all } J \in M \\ \mathbf{u} & \text{otherwise} \end{cases}
 \end{aligned}$$

The operator max chooses the greatest element with respect to the truth ordering given above and likewise min chooses the least one. We can see that the truth of modal atoms is evaluated just as in the two-valued case (see [10]), we only have to separate additionally false from undefined modal atoms which is done by means of the other set of interpretations in the structure. Note that implications and objective MKNF formulas can never be undefined.

**Definition 3.1.** An interpretation pair  $(M, N)$  consists of two MKNF interpretations  $M, N$  and models a closed MKNF formula  $\varphi$ , written  $(M, N) \models \varphi$ , if and only if  $(I, M, N)(\varphi) = \mathbf{t}$  for each  $I \in M$ . We call  $\varphi$  consistent if there exists an interpretation pair modeling it.

It is straightforward to see (cf. [10]) that  $(M, M)$  corresponds to the (two-valued) MKNF interpretation  $M$ , i.e. a nonempty set of Herbrand first-order interpretations over  $\Delta$ , since there are no undefined modal atoms in it. In this case, recalling from [10],  $M$  is additionally an MKNF model if (1)  $(I, M, M)(\varphi) = \mathbf{t}$  for all  $I \in M$  and (2) for each MKNF interpretation  $M'$  such that  $M' \supset M$  we have  $(I', M', M)(\varphi) = \mathbf{f}$  for some  $I' \in M'$ .

*Example 3.1.* Let us consider the following hybrid MKNF knowledge base

$$\text{NaturalDeath} \sqsubseteq \text{Pay} \quad \text{Suicide} \sqsubseteq \neg\text{Pay}$$

$$\mathbf{K} \text{Pay}(x) \leftarrow \mathbf{K} \text{murdered}(x), \mathbf{K} \text{benefits}(y, x), \mathbf{not} \text{responsible}(y, x)$$

$$\mathbf{K} \text{Suicide}(x) \leftarrow \mathbf{not} \text{NaturalDeath}(x), \mathbf{not} \text{murdered}(x)$$

$$\mathbf{K} \text{murdered}(x) \leftarrow \mathbf{not} \text{NaturalDeath}(x), \mathbf{not} \text{Suicide}(x)$$

based on which a life insurance company decides whether to pay or not the insurance. Additionally, we know that Mr. Jones who owned a life insurance was found death in his living room, the revolver still in his hand. Thus we add  $\neg\text{NaturalDeath}(\text{jones})$  and the last two rules offer us a choice between commitment of suicide or murder. While immediately obtaining two MKNF models in such a scenario, the three-valued framework allows to assign  $\mathbf{u}$  to both so that we delay this decision until the evidence is evaluated. Until then, by the first rule, also no payment is possible.

### 3.2 Alternating Fixpoint for Hybrid MKNF

As discussed in [9], since an MKNF model  $M$  is in general infinite, instead of representing  $M$  directly, a first-order formula  $\varphi$  is computed such that  $M$  is exactly the set of first-order models of  $\varphi$ . This is possible for modally closed MKNF formulae and the ideas from [11] are applied to provide a partition  $(P, N)$  of modal atoms which uniquely defines  $\varphi$ . We extend this idea by allowing partitions to be partial in the sense that modal atoms may occur neither in  $P$  nor in  $N$ , i.e. are neither true nor false but supposed to be undefined. To obtain the unique desired partial partition we apply a technique known from logic programming: stable models ([6]) for normal logic programs correspond one-to-one to MKNF models of programs of MKNF rules (see [8]). The well-founded model ([14]) for normal logic programs can be computed by an alternating fixpoint of the operator used to define stable models ([13]).

Here we proceed similarly: we define an operator providing a stable condition for nondisjunctive hybrid MKNF knowledge bases and use it to obtain an alternating fixpoint, the well-founded semantics. We thus start by adapting some notions from [10] formalizing partitions and related concepts.

**Definition 3.2.** Let  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$  be a hybrid MKNF knowledge base. The set of  $\mathbf{K}$ -atoms of  $\mathcal{K}$ , written  $\text{KA}(\mathcal{K})$ , is the smallest set that contains (i) all  $\mathbf{K}$ -atoms of  $\mathcal{P}_G$ , and (ii) a modal atom  $\mathbf{K}\xi$  for each modal atom **not**  $\xi$  occurring in  $\mathcal{P}_G$ .

For a subset  $P$  of  $\text{KA}(\mathcal{K})$ , the objective knowledge of  $P$  is the formula  $\text{ob}_{\mathcal{K}, P} = \mathcal{O} \cup \bigcup_{\mathbf{K}\xi \in P} \xi$ . A (partial) partition  $(P, N)$  of  $\text{KA}(\mathcal{K})$  is consistent if  $\text{ob}_{\mathcal{K}, P} \not\models \xi$  for each  $\mathbf{K}\xi \in N$ .

For a set of modal atoms  $S$ ,  $S_{DL}$  is the subset of DL-atoms of  $S$  and  $\widehat{S} = \{\xi \mid \mathbf{K}\xi \in S\}$ .

An MKNF interpretation  $M$  induces the partition  $(P, N)$  of  $\text{KA}(\mathcal{K})$  if  $\mathbf{K}\xi \in P$  implies  $(M, M) \models \mathbf{K}\xi$  and  $\mathbf{K}\xi \in N$  implies  $(M, M) \models \text{not } \xi$ .

We now adapt the operators from [10] which allow to draw conclusions from positive hybrid MKNF knowledge bases similarly to the immediate consequence operator for definite logic programs, only that the operators below also are “aware” of possible consequences including the DL knowledge base  $\mathcal{O}$ .

**Definition 3.3.** For  $\mathcal{K}$  a positive nondisjunctive DL-safe hybrid MKNF knowledge base,  $R_{\mathcal{K}}$ ,  $D_{\mathcal{K}}$ , and  $T_{\mathcal{K}}$  are defined on the subsets of  $\text{KA}(\mathcal{K})$  as follows:

$$\begin{aligned}
 R_{\mathcal{K}}(S) &= S \cup \{\mathbf{K}H \mid \mathcal{K} \text{ contains a rule of the form (1) such that } \mathbf{K}A_i \in S \\
 &\text{for each } 1 \leq i \leq n\} \\
 D_{\mathcal{K}}(S) &= \{\mathbf{K}\xi \mid \mathbf{K}\xi \in \text{KA}(\mathcal{K}) \text{ and } \mathcal{O} \cup \widehat{S}_{DL} \models \xi\} \cup \{\mathbf{K}Q(b_1, \dots, b_n) \mid \\
 &\mathbf{K}Q(a_1, \dots, a_n) \in S \setminus S_{DL}, \mathbf{K}Q(b_1, \dots, b_n) \in \text{KA}(\mathcal{K}), \text{ and } \mathcal{O} \cup \widehat{S}_{DL} \models a_i \approx b_i \\
 &\text{for } 1 \leq i \leq n\} \\
 T_{\mathcal{K}}(S) &= R_{\mathcal{K}}(S) \cup D_{\mathcal{K}}(S)
 \end{aligned}$$

The difference to the operators in [10] is that given e.g. only  $a \approx b$  and  $\mathbf{K}Q(a)$  we do not derive  $\mathbf{K}Q(b)$  explicitly but only as a consequence of  $\text{ob}_{\mathcal{K},P}$ .

As in [9], it can be shown that  $T_{\mathcal{K}}$  is monotonic and yields a least fixpoint  $T_{\mathcal{K}} \uparrow \omega$  in the usual manner. We can therefore, in the style of stable models, define a transformation which turns a nondisjunctive hybrid MKNF knowledge base into a positive one allowing to apply the previous operators.

**Definition 3.4.** Let  $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$  be a ground nondisjunctive DL-safe hybrid MKNF knowledge base and  $S \subseteq \text{KA}(\mathcal{K})$ . The MKNF transform  $\mathcal{K}_G/S = (\mathcal{O}, \mathcal{P}_G/S)$  is obtained by  $\mathcal{P}_G/S$  containing all rules  $\mathbf{K}H \leftarrow \mathbf{K}A_1, \dots, \mathbf{K}A_n$  for which there exists a rule  $\mathbf{K}H \leftarrow \mathbf{K}A_1, \dots, \mathbf{K}A_n, \text{not } B_1, \dots, \text{not } B_m$  in  $\mathcal{P}_G$  with  $\mathbf{K}B_j \notin S$  for all  $1 \leq j \leq m$ .

On top of that, an operator yielding the fixpoint of  $T_{\mathcal{K}}$  is defined.

**Definition 3.5.** Let  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$  be a nondisjunctive DL-safe hybrid MKNF knowledge base and  $S \subseteq \text{KA}(\mathcal{K})$ . We define:

$$\Gamma_{\mathcal{K}}(S) = T_{\mathcal{K}_G/S} \uparrow \omega$$

This operator is antitonic (cf. extended technical report), so applying  $\Gamma_{\mathcal{K}}(S)$  twice is a monotonic operation yielding a least fixpoint by the Knaster-Tarski theorem (and dually a greatest one) and we can iterate as follows:  $\Gamma_{\mathcal{K}}^2 \uparrow 0 = \emptyset$ ,  $\Gamma_{\mathcal{K}}^2 \uparrow (n+1) = \Gamma_{\mathcal{K}}^2(\Gamma_{\mathcal{K}}^2 \uparrow n)$ , and  $\Gamma_{\mathcal{K}}^2 \uparrow \omega = \bigcup \Gamma_{\mathcal{K}}^2 \uparrow i$ , and dually  $\Gamma_{\mathcal{K}}^2 \downarrow 0 = \text{KA}(\mathcal{K})$ ,  $\Gamma_{\mathcal{K}}^2 \downarrow (n+1) = \Gamma_{\mathcal{K}}^2(\Gamma_{\mathcal{K}}^2 \downarrow n)$ , and  $\Gamma_{\mathcal{K}}^2 \downarrow \omega = \bigcap \Gamma_{\mathcal{K}}^2 \downarrow i$ . The least and the greatest fixpoint then define the well-founded partition.

**Definition 3.6.** Let  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$  be a nondisjunctive DL-safe hybrid MKNF knowledge base and let  $\mathbf{P}_{\mathcal{K}}, \mathbf{N}_{\mathcal{K}} \subseteq \text{KA}(\mathcal{K})$  with  $\mathbf{P}_{\mathcal{K}} = \Gamma_{\mathcal{K}}^2 \uparrow \omega$  and  $\mathbf{N}_{\mathcal{K}} = \Gamma_{\mathcal{K}}^2 \downarrow \omega$ . Then  $(P_W, N_W) = (\mathbf{P}_{\mathcal{K}} \cup \{\mathbf{K}\pi(\mathcal{O})\}, \text{KA}(\mathcal{K}) \setminus \mathbf{N}_{\mathcal{K}})$  is the well-founded partition of  $\mathcal{K}$ .

*Example 3.2.* Continuing our example, the investigation of the police reveals that the known criminal Max is responsible for the murder, though not being detectable, so we cannot conclude  $\text{Suicide}(\text{jones})$  while  $\mathbf{K}\text{responsible}(\text{max}, \text{jones})$  and  $\mathbf{K}\text{murdered}(\text{jones})$  hold. Unfortunately, the person benefitting from the insurance is the nephew Thomas who many years ago left the country, i.e.  $\mathbf{K}\text{benefits}(\text{thomas}, \text{jones})$ . Computing the well-founded partition yields thus  $\mathbf{K}\text{Pay}(\text{jones})$ , so the company contacts the nephew outside the country. However, they also hire a private detective who finds out that Thomas is max, having altered his personality, i.e. we can add  $\text{thomas} \approx \text{max}$  to the hybrid KB. Due

to  $D_{\mathcal{K}}$  and grounding we now obtain a well-founded partition which contains  $\mathbf{K} \text{responsible}(\text{thomas}, \text{jones})$  and  $\mathbf{K} \text{benefits}(\text{max}, \text{jones})$  being true and the insurance is not paid any longer.

One of the results shown in the extended paper is that the well-founded partition is consistent. Besides that, similarly to stable models, we can compute one fixpoint defining the well-founded partition directly from the other.

**Proposition 3.1.** *Let  $\mathcal{K}$  be a nondisjunctive DL-safe hybrid MKNF knowledge base. Then  $\mathbf{P}_{\mathcal{K}} = \Gamma_{\mathcal{K}}(\mathbf{N}_{\mathcal{K}})$  and  $\mathbf{N}_{\mathcal{K}} = \Gamma_{\mathcal{K}}(\mathbf{P}_{\mathcal{K}})$ .*

Knowing this, we can use  $\Gamma_{\mathcal{K}}$  as an alternative characterization of MKNF models if the considered KB is consistent. It should be noted that in case of an inconsistent hybrid MKNF KB due to the operator  $\mathcal{D}_{\mathcal{K}}$  we obtain a well-founded partition where all modal  $\mathbf{K}$ -atoms are true. I.e., even though we always obtain a well-founded partition for any  $\mathcal{K}$  the result may not be a desired one.

It was also shown that the information derived in the well-founded partition is contained in any MKNF model.

**Theorem 3.1.** *Let  $\mathcal{K}$  be a nondisjunctive DL-safe hybrid MKNF knowledge base,  $M$  an MKNF model of  $\mathcal{K}$  with  $(P, N)$  induced by  $M$ , and  $(P_W, N_W)$  the well-founded partition of  $\mathcal{K}$ . Then  $P_W \subseteq (P \cup \{\mathbf{K} \pi(\mathcal{O})\})$  and  $N_W \subseteq N$ .*

Furthermore, the well-founded partition yields a model in the three-valued framework we defined in the previous subsection.

**Theorem 3.2.** *Let  $\mathcal{K}$  be a consistent nondisjunctive DL-safe hybrid MKNF KB and  $(\mathbf{P}_{\mathcal{K}} \cup \{\mathbf{K} \pi(\mathcal{O})\}, \mathbf{KA}(\mathcal{K}) \setminus \mathbf{N}_{\mathcal{K}})$  be the well-founded partition of  $\mathcal{K}$ . Then  $(I_P, I_N) \models \pi(\mathcal{K})$  where  $I_P = \{I \mid I \models \text{ob}_{\mathcal{K}, \mathbf{P}_{\mathcal{K}}}\}$  and  $I_N = \{I \mid I \models \text{ob}_{\mathcal{K}, \mathbf{N}_{\mathcal{K}}}\}$ .*

One of the open questions mentioned in [10] was that MKNF models are not compatible with the well-founded model for logic programs. Our approach, regarding knowledge bases just consisting of rules, does coincide with the well-founded model for the corresponding (normal) logic program.

Finally, though not providing here a detailed study of complexity issues we can recall from [9], assuming that entailment of first-order formulas encountered while computing  $T_{\mathcal{K}}$  is decidable in  $\mathcal{C}$ , that the data complexity of computing  $T_{\mathcal{K}}$  is in  $\mathbf{P}^{\mathcal{C}}$  (for positive nondisjunctive programs). Since we just apply the same operator  $n$ -times we remain in the same complexity class while the data complexity for reasoning with MKNF models in nondisjunctive programs is shown to be  $\mathcal{E}^{\mathbf{P}^{\mathcal{C}}}$  where  $\mathcal{E} = \mathbf{NP}$  if  $\mathcal{C} \subseteq \mathbf{NP}$ , and  $\mathcal{E} = \mathcal{C}$  otherwise. Thus computing the well-founded partition ends up in a strictly smaller complexity class than deriving the MKNF models.

## 4 Conclusions and Future Work

We have continued the work on hybrid MKNF knowledge bases providing an alternating fixpoint restricted to nondisjunctive rules. We basically achieve better complexity results by having only one model which is semantically weaker

than any MKNF model defined in [10] but bottom-up computable. The well-founded semantics is not only a sound approximation of any MKNF model but a partition of modal atoms which can seamlessly be integrated in the reasoning algorithms presented for MKNF models in [10] thus reducing the difficulty of guessing the ‘right’ model. Future work shall include the extension to disjunctive rules, a study on top-down querying procedures, and further investigations on the well-founded model in the three-valued framework.

## References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. J. de Bruijn, T. Eiter, A. Polleres, and H. Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge-base combination. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, Hyderabad, India, January 6–12 2007. AAAI Press.
3. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. In D. Dubois, C. Welty, and M.-A. Williams, editors, *KR’04*, pages 141–151. AAAI Press, 2004.
4. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-founded semantics for description logic programs in the semantic web. In G. Antoniou and H. Boley, editors, *RuleML’04*, pages 81–97. Springer, LNCS, 2004.
5. M. Fitting. The family of stable models. *Journal of Logic Programming*, 17(2/3&4):197–225, 1993.
6. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *ICLP*. MIT Press, 1988.
7. S. Heymans, D. V. Nieuwenborgh, and D. Vermeir. Guarded open answer set programming. In C. Baral, G. Greco, N. Leone, and G. Terracina, editors, *LPNMR’05*, pages 92–104. Springer, LNAI, 2005.
8. V. Lifschitz. Nonmonotonic databases and epistemic queries. In *IJCAI’91*, pages 381–386, 1991.
9. B. Motik and R. Rosati. Closing semantic web ontologies. Technical report, University of Manchester, UK, 2006.
10. B. Motik and R. Rosati. A faithful integration of description logics with logic programming. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 477–482, Hyderabad, India, January 6–12 2007. AAAI Press.
11. R. Rosati. Reasoning about minimal belief and negation as failure. *J. of Artificial Intelligence Research*, 11:277–300, 1999.
12. R. Rosati. DL+Log: A tight integration of description logics and disjunctive datalog. In P. Doherty, J. Mylopoulos, and C. Welty, editors, *KR’06*. AAAI Press, 2006.
13. A. van Gelder. The alternating fixpoint of logic programs with negation. In *Principles of Database Systems*, pages 1–10. ACM Press, 1989.
14. A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

## Conjunctive Queries for $\mathcal{EL}$ with Role Composition

Markus Krötzsch and Sebastian Rudolph

Institute AIFB, Universität Karlsruhe, Germany  
 {mak|sru}@aifb.uni-karlsruhe.de

**Abstract.**  $\mathcal{EL}^{++}$  is a rather expressive description logic (DL) that still admits polynomial time inferencing for many reasoning tasks. Conjunctive queries are an important means for expressive querying of DL knowledge bases. We address the problem of computing conjunctive query entailment for  $\mathcal{EL}^{++}$  knowledge bases. As it turns out, querying unrestricted  $\mathcal{EL}^{++}$  is actually undecidable, but we identify restrictions under which query answering becomes decidable and even tractable. We give precise characterisations of schema, query, and combined complexity. To the best of our knowledge, the presented algorithm is the first to answer conjunctive queries in a DL that admits complex role inclusion axioms.

### 1 Introduction

Conjunctive queries originated from research in relational databases [1], and, more recently, have been considered for expressive description logics (DLs) as well [2–6]. Algorithms for answering (extensions of) conjunctive queries in the expressive DL *SHIQ* have been discussed in [3, 4], but the first algorithm that supports queries for transitive roles was presented only very recently [6].

Modern DLs, however, allow for complex role inclusion axioms that encompass role composition and further generalise transitivity. To the best of our knowledge, no algorithms for answering conjunctive queries in those cases have been proposed yet. A relevant logic of this kind is *SROIQ* [7], the basic DL considered for OWL 1.1.<sup>1</sup> Another interesting DL that admits complex role inclusions is  $\mathcal{EL}^{++}$  [8, 9], which has been proposed as a rather expressive logic for which many inference tasks can be computed in polynomial time. In this paper, we present a novel algorithm for answering conjunctive queries in  $\mathcal{EL}^{++}$ , which is based on an automata-theoretic formulation of complex role inclusion axioms that was also found useful in reasoning with *SROIQ* [10, 7].

Our algorithm in particular allows us to derive a number of complexity results related to conjunctive query answering in  $\mathcal{EL}^{++}$ . We first show that conjunctive queries in  $\mathcal{EL}^{++}$  are undecidable in general, and identify the  $\mathcal{EL}^{++}$ -fragment of *SROIQ* as an appropriate decidable sub-DL. Under some related restrictions of role inclusion axioms, we show that conjunctive query answering in general is PSPACE-complete. Query answering for fixed knowledge bases (query complexity) is shown to be NP-complete, whereas for fixed queries (schema complexity) it is merely P-complete.

After introducing some preliminaries in Section 2, we present a general undecidability result for conjunctive queries  $\mathcal{EL}^{++}$  in Section 3. Thereafter, we present a modified, automata-based inferencing procedure for  $\mathcal{EL}^{++}$  in Section 4. This will be the basis for

<sup>1</sup> <http://owl1.1.cs.manchester.ac.uk/>

the algorithm for checking entailment of conjunctive queries as presented in Section 5, which operates on a fragment of  $\mathcal{EL}^{++}$  for which this problem is decidable. Finally, we derive a number of complexity results related to conjunctive queries in  $\mathcal{EL}^{++}$ . Proofs are usually omitted in the extended abstract for reasons of space. They are found in the accompanying technical report [11].

## 2 Preliminaries

We assume the reader to be familiar with the basic notions of description logics (DLs). The DLs that we will encounter in this paper are  $\mathcal{EL}^{++}$  [8] and, marginally,  $\mathcal{SROIQ}$  [7]. A *signature* of DL consists of a finite set of *role names*  $\mathbf{R}$ , a finite set of *individual names*  $\mathbf{I}$ , and a finite set of *concept names*  $\mathbf{C}$ , and we will use this notation throughout the paper.  $\mathcal{EL}^{++}$  supports *nominals*, which we conveniently represent as follows: for any  $a \in \mathbf{I}$ , there is a concept  $\{a\} \in \mathbf{C}$  such that  $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$  (for any interpretation  $\mathcal{I}$ ). As shown in [8], any  $\mathcal{EL}^{++}$  knowledge base is equivalent to one in *normal form*, only containing the following axioms:

$$\begin{array}{llll} \text{TBox: } & A \sqsubseteq C & A \sqcap B \sqsubseteq C & A \sqsubseteq \exists R.C \quad \exists R.A \sqsubseteq C \\ \text{RBox: } & R \sqsubseteq T & R \circ S \sqsubseteq T & \end{array}$$

where  $A, B \in \mathbf{C} \cup \{\top\}$ ,  $C \in \mathbf{C} \cup \{\perp\}$ , and  $R, S, T \in \mathbf{R}$ . Note that ABox statements of the forms  $C(a)$  and  $R(a, b)$  are internalised into the TBox. The standard model theoretic semantics of  $\mathcal{EL}^{++}$  can be found in [8]. Unless otherwise specified, the letters  $C, D, E$  in the remainder of this work always denote (arbitrary) concept names, and the letters  $R, S$  denote (arbitrary) role names. We do not consider concrete domains in this paper, but are confident that our results can be extended accordingly.

For *conjunctive queries*, we largely adopt the notation of [6] but directly allow for individuals in queries. Let  $\mathbf{V}$  be a countable set of *variable names*. Given elements  $x, y \in \mathbf{V} \cup \mathbf{I}$ , a *concept atom* (*role atom*) is an expression  $C(x)$  with  $C \in \mathbf{C}$  ( $R(x, y)$  with  $R \in \mathbf{R}$ ). A *conjunctive query*  $q$  is a set of concept and role atoms, read as a conjunction of its elements. By  $\text{Var}(q)$  we denote the set of variables occurring in  $q$ . Consider an interpretation  $\mathcal{I}$  with domain  $\Delta^{\mathcal{I}}$ , and a function  $\pi : \text{Var}(q) \cup \mathbf{I} \rightarrow \Delta^{\mathcal{I}}$  such that  $\pi(a) = a^{\mathcal{I}}$  for all  $a \in \mathbf{I}$ . We define

$$\mathcal{I}, \pi \models C(x) \text{ if } \pi(x) \in C^{\mathcal{I}}, \quad \text{and} \quad \mathcal{I}, \pi \models R(x, y) \text{ if } (\pi(x), \pi(y)) \in R^{\mathcal{I}}.$$

If there is some  $\pi$  such that  $\mathcal{I}, \pi \models A$  for all atoms  $A \in q$ , we write  $\mathcal{I} \models q$  and say that  $\mathcal{I}$  *models*  $q$ . A knowledge base  $KB$  entails  $q$  if all models of  $KB$  entail  $q$ .

## 3 Conjunctive Queries in $\mathcal{EL}^{++}$

We first investigate the complexity of conjunctive queries in general  $\mathcal{EL}^{++}$  as defined in [8]. The following result might be mildly surprising, but is in fact closely related to similar results for logics with complex role expressions (see, e.g., [12]).

**Theorem 1.** *For an  $\mathcal{EL}^{++}$  knowledge base  $KB$  and a conjunctive query  $q$ , the entailment problem  $KB \models q$  is undecidable. Likewise, checking class subsumptions in  $\mathcal{EL}^{++}$  extended with inverse roles or role conjunctions is undecidable, even if those operators occur only in the concepts whose subsumption is checked.*

*Proof.* Intuitively, the result holds since RBoxes can encode context-free languages, the intersection of which can then be checked with conjunctive queries/inverse roles/role conjunctions. This problem is undecidable. The proof in [11] uses an even simpler reduction of the undecidable Post correspondence problem.  $\square$

Clearly, arbitrary role compositions are overly expressive when aiming for a decidable (or even tractable) logic that admits conjunctive queries. We thus restrict our attention to the fragment of  $\mathcal{EL}^{++}$  that is in the (decidable) description logic  $\mathcal{SROIQ}$  [7], and investigate its complexity with respect to conjunctive query answering.

**Definition 1.** An  $\mathcal{EL}^{++}$  RBox in normal form is regular if there is a strict partial order  $<$  on  $\mathbf{R}$  such that, for all role inclusion axioms  $R_1 \sqsubseteq S$  and  $R_1 \circ R_2 \sqsubseteq S$ , we find  $R_i < S$  or  $R_i = S$  ( $i = 1, 2$ ). An  $\mathcal{EL}^{++}$  knowledge base is regular if it has a regular RBox.

The existence of  $<$  ensures that the role hierarchy does not contain cyclic dependencies other than through direct recursion of a single role.

#### 4 Reasoning Automata for $\mathcal{EL}^{++}$

In this section, we describe the construction of an automaton that encodes certain concept subsumptions entailed by an  $\mathcal{EL}^{++}$  knowledge base. The automaton itself is closely related to the reasoning algorithm given in [8], but the representation of entailments via nondeterministic finite automata (NFA) will be essential for the query answering algorithm in the following section. We describe an NFA  $\mathcal{A}$  as a tuple  $(Q_{\mathcal{A}}, \Sigma_{\mathcal{A}}, \delta_{\mathcal{A}}, i_{\mathcal{A}}, F_{\mathcal{A}})$ , where  $Q_{\mathcal{A}}$  is a finite set of states,  $\Sigma_{\mathcal{A}}$  is a finite alphabet,  $\delta_{\mathcal{A}} : Q_{\mathcal{A}} \times Q_{\mathcal{A}} \rightarrow 2^{\Sigma_{\mathcal{A}}}$  is a transition function that maps pairs of states to sets of alphabet symbols,<sup>2</sup>  $i_{\mathcal{A}}$  is the initial state, and  $F_{\mathcal{A}}$  is a set of final states.

Consider an  $\mathcal{EL}^{++}$  knowledge base  $KB$ . Given a concept name  $A \in \mathbf{C}$ , we construct an NFA  $\mathcal{A}_{KB}(A) = (Q, \Sigma, \delta, i, F)$  that computes superconcepts of  $A$ , where we omit the subscript if  $KB$  is clear from the context. Set  $Q = F = \mathbf{C} \cup \{\top\}$ ,  $\Sigma = \mathbf{C} \cup \mathbf{R} \cup \{\top, \perp\}$ , and  $i = A$ . The transition function  $\delta$  is initially defined as  $\delta(C, C) := \{C, \top\}$  (for all  $C \in Q$ ), and extended iteratively by applying the rules in Table 1. The rules correspond to completion rules in [8, Table 2], though the conditions for (CR6) are slightly relaxed, fixing a minor glitch in the original algorithm.

It is easy to see that the rules of Table 1 can be applied at most a polynomial number of times. The words accepted by  $\mathcal{A}(A)$  are strings of concept and role names. For each such word  $w$  we inductively define a concept expression  $C_w$  as follows:

- if  $w$  is empty, then  $C_w = \top$ ,
- if  $w = Rv$  for some  $R \in \mathbf{R}$  and word  $v$ , then  $C_w = \exists R.(C_v)$ ,
- if  $w = Cv$  for some  $C \in \mathbf{C}$  and word  $v$ , then  $C_w = C \sqcap C_v$ .

For instance, the word  $CRDES$  translates into  $C_{CRDES} = C \sqcap \exists R.(D \sqcap E \sqcap \exists S.\top)$ . Based on the close correspondence of the above rules to the derivation rules in [8], we can now establish the main correctness result for the automaton  $\mathcal{A}(A)$ .

<sup>2</sup> A possibly more common definition is to map pairs of states and symbols to sets of states, but the above is more convenient for our purposes.

**Table 1.** Completion rules for constructing an NFA from an  $\mathcal{EL}^{++}$  knowledge base  $KB$ .

- (CR1) If  $C' \in \delta(C, C)$ ,  $C' \sqsubseteq D \in KB$ , and  $D \notin \delta(C, C)$  then  $\delta(C, C) := \delta(C, C) \cup \{D\}$ .
- (CR2) If  $C_1, C_2 \in \delta(C, C)$ ,  $C_1 \sqcap C_2 \sqsubseteq D \in KB$ , and  $D \notin \delta(C, C)$  then  $\delta(C, C) := \delta(C, C) \cup \{D\}$ .
- (CR3) If  $C' \in \delta(C, C)$ ,  $C' \sqsubseteq \exists R.D \in KB$ , and  $R \notin \delta(C, D)$  then  $\delta(C, D) := \delta(C, D) \cup \{R\}$ .
- (CR4) If  $R \in \delta(C, D)$ ,  $D' \in \delta(D, D)$ ,  $\exists R.D' \sqsubseteq E \in KB$ , and  $E \notin \delta(C, C)$  then  $\delta(C, C) := \delta(C, C) \cup \{E\}$ .
- (CR5) If  $R \in \delta(C, D)$ ,  $\perp \in \delta(D, D)$ , and  $\perp \notin \delta(C, C)$  then  $\delta(C, C) := \delta(C, C) \cup \{\perp\}$ .
- (CR6) If  $\{a\} \in \delta(C, C) \cap \delta(D, D)$ , and there are states  $C_1, \dots, C_n$  such that
  - $C_1 \in \{C, \top, A\} \cup \{\{b\} \mid b \in \mathbf{I}\}$ ,
  - $\delta(C_j, C_{j+1}) \neq \emptyset$  for all  $j = 1, \dots, n-1$ ,
 as well as  $C_n = D$ , and  $\delta(D, D) \not\subseteq \delta(C, C)$  then  $\delta(C, C) := \delta(C, C) \cup \delta(D, D)$ .
- (CR7) If  $R \in \delta(C, D)$ ,  $R \sqsubseteq S$ , and  $S \notin \delta(C, D)$  then  $\delta(C, D) := \delta(C, D) \cup \{S\}$ .
- (CR8) If  $R_1 \in \delta(C, D)$ ,  $R_2 \in \delta(D, E)$ ,  $R_1 \circ R_2 \sqsubseteq S$ , and  $S \notin \delta(C, E)$  then  $\delta(C, E) := \delta(C, E) \cup \{S\}$ .

**Theorem 2.** Consider a knowledge base  $KB$ , concept  $A$ , and NFA  $\mathcal{A}(A)$  as above, and let  $w$  be some word over the associated alphabet. Then  $KB \models A \sqsubseteq C_w$  iff one of the following holds:

- $\mathcal{A}(A)$  accepts the word  $w$ , or
- there is a transition  $\perp \in \delta(C, C)$  where  $C = \top$ ,  $C = A$ , or  $C = \{a\}$  for some individual  $a$ .

In particular,  $\mathcal{A}(A)$  can be used to check all subsumptions between  $A$  and some atomic concept  $B$ .

The second item of the theorem addresses the cases where  $A$  is inferred to be empty (i.e. inconsistent) or where the whole knowledge base is empty. While the above yields an alternative formulation of the  $\mathcal{EL}^{++}$  reasoning algorithm presented in [8], it has the advantage that it also encodes all *paths* within the inferred models. This will be essential for our results in the next section where we will use the following convenient definition.

**Definition 2.** Consider a knowledge base  $KB$ , concepts  $A, B \in \mathbf{C}$ , and the NFA  $\mathcal{A}(A) = (Q, \Sigma, \delta, i, F)$ . The automaton  $\mathcal{A}_{KB}(A, B)$  (or just  $\mathcal{A}(A, B)$ ) is defined as  $(Q, \mathbf{R}, \delta, i, F')$  where  $F' = \emptyset$  whenever  $\perp \in \delta(A, A)$ , and  $F' = \{B\}$  otherwise.

$\mathcal{A}(A, B)$  obviously accepts all words  $R_1, \dots, R_n$  such that  $A \sqsubseteq \exists R_1(\dots \exists R_n.B \dots)$  is a consequence of  $KB$ , with the border case where  $n = 0$  and  $KB \models A \sqsubseteq B$ . Moreover, the language accepted by the NFA is empty whenever  $A \sqsubseteq \perp$  has been inferred.

## 5 Deciding Conjunctive Queries for $\mathcal{EL}$

In this section, we present a nondeterministic algorithm that decides the entailment of a query  $q$  with respect to some knowledge base  $KB$ . This is done by constructing a so-called *proof graph* which establishes, for any interpretation  $\mathcal{I}$  of  $KB$ , the existence of a suitable function  $\pi$  that shows query entailment.

Formally, a proof graph is a tuple  $(N, L, E)$  consisting of a set of nodes  $N$ , a labelling function  $L : N \rightarrow \mathbf{C} \cup \{\top\}$ , and a partial transition function  $E : N \times N \rightarrow \mathbf{A}$ , where  $\mathbf{A}$  is the set of all NFA over the alphabet  $\mathbf{C} \cup \{\top, \perp\} \cup \mathbf{R}$ . The nodes of the proof graph

are abstract representations of elements in the domain of some model of  $KB$ . The labels assign a concept to each node, and our algorithm ensures that the represented element is necessarily contained in the interpretation of this concept. Intuitively, the label encodes all relevant concept information about one node. This is only possible since (1)  $KB$  is in normal form and thus supplies concept names for all composite concept expressions such as conjunctions, and (2)  $\mathcal{EL}^{++}$  does not allow inverse roles or number restrictions that could be used to infer further information based on the relationship of an element to elements in the model. Finally, the transition function encodes paths in each model, which provide the basis for inferencing about role relationships between elements. It would be possible to adopt a more concrete representation for role paths (e.g. by guessing a single path), but our formulation reduces nondeterminism and eventually simplifies our investigation of algorithmic complexity.

The automaton of Definition 2 encodes concept subsumptions based on TBox and RBox. For deciding query entailment we also require automata that represent the content of the RBox.

**Proposition 1.** *Given a regular  $\mathcal{EL}^{++}$  RBox, and some role  $R \in \mathbf{R}$ , there is an NFA  $\mathcal{A}(R)$  over the alphabet  $\mathbf{R}$  which accepts a word  $R_1 \dots R_n$  iff  $R_1 \circ \dots \circ R_n \sqsubseteq R$  is a consequence of every  $\mathcal{EL}^{++}$  knowledge base with the given RBox.*

*Proof sketch.* One possible construction for the required automaton is discussed in [7]. Intuitively, the RBox can be understood as a grammar for a regular language, for which an automaton can be constructed in a canonical way.  $\square$

The above construction might be exponential for some RBoxes. In [10], restrictions have been discussed that prevent this blow-up, leading to NFA of only polynomial size w.r.t. the RBox. Accordingly, an RBox is *simple* whenever, for all axioms of the form  $R_1 \circ S \sqsubseteq S$ ,  $S \circ R_2 \sqsubseteq S$ , the RBox does not contain a common subrole  $R$  of  $R_1$  and  $R_2$  for which there is an axiom of the form  $R \circ S' \sqsubseteq R'$  or  $S' \circ R \sqsubseteq R'$ . We will usually consider only such simple RBoxes whenever the size of the constructed automata matters.

We are now ready to present the algorithm. It proceeds in various consecutive steps:

*Query factorisation.* The algorithm nondeterministically selects a variable  $x \in \text{Var}(q)$  and some element  $e \in \text{Var}(q) \cup \mathbf{I}$ , and replaces all occurrences of  $x$  in  $q$  with  $e$ . This step can be executed an arbitrary number of times (including zero).

*Proof graph initialisation.* The proof graph  $(N, L, E)$  is initialised by setting  $N := \{\top\} \cup \mathbf{I} \cup \text{Var}(q)$ .  $L$  is initialised by  $L(\top) := \top$ ,  $L(a) := \{a\}$  for each  $a \in \mathbf{I}$ . For each  $x \in \text{Var}(q)$ , the algorithm selects a label  $L(x) \in \mathbf{C} \cup \{\top\}$ . Finally,  $E$  is initialised by setting  $E(n, a) := \mathcal{A}(L(n), L(a))$  for each  $n \in N$ ,  $a \in \mathbf{I}$ . A node  $m \in N$  is *reachable* if there is some node  $n \in N$  such that  $E(n, m)$  is defined, and *unreachable* otherwise. Clearly, all nominal nodes are reachable by the initialisation of  $E$ . Now as long as there is some unreachable node  $x \in \text{Var}(q)$ , the algorithm nondeterministically selects one such  $x$  and some node  $n \in N$  that is either reachable or  $\top$ , and sets  $E(n, x) := \mathcal{A}(L(n), L(x))$ . After this procedure, the graph  $(N, L, E)$  is such that all nodes other than  $\top$  are reachable. Finally, the algorithm checks whether any of the automata  $E(n, m)$  ( $n, m \in N$ ) accepts the empty language, and aborts with failure if this is the case.

*Checking concept entailment.* For all concept atoms  $C(n) \in q$ , the algorithm checks whether  $L(n) \models C$  with respect to  $KB$ .

For the remaining steps of the algorithm, some preliminary definitions and observations are needed. The automata  $E(n, m)$  of the proof graph represent chains of existential role restrictions that exist within any model. If  $m \in \text{Var}(q)$ , then the automaton encodes (all possible) ways of constructing an element that belongs to the interpretation of  $L(m)$  in each model. The role automata  $\mathcal{A}(R)$  in turn encode possible chains of roles that suffice to establish role  $R$  along some such path. To show that an atom  $R(n, m)$  is entailed, one thus merely has to check whether the automata  $E(n, m)$  and  $\mathcal{A}(R)$  have a non-empty intersection. Two issues must be taken into account. Firstly, not every pair of nodes is linked via some edge  $E(n, m)$ , so one might have to look for a longer path and check non-emptiness of its intersection with  $\mathcal{A}(R)$ . Secondly, there might be many role atoms that affect the path between  $n$  and  $m$ , and all of them must be satisfied concurrently. Hence, one either needs to check intersections of many automata concurrently, or one needs to retain the restrictions imposed by one (processed) role atom before treating further atoms. The following is easy to prove.

**Proposition 2.** *For every pair of nodes  $n, m \in N$ , there either is a unique shortest connecting path  $n_0 = n, n_1, \dots, n_k = m$  with  $n_i \in N$  and  $E(n_i, n_{i+1})$  defined, or there is no connecting path at all. If it exists, this path can be computed by a deterministic algorithm in polynomial time.*

Now any role atom in the query should span over some existing path, and we need to check whether this path suffices to establish the required role. To do this, we nondeterministically split the role automaton into parts that are distributed along the path.

**Definition 3.** *Consider an NFA  $\mathcal{A} = (Q, \Sigma, \delta, i, \{f\})$ . A split of  $\mathcal{A}$  into  $k$  parts is given by NFA  $\mathcal{A}_1, \dots, \mathcal{A}_k$  that are constructed as follows. For every  $j = 0, \dots, k$ , there is some state  $q_j \in Q$  such that  $q_0 = i$  and  $q_k = f$ . The NFA  $\mathcal{A}_j$  has the form  $(Q, \Sigma, \delta, q_{j-1}, \{q_j\})$ .*

It is easy to see that, if each split automaton  $\mathcal{A}_j$  accepts some word  $w_j$ , we find that  $w_1 \dots w_k$  is accepted by  $\mathcal{A}$ . Likewise, any word accepted by  $\mathcal{A}$  is also accepted in this sense by split of  $\mathcal{A}$ . Since the combination of any split in general accepts less words than  $\mathcal{A}$ , splitting an NFA usually involves some don't-know nondeterminism. We can now proceed with the steps of the algorithm.

*Splitting of role automata.* For each role atom  $R(n, m)$  within the query, the algorithm computes the shortest path  $n = n_0, \dots, n_k = m$  from  $n$  to  $m$ , or aborts with failure if no such path exists. Next, it splits the NFA  $\mathcal{A}(R)$  into  $k$  automata  $\mathcal{A}(R(n, m), n_0, n_1), \dots, \mathcal{A}(R(n, m), n_{k-1}, n_k)$ , and aborts with failure if any of the split automata is empty.

*Check role entailment.* Finally, for each  $n, m \in N$  with  $E(n, m)$  defined, the algorithm executes the following checks:

- If  $m \in \mathbf{I}$ , it checks whether the intersection of the edge automaton  $E(n, m)$  with any single split automaton of the form  $\mathcal{A}(F, n, m)$  is empty.
- If  $m \in \text{Var}(q)$ , it checks whether the simultaneous intersection of the edge automaton  $E(n, m)$  with all split automata of the form  $\mathcal{A}(F, n, m)$  is empty.

If all those intersections have been shown to be non-empty, the algorithm confirms the entailment of the query (we say that the algorithm *accepts* the query). Otherwise it terminates with failure.

Formal proofs of soundness and completeness of the algorithm are given in [11]. Soundness is established by showing that acceptance implies the existence of a match for the query w.r.t. any model of  $KB$ . Indeed, a suitable section of the model can be found by retracting the algorithm's construction of the proof graph to find suitable domain element, and by noting that the properties that the algorithm has inferred ensure that all conditions imposed by the query are satisfied for this match. For completeness, a canonical model is constructed and this model is used to guide the choices of the algorithm to successful acceptance. Similar to the constructed proof graph, the canonical model exposes a certain local “tree-likeness”: while the presence of nominals prevents the model from being a tree, all cycles in the model must involve a named constant (and thus a nominal). This fact is exploited by the algorithm in its construction of shortest paths and allows us to focus on only one unique such path for showing the entailment of all role atoms in the query.

## 6 Complexity of Query Answering for $\mathcal{EL}^{++}$

Finally, we harvest a number of complexity results from the algorithm of Section 5.

**Theorem 3.** *The complexities of conjunctive query entailment for regular  $\mathcal{EL}^{++}$  knowledge bases – estimated w.r.t. the size of the variable input – are shown in the following table. Whenever the  $RBox$  is variable, we assume that it is simple.*

	Variable parts:				Complexity
	Query	$RBox$	$TBox$	$ABox$	
Combined complexity	×	×	×	×	$\text{PSPACE-complete}$
Query complexity	×				$\text{NP-complete}$
Schema complexity		×	×	×	$\text{P-complete}$
Data complexity				×	$\text{P-complete}$

*Proof.* The hardness proofs detailed in [11] apply known hardness results for the data-complexity of instance checking in fragments of  $\mathcal{EL}$  [13], evaluation of single Data-log clauses ( $\text{NP-complete}$ , [14]), and emptiness of the intersection of finite automata ( $\text{PSPACE-complete}$ , [15]). For containment in the respective complexity classes, one carefully estimates complexity boundaries for the algorithm of Section 5.  $\square$

We remark that the above results are quite generic, and can be established for many other DLs. Especially,  $\text{NP-hardness}$  w.r.t. knowledge base size can be shown for any logic that admits an  $ABox$ , whereas  $\text{PSPACE-hardness}$  of the combined problem follows whenever the DL additionally admits role composition and existential role restrictions.

## 7 Conclusion

We have proposed a novel algorithm for answering conjunctive queries in  $\mathcal{EL}^{++}$  KBs, which is worst-case optimal under various assumptions. Apparently, this also constitutes the first inference procedure for conjunctive queries in a DL that supports complex

role inclusions (including composition). Showing undecidability of conjunctive queries for unrestricted  $\mathcal{EL}^{++}$ , we illustrated that combining role atoms in queries and complex role inclusion axioms can make reasoning significantly more difficult.

A compact automata-based representation of role chains *and* (parts of) models allowed us to establish polynomial bounds for inferencing in various cases, thus identifying querying scenarios that are still tractable for  $\mathcal{EL}^{++}$ . Conjunctive queries inherently introduce some nondeterminism, but automata can conveniently represent sets of possible solutions instead of considering each of them separately. We therefore believe that central methods from the presented algorithm can be a basis for actual implementation that introduces additional heuristics to ameliorate nondeterminism.

## References

1. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In Hopcroft, J.E., Friedman, E.P., Harrison, M.A., eds.: Proc. STOC'77, ACM Press (1977) 77–90
2. Horrocks, I., Sattler, U., Tessaris, S., Tobies, S.: How to decide query containment under constraints using a description logic. In Parigot, M., Voronkov, A., eds.: Proc. LPAR 2000. Volume 1955 of LNAI., Springer (2000) 326–343
3. Hustadt, U., Motik, B., Sattler, U.: A decomposition rule for decision procedures by resolution-based calculi. In: Proc. LPAR 2004. (2005) 21–35
4. Ortiz, M.M., Calvanese, D., Eiter, T.: Data complexity of answering unions of conjunctive queries in *SHIQ*. In: Proc. DL 2006, CEUR Electronic Workshop Proceedings (2006)
5. Ortiz, M.M., Calvanese, D., Eiter, T.: Characterizing data complexity for conjunctive query answering in expressive description logics. In: Proc. AAAI'06. (2006)
6. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic *SHIQ*. In: Proc. IJCAI-07, Hyderabad, India (2007)
7. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SROIQ*. In: Proc. KR2006, AAAI Press (2006) 57–67
8. Baader, F., Brandt, S., Lutz, C.: Pushing the  $\mathcal{EL}$  envelope. In: Proc. IJCAI-05, Edinburgh, UK, Morgan-Kaufmann Publishers (2005)
9. Krisnadhi, A., Lutz, C.: Data complexity in the  $\mathcal{EL}$  family of DLs. In: Proc. DL 2007, CEUR Electronic Workshop Proceedings (2007)
10. Horrocks, I., Sattler, U.: Decidability of *SHIQ* with complex role inclusion axioms. In: Proc. IJCAI-03, Acapulco, Mexico, Morgan-Kaufmann Publishers (2003) 343–348
11. Krötzsch, M., Rudolph, S.: Conjunctive queries for  $\mathcal{EL}$  with role composition. Technical report, Universität Karlsruhe (TH), Germany (2007) Available at [http://www.aifb.uni-karlsruhe.de/Publikationen/showPublikation?publ\\_id=1463](http://www.aifb.uni-karlsruhe.de/Publikationen/showPublikation?publ_id=1463).
12. Wessel, M.: Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In: Proc. DL 2001. (2001)
13. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. KR 2006. (2006) 260–270
14. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. *ACM Computing Surveys* **33** (2001) 374–425
15. Kozen, D.: Lower bounds for natural proof systems. In: Proc. 18th Symp. on the Foundations of Computer Science. (1977) 254–266

## Semantic difference in $\mathcal{ALN}$

Alain Léger<sup>1</sup>, Christophe Rey<sup>2</sup>, and Farouk Toumani<sup>2</sup>

<sup>1</sup> France Télécom R&D, Rennes, France

`alain.leger@francetelecom.com`

<sup>2</sup> LIMOS, CNRS, Université Blaise Pascal, Clermont-Ferrand, France

`christophe.rey@univ-bpclermont.fr, ftoumani@isima.fr`

**Abstract.** We study the semantic difference operator defined in [16] for  $\mathcal{ALN}$ . We give a polynomial-time algorithm to compute it. We compare it with the syntactic difference operator defined in [7], for which we also give a polynomial-time algorithm for  $\mathcal{ALN}$ .

### 1 Introduction

Among non standard reasonings for description logics, the subtraction or difference operation addresses the problem of computing descriptions that are part of one concept and not part of another one. Two definitions of the difference operation have been used in the literature, namely the *semantic* and the *syntactic* difference. The semantic difference between two descriptions  $B$  and  $A$ , noted  $B - A$ , has been defined in [16] as the most general descriptions  $C$  such that  $C \sqcap A \equiv B$ . Two general kinds of applications of the semantic difference have been mentioned in [16]: removing specific information from a description and description decomposition. Such an inference mechanisms can, for example, be useful in tutorial systems that have to explain concepts to users. Recently, motivated by two applications in the areas of *semantic web service discovery* and *querying e-catalog communities*, we have used the semantic difference operation to define a new more flexible concept rewriting approach, called *best covering concepts* using terminologies [10, 4, 5].

However, the difference operation suffers from some drawbacks. First, in the languages that provide full negation the difference  $B - A$  is always equal to  $\neg(A \sqcap \neg B)$ , a description which is not very useful in practice [16, 7]. Second, in many description logics, e.g.,  $\mathcal{ALN}$ , the difference operations is not semantically unique, i.e., it yields to a set of descriptions that are not semantically equivalent to each other. In this case, the semantic difference is a set-valued operation and gives rise to two main difficulties: (i) computation of the semantic difference, and (ii) manipulation of the results in practical cases.

To cope with these limitations, a *syntactic* difference operation has been proposed in [7]. In this case, the difference  $B - A$  yield to a syntactically minimal<sup>3</sup> description  $C$  such that  $C \sqcap A \equiv B \sqcap A$ . The syntactic difference has been used in [7] to measure the accuracy of an approximation of a given concept.

<sup>3</sup> That is a description containing less syntactic redundancies possible.

In contrast with its semantic version, the syntactic difference operation always produce a unique description and hence it is usually easier to compute. However, as it can be expected, the result of a syntactic operation is less accurate than the one produced by the semantic one. The overall conclusion is that the choice of a difference operator, i.e., semantic v.s. syntactic, is application dependent.

In this paper, we investigate the problem of computing the difference descriptions in the context of the  $\mathcal{ALN}$  language. The motivation is the extension of our previous work on the concept covering problem [10] to  $\mathcal{ALN}$ . We first recall some useful results about  $\mathcal{ALN}$  in section 2. Then, in section 3, we provide a polynomial-time algorithm to compute the semantic difference in  $\mathcal{ALN}$ . In section 4, we provide another polynomial-time algorithm to compute the syntactic difference in  $\mathcal{ALN}$ , then we compare both operators, and justify why we have chosen the semantic one in the study of concept covers. We conclude in section 5.

## 2 Preliminaries

We assume the reader familiar with the  $\mathcal{ALN}$  description logic. Let  $C$  an  $\mathcal{ALN}$ -concept description. The normal form of  $C$ , noted  $\hat{C}$ , is obtained as usual by applying the set of normalization rules given in [2, 12] that aim at removing redundancies and making explicit all implicit inconsistencies due to interactions between constructors. For more conveniency, we define in this paper a set-oriented representation of (normalized)  $\mathcal{ALN}$ -descriptions. To this end, we introduce below the notions of  $\mathcal{ALN}$ -clause and clausal form.

Let  $P$  be an atomic concept, the negation of an atomic concept or a number restriction. An  $\mathcal{ALN}$ -clause, or more simply a clause, is either: (i) a description  $P$ , or (ii) a description of the form  $\forall R_1.(\dots(\forall R_n.P))$ . In the following, a clause of the form  $\forall R_1.(\dots(\forall R_n.P)\dots)$  is written  $\forall R_1\dots R_n.P$ .

Let  $C$  be an  $\mathcal{ALN}$ -description. A clausal form of  $C$ , noted  $\hat{G}_C^\#$ , is the set made of all the clauses that appear in the description obtained by recursively applying the following rule on  $\hat{C}$ :  $(\forall R.(E \sqcap F)) \xrightarrow{\equiv} (\forall R.E) \sqcap (\forall R.F)$ .

*Example 1.* The following example shows the normal form and the clausal form of an  $\mathcal{ALN}$ -descriptions  $C$ .

$$\begin{aligned} C &\equiv \forall T.((\geq 4 R) \sqcap (\exists S) \sqcap (\leq 1 R) \sqcap (\geq 2 R) \sqcap (\forall Q.A) \sqcap (\forall Q.(\forall R.(\forall S.(B \sqcap \neg B)))) \\ \hat{C} &= (\leq 0 T) \sqcap (\geq 2 R) \sqcap (\forall Q.(A \sqcap \forall R.(\leq 0 S))) \\ \hat{G}_C^\# &= \{\leq 0 T, \geq 2 R, \forall Q.A, \forall QR. \leq 0 S\} \end{aligned}$$

Clausal forms enable a set-oriented representation of concept descriptions that is easy to understand and manipulate (especially from an algorithmic point of view). Moreover, previous results regarding subsumption and lcs in  $\mathcal{ALN}$ , achieved using different formal frameworks such as description graphs [14, 11] or automata theory [1, 11], can be easily translated in our context as shown below.

**Theorem 1 (Structural subsumption and lcs in  $\mathcal{ALN}$ ).** *Let  $C$  and  $D$  two  $\mathcal{ALN}$ -descriptions. There is :*

1.  $C \sqsubseteq D \Leftrightarrow \forall c_D \in \widehat{\mathcal{G}}_D^\#, \exists c_C \in \widehat{\mathcal{G}}_C^\# \mid c_C \sqsubseteq c_D$
2.  $\widehat{\mathcal{G}}_{lcs(C,D)}^\# = \{c_1 \mid (c_1, c_2) \in (\widehat{\mathcal{G}}_C^\# \times \widehat{\mathcal{G}}_D^\#) \cup (\widehat{\mathcal{G}}_D^\# \times \widehat{\mathcal{G}}_C^\#) \text{ and } c_2 \sqsubseteq c_1\}$

We recall that, building  $\widehat{C}$  from an  $\mathcal{ALN}$ -description  $C$  can be achieved in polynomial time in the size of  $C$  [6]. So  $\widehat{\mathcal{G}}_C^\#$  can also be computed in polynomial time in the size of  $C$ . Consequently, testing subsumption between two  $\mathcal{ALN}$  descriptions  $C$  and  $D$  using theorem 1 can also be achieved in polynomial time in the sizes of the inputs. Moreover, it is shown in [11] that the lcs of two  $\mathcal{ALN}$  descriptions always exists and it can be computed in polynomial time in the sizes of the inputs.

In order to study the semantic difference in  $\mathcal{ALN}$ , we need to recall the notion of weak approximation defined in [8, 9]. The  $\mathcal{L}_2$ -description  $D$  is a weak approximation of the  $\mathcal{L}_1$ -description  $C$  if  $D$  is the maximal description w.r.t. subsumption being subsumed by  $C$ . In this case, we write  $D = Approx_1(C)$ . In fact, as will be seen later, in our work we are only interested in computing the weak approximation of the negation of an  $\mathcal{ALN}$ -clause (which is an  $\mathcal{ALEN}$ -description) by an  $\mathcal{ALN}$ -description. To this end, we reuse the following result [8, 9]:

**Lemma 1 (Weak approximation of  $\forall R_1 R_2 \dots R_n . P$ ).** *Let  $C$  be a  $\mathcal{ALN}$ -clause, i.e.  $C \equiv \forall R_1 R_2 \dots R_n . P$  with  $P$  an atomic concept, the negation of an atomic concept or a number restriction. There is:*

$$\widehat{\mathcal{G}}_{Approx_1(\neg C)}^\# = \{ \forall R_1 R_2 \dots R_n . (\neg P), \forall R_1 R_2 \dots R_{n-1} . (\geq 1 R_n), \\ \forall R_1 R_2 \dots R_{n-2} . (\geq 1 R_{n-1}), \dots, \forall R_1 . (\geq 1 R_2), (\geq 1 R_1) \}$$

All previous recalls and following results about semantic and syntactic difference in  $\mathcal{ALN}$  can be extended to take into account an  $\mathcal{ALN}$ -terminology  $\mathcal{T}$  containing either concept definitions of the form  $A \equiv C$  or atomic concept inclusions of the form  $A \sqsubseteq C$ , with  $A$  an atomic concept and  $C$  an  $\mathcal{ALN}$ -description. This is due to the fact that testing the subsumption of two  $\mathcal{ALN}$ -descriptions wrt  $\mathcal{T}$  amounts to testing the subsumption of the same but unfolded descriptions wrt to the empty terminology (i.e.  $(C \sqsubseteq_{\mathcal{T}} D) \Leftrightarrow (\mathcal{T}(C) \sqsubseteq \mathcal{T}(D))$ ). Thus, to take into account concept definitions, a first unfolding step is mandatory (which can lead to an exponential blow-up [15]). Taking into account atomic concept inclusions can be achieved by replacing them by concept definitions adding a new atomic concept. For example,  $A \sqsubseteq P_1 \sqcap \forall R . P_2$  would be replaced by  $A \equiv P_1 \sqcap \forall R . P_2 \sqcap A'$ . In the sequel, we suppose that we work on unfolded descriptions, so we do not talk about terminologies any more.

### 3 Semantic difference in $\mathcal{ALN}$

Given two concept descriptions  $B$  and  $A$ , the semantic difference  $B - A$  amounts to computing all the maximal, w.r.t. subsumption, descriptions  $C$  s.t.  $C \sqcap A$  is equivalent to  $B$ . So  $C$  can be seen as (i) what has to be added to  $A$  in order to get back  $B$ , and as (ii) the rest of  $B$  after removing its common information with  $A$ . The fact that  $C$  must be maximal with respect to subsumption ensures that

there is no semantic redundancy in  $C$ , which means (i)  $C$  is only what is strictly necessary to add to  $A$  to get  $B$ , and (ii)  $C$  describes what is really specific in  $B$  w.r.t.  $A$ . Primarily defined with the constraint  $B \sqsubseteq A$ , the semantic difference is generalized to all couples of descriptions using their least common subsumer, if it exists [16]. The formal definition of the semantic difference is now given.

**Definition 1 (Semantic difference [16]).** *Let  $\mathcal{L}$  a description logic,  $B$  and  $A$  two  $\mathcal{L}$ -concept descriptions such that  $B \sqsubseteq A$ . The semantic difference between  $B$  and  $A$ , noted  $B - A$ , is defined by :*

$$B - A := \text{Max}_{\sqsubseteq} \{C \mid C \sqcap A \equiv B\}$$

*If the lcs always exists between two  $\mathcal{L}$ -descriptions (for example in  $\mathcal{ALN}$ ), then this definition is extended to couples of descriptions  $B$  and  $A$  with  $B \not\sqsubseteq A$  by :*

$$B - A := B - \text{lcs}(B, A)$$

Note that this definition is independent of  $\mathcal{L}$ , and the result of a semantic difference may be a set of descriptions. Whereas, in [16] the semantic difference is especially studied for languages having a special property<sup>4</sup> ensuring a unique description in the result, we study here the semantic difference for  $\mathcal{ALN}$ . The difference of two  $\mathcal{ALN}$  descriptions may lead to potentially numerous non equivalent  $\mathcal{ALN}$  descriptions (see example 2). This is due to the possibility to decompose the empty concept  $\perp$  into non trivial conjunctions. Up to our knowledge, this is the first time that the semantic difference is studied for a language that implies a non unique difference.

*Example 2.* Let us consider the following two  $\mathcal{ALN}$  descriptions:

$$Q \equiv A \sqcap \forall R_1.(B \sqcap \leq 4R_2) \sqcap \leq 0R_3$$

$$S \equiv A \sqcap \forall R_1.(B \sqcap \forall R_4.C) \sqcap \forall R_3.(D \sqcap \forall R_5.E \sqcap \leq 2R_6)$$

The lcs of  $Q$  and  $S$  is:

$$\text{lcs}(Q, S) \equiv A \sqcap \forall R_1.B \sqcap \forall R_3.(D \sqcap \forall R_5.E \sqcap \leq 2R_6)$$

Hence, the semantic difference between  $Q$  and  $S$  is given by the set:

$$Q - S = \{\forall R_1. \leq 4R_2 \sqcap \forall R_3 \forall R_5. \neg E \sqcap \forall R_3. \geq 1R_5, \\ \forall R_1. \leq 4R_2 \sqcap \forall R_3. \neg D, \\ \forall R_1. \leq 4R_2 \sqcap \forall R_3. \geq 3R_6\}$$

We now see algorithm **compute $\mathcal{ALN}$ SemDiff** to compute the semantic difference of two  $\mathcal{ALN}$ -descriptions  $A$  and  $B$ . Due to lack of space, its detailed form is given in [13], but its underlying principle are given in lemma 2 and its soundness and completeness is given in theorem 2. This is the main contribution of this paper.

**Lemma 2 (Building one description of the semantic difference).** *Let  $B$  and  $A$  two  $\mathcal{ALN}$ -descriptions such that  $B \sqsubseteq A$ . Let  $C$  be an  $\mathcal{ALN}$ -description in  $B - A$ . Let  $P$  be any atomic concept, the negation of any atomic concept or any number restriction.  $\widehat{\mathcal{G}}_C^\#$  can be built as follows:*

<sup>4</sup> This is the so-called structural subsumption property in the sense of [16] which is stronger than the usual notion of a structural subsumption algorithm.

First, if  $\widehat{\mathcal{G}}_A^\# = \widehat{\mathcal{G}}_B^\#$ , then  $\widehat{\mathcal{G}}_C^\#$  must be  $\{\top\}$ . Else,  $\widehat{\mathcal{G}}_C^\#$  is initialized at  $\emptyset$ , and then, for all  $c_B$  in  $\widehat{\mathcal{G}}_B^\#$ :

• **Inconsistency case:**

if  $c_B = \forall R_1 R_2 \dots R_{n-1}. (\leq 0 R_n)$ , with  $n \geq 0$  (if  $n = 0$  then  $c_B = \perp$ )

and  $\exists c_A \in \widehat{\mathcal{G}}_A^\# \mid c_A = \forall R_1 R_2 \dots R_n R_{n+1} \dots R_{n+m}. P$ ,  $m \geq 0$

then we add to  $\widehat{\mathcal{G}}_C^\#$  all clauses  $c$  verifying:

$\left( c = \forall R_1 R_2 \dots R_n. c', c' \in \widehat{\mathcal{G}}_{\text{Approx}_\top(\neg \forall R_{n+1} \dots R_{n+m}. P)}^\# \right)$  and  $(A \not\sqsubseteq c)$

• **General case:**

else if  $c_B \notin \widehat{\mathcal{G}}_A^\#$ , then we add  $c_B$  to  $\widehat{\mathcal{G}}_C^\#$ .

Thus, for each clause  $c_B$  of  $\widehat{\mathcal{G}}_B^\#$ , zero, one or many clauses of  $\widehat{\mathcal{G}}_C^\#$  will be generated such that the conjunction of these generated clauses and some clauses in  $\widehat{\mathcal{G}}_A^\#$  gives back  $c_B$  after normalization. The reason for having a set of descriptions in the result is due to some clauses in  $\widehat{\mathcal{G}}_B^\#$  that may lead to different possibilities to generate clauses of  $\widehat{\mathcal{G}}_C^\#$ . More precisely, only clauses  $c_B$  of  $\widehat{\mathcal{G}}_B^\#$  that have the form  $\forall R_1 R_2 \dots R_{n-1}. (\leq 0 R_n)$ , with  $n \geq 0$ <sup>5</sup> (i.e.  $\forall R_1 R_2 \dots R_n. \perp$ ) may lead to such situations (this is the so-called "Inconsistency case" in theorem 2). All other configurations for  $c_B$  are trivially solved (this is the so-called "General case" in theorem 2).

Elements of proof are now given. In the inconsistency case, the weak approximation is used to generate clauses that stay in  $\mathcal{ALN}$  (because we can't use the full negation constructor in  $\mathcal{ALN}$ ). The other interesting point concerning the weak approximation is that it ensures the property of maximality w.r.t. subsumption that is required. Last but not least, we use the characterization of inconsistency in  $\mathcal{ALN}$  showed in lemmas 4.2.2 and 6.1.4 of [11] as the main argument of completeness for this theorem. The detailed proof of lemma 2 is given in [13].

Based on lemma 2, the **computeALNSemDiff** algorithm computes *all and only all* descriptions in the semantic difference by computing all possible combinations of multiple clauses cases. Theorem 2 proves its soundness and completeness and gives its complexity (see [13] for the proof).

**Theorem 2 (computeALNSemDiff characteristics).** *Let B and A be two ALN-descriptions, given in their clausal forms, such that  $B \sqsubseteq A$ . Algorithm **computeALNSemDiff** (given in [13]) computes the clausal form of exactly all ALN-descriptions that belong to the semantic difference of B and A as defined in definition 1. This computation is PTIME wrt the sizes of B and A (i.e. the numbers of clauses in their clausal forms and the maximal number of roles in any of their clauses).*

<sup>5</sup> If  $n = 0$ , then it is the clause  $\perp$ .

## 4 Semantic versus syntactic difference in $\mathcal{ALN}$

In this section, we focus on the comparison between the semantic difference and the syntactic difference in  $\mathcal{ALN}$ . We first show how to compute the syntactic difference in  $\mathcal{ALN}$ , and then we compare both operators.

### 4.1 Syntactic difference in $\mathcal{ALN}$

Syntactic difference has been defined in [11, 7]. The aim of the syntactic difference  $B - A$  is to remove from  $B$  all its *subdescriptions* that are redundant with  $A$  (i.e. that are also in  $A$ ). The consequence is that the result is minimal in size. This operator was initially defined to evaluate the loss of information when approximating an  $\mathcal{ALC}$ -description by an  $\mathcal{ALE}$ -description [7].

The syntactic difference relies on the notion of subdescription. Intuitively,  $D$  is a subdescription of  $E$  if  $D$  can be obtained from  $E$  by removing conjuncts or disjuncts that are in  $E$ , replacing parts of the description of  $E$  by  $\perp$  or by subdescriptions of these parts. This notion of subdescription defines the partial order  $\preceq_d : D \preceq_d E$  iff  $D$  is a subdescription of  $E$  [3, 11, 7]. This partial order is used to define the syntactic difference  $B - A$ . Thereafter, we recall the definition of this operation using  $\mathcal{ALE}$  for both  $B$  and  $A$ . It is the only case for which the difference is uniquely determined, modulo associativity and commutativity of concept conjunction, and for which a sound and complete algorithm exists [11, 7]<sup>6</sup>.

**Definition 2 (Syntactic difference in  $\mathcal{ALE}$ ).** *Let  $A$  and  $B$  be two  $\mathcal{ALE}$  descriptions. The syntactic difference  $B - A$  of  $B$  and  $A$  is defined as the  $\mathcal{ALE}$  description  $C$  which is minimal w.r.t.  $\preceq_d$  such that  $C \sqcap A \equiv A \sqcap B$ .*

Looking at the previous definition, it seems that, in  $\mathcal{ALN}$ , the only difference between semantic and syntactic difference is how  $\perp$  is processed: in the semantic difference, non trivial conjunctions equivalent to  $\perp$  are computed, while they are not in the syntactic difference. If we extend the definition of  $\preceq_d$  to  $\mathcal{ALN}$ , as well as the definition of the syntactic difference to  $\mathcal{ALN}$ , we can prove the previous intuition by the following theorem which is the second contribution of this paper. This theorem shows that the syntactic difference is basically a set difference between clauses of  $A \sqcap B$  and  $A$  (see [13] for the full proof).

**Theorem 3 (Building the syntactic difference in  $\mathcal{ALN}$ ).** *Let  $A$  and  $B$  be two  $\mathcal{ALN}$ -descriptions given in their clausal form. The syntactic difference  $B - A$  defined in definition 2 is a unique  $\mathcal{ALN}$ -description  $C$  such that if  $A \sqsubseteq B$ , then  $C = \top$ , else  $\widehat{\mathcal{G}}_C^\# = \widehat{\mathcal{G}}_{A \sqcap B}^\# \setminus \widehat{\mathcal{G}}_A^\#$ .*

Thus, computing the syntactic difference between two  $\mathcal{ALN}$  descriptions is PTIME wrt the numbers of clauses in their clausal forms.

<sup>6</sup> The case where  $B$  is an  $\mathcal{ALC}$  description and  $A$  an  $\mathcal{ALE}$  has been studied in [7], but only a heuristic has been given to compute it.

## 4.2 Comparing semantic and syntactic operators

Theorems 2 and 3 show that, for  $\mathcal{ALN}$ -descriptions, *both difference compute the same result*, except for  $\perp$  in the two special cases presented below.

– **Case 1:**

Let  $B \equiv \forall R.\perp$  and  $A \equiv \forall R.(\neg P \sqcap P')$ . There is:

Semantic difference:  $B - A = \{\forall R.P, \forall R.\neg P'\}$ .

Syntactic difference:  $B - A = \forall R.\perp$ .

This case has already been studied in [11], and the conclusion is the following. It is true that the semantic difference does give the semantic gap between  $B$  and  $A$ , i.e. what has to be added to  $A$  to get  $B$ . But multiple results are less easy to figure out by a user (e.g. by a knowledge engineer) than the unique one of the syntactic difference. Moreover, in this case, the result of the syntactic difference is more intuitive since it doesn't refer to any decomposition of  $\perp$ . Nevertheless, we could add that getting  $B$  as the result of  $B - A$  amounts to say that there is no common point between  $B$  and  $A$ , whereas  $lcs(B, A) \equiv A$ .

– **Case 2:**

Let  $B \equiv \forall R.P$  and  $A \equiv \forall R.\neg P$ . There is:

Semantic difference:  $B - lcs(B, A) = \{\forall R.P\}$ .

Syntactic difference:  $B - A = \forall R.\perp$ .

In that case, the result of the semantic difference seems to be more intuitive. Indeed, getting  $B$  as the result of  $B - A$  amounts to say that there is no common point between  $B$  and  $A$ , which is verified in that case since  $lcs(B, A) \equiv \top$ . By the contrary, the result of the syntactic difference is harder to interpret, since it cannot be interpreted neither as what remains of  $B$  once  $A$  has been removed, nor as what to add to  $A$  to get  $B$ .

So none of the two operators always produces more intuitive or understandable results. On the one hand, syntactic one generates a unique result which can be easier to manipulate (especially by a human). On the other hand, the semantic operator really computes the semantic gap between two description, by possibly generating many results and handling non intuitive decompositions of  $\perp$ . These multiple results may be more difficult to manipulate, but can allow a more exhaustive processing of some task. Hence the overall conclusion is that the choice of a difference operator in  $\mathcal{ALN}$  will eventually depend more on the applicative context than on other technical criteria. None is a priori better. However, in our application context [4, 5], the difference operation is used to define the notion of best cover of a concept using a terminology. The aim there is to reformulate a query  $Q$  into a description that contain as much as possible of *common information* with  $Q$  and as less as possible of *extra information* with respect to  $Q$ . Such a description is called a best cover of  $Q$ . In [5], the extra information contained in a query  $Q$  and not in its best cover  $E$ , computed using the difference  $Q - E$ , is used to query remote sources in a peer-to-peer integration system. In such a context, using the semantic difference turns out to be more adequate than the syntactic one as it enables to query more relevant sources than what is enabled by the syntactic difference.

## 5 Conclusion

In this paper, we investigated the problem of computing the semantic and the syntactic difference operators in the context of the  $\mathcal{ALN}$  language. We provide two polynomial-time algorithms to compute them. We compare both and argue that the semantic one is better suited to extend the notion of concept covers previously studied in [10, 4, 5].

## References

1. F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 1999.
2. F. Baader, R. Küsters, and R. Molitor. Computing Least Common Subsumer in Description Logics with Existential Restrictions. In *Proceedings of IJCAI'99*.
3. F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In *Proc. of KR2000*.
4. B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani. On automating web services discovery. *VLDB J.*, 14(1):84–96, 2005.
5. B. Benatallah, M.-S. Hacid, H. Paik, C. Rey, and F. Toumani. Towards semantic-driven, flexible and scalable framework for peering and querying e-catalog communities. *Information Systems, Special issue on semantic web and web services*, 31(4-5), 2006.
6. A. Borgida and P.F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *JAIR*, 1:277–308, 1994.
7. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and Difference in Description Logics. In *Proc. of KR2002*.
8. F. Goasdoué. *Réécriture de requêtes en termes de vues dans CARIN et intégration d'informations*. PhD thesis, Université Paris XI Orsay, 2001.
9. F. Goasdoué and M.-C. Rousset. Compilation and approximation of conjunctive queries by concept descriptions. In *Proc. of DL2002*, 2002.
10. M.S. Hacid, A. Léger, C. Rey, and F. Toumani. Computing concept covers: a preliminary report. In *Proc. of DL02 Toulouse. France*, April 2002.
11. R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001. Ph.D. thesis.
12. R. Küsters and R. Molitor. Computing least common subsumers in alen. LTCS-Report 00-07, LTCS, RWTH Aachen, Germany, 2000.
13. A. Léger, C. Rey, and F. Toumani. Semantic difference in the aln description logic. Research report, LIMOS, 2007. see <http://www.isima.fr/~rey/RRSemDiff.pdf>.
14. R. Molitor. Structural subsumption for  $\mathcal{ALN}$ . Technical Report LTCS-98-03, LuFG Theoretical Computer Science, RWTH Aachen, March 1998.
15. B. Nebel. Terminological Reasoning is Inherently Intractable. *ai*, 43:235–249, 1990.
16. G. Teege. Making the difference: A subtraction operation for description logics. In *Proc. of KR'94*, May 1994.

## Consistent Query Answering over Description Logic Ontologies\*

Domenico Lembo and Marco Ruzzi

Dipartimento di Informatica e Sistemistica,  
Sapienza Università di Roma  
*lembo, ruzzi@dis.uniroma1.it*

**Abstract.** Description Logics (DLs) have been widely used in the last years as formal language for specifying ontologies. In several contexts, as ontology-based data integration, it may frequently happen that data contradict the intensional knowledge provided by the ontology through which they are accessed, which therefore may result inconsistent. In this paper, we analyze the problem of consistent query answering (CQA) over DL ontologies, i.e., the problem of providing meaningful answers to queries posed over inconsistent ontologies. We provide inconsistency-tolerant semantics for DLs, and study the computational complexity of CQA under such semantics for the case of conjunctive queries posed over ontologies specified in *DL-Lite<sub>F</sub>*, a DL specifically tailored to deal with large amounts of data. We show that the above problem is coNP-complete w.r.t. data complexity. We then study the problem of consistent instance checking for *DL-Lite<sub>F</sub>* ontologies, i.e., the instance checking problem considered under our inconsistency-tolerant semantics, and we show that such problem is in PTIME w.r.t. data complexity.

### 1 Introduction

Description Logics (DLs) have been widely used in the last years as formal language for specifying ontologies, for their ability of combining modelling power and decidability of reasoning [10]. Recently, besides expressive DLs, which suffer from inherently worst-case exponential time behavior of reasoning [3], also DLs that allow for tractable reasoning have been proposed for ontology modelling [2, 7]. Such DLs are particularly suited for management of large amounts of data (e.g., from thousands to millions of instances). Then, a challenging use of them is represented by ontology-based data integration, an issue that has recently received a growing attention in the Semantic Web community [15]. Indeed, integrating data in the Semantic Web context mainly means accessing, collecting, and exchanging data distributed over the web through mediated schemas given in terms of ontologies (i.e., DL TBoxes).

Due to the dynamic nature of the setting described above, it may frequently happen that data contradict the intensional knowledge provided by the ontology through which they are accessed, especially in those cases in which the ontology provides a conceptual view of a number of autonomous data sources, heterogeneous and widely distributed.

---

\* The present work is an extended abstract of [12].

In the above situation, ontologies may result inconsistent, and reasoning over them according to classical DLs may become meaningless, since whatever conclusion may be derived from an inconsistent theory. Then, besides handling inconsistency at the terminological/schema level, which has been a subject recently investigated for ontology-based applications [14, 11], the need arises in this context to deal with inconsistency at the instance/data level. In the present paper we study this problem.

The approach commonly adopted to solve data inconsistency is through data cleaning [5]. This approach is procedural, and is based on domain-specific transformation mechanisms applied to the data. One of its problems is incomplete information on how certain conflicts should be resolved. This typically happens in systems which are not tailored for business logic support at the enterprise level, like systems for information integration over the web. In the last years, an alternative declarative approach has been investigated in the area of consistent query answering (CQA) [1, 4, 6]. Such an approach relies on the notion of *repair* for a database instance that may violate integrity constraints (ICs) specified over its schema. Roughly speaking, a repair is a new database instance which satisfies the constraints in the schema and minimally differs from the original one. In general multiple repairs are possible. Then, CQA amounts to compute those answers to a user query that are in the evaluation of the query over each repair. It is well-known [6, 9] that CQA of Conjunctive Queries (CQs) expressed over database schemas with (even simple forms of) ICs is a coNP-complete problem in data complexity, i.e., the complexity measured only with respect to the size of the database instance.

In this paper, we study CQA over DL ontologies. In particular, we provide a new semantic characterization for DLs, based on the notion of repair. We focus on  $DL-Lite_{\mathcal{F}}$ , a DL of the  $DL-Lite$  family [7, 8]. The  $DL-Lite$  family comprises a series of DLs that are specifically tailored to deal with large amounts of data and reasoning over queries. While the expressive power of the DLs in the  $DL-Lite$  family is carefully controlled to maintain low the complexity of reasoning, such DLs are expressive enough to capture the main notions of both ontologies, and conceptual modelling formalisms used in databases and software engineering (i.e., ER and UML class diagrams). We study CQA for the class of union of conjunctive queries (UCQs), which is the most expressive class of queries for which decidability of query answering has been proved in DLs [13]. Notably, standard query answering of UCQs over  $DL-Lite_{\mathcal{F}}$  can be solved by means of evaluation of suitable first-order logic queries over the underlying  $DL-Lite$  ABox considered as a flat relational database [7, 8]. This allows for using well established Relational Database Management System technology for reasoning over queries in such DLs.

The contributions of the present paper can then be summarized as follows.

- We provide an inconsistency-tolerant semantics for DLs, which relies on the notion of repair of a DL ontology, and allows for meaningful reasoning in the presence of inconsistency (Section 3);
- We study computational complexity of CQA for conjunctive queries expressed over  $DL-Lite_{\mathcal{F}}$  ontologies, and show that such a problem is coNP-complete w.r.t. data complexity (Section 4);
- Towards identification of tractable cases of CQA for  $DL-Lite$ , we study consistent instance checking over  $DL-Lite_{\mathcal{F}}$  ontologies, i.e., the instance checking problem

under our inconsistency-tolerant semantics. and show that such a problem is in PTIME w.r.t. data complexity (Section 5).

## 2 The Description Logic $DL-Lite_{\mathcal{F}}$

In this section we present the syntax and the semantics of  $DL-Lite_{\mathcal{F}}$  [7].  $DL-Lite_{\mathcal{F}}$  concepts and roles are defined as follows:

$$B \longrightarrow A \mid \exists R \quad R \longrightarrow P \mid P^- \quad C \longrightarrow B \mid \neg B$$

where  $A$  denotes an atomic concept,  $P$  an atomic role, and  $P^-$  the inverse of the atomic role  $P$ .  $B$  denotes a *basic concept*, i.e., a concept that can be either an atomic concept or a concept of the form  $\exists R$ , and  $R$  denotes a *basic role*, i.e., a role that is either an atomic role or the inverse of an atomic role. Finally,  $C$  denotes a (*general*) *concept*, which can be a basic concept or its negation. Let  $B_1$  and  $B_2$  be basic concepts, we call *positive inclusions (PIs)* assertions of the form  $B_1 \sqsubseteq B_2$ , whereas we call *negative inclusions (NIs)* assertions of the form  $B_1 \sqsubseteq \neg B_2$ .

A DL *knowledge base (KB)*  $\mathcal{K}$  is a pair  $\langle \mathcal{T}, \mathcal{A} \rangle$  which represents the domain of interest in terms of two parts, a *TBox*  $\mathcal{T}$ , specifying the intensional knowledge, and an *ABox*  $\mathcal{A}$ , specifying extensional knowledge. A  $DL-Lite_{\mathcal{F}}$  TBox is formed by: (i) a finite set of *concept inclusion assertions*, i.e., expressions of the form  $B \sqsubseteq C$ , meaning that all instances of the basic concept  $B$  are also instances of the generic concept  $C$ , and (ii) a finite set of *functionality assertions* on roles or on their inverses of the form (funct  $P$ ) or (funct  $P^-$ ), respectively, meaning that a relation  $P$  (resp.  $P^-$ ) is functional.  $DL-Lite_{\mathcal{F}}$  ABoxes are formed by a finite set of membership assertions on atomic concepts and atomic roles, of the form  $A(a)$  and  $P(a, b)$ , stating respectively that the object denoted by the constant  $a$  is an instance of the atomic concept  $A$  and that the pair of objects denoted by the pair of constants  $(a, b)$  is an instance of the role  $P$ .

The semantics of a DL is given in terms of interpretations, where an *interpretation*  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  consists of a non-empty interpretation domain  $\Delta^{\mathcal{I}}$  and an *interpretation function*  $\cdot^{\mathcal{I}}$  that assigns to each concept  $C$  a subset  $C^{\mathcal{I}}$  of  $\Delta^{\mathcal{I}}$ , and to each role  $R$  a binary relation  $R^{\mathcal{I}}$  over  $\Delta^{\mathcal{I}}$ . In particular we have:

$$\begin{aligned} A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} & (\exists R)^{\mathcal{I}} &= \{o \mid \exists o'. (o, o') \in R^{\mathcal{I}}\} \\ P^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} & (\neg B)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}} \\ (P^-)^{\mathcal{I}} &= \{(o_2, o_1) \mid (o_1, o_2) \in P^{\mathcal{I}}\} \end{aligned}$$

Furthermore, an interpretation  $\mathcal{I}$  is a model of a concept inclusion assertion  $B \sqsubseteq C$ , if  $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ , and  $\mathcal{I}$  is a model of an assertion (funct  $P$ ) if  $(o, o_1) \in P^{\mathcal{I}}$  and  $(o, o_2) \in P^{\mathcal{I}}$  implies  $o_1 = o_2$ . Analogously for (funct  $P^-$ ).

To specify the semantics of membership assertions, we extend the interpretation function to constants, by assigning to each constant  $a$  a *distinct* object  $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ . Note that this implies that we enforce the *unique name assumption* on constants [3]. An interpretation  $\mathcal{I}$  is a model of a membership assertion  $A(a)$ , (resp.,  $P(a, b)$ ) if  $a^{\mathcal{I}} \in A^{\mathcal{I}}$  (resp.,  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$ ).

Given an (inclusion, functionality, or membership) assertion  $\alpha$ , and an interpretation  $\mathcal{I}$ , we denote by  $\mathcal{I} \models \alpha$  the fact that  $\mathcal{I}$  is a model of  $\alpha$ , and also say that  $\alpha$  is satisfied by

$\mathcal{I}$ . Given a (finite) set of assertions  $\kappa$ , we denote by  $\mathcal{I} \models \kappa$  the fact that  $\mathcal{I}$  is a model of every assertion in  $\kappa$ . A *model of a DL-Lite KB*  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  is an interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models \mathcal{T}$  and  $\mathcal{I} \models \mathcal{A}$ . With a little abuse of notation, we also write  $\mathcal{I} \models \mathcal{K}$ . A KB is *satisfiable* if it has at least one model, otherwise it is *unsatisfiable*.

*Example 1.* Consider the atomic concepts *Cat*, *Dog*, *Pet* and *Person* and the roles *hasOwner* and *feeds*. The following TBox  $\mathcal{T}$  is an example of *DL-Lite<sub>F</sub>* TBox:

$$\begin{array}{ll} \text{Dog} \sqsubseteq \text{Pet} & \exists \text{hasOwner}^- \sqsubseteq \text{Person} \\ \text{Cat} \sqsubseteq \text{Pet} & \text{Cat} \sqsubseteq \neg \text{Dog} \\ \text{Pet} \sqsubseteq \exists \text{hasOwner} & (\text{funct } \text{hasOwner}). \end{array}$$

In the TBox above we say that cats and dogs are pets, every pet has an owner, a cat is not a dog, the owner of an animal is a person, and that a pet cannot have more than one owner. Finally, we show a simple *DL-Lite* ABox  $\mathcal{A}$ : *Person*(John), *Dog*(Bruto), *hasOwner*(Tom, Leo).  $\square$

A *union of conjunctive queries (UCQ)*  $q$  over a *DL-Lite<sub>F</sub>* KB  $\mathcal{K}$  is an expression of the form

$$q(\mathbf{x}) \leftarrow \bigvee_{i=1, \dots, n} \exists \mathbf{y}_i. \text{conj}_i(\mathbf{x}, \mathbf{y}_i), \quad (1)$$

where each  $\text{conj}_i(\mathbf{x}, \mathbf{y}_i)$  is a conjunction of atoms and equalities, with free variables  $\mathbf{x}$  and  $\mathbf{y}_i$ . Variables in  $\mathbf{x}$  are called *distinguished*, and the size of  $\mathbf{x}$  is called the *arity* of  $q$ . The right-hand side of the Formula (1) is called the *body* of  $q$ . Atoms in each  $\text{conj}_i$  are of the form  $A(z)$  or  $P(z_1, z_2)$ , where  $A$  and  $P$  are respectively an atomic concept and an atomic role of  $\mathcal{K}$ , and  $z, z_1, z_2$  are either constants in  $\mathcal{K}$  or variables. A *Boolean UCQ* is a query with arity 0, written simply as a sentence of the form  $\bigvee_{i=1, \dots, n} \exists \mathbf{y}_i. \text{conj}_i(\mathbf{y}_i)$ . A UCQ with a single conjunction of atoms, i.e., with  $n = 1$  in the Formula (1), is called *conjunctive query (CQ)*.

Let  $q$  be a Boolean UCQ over a *DL-Lite<sub>F</sub>* KB  $\mathcal{K}$ . We say that  $q$  is *entailed* by  $\mathcal{K}$ , and write  $\mathcal{K} \models q$ , if, for every model  $\mathcal{M}$  of  $\mathcal{K}$ ,  $\mathcal{M} \models q$ , where  $\models$  is the standard evaluation of first-order sentences in an interpretation. The *instance checking problem* corresponds to entailment of a Boolean ground CQ consisting of a single atom, i.e., a membership assertion of the form  $A(a)$  or  $P(a, b)$ . Let  $q$  be a non-Boolean UCQ of arity  $n$  over  $\mathcal{K}$ , and let  $\mathbf{t}$  be an  $n$ -tuple of constants. We say that  $\mathbf{t}$  is a *certain answer* to  $q$  in  $\mathcal{K}$  if  $\mathcal{K} \models q(\mathbf{t})$ , where  $q(\mathbf{t})$  is the sentence obtained from the body of  $q$  by replacing its distinguished variables by constants in  $\mathbf{t}$ . We denote by  $\text{Ans}(q, \mathcal{K})$  the set of certain answers to  $q$  in  $\mathcal{K}$ .

*Example 2.* Let us consider the DL KB  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  where the TBox  $\mathcal{T}$  and the ABox  $\mathcal{A}$  are as defined in Example 1. The following query  $q$  is a CQ over  $\mathcal{K}$ :  $q(x) \leftarrow \text{Person}(x)$ . It is easy to see that the set of certain answers to  $q$  in  $\mathcal{K}$  is  $\text{Ans}(q, \mathcal{K}) = \{\text{John}, \text{Leo}\}$  where John can be directly derived from the membership assertions of  $\mathcal{A}$ , whereas Leo is implied by the inclusion assertion  $\exists \text{hasOwner}^- \sqsubseteq \text{Person}$  and by the role membership assertion *hasOwner*(Tom, Leo).  $\square$

### 3 Inconsistency-tolerant Semantics

Let us now consider the case in which a DL KB  $\mathcal{K}$  is unsatisfiable, i.e.,  $\mathcal{K}$  does not have any model. As already said in the introduction, reasoning over such a  $\mathcal{K}$  is meaningless, since whatever consequence can be deduced from  $\mathcal{K}$ . In this section, we provide a new semantics for DL KB that is inconsistency-tolerant, i.e., it allows for “meaningful” reasoning over KBs that are unsatisfiable according to the classical first-order based semantics, as that considered in Section 2 for  $DL-Lite_{\mathcal{F}}$ . In particular, our semantics is tolerant to the inconsistency that arises in a DL knowledge base  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  in which a satisfiable TBox  $\mathcal{T}$  may be contradicted by the extensional assertions in the ABox  $\mathcal{A}$ , thus resulting in possibly unsatisfiable KBs. This situation frequently happens in those systems that provide access to data (possibly integrated from autonomous sources) through DL ontologies, as in Semantic Web applications.

Formally, let  $\mathcal{I}$  be an interpretation and let  $\mathcal{A}$  be an ABox. We denote by  $Sat(\mathcal{I}, \mathcal{A})$  the set of membership assertions from  $\mathcal{A}$  that are satisfied in  $\mathcal{I}$ , i.e.,  $Sat(\mathcal{I}, \mathcal{A}) = \{\alpha \mid \alpha \in \mathcal{A} \text{ and } \mathcal{I} \models \alpha\}$ .

**Definition 1.** Let  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  be a DL KB and  $\mathcal{R}$  an interpretation. We say that  $\mathcal{R}$  is a repair of  $\mathcal{K}$  if: (i)  $\mathcal{R}$  is a model for  $\mathcal{T}$ ; (ii) there exists no interpretation  $\mathcal{R}'$  such that  $\mathcal{R}'$  is a model for  $\mathcal{T}$  and  $Sat(\mathcal{R}', \mathcal{A}) \supset Sat(\mathcal{R}, \mathcal{A})$ .

In the following, we denote by  $Rep(\mathcal{K})$  the set of repairs of  $\mathcal{K}$ . It is easy to see that when a KB  $\mathcal{K}$  is satisfiable, repairs of  $\mathcal{K}$  coincide with models of  $\mathcal{K}$ . Also, when the TBox of  $\mathcal{K}$  is satisfiable,  $\mathcal{K}$  has always at least one repair.

Following the line of research of CQA [1, 4, 6], in our semantics, intensional knowledge specified by the TBox of a knowledge base is considered stronger than data, i.e., the extensional knowledge provided by the ABox. Indeed, a repair  $\mathcal{R}$  of a knowledge base  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  is an interpretation that needs to satisfy  $\mathcal{T}$  and that at the same time satisfies a maximal set  $\mathcal{A}_m$  of the membership assertions in  $\mathcal{A}$ , i.e.,  $\mathcal{R}$  is a model of the knowledge base  $\langle \mathcal{T}, \mathcal{A}_m \rangle$ .

Let  $q$  be Boolean UCQ over a  $DL-Lite$  KB  $\mathcal{K}$ , we say that  $q$  is *consistently entailed* by  $\mathcal{K}$ , and write  $\mathcal{K} \models_{cons} q$  if, for every  $\mathcal{R} \in Rep(\mathcal{K})$ ,  $\mathcal{R} \models q$ . Then, given a non-Boolean UCQ  $q$  of arity  $n$  over  $\mathcal{K}$ , we say that an  $n$ -tuple  $t$  of constants is a *consistent answer* to  $q$  in  $\mathcal{K}$  if  $\mathcal{K} \models_{cons} q(t)$ . We denote by  $ConsAns(q, \mathcal{K})$  the set of consistent answers to  $q$  in  $\mathcal{K}$ . Furthermore, the *consistent instance checking* problem corresponds to consistent entailment of a membership assertion.

We finally notice that, when a DL knowledge base  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  is a  $DL-Lite$  KB,  $\mathcal{K}$  may result unsatisfiable only if the ABox  $\mathcal{A}$  contradicts the intensional knowledge of the TBox  $\mathcal{T}$ . Indeed, it is possible to show that a  $DL-Lite$  TBox admits always at least one model. As a consequence, we have that our inconsistency-tolerant semantics ensures that every  $DL-Lite$  KB has always at least one repair.

*Example 3.* Let us consider again the  $DL-Lite_{\mathcal{F}}$  TBox  $\mathcal{T}$  described in Example 1 and the ABox  $\mathcal{A}'$  containing the facts  $Person(John)$ ,  $hasOwner(Tom, John)$ , and  $hasOwner(Tom, Leo)$ . It is easy to see that the knowledge base  $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$  is unsatisfiable, since the functionality assertion on  $hasOwner$  is violated. Then, each repair  $\mathcal{R}$  of  $\mathcal{K}'$  is such that  $\mathcal{R} \models \mathcal{T}$ , and either  $Sat(\mathcal{R}, \mathcal{A}') = \{hasOwner(Tom, John)\}$ ,

or  $Sat(\mathcal{R}, \mathcal{A}') = \{hasOwner(Tom, Leo)\}$ . Let us now consider the Boolean CQ  $q' = \exists y.hasOwner(Tom, y)$  over  $\mathcal{K}'$ , asking whether Tom is owned by someone. It is easy to see that  $q$  is consistently entailed by  $\mathcal{K}'$ . However, for the query  $q'' = \{x \mid hasOwner(Tom, x)\}$ , we have that  $ConsAns(q'', \mathcal{K}') = \emptyset$ . In words, we cannot establish who is the owner of Tom, but we can state that Tom has an owner.  $\square$

#### 4 Consistent Query Answering

In this section we consider the problem of CQA for CQs over  $DL-Lite_{\mathcal{F}}$  KBs and show that such a problem is coNP-complete w.r.t. data complexity, i.e., the complexity measured w.r.t. the size of the ABox only.

**Theorem 1.** *Let  $\mathcal{K}$  be a  $DL-Lite_{\mathcal{F}}$  KB,  $q$  a CQ of arity  $n$  over  $\mathcal{K}$ , and  $\mathbf{t}$  an  $n$ -tuple of constants. Then, the problem of establishing whether  $\mathbf{t} \in ConsAns(q, \mathcal{K})$  is coNP-complete with respect to data complexity.*

*Proof (sketch)* CoNP hardness can be proved by a reduction from the 3-colorability problem, whereas membership in coNP follows from the results in [9].

The above result tells us that CQA over  $DL-Lite_{\mathcal{F}}$  KBs is in general intractable w.r.t. data complexity, differently from the problem of (standard) query answering over  $DL-Lite_{\mathcal{F}}$  KBs [7]. Notice that tractability of query answering (and of classical DL reasoning services) is a crucial property for DLs of the  $DL-Lite$  family which makes them particularly suited for dealing with big amounts of data. Then, to preserve this nice behavior, we aim at identifying interesting cases in which CQA is tractable. As we will show in the next section, consistent instance checking is in fact tractable over  $DL-Lite_{\mathcal{F}}$  KBs.

#### 5 Consistent Instance Checking

We now address consistent instance checking over  $DL-Lite_{\mathcal{F}}$  KBs, and show that such a problem is in PTIME w.r.t. data complexity. To this aim, we define a polynomial time algorithm that takes as input a membership assertion  $\alpha$  and a  $DL-Lite_{\mathcal{F}}$  KB  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  and returns *true* if  $\alpha$  is consistently entailed by  $\mathcal{K}$ , *false* otherwise.

In the following we only sketch our algorithm and refer the reader to [12] for a more detailed description of it. We assume that input KB  $\mathcal{K}$  is closed w.r.t. logical implication of NIs by PIs and NIs specified in the TBox  $\mathcal{T}$ , i.e., we assume that all NIs logically implied by  $\mathcal{T}$  are explicitly asserted in  $\mathcal{T}$ . Furthermore, we assume that the ABox  $\mathcal{A}$  does not contain any membership assertion  $\beta$  such that the KB  $\langle \mathcal{T}, \{\beta\} \rangle$  is inconsistent, i.e.,  $\mathcal{A}$  does not contain membership assertions that not belong to any repair of  $\mathcal{K}$ . Notice that each  $DL-Lite_{\mathcal{F}}$  KB can be transformed in a KB of the above form through simple pre-processing steps [12].

Our technique can be summarized in the two steps described below. In the first step, we only take care of the PIs in  $\mathcal{K}$  by means of the algorithm **PerfectRef** presented in [7]. Informally, such an algorithm takes a  $DL-Lite_{\mathcal{F}}$  TBox  $\mathcal{T}$  and a UCQ  $q$  as input,

and reformulates  $q$  according to the PIs in  $\mathcal{T}$ , used as rewriting rules, iteratively applied from right to left to the atoms occurring in  $q$ , thus producing a new UCQ  $q_r$  over  $\mathcal{K}$ . In words, **PerfectRef** encodes in  $q_r$  the intensional knowledge of  $\mathcal{T}$  in such a way that the answers to  $q$  over  $\mathcal{K}$  correspond to the answers to  $q_r$  over the *DL-Lite<sub>F</sub>* KB  $\mathcal{K}^-$  that is obtained from  $\mathcal{K}$  by removing all PIs in  $\mathcal{T}$ . We execute the algorithm **PerfectRef** with the membership assertion  $\alpha$  and the TBox  $\mathcal{T}$  as input. Due to the particular form of the input query, we obtain as result a Boolean union of atoms  $q_r$  over  $\mathcal{K}$ , i.e.,  $q_r$  is a CQ of the form  $\bigvee_{i=1,\dots,n} \exists \mathbf{y}_i \cdot \text{conj}_i(\mathbf{y}_i)$ , in which every  $\text{conj}_i$  consists of a single atom, denoted in the following by  $\text{atom}_i$ .

In the second step we take into account only NIs and functionalities in  $\mathcal{T}$ . To this aim, we use the algorithm **ConsAnswer** which takes as input the Boolean union of atoms  $q_r$  and the *DL-Lite<sub>F</sub>* KB  $\mathcal{K}^-$ , and returns *true* if  $\mathcal{K}^- \models_{\text{cons}} q_r$ , *false* otherwise. To explain the algorithm **ConsAnswer** more in detail, we need two preliminary definitions. A  $\mathcal{K}$ -opponent to a membership assertion  $\alpha$  is a membership assertion  $\beta \in \mathcal{A}$  that together with  $\alpha$  contradicts a functionality or a NI assertion of  $\mathcal{T}$ , i.e., the KB  $\langle \mathcal{T}, \{\alpha, \beta\} \rangle$  is unsatisfiable. Then, let  $q$  be a Boolean union of atoms  $\bigvee_{i=1,\dots,n} \exists \mathbf{y}_i \cdot \text{atom}_i(\mathbf{y}_i)$ . A membership assertion  $\gamma$  is an *image* of  $q$  if there is an  $i \in \{1, \dots, n\}$  such that there exists a substitution  $\sigma$  from the variables in  $\text{atom}_i(\mathbf{y}_i)$  to constants in  $\gamma$  such that  $\sigma(\text{atom}_i(\mathbf{y}_i)) = \gamma$ . Roughly speaking, an image of  $q$  is a membership assertion  $\gamma$  such that  $q$  is entailed by the knowledge base constituted only by the assertion  $\gamma$ . With these notions in place, we can intuitively describe the behavior of **ConsAnswer**. The algorithm verifies whether there exists an image  $\gamma$  of the input query  $q_r$  that belongs to  $\mathcal{A}$  such that either (a)  $\gamma$  has no  $\mathcal{K}$ -opponents or (b) every  $\mathcal{K}$ -opponent  $\beta$  to  $\gamma$  is such that  $\beta$  has at least one  $\mathcal{K}$ -opponent  $\beta'$  which is not  $\mathcal{K}$ -opponent to  $\gamma$  and is in turn  $\mathcal{K}$ -opponent to a different image  $\gamma'$ . If the condition (a) succeeds, then the query  $q_r$  is consistently entailed by  $\mathcal{K}$  since every repair of  $\mathcal{K}$  satisfies the same image of  $q_r$ . As for condition (b), it ensures that if a repair  $\mathcal{R}$  does not satisfy the image  $\gamma$ , since it satisfies the  $\mathcal{K}$ -opponent  $\beta$  to  $\gamma$ , then  $\mathcal{R}$  satisfies another image  $\gamma'$ , whose satisfaction is guaranteed by the fact that  $\mathcal{R}$  does not satisfy  $\beta'$ .

It is possible to prove that the procedure described above is sound and complete w.r.t. finding a solution to the consistent instance checking problem, and that it runs in time polynomial in the size of the ABox  $\mathcal{A}$ . Therefore, we can give the following notable result.

**Theorem 2.** *Let  $\mathcal{K}$  be a *DL-Lite<sub>F</sub>* KB and  $\alpha$  a membership assertion. Then, the problem of establishing whether  $\mathcal{K} \models_{\text{cons}} \alpha$  is in PTIME with respect to data complexity.*

## 6 Conclusions

The work in the present paper can be extended in several directions. We are currently developing a completely intensional technique for consistent instance checking, with the aim of reducing this problem to query evaluation over a database instance representing the ABox of the knowledge base. Such a technique would allow us to maintain reasoning at the intensional level, as it can be already done for standard query answering over *DL-Lite<sub>F</sub>* KBs. We are also working in the direction of identifying other tractable

cases of consistent query answering over  $DL-Lite_{\mathcal{F}}$  KBs. In this respect, we point out that results of the present paper immediately imply that consistent query answering of Boolean ground unions of conjunctive queries is tractable. The same analysis is being carried out over other DLs that allow for tractable reasoning [2]. Finally, we are also studying the problem of consistent query answering over more expressive DLs.

**Acknowledgments.** This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU under contract number FP6-7603, by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant, and by MIUR FIRB 2005 project “Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet” (TOCAL.IT). We finally thank Riccardo Rosati for fruitful discussions on the matter of the present paper, and his helpful comments.

## References

1. M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS'99*, pages 68–79, 1999.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the  $\mathcal{EL}$  envelope. In *Proc. of IJCAI 2005*, pages 364–369, 2005.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
4. L. E. Bertossi, A. Hunter, and T. Schaub, editors. *Inconsistency Tolerance*, volume 3300 of LNCS. Springer, 2005.
5. M. Bouzeghoub and M. Lenzerini. Introduction to the special issue on data extraction, cleaning, and reconciliation. *Information Systems*, 26(8):535–536, 2001.
6. A. Cali, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS 2003*, pages 260–271, 2003.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, pages 602–607, 2005.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
9. L. Grieco, D. Lembo, M. Ruzzi, and R. Rosati. Consistent query answering under key and exclusion dependencies: Algorithms and experiments. In *Proc. of CIKM 2005*, pages 792–799, 2005.
10. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From  $SHIQ$  and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.
11. Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI 2003*, 2003.
12. D. Lembo and M. Ruzzi. Consistent query answering over description logic ontologies. In *Proc. of the 1st Int. Conf. on Web Reasoning and Rules System (RR 2007)*, 2007. To appear. Available at: <http://www.dis.uniroma1.it/~ruzzi/full.pdf>.
13. M. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in  $SHIQ$ . In *Proc. of DL 2006*. CEUR Electronic Workshop Proceedings, <http://ceur-ws.org/Vol-189/>, 2006.
14. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proc. of the 14th Int. World Wide Web Conf. (WWW 2005)*, 2005.
15. N. Shadbolt, W. Hall, and T. Berners-Lee. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

## Exploiting Description Logic Reasoners in Inductive Logic Programming Systems: An Experience within the Semantic Web area

Francesca A. Lisi

Dipartimento di Informatica, Università degli Studi di Bari,  
Via E. Orabona 4, 70125 Bari, Italy  
lisi@di.uniba.it

**Abstract.** In spite of the increasing effort spent on building ontologies for the Semantic Web, little attention has been paid to the impact of these ontologies on knowledge-based intelligent systems such as Inductive Logic Programming (ILP) systems which were not conceived for dealing with DL knowledge bases. In this paper, we describe an extension of the ILP system  $\mathcal{AL}$ -QUIN to deal with a background knowledge in the form of OWL DL ontology. The extension consists of a preprocessing of the ontology that mainly relies on the services of the DL reasoner Pellet.

### 1 Introduction

Description Logics (DLs) are the most currently used among the logical formalisms proposed by Ontological Engineering [5]. Also the DL-based approach to Ontological Engineering is playing a relevant role in the definition of the Semantic Web. The Semantic Web is the vision of the World Wide Web enriched by machine-processable information which supports the user in his tasks [2]. Its architecture consists of several layers, each of which is equipped with an ad-hoc mark-up language. DLs, more precisely the very expressive DL  $\mathcal{SHIQ}$ , have guided the design of the mark-up language OWL for the *ontological layer* [6]. A DL reasoner, Pellet [16], has been recently proposed for OWL. In spite of the increasing effort spent on building ontologies for the Semantic Web, little attention has been paid to the impact of these ontologies on knowledge-based intelligent systems such as Inductive Logic Programming (ILP) systems which were not conceived for dealing with DL knowledge bases. Note that the use of background knowledge has been widely recognized as one of the strongest points of ILP when compared to other forms of concept learning and has been empirically studied in several application domains [11]. Yet the background knowledge in ILP systems is often not organized around a well-formed conceptual model and still ignores the latest developments in Knowledge Engineering such as ontologies and ontology languages based on DLs. In a recent position paper, Page and Srinivasan have pointed out that the use of special-purpose reasoners in ILP is among the pressing issues that have arisen from the most challenging ILP applications of today [12]. We think that this is the case for ILP applications in the

Semantic Web area. In this paper we report on an experience with DL reasoners in ILP within the Semantic Web application area. In particular, we describe an extension of the ILP system  $\mathcal{AL}$ -QUIN [10] to deal with a background knowledge in the form of OWL DL ontology. The extension consists of a preprocessing of the ontology that mainly relies on the reasoning services of Pellet.

The paper is structured as follows. Section 2 briefly describes  $\mathcal{AL}$ -QUIN. Section 3 illustrates the use of Pellet in  $\mathcal{AL}$ -QUIN. Section 4 concludes the paper.

## 2 The ILP system $\mathcal{AL}$ -QuIn

The ILP system  $\mathcal{AL}$ -QUIN ( $\mathcal{AL}$ -log Query Induction) [10] supports a data mining task known under the name of *frequent pattern discovery*. In data mining a *pattern* is considered as an intensional description (expressed in a given language  $\mathcal{L}$ ) of a subset of a given data set  $\mathbf{r}$ . The *support* of a pattern is the relative frequency of the pattern within  $\mathbf{r}$  and is computed with the evaluation function *supp*. The task of frequent pattern discovery aims at the extraction of all *frequent* patterns, i.e. all patterns whose support exceeds a user-defined threshold of *minimum support*.  $\mathcal{AL}$ -QUIN solves a variant of the frequent pattern discovery problem which takes concept hierarchies into account during the discovery process, thus yielding descriptions at multiple granularity levels up to a maximum level *maxG*. More formally, given

- a data set  $\mathbf{r}$  including a taxonomy  $\mathcal{T}$  where a reference concept  $C_{ref}$  and task-relevant concepts are designated,
- a multi-grained language  $\{\mathcal{L}^l\}_{1 \leq l \leq maxG}$  of patterns
- a set  $\{minsup^l\}_{1 \leq l \leq maxG}$  of user-defined minimum support thresholds

the problem of *frequent pattern discovery at  $l$  levels of description granularity*,  $1 \leq l \leq maxG$ , is to find the set  $\mathcal{F}$  of all the patterns  $P \in \mathcal{L}^l$  that describe the reference concept w.r.t. the task-relevant concepts and turn out to be frequent in  $\mathbf{r}$ . Note that  $P$ 's with support  $s$  such that (i)  $s \geq minsup^l$  and (ii) all ancestors of  $P$  w.r.t.  $\mathcal{T}$  are frequent in  $\mathbf{r}$ . Note that a pattern  $Q$  is considered to be an ancestor of  $P$  if it is a coarser-grained version of  $P$ .

*Example 1.* As a showcase we consider the task of finding frequent patterns that describe Middle East countries (reference concept) w.r.t. the religions believed and the languages spoken (task-relevant concepts) at three levels of granularity ( $maxG = 3$ ). Minimum support thresholds are set to the following values:  $minsup^1 = 20\%$ ,  $minsup^2 = 13\%$ , and  $minsup^3 = 10\%$ . The data set and the language of patterns will be illustrated in Example 2 and 3, respectively.

In  $\mathcal{AL}$ -QUIN data and patterns are represented according to the hybrid knowledge representation and reasoning system  $\mathcal{AL}$ -log [4]. In particular, the data set  $\mathbf{r}$  is represented as an  $\mathcal{AL}$ -log knowledge base  $\mathcal{B}$ , thus composed of a structural part and a relational part. The structural subsystem  $\Sigma$  is based on  $\mathcal{ALC}$  [14] whereas the relational subsystem  $\Pi$  is based on an extended form of DATALOG [3] that is obtained by using  $\mathcal{ALC}$  concept assertions essentially as type constraints on variables.

*Example 2.* For the task of interest, we consider an  $\mathcal{AL}$ -log knowledge base  $\mathcal{B}_{CIA}$  that integrates a  $\mathcal{ALC}$  component  $\Sigma_{CIA}$  containing taxonomies rooted into the concepts **Country**, **EthnicGroup**, **Language** and **Religion** and a DATALOG component  $\Pi_{CIA}$  containing facts<sup>1</sup> extracted from the on-line 1996 CIA World Fact Book<sup>2</sup>. Note that Middle East countries have been defined as Asian countries that host at least one Middle Eastern ethnic group:

$\text{MiddleEastCountry} \equiv \text{AsianCountry} \sqcap \exists \text{Hosts.MiddleEastEthnicGroup}$ .

In particular, Armenia ('ARM') and Iran ('IR') are classified as Middle East countries because the following membership assertions hold in  $\Sigma_{CIA}$ :

```
'ARM':AsianCountry.
'IR':AsianCountry.
'Arab':MiddleEastEthnicGroup.
'Armenian':MiddleEastEthnicGroup.
<'ARM','Armenian':>:Hosts.
<'IR','Arab':>:Hosts.
```

Also  $\Pi_{CIA}$  includes constrained DATALOG clauses such as:

```
believes(Code, Name) ←
    religion(Code, Name, Percent) & Code:Country, Name:Religion.
speaks(Code, Name) ←
    language(Code, Name, Percent) & Code:Country, Name:Language.
```

that define views on the relations **religion** and **language**, respectively.

The language  $\mathcal{L} = \{\mathcal{L}^l\}_{1 \leq l \leq \max G}$  of patterns allows for the generation of  $\mathcal{AL}$ -log unary conjunctive queries, called  $\mathcal{O}$ -queries. Given a reference concept  $C_{ref}$ , an  $\mathcal{O}$ -query  $Q$  to an  $\mathcal{AL}$ -log knowledge base  $\mathcal{B}$  is a (linked and connected)<sup>3</sup> constrained DATALOG clause of the form

$$Q = q(X) \leftarrow \alpha_1, \dots, \alpha_m \& X : C_{ref}, \gamma_1, \dots, \gamma_n$$

where  $X$  is the *distinguished variable* and the remaining variables occurring in the body of  $Q$  are the *existential variables*. Note that  $\alpha_j$ ,  $1 \leq j \leq m$ , is a DATALOG literal whereas  $\gamma_k$ ,  $1 \leq k \leq n$ , is an assertion that constrains a variable already appearing in any of the  $\alpha_j$ 's to vary in the range of individuals of a concept defined in  $\mathcal{B}$ . The  $\mathcal{O}$ -query

$$Q_t = q(X) \leftarrow \& X : C_{ref}$$

is called *trivial* for  $\mathcal{L}$  because it only contains the constraint for the *distinguished variable*  $X$ . Furthermore the language  $\mathcal{L}$  is *multi-grained*, i.e. it contains expressions at multiple levels of description granularity. Indeed it is implicitly defined

<sup>1</sup> <http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-rel-facts.flp>

<sup>2</sup> <http://www.odci.gov/cia/publications/factbook/>

<sup>3</sup> For the definition of linkedness and connectedness see [11].

by a *declarative bias specification* which consists of a finite alphabet  $\Delta$  of DATA-LOG predicate names and finite alphabets  $\Gamma^l$  (one for each level  $l$  of description granularity) of  $\mathcal{ALC}$  concept names. Note that the  $\alpha_i$ 's are taken from  $\mathcal{A}$  and  $\gamma_j$ 's are taken from  $\Gamma^l$ . We impose  $\mathcal{L}$  to be finite by specifying some bounds, mainly  $maxD$  for the maximum depth of search and  $maxG$  for the maximum level of granularity.

*Example 3.* To accomplish the task of Example 1 we define  $\mathcal{L}_{CIA}$  as the set of  $\mathcal{O}$ -queries with  $C_{ref} = \text{MiddleEastCountry}$  that can be generated from the alphabet  $\Delta = \{\text{believes}/2, \text{speaks}/2\}$  of DATA-LOG binary predicate names, and the alphabets

$$\begin{aligned} \Gamma^1 &= \{\text{Language}, \text{Religion}\} \\ \Gamma^2 &= \{\text{IndoEuropeanLanguage}, \dots, \text{MonotheisticReligion}, \dots\} \\ \Gamma^3 &= \{\text{IndoIranianLanguage}, \dots, \text{MuslimReligion}, \dots\} \end{aligned}$$

of  $\mathcal{ALC}$  concept names for  $1 \leq l \leq 3$ , up to  $maxD = 5$ . Examples of  $\mathcal{O}$ -queries in  $\mathcal{L}_{CIA}$  are:

$$\begin{aligned} Q_t &= \text{q}(X) \leftarrow \& X:\text{MiddleEastCountry} \\ Q_1 &= \text{q}(X) \leftarrow \text{speaks}(X, Y) \& X:\text{MiddleEastCountry}, Y:\text{Language} \\ Q_2 &= \text{q}(X) \leftarrow \text{speaks}(X, Y) \& X:\text{MiddleEastCountry}, Y:\text{IndoEuropeanLanguage} \\ Q_3 &= \text{q}(X) \leftarrow \text{believes}(X, Y) \& X:\text{MiddleEastCountry}, Y:\text{MuslimReligion} \end{aligned}$$

where  $Q_t$  is the trivial  $\mathcal{O}$ -query for  $\mathcal{L}_{CIA}$ ,  $Q_1 \in \mathcal{L}_{CIA}^1$ ,  $Q_2 \in \mathcal{L}_{CIA}^2$ , and  $Q_3 \in \mathcal{L}_{CIA}^3$ . Note that  $Q_1$  is an ancestor of  $Q_2$ .

The *support* of an  $\mathcal{O}$ -query  $Q \in \mathcal{L}^l$  w.r.t an  $\mathcal{AL}$ -log knowledge base  $\mathcal{B}$  is defined as

$$supp(Q, \mathcal{B}) = | \text{answerset}(Q, \mathcal{B}) | / | \text{answerset}(Q_t, \mathcal{B}) |$$

where  $\text{answerset}(Q, \mathcal{B})$  is the set of correct answers to  $Q$  w.r.t.  $\mathcal{B}$ . An *answer* to  $Q$  is a ground substitution  $\theta$  for the distinguished variable of  $Q$ . An answer  $\theta$  to  $Q$  is a *correct (resp. computed) answer* w.r.t.  $\mathcal{B}$  if there exists at least one correct (resp. computed) answer to  $body(Q)\theta$  w.r.t.  $\mathcal{B}$ . Thus the computation of support relies on query answering in  $\mathcal{AL}$ -log.

*Example 4.* The pattern  $Q_2$  turns out to be frequent because it has support  $supp(Q_2, \mathcal{B}_{CIA}) = (2/15)\% = 13.3\% (\geq minsup^2)$ . It is to be read as '13.3 % of Middle East countries speak an Indoeuropean language'. The two correct answers to  $Q_2$  w.r.t.  $\mathcal{B}_{CIA}$  are 'ARM' and 'IR'.

### 3 Exploiting Pellet in $\mathcal{AL}$ -QuIn

#### 3.1 Coverage of observations

In ILP the evaluation of inductive hypotheses (like candidate patterns in frequent pattern discovery) w.r.t. a set of observations (data units) is usually referred to as

the *coverage test* because it checks which observations satisfy (are covered by) the hypothesis. Since evaluation is the most computationally expensive step when inducing hypotheses expressed in (fragments of) first-order logic, an appropriate choice of representation for observations can help speeding up this step. In  $\mathcal{AL}$ -QUIN the extensional part of  $\Pi$  is partitioned into portions  $\mathcal{A}_i$  each of which refers to an individual  $a_i$  of  $C_{ref}$ . The link between  $\mathcal{A}_i$  and  $a_i$  is represented with the DATALOG literal  $q(a_i)$ . The pair  $(q(a_i), \mathcal{A}_i)$  is called *observation*.

*Example 5.* By assuming `MiddleEastCountry` as reference concept, the observation  $\mathcal{A}_{ARM}$  contains DATALOG facts such as

```
language('ARM', 'Armenian', 96).
language('ARM', 'Russian', 2).
```

concerning the individual 'ARM' whereas  $\mathcal{A}_{IR}$  consists of facts like

```
language('IR', 'Turkish', 1).
language('IR', 'Kurdish', 9).
language('IR', 'Baloch', 1).
language('IR', 'Arabic', 1).
language('IR', 'Luri', 2).
language('IR', 'Persian', 58).
language('IR', 'Turkic', 26).
```

related to the individual 'IR'.

In ILP the coverage test must take the background knowledge into account. The portion  $\mathcal{K}$  of  $\mathcal{B}$  which encompasses the whole  $\Sigma$  and the intensional part (IDB) of  $\Pi$  is considered as *background knowledge* for  $\mathcal{AL}$ -QUIN. Therefore proving that an  $\mathcal{O}$ -query  $Q$  covers an observation  $(q(a_i), \mathcal{A}_i)$  w.r.t.  $\mathcal{K}$  equals to proving that  $\theta_i = \{X/a_i\}$  is a correct answer to  $Q$  w.r.t.  $\mathcal{B}_i = \mathcal{K} \cup \mathcal{A}_i$ .

*Example 6.* Checking whether  $Q_2$  covers the observation  $(q('ARM'), \mathcal{A}_{ARM})$  w.r.t.  $\mathcal{K}_{CIA}$  is equivalent to answering the query

$$Q_2^{(0)} = \leftarrow q('ARM')$$

w.r.t.  $\mathcal{K}_{CIA} \cup \mathcal{A}_{ARM} \cup Q_2$ . The coverage test for  $(q('IR'), \mathcal{A}_{IR})$  is analogous.

A common practice in ILP is to use a reformulation operator, called *saturation* [13], to speed-up the coverage test. It enables ILP systems to make background knowledge explicit within the observations instead of implicit and apart from the observations. In the following we will discuss the implementation of the coverage test in  $\mathcal{AL}$ -QUIN and clarify the role of Pellet in supporting the saturation of observations w.r.t. a OWL-DL background knowledge  $\Sigma$ .

### 3.2 Saturation and instance retrieval

$\mathcal{AL}$ -QUIN is implemented with Prolog as usual in ILP. Thus, the *actual* representation language in  $\mathcal{AL}$ -QUIN is a kind of DATALOG<sup>OI</sup> [15], i.e. the subset of

DATALOG<sup>≠</sup> equipped with an equational theory that consists of the axioms of Clark’s Equality Theory augmented with one rewriting rule that adds *inequality atoms*  $s \neq t$  to any  $P \in \mathcal{L}$  for each pair  $(s, t)$  of distinct terms occurring in  $P$ . Note that concept assertions are rendered as *membership atoms*, e.g.  $a : C$  becomes  $c.C(a)$ .

*Example 7.* The following query

```
q(X) ← c.MiddleEastCountry(X), believes(X,Y), c.MonotheisticReligion(Y),
       believes(X,Z), Y≠Z
```

is the DATALOG<sup>OI</sup> rewriting of:

```
q(X) ← believes(X,Y), believes(X,Z) &
       X:MiddleEastCountry, Y:MonotheisticReligion
```

where the absence of a  $\mathcal{ALC}$  constraint for the variable Z explains the need for the inequality atom.

When implementing the coverage test in  $\mathcal{AL}$ -QUIN, the goal has been to reduce constrained SLD-resolution of  $\mathcal{AL}$ -log to SLD-resolution on DATALOG<sup>OI</sup>. A crucial issue in this mapping is to deal with the satisfiability tests of  $\mathcal{ALC}$  constraints w.r.t.  $\Sigma$  which are required by constrained SLD-resolution because they are performed by applying the tableau calculus for  $\mathcal{ALC}$ . The reasoning on the constraint part of  $\mathcal{O}$ -queries has been replaced by preliminary saturation steps of the observations w.r.t. the background knowledge  $\Sigma$ . By doing so, the observations are completed with concept assertions that can be derived from  $\Sigma$  by posing *instance retrieval* problems to a DL reasoner. Here, the retrieval is called *levelwise* because it follows the layering of  $\mathcal{T}$ : individuals of concepts belonging to the  $l$ -th layer  $\mathcal{T}^l$  of  $\mathcal{T}$  are retrieved all together. Conversely the retrieval for the *reference concept* is made only once at the beginning of the whole discovery process because it makes explicit knowledge of interest to all the levels of granularity. This makes SLD-refutations of queries in  $\mathcal{L}^l$  work only on extensional structural knowledge at the level  $l$  of description granularity.

A Java application, named OWL2DATALOG, has been developed to support the saturation of observations w.r.t. a OWL-DL background knowledge  $\Sigma$  in  $\mathcal{AL}$ -QUIN. To achieve this goal, it supplies the following functionalities:

- levelwise retrieval w.r.t.  $\Sigma$
- DATALOG<sup>OI</sup> rewriting of (asserted and derived) concept assertions of  $\Sigma$

Note that the former is implemented by a client for the DIG server Pellet.

*Example 8.* The DATALOG<sup>OI</sup> rewriting of the concept assertions derived for  $\mathcal{T}^2$  produces facts like:

```
c_AfroAsiaticLanguage('Arabic').
...
c_IndoEuropeanLanguage('Armenian').
...
c_MonotheisticReligion('ShiaMuslim').
...
```

to be considered during coverage tests of  $\mathcal{O}$ -queries in  $\mathcal{L}^2$ .

The concept assertions, once translated to  $\text{DATALOG}^{OI}$ , are added to the facts derived from the IDB of  $\Pi$  at the loading of each observation. The coverage test therefore concerns  $\text{DATALOG}^{OI}$  rewritings of both  $\mathcal{O}$ -queries and saturated observations.

*Example 9.* The  $\text{DATALOG}^{OI}$  rewriting

$q(X) \leftarrow \text{c\_MiddleEastCountry}(X), \text{speaks}(X, Y), \text{c\_IndoEuropeanLanguage}(Y)$

of  $Q_2$  covers the  $\text{DATALOG}^{OI}$  rewriting:

$\text{c\_MiddleEastCountry}('ARM')$ .  
 $\text{speaks}('ARM', 'Armenian')$ .  
 $\dots$   
 $\text{c\_IndoEuropeanLanguage}('Armenian')$ .  
 $\dots$

of the saturated observation  $\hat{\mathcal{A}}_{ARM}$ .

Note that the translation from OWL-DL to  $\text{DATALOG}^{OI}$  is possible because we assume that *all* the concepts are named. This means that an equivalence axiom is required for each complex concept in the knowledge base. Equivalence axioms help keeping concept names (used within constrained  $\text{DATALOG}$  clauses) independent from concept definitions.

## 4 Final remarks

In this paper we have shown how to exploit DL reasoners to make existing ILP systems compliant with the latest developments in Ontological Engineering. We would like to emphasize that  $\mathcal{AL}$ -QUIN was originally conceived to deal with background knowledge in the form of  $\mathcal{ALC}$  taxonomic ontologies but the implementation of this feature was still lacking<sup>4</sup>. Therefore, Pellet makes  $\mathcal{AL}$ -QUIN fulfill its design requirements. More precisely, the instance retrieval problems solved by Pellet support the saturation phase in  $\mathcal{AL}$ -QUIN. Saturation then compiles DL-based background knowledge down to the usual  $\text{DATALOG}$ -like formalisms of ILP systems. In this respect, the pre-processing method proposed in [8] to enable legacy ILP systems to work within the framework of CARIN [9] is related to ours but it lacks an implementation. Analogously, the method proposed in [7] for translating OWL-DL to disjunctive  $\text{DATALOG}$  is far too general with respect to the specific needs of our application. Rather, the proposal of interfacing existing reasoners to combine ontologies and rules [1] is more similar to ours in the spirit. For the future we intend to compare  $\mathcal{AL}$ -QUIN with other ILP systems able to deal with ontological background knowledge as soon as they are implemented and deployed.

<sup>4</sup>  $\mathcal{AL}$ -QUIN could actually deal only with concept hierarchies in  $\text{DATALOG}^{OI}$ .

## References

1. U. Assmann, J. Henriksson, and J. Maluszynski. Combining safe rules and ontologies by interfacing of reasoners. In J.J. Alferes, J. Bailey, W. May, and U. Schertel, editors, *Principles and Practice of Semantic Web Reasoning*, volume 4187 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2006.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May, 2001.
3. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.
4. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf.  $\mathcal{AL}$ -log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
5. A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2004.
6. I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
7. U. Hustadt, B. Motik, and U. Sattler. Reducing *SHIQ*-description logic to disjunctive datalog programs. In D. Dubois, C.A. Welty, and M.-A. Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 152–162. AAAI Press, 2004.
8. J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, pages 117–132. Springer, 2003.
9. A.Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
10. F.A. Lisi. Data Mining in Hybrid Languages with ILP. In D. Calvanese, G. De Giacomo, and E. Franconi, editors, *Proc. 2003 International Workshop on Description Logics*. <http://SunSITE.Informatik.RWTH-Aachen.de/Publications/CEUR-WS/Vol-81/lisi.pdf>, 2003.
11. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
12. D. Page and A. Srinivasan. ILP: A short look back and a longer look forward. *Journal of Machine Learning Research*, 4:415–430, 2003.
13. C. Rouveirol. Flattening and saturation: Two representation changes for generalization. *Machine Learning*, 14(1):219–232, 1994.
14. M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
15. G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli. A logic framework for the incremental inductive synthesis of Datalog theories. In N.E. Fuchs, editor, *Proceedings of 7th International Workshop on Logic Program Synthesis and Transformation*, volume 1463 of *Lecture Notes in Computer Science*, pages 300–321. Springer, 1998.
16. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 2006.

# Extracting Ontologies from Relational Databases

Lina Lubyte and Sergio Tessaris

Faculty of Computer Science – Free University of Bozen-Bolzano

## 1 Introduction

The use of a conceptual model or an ontology over data sources has been shown to be necessary to overcome many important database problems (for a survey see [1]). Since ontologies provide a conceptual view of the application domain, the recent trend to employ such ontologies for navigational (and reasoning) purposes when accessing the data gives additional motivation for the problem of extracting the ontology from database schema [2]. When such an ontology exists, modelling the relation between the data sources and an ontology is a crucial aspect in order to capture the semantics of the data.

In this paper we define the framework for extracting from a relational database an ontology that is to be used as a conceptual view over the data, where the semantic mapping between the database schema and the ontology is captured by associating a view over the source data to each element of the ontology. Thus, the vocabulary over the ontology can be seen as a set of (materialised) views over the vocabulary of the data source; i.e., a technique known as GAV approach in the information integration literature [3]. To describe the extracted conceptual model, we provide an expressive ontology language which can capture features from Entity-Relationship and UML class diagrams, as well as variants of Description Logics. The heuristics underlying the ontology extraction process are based on ideas of standard relational schema design from ER diagrams in order to uncover the connections between relational constructs and those of ontologies. Besides the latter assumption the procedure presented in this paper takes into consideration relations being in third normal form (3NF). Under this assumption we can formally prove that the conversion preserves the semantics of the constraints in the relational database. Therefore, there is no data loss, and the extracted model constitutes a faithful wrapper of the relational database.

## 2 Preliminaries

We assume that the reader is familiar with standard relational database notions as presented, for example, in [4]. We assume that the database domain is a fixed denumerable set of elements  $\Delta$  and that every such element is denoted uniquely by a constant symbol, called its *standard name* [5]. We make use of the standard notion of relational model by using named attributes, each with an associated datatype, instead of tuples.

A *relational schema*  $\mathcal{R}$  is a set of relationships, each one with a fixed set of attributes (assumed to be pairwise distinct) with associated datatypes. We

use  $[s_1 : D_1, \dots, s_n : D_n]$  to denote that a relationship has attributes  $s_1, \dots, s_n$  with associated data types  $D_1, \dots, D_n$ . We interpret relationships over a fixed countable domain  $\Delta$  of datatype elements, which we consider partitioned into the datatypes  $D_i$ . A *database instance* (or simply *database*)  $\mathcal{D}$  over a relational schema  $\mathcal{R}$  is an (interpretation) function that maps each relationship  $R$  in  $\mathcal{R}$  into a set  $R^{\mathcal{D}}$  of total functions from the set of attributes of  $R$  to  $\Delta$ . Let  $A = [s_1, \dots, s_m]$  be a sequence of  $m$  attribute names of a relationship  $R$  of a schema  $\mathcal{R}$ . The *projection* of  $R^{\mathcal{D}}$  over  $A$  is the relation  $\pi_A R^{\mathcal{D}} \subseteq \Delta_D^m$ , satisfying the condition that  $\phi \in R^{\mathcal{D}}$  iff  $(\phi(s_1), \dots, \phi(s_m)) \in \pi_A R^{\mathcal{D}}$ .

The ontology extraction task takes as input a relational source; e.g. a DBMS. We abstract from any specific database implementation by considering an abstract *relational source*  $\mathcal{DB}$ , which is a pair  $(\mathcal{R}, \Sigma)$ , where  $\mathcal{R}$  is a relational schema and  $\Sigma$  is a set of *integrity constraints*. The semantics of relational schemata is provided in the usual way by means of the relational model. Below we briefly list the kind of database integrity constraints we consider in our framework (for more details the reader is referred to [6]). *Nulls-not-allowed constraints*: satisfied in a database when null are not contained in any indicated attribute. *Unique constraints*: satisfied when the sequence of attributes are unique in a relation. Together with nulls-not-allowed constraints they correspond to *key constraints*. *Inclusion dependencies*: satisfied when the projection of two relations are included one in the other. When the attributes of the target relation are a candidate key as well, we call them *foreign key constraints*. *Exclusion dependencies*: satisfied when the intersection of the projection of two relations is the empty set. *Covering constraints*:<sup>1</sup> between a relation and a set of relations, satisfied when the projection of the relation over the specified attributes is included in the union of the projections of the relations in the set.

We call a *DLR-DB system*  $\mathcal{S}$  a triple  $\langle \mathcal{R}, \mathcal{P}, \mathcal{K} \rangle$ , where  $\mathcal{R}$  is a *relational schema*,  $\mathcal{P}$  is a *component structure* over  $\mathcal{R}$ , and  $\mathcal{K}$  is a set of assertions involving names in  $\mathcal{R}$ . The intuition behind a named component is the role name of a relationship in an ER schema (or UML class-diagram). The *component structure*  $\mathcal{P}$  associates to each relationship a mapping from *named components* to sequences of attributes. Let  $R$  be a relationship in  $\mathcal{R}$ , to ease the notation we write  $\mathcal{P}_R$  instead of  $\mathcal{P}(R)$ .

Let  $R$  be a relationship in  $\mathcal{R}$ , with attributes  $[s_1 : D_1, \dots, s_n : D_n]$ .  $\mathcal{P}_R$  is a non-empty (partial) function from a set of named components to the set of nonempty sequences of attributes of  $R$ . The domain of  $\mathcal{P}_R$ , denoted  $\mathcal{C}_R$ , is called the set of *components of  $R$* . For a named component  $c \in \mathcal{C}_R$ , the sequence  $\mathcal{P}_R(c) = [s_{i_1}, \dots, s_{i_m}]$ , where each  $i_j \in \{1, \dots, n\}$ , is called the  *$c$ -component of  $R$* . We require that the sequences of attributes for two different named components are not overlapping, and that each attribute appears at most once in each sequence. The *signature* of a component  $\mathcal{P}_R(c)$ , denoted  $\tau(\mathcal{P}_R(c))$ , is the sequence of types of the attributes of the component. Two components  $\mathcal{P}_R(c_1)$  and  $\mathcal{P}_R(c_2)$  are *compatible* if the two signatures  $\tau(\mathcal{P}_R(c_1))$  and  $\tau(\mathcal{P}_R(c_2))$  are equal.

<sup>1</sup> In ER terminology, this may also be indicated as *mandatory* for an IS-A relationship.

The *DLR-DB* ontology language, used to express the assertions in  $\mathcal{K}$ , is based on the idea of modelling the domain by means of *axioms* involving the projection of the relationship over the named components. An *atomic formula* is a projection of a relationship  $R$  over one of its components. The projection of  $R$  over the  $c$ -component is denoted by  $R[c]$ . When the relationship has a single component, then this can be omitted and the atomic formula  $R$  corresponds to its projection over the single component. Given the atomic formulae  $R[c], R'[c'], R_i[c_i]$ , an *axiom* is an assertion of the form specified below; where all the atomic formulae involved in the same axiom must be compatible. The semantics is provided in terms of relational models for  $\mathcal{R}$ , where  $\mathcal{K}$  plays the role of constraining the set of “admissible” models.

$R[c] \sqsubseteq R'[c'] \quad \pi_c R^{\mathcal{D}} \subseteq \pi_{c'} R'^{\mathcal{D}}$	Subclass
$R[c] \text{ disj } R'[c'] \quad \pi_c R^{\mathcal{D}} \cap \pi_{c'} R'^{\mathcal{D}} = \emptyset$	Disjointness
$\text{funct}(R[c])$ for all $\phi_1, \phi_2 \in R^{\mathcal{D}}$ with $\phi_1 \neq \phi_2$ , we have $\phi_1(s) \neq \phi_2(s)$ for some $s$ in $c$	Functionality
$R_1[c_1], \dots, R_k[c_k] \text{ cover } R[c] \quad \pi_c R^{\mathcal{D}} \subseteq \bigcup_{i=1..k} \pi_{c_i} R_i^{\mathcal{D}}$	Covering

A database  $\mathcal{D}$  is said to be a *model* for  $\mathcal{K}$  if it satisfies all its axioms, and for each relationship  $R$  in  $\mathcal{R}$  with components  $c_1, \dots, c_k$ , for any  $\phi_1, \phi_2 \in R^{\mathcal{D}}$  with  $\phi_1 \neq \phi_2$ , there is some  $s$  in  $c_i$  s.t.  $\phi_1(s) \neq \phi_2(s)$ . The above conditions are well defined because we assumed the compatibility of the atomic formulae involved in the constraints.

The *DLR-DB* ontology language enables the use of the most commonly used constructs in conceptual modelling (see [7]). Note that by taking away the covering axioms and considering only components containing single attributes this ontology language corresponds exactly to *DLR-Lite* (see [8]). By virtue of the assumption that components do not share attributes, it is not difficult to show that the same reasoning mechanism of *DLR-Lite* can be used in our case. The discussion on the actual reasoning tasks which can be employed in the context of *DLR-DB* systems is out of the scope of this paper. Herewith we are mainly interested of the use of the language to express data models extracted from the relational data sources.

### 3 Ontology Extraction

Our proposed ontology extraction algorithm works in two phases. Firstly, a classification scheme for relations from the relational source is derived. Secondly, based on this classification, the ontology describing the data source is extracted. Moreover, the process generates a set of view definitions, expressing the mapping between the database schema and the ontology. In this section we briefly sketch the procedure, more details and the algorithms can be found in [6].

The principles upon our technique are based on best practices on relational schema design from ER diagrams – a standard database modelling technique [9]. One benefit of this approach is that it can be shown that our algorithm, though heuristic in general, is able to reconstruct the original ER diagram under some

assumptions on the latter. Specifically, we consider ER models that support entities with attributes,<sup>2</sup>  $n$ -ary relationships which are subject to cardinality constraints, and inheritance hierarchies (IS-A) between entities (including multiple inheritance) which may be constrained to be disjoint or covering. Roughly speaking, we reverse the process of translating ER model to relational model. As a result, we identify that relations representing entities have keys which are not part of their foreign keys, and every such foreign key represents functional binary relationship (i.e., one-to-one or one-to-many) with another entity. On the other hand, relations that correspond to  $n$ -ary relationships with cardinalities “many” for all participating entities have keys composed of their foreign keys. Since we assume there is no IS-A between relationships, every such foreign key references key of a relation resulting from (sub-)entity. When a relation has key that is also its foreign key, and no other non-key foreign keys appear in that relation, then, clearly, an inheritance relationship exists. If instead non-key foreign keys are present but the relation is the target of some foreign key, we are sure that this relation corresponds to sub-entity. Otherwise, such relation might also “look like” functional relationship (binary or  $n$ -ary), mapped directly to a relation, and therefore relations of this type are classified as ambiguous relations (see below).

The classification of the relations, based on their keys and foreign keys, can be summarised as: *base relation* when the primary key is disjoint with every foreign key; *specific relation* when the primary key is also a foreign key and it has a single foreign key, or it is referred to by some relation;<sup>3</sup> *relationship relation* when the primary key is composed by all the foreign keys, which are more than 1. Any other relation is classified as *ambiguous*.

Once the relations in  $\mathcal{DB}$  are classified according to the conditions defined above, then the actual ontology extraction process returns a *DLR-DB* system as output. For every base and specific relation  $r_i$  the algorithm generates a relationship  $R_i$  with the attributes in a one-to-one correspondence with non-foreign key attributes of  $r_i$ , and a single  $c$ -component, where  $\mathcal{P}_{R_i}(c)$  corresponds to the key attributes of  $r_i$  and thus functionality axiom  $\text{funct}(R_i[c])$  is added to  $\mathcal{K}$ ; a view is defined by projecting on all non-key foreign key attributes of  $r_i$ .

Once relationships for base and specific relations are defined, associations between those relationships must be identified. Specifically, a non-key foreign key in a relation  $r_i$  referencing relation,  $r_j$ , determines the association between relationships  $R_i$  and  $R_j$ . Thus, for each such foreign key, a relationship  $R_k$  is generated, having two components,  $c_i$ -component and  $c_j$ -component, where  $\mathcal{P}_{R_k}(c_i)$  and  $\mathcal{P}_{R_k}(c_j)$  correspond to key attributes of  $r_i$  and  $r_j$ , respectively, where  $c_i$ -component is functional, i.e., we have  $\text{funct}(R_k[c_i])$  in  $\mathcal{K}$ ; a corresponding view is defined by joining  $r_j$  with  $r_i$  and projecting on their keys. For expressing an association, determined by  $R_k$ , between  $R_i$  and  $R_j$  the axioms of the form  $R_k[c_i] \sqsubseteq R_i$  and  $R_k[c_j] \sqsubseteq R_j$  are added to  $\mathcal{K}$ . Furthermore, whenever the latter foreign key of  $r_i$  participates in a nulls-not-allowed constraint, the axiom  $R_i \sqsubseteq R_k[c_i]$  is generated stating mandatory participation for instances of  $R_i$  to

<sup>2</sup> We do not deal with multi-valued attributes in this paper.

<sup>3</sup> I.e., the relation appears on the right-hand side of some foreign key constraint.

$R_k$  as values for the  $c_i$ -component; its participation to a unique constraint determines instead the functionality axiom  $\text{funct}(R_k[c_j])$  meaning that every value of the  $c_j$ -component appears in it at most once; finally, appearance of the foreign key of  $r_i$  in the right-hand side of an inclusion dependency determines mandatory participation for values of the only component of  $R_j$  to the relationship  $R_k$  as values for the  $c_j$ -component, and thus the axiom  $R_j \sqsubseteq R_k[c_j]$  is added to  $\mathcal{K}$ . For expressing an ISA between classes, for every specific relation  $r_i$  the subclass axiom  $R_i \sqsubseteq R_j$  is added to  $\mathcal{K}$ , where  $R_j$  is the relationship corresponding to (base or specific) relation,  $r_j$ , that the key foreign key of  $r_i$  references. Additionally, each exclusion dependency on the set of specific relations induces the disjointness axioms  $R_i \text{ disj } R_k$ , for every pair of relations  $r_i, r_k$  appearing in the exclusion dependency. Similarly, every covering constraint on the set of specific relations induces the corresponding covering axiom in  $\mathcal{K}$ .

Each relationship relation  $r_i$  is accounted for by generating a relationship  $R_i$ , with attributes in a one-to-one correspondence with those of  $r_i$ , and  $n$  components, where  $n$  is the number of foreign keys of  $r_i$ . Each  $\mathcal{P}_{R_i}(c_{i_l})$  ( $l \in \{1, \dots, n\}$ ) has sequence of attributes corresponding to the  $l$ -th foreign key attributes of  $r_i$ ; the corresponding view is defined by projecting on all attributes of  $r_i$ . Then, for each foreign key of  $r_i$  referencing relation  $r_j$  (that is already represented with a relationship  $R_j$  having a single component), the algorithm generates an axiom  $R_i[c_{i_l}] \sqsubseteq R_j$  stating that the role corresponding to the  $c_{i_l}$ -component of  $R_i$  is of type  $R_j$ . Furthermore, if this foreign key appears on the right-hand side of an inclusion dependency, the axiom  $R_j \sqsubseteq R_i[c_{i_l}]$  is added to  $\mathcal{K}$  that states mandatory participation for instances of  $R_j$  to the relationship  $R_i$  as values for the  $c_{i_l}$  component.

Finally, the appropriate structures for ambiguous relations must be identified. As already discussed before, an ambiguous relation may correspond in ER schema to either sub-entity, which also participates with cardinality “one” in a binary relationship, or a functional relationship that was directly mapped to a relation. Following the idea that all functional binary relationships should be represented in a relational model with an embedded foreign key, e.g., in order to obtain the relational schema with a minimum number of relations, and that  $n$ -ary relationships ( $n \geq 3$ ) are relatively unusual, our heuristics “prefers” to recover an inheritance relationship, and thus the algorithm generates the structures corresponding to those defined for specific relations. On the other hand, a user could decide which is the “best” structure for ambiguous relations. In this way, the ontology extraction task may be a completely automated procedure, or semi-automated process with a user intervention.

As an example of the ontology extraction process, consider the relational schema (primary keys are underlined) with constraints of Figure 1. At the initial step of extraction process, relations Scholar, Publication and Department are classified as base relations, i.e. their keys and foreign keys do not share any attributes; IsAuthorOf relation is classified as relationship relation – its key is entirely composed from foreign keys; while relations PostDoc and Professor satisfy the conditions required for specific relations, i.e. the key ssn is their single foreign key.

Without going into details of the algorithm, in Table 1 we list the extracted relationships of *DLR-DB*  $\mathcal{R}$  together with the devised component structure  $\mathcal{P}$ , by considering the relation names and their corresponding attributes in the input relational source. Starting with base and specific relations, we have the corresponding relationships with single components. Since the component names for the latter relationships are not relevant (they can be omitted), we choose a common name *id* for all the five of them.<sup>4</sup> Figure 2 shows the extracted ontology together with the corresponding ER diagram.

Relationship	Component $c$	$\mathcal{P}_R(c)$	Additional attr.	View definition
Scholar	id	ssn	name	$\pi_{ssn,name}(Scholar_r)$
Publication	id	id	title, year	$\pi_{id,title,year}(Publication_r)$
Department	id	no	name	$\pi_{no,name}(Department_r)$
PostDoc	id	ssn	scholarship	$\pi_{ssn,scholarship}(PostDoc_r)$
Professor	id	ssn	salary	$\pi_{ssn,salary}(Professor_r)$
WorksFor	employee dept	ssn no		$\pi_{ssn,no}(Department_r \bowtie Scholar_r)$
IsAuthorOf	author publication	schSsn pubId		$\pi_{schSsn,pubId}(IsAuthorOf_r)$

Table 1. Extracted Schema.

#### 4 Discussion and Related Work

Much work has been addressed on the issue of explicitly defining semantics in database schemas [10, 7] and extracting semantics out of database schemas [11, 12]. The work described in [10] provides algorithms that investigate data instances of an existing legacy database in order to identify candidate keys of

<sup>4</sup> For the sake of clarity, the naming of the components for relationships WorksFor and IsAuthorOf, as well as the name of the WorksFor relationship itself, are determined by domain knowledge.

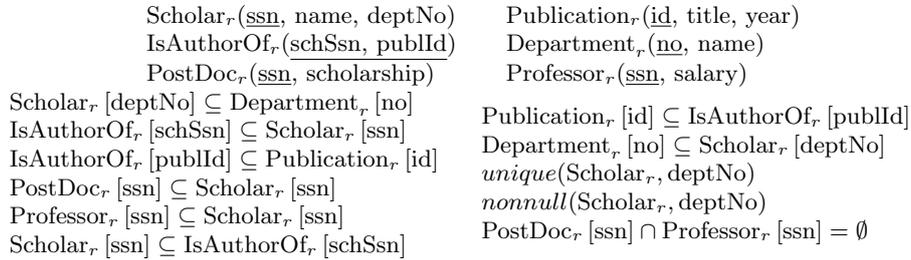


Fig. 1. Relational schema with constraints.

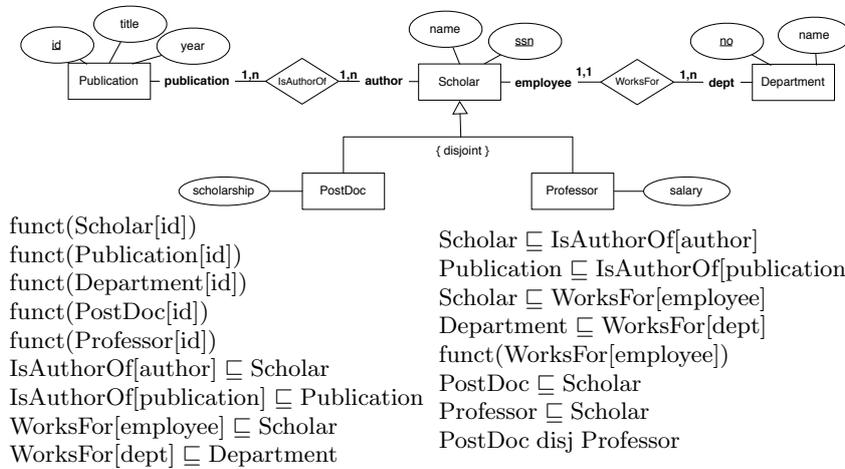


Fig. 2. Extracted ontology and corresponding ER diagram.

relations, to locate foreign keys, and to decide on the appropriate links between the given relations. As a result, user involvement is always required. In our work we instead assume the knowledge on key and foreign key constraints, as well as non null and unique values on attributes, inclusion and disjointness between relations, etc. exist in the schema.

The work in [12] propose transformations that are applied to produce the re-engineered schema and handles the establishment of inheritance hierarchies. However, it considers relations in BCNF and thus every relation is in a one-to-one correspondence with an object in the extracted schema. The main idea of the methodology described in [11] comes close to ours in the sense that it derives classification for relations and attributes based on heuristics of what kind of ER components would give rise to particular relations. Unlike the latter, our proposed technique can be seen as a *schema transformation* as defined in [15]. Because of lack of space, we omit the proof that this transformation is indeed *equivalence preserving* (for the actual proof and details see [6]).

The recent call for a Semantic Web arose several approaches in bringing together relational databases and ontologies. Among them we mention [13], where the authors describe an automatic mapping between relations and ontologies, when given as input simple correspondences from attributes of relations to datatype properties of classes in an ontology. Unlike our approach, it requires a target ontology onto which the relations are mapped to. On the other hand, the approach of [14] extracts the schema information of the data source and converts it into an ontology. However, the latter technique extracts only the structural information about the ontology, so the constraints are not taken into account.

This paper describes an heuristic procedure for extracting from relational database its conceptual view, where the wrapping of relational data sources by means of an extracted ontology is done by associating view over the original data to each element of the ontology. To represent the extracted ontology, instead of

a graphical notation, we employ an ontology language thus providing a precise semantics to extracted schema. Our extraction procedure relies on information from the database schema and automatically extracts all the relevant semantics if an input relational schema was designed using a standard methodology.

We are currently following several directions to continue the work reported in this paper. First, conceptual modelling constructs as multi-valued attributes and weak entities, alternative techniques for inheritance representation in relational tables. To this purpose we are starting to experiment with real database schemas to evaluate the quality of the extracted ontologies.

## References

1. Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hubner, S.: Ontology-based integration of information - a survey of existing approaches. In: Proc. of IJCAI-01 Workshop: Ontologies and Information Sharing. (2001) 108–117
2. Lembo, D., Lutz, C., Suntisrivaraporn, B.: Tasks for ontology design and maintenance. Deliverable D05, TONES EU-IST STREP FP6-7603 (2006)
3. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of PODS02. (2002) 233–346
4. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. (1995)
5. Levesque, H.J., Lakemeyer, G.: The Logic of Knowledge Bases. The MIT Press (2001)
6. Lubyte, L., Tessaris, S.: Extracting ontologies from relational databases. Technical report, Free University of Bozen-Bolzano (2007) Available at <http://www.inf.unibz.it/krdp/pub/index.php>.
7. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on uml class diagrams. Artificial Intelligence **168**(1) (2005) 70–118
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR 2006. (2006) 260–270
9. Batini, C., Ceri, S., Navathe, S.B.: Conceptual Database Design. An Entity-Relationship Approach. Benjamin/Cummings Publishing Company, Inc. (1992)
10. Alhajj, R.: Extracting an extended entity-relationship model from a legacy relational database. Information Systems **26**(6) (2003) 597–618
11. Chiang, R.H.L., Barron, T.M., Storey, V.C.: Reverse engineering of relational databases: extraction of an eer model from a relational database. Data and Knowledge Engineering **12**(2) (1994) 107–142
12. Markowitz, V.M., Makowsky, J.A.: Identifying extended entity-relationship object structures in relational schemas. IEEE Transactions on Software Engineering **16**(8) (1990) 777–790
13. An, Y., Borgida, A., Mylopoulos, J.: Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In: ODBASE’05. (2005) 1152–1169
14. Laborda, C.P., Conrad, S.: Bringing relational data into the semantic web using sparql and relational.owl. In: Proc. of the 22nd Int. Conf. on Data Engineering Workshops (ICDEW’06). (2006) 55–62
15. Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R.: The use of information capacity in schema integration and translation. In: Proc. of VLDB’93, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1993) 120–133

## Paraconsistent Resolution for Four-valued Description Logics <sup>\*</sup>

Yue Ma<sup>1,2</sup>, Pascal Hitzler<sup>2</sup>, and Zuoquan Lin<sup>1</sup>

<sup>1</sup>Department of Information Science, Peking University, China

<sup>2</sup>AIFB, Universität Karlsruhe, Germany

{mayue,lz}@is.pku.edu.cn, {yum,hitzler}@aifb.uni-karlsruhe.de

**Abstract.** In this paper, we propose an approach to translating any  $\mathcal{ALC}$  ontology (possible inconsistent) into a logically consistent set of disjunctive datalog rules. We achieve this in two steps: First we give a simple way to make any  $\mathcal{ALC}$  based ontology 4-valued satisfiable, and then we study a sound and complete paraconsistent ordered-resolution decision procedure for our 4-valued  $\mathcal{ALC}$ . Our approach can be viewed as a paraconsistent version of KAON2 algorithm.

### 1 Introduction

The study of inconsistency handling in description logics can be divided into two fundamentally different approaches. The first is based on the assumption that inconsistencies indicate erroneous data which is to be repaired in order to obtain a consistent knowledge base [1]. The other approach yields to the insight that inconsistencies are a natural phenomenon in realistic data which are to be handled by a logic, such as paraconsistent logics which tolerates it [2–6]. Comprised with the former, the latter acknowledges and distinguishes the different epistemic statuses between “the assertion is true” and “the assertion is true with conflict”. In this paper, following [6], we study an approach to reasoning with an inconsistent  $\mathcal{ALC}$  ontology, which belongs to the second.

Considering applications of DLs, the reasoning algorithm is as important as the semantics definition. Compared to the algorithm implemented in [6] by employing a polynomial transformation algorithm which still may be time consuming for large ontologies, the process described in this paper which translate an ontology directly into a satisfiable set of rules saves the preprocessing time. Compared to the algorithm proposed in [4] which is a sequence calculus based procedure, our algorithm can benefit directly from the technical details of the KAON2 implementation. Compared to the work given in [3] where a tractable subsumption is discussed, our approach spells out for both subsumption and instant checking reasoning tasks.

There exist two fundamentally reasoning algorithms which have been implemented in DLs reasoners. The first historic approach is based on tableaux algorithms [7]. The

---

<sup>\*</sup> We acknowledge support by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project (grant 01 IMD01 B), by the EU under the IST project NeOn (IST-2006-027595, <http://www.neon-project.org/>), by the Deutsche Forschungsgemeinschaft (DFG) in the ReaSem project, and by China Scholarship Council.

second approach is based on basic superposition (ordered resolution for  $\mathcal{ALC}$ ) realized by KAON2 reasoner [8]. Our algorithm is based on an adaptation of the algorithm underlying KAON2 for dealing with  $\mathcal{ALC4}$ , a paraconsistent  $\mathcal{ALC}$ . Theoretically, we design a paraconsistent ordered-resolution for  $\mathcal{ALC4}$  which is different from other algorithms in [3, 4] and which provides a way to extend KAON2 algorithm to reason with inconsistent ontologies. Due to space limitations, proofs are omitted. They can be found in a technical report under <http://www.aifb.uni-karlsruhe.de/WBS/phi/pub/parowltr.pdf>.

The paper is structured as follows. In Section 2, we review briefly the syntax and semantics of the paraconsistent description logic defined in our previous work [6], where more technical details and intuitions can be found. Section 3 gives the paraconsistent resolution decision procedure and Section 4 studies how a set of consistent disjunctive datalog rules can be obtained from a four-valued  $\mathcal{ALC}$  ontology. Finally we conclude this paper in Section 5.

## 2 The Four-valued Description Logic $\mathcal{ALC4}$

### 2.1 Syntax and Semantics

Syntactically,  $\mathcal{ALC4}$  hardly differs from  $\mathcal{ALC}$ . Complex concepts and assertions are defined in exactly the same way. However, we allow three kinds of class inclusions, corresponding to the three implication connectives in four-valued logic case. They are called material inclusion axiom, internal inclusion axiom, and strong inclusion axiom, denoted as  $C \mapsto D$ ,  $C \sqsubseteq D$ , and  $C \rightarrow D$ , respectively.

Semantically, four-valued interpretations map individuals to elements of the domain of the interpretation, as usual. For concepts, however, to allow for reasoning with inconsistencies, a four-valued interpretation over domain  $\Delta^I$  assigns to each concept  $C$  a pair  $\langle P, N \rangle$  of (not necessarily disjoint) subsets of  $\Delta^I$ . Intuitively,  $P$  is the set of elements known to belong to the extension of  $C$ , while  $N$  is the set of elements known to be not contained in the extension of  $C$ .  $P$  and  $N$  are not necessarily disjoint and mutual complementary with respect to the domain.

Formally, a four-valued interpretation is a pair  $I = (\Delta^I, \cdot^I)$  with  $\Delta^I$  as domain, where  $\cdot^I$  is a function assigning elements of  $\Delta^I$  to individuals, and subsets of  $(\Delta^I)^2$  to concepts, such that the conditions in Table 1 are satisfied, where functions  $\text{proj}^+(\cdot)$  and  $\text{proj}^-(\cdot)$  are defined by  $\text{proj}^+(\langle P, N \rangle) = P$  and  $\text{proj}^-(\langle P, N \rangle) = N$ .

For the semantics defined above, we ensure that a number of useful equivalences from classical DLs, such as double negation law and Demorgan Law, hold, see [6] for details.

The semantics of the three different types of inclusion axioms is formally defined in Table 2 (together with the semantics of concept assertions). We say that a four-valued interpretation  $I$  satisfies a four-valued ontology  $\mathcal{O}$  (i.e. is a model of it) iff it satisfies each assertion and each inclusion axiom in  $\mathcal{O}$ . An ontology  $\mathcal{O}$  is 4-valued satisfiable (unsatisfiable) iff there exists (does not exist) such a model.

**Table 1.** Semantics of  $\mathcal{ALC}4$  Concepts

Constructor Syntax	Semantics
$A$	$A^I = \langle P, N \rangle$ , where $P, N \subseteq \Delta^I$
$R$	$R^I = \langle R_P, R_N \rangle$ , where $R_P, R_N \subseteq \Delta^I \times \Delta^I$
$o$	$o^I \in \Delta^I$
$\top$	$\langle \Delta^I, \emptyset \rangle$
$\perp$	$\langle \emptyset, \Delta^I \rangle$
$C_1 \sqcap C_2$	$\langle P_1 \cap P_2, N_1 \cup N_2 \rangle$ , if $C_i^I = \langle P_i, N_i \rangle$ for $i = 1, 2$
$C_1 \sqcup C_2$	$\langle P_1 \cup P_2, N_1 \cap N_2 \rangle$ , if $C_i^I = \langle P_i, N_i \rangle$ for $i = 1, 2$
$\neg C$	$(\neg C)^I = \langle N, P \rangle$ , if $C^I = \langle P, N \rangle$
$\exists R.C$	$\{x \mid \exists y, (x, y) \in \text{proj}^+(R^I) \text{ and } y \in \text{proj}^+(C^I)\},$ $\{x \mid \forall y, (x, y) \in \text{proj}^+(R^I) \text{ implies } y \in \text{proj}^-(C^I)\}$
$\forall R.C$	$\{x \mid \forall y, (x, y) \in \text{proj}^+(R^I) \text{ implies } y \in \text{proj}^-(C^I)\},$ $\{x \mid \exists y, (x, y) \in \text{proj}^+(R^I) \text{ and } y \in \text{proj}^-(C^I)\}$

**Table 2.** Semantics of inclusion axioms in  $\mathcal{ALC}4$

Axiom Name	Syntax	Semantics
material inclusion	$C_1 \mapsto C_2$	$\Delta^I \setminus \text{proj}^-(C_1^I) \subseteq \text{proj}^+(C_2^I)$
internal inclusion	$C_1 \sqsubset C_2$	$\text{proj}^+(C_1^I) \subseteq \text{proj}^+(C_2^I)$
strong inclusion	$C_1 \rightarrow C_2$	$\text{proj}^+(C_1^I) \subseteq \text{proj}^+(C_2^I)$ and $\text{proj}^-(C_2^I) \subseteq \text{proj}^-(C_1^I)$
concept assertion	$C(a)$	$a^I \in \text{proj}^+(C^I)$
role assertion	$R(a, b)$	$(a^I, b^I) \in \text{proj}^+(R^I)$

## 2.2 The satisfiability of $\mathcal{ALC}4$ ontologies

Note that the four-valued semantics given in previous subsection does not assure that every  $\mathcal{ALC}4$  ontology has 4-valued models.<sup>1</sup> The following example illustrates this.

**Example 1** Consider  $\mathcal{T} = \{\top \sqsubset \perp\}$ . Since for any four-valued interpretation  $I$ ,  $\top^I = \langle \Delta^I, \emptyset \rangle$  and  $\perp^I = \langle \emptyset, \Delta^I \rangle$ , where  $\Delta^I \neq \emptyset$  for DL interpretations,  $\mathcal{T}$  has no four-valued model according to Table 2. The conclusion remains the same if other two kinds of inclusion are used in  $\mathcal{T}$ .

To make every ontology 4-valued satisfiable, we introduce the following substitution (Definition 1). The underlying intuition is that  $\perp \equiv A \sqcap \neg A$  and  $\top \equiv A \sqcup \neg A$  hold for any concept  $A$  under the classical semantics w.r.t. any ontology.

**Definition 1** Given an ontology  $O$ , the satisfiable form of  $O$ , denote  $SF(O)$ , is the ontology obtained by replacing each occurrence of  $\perp$  in  $(O)$  with  $NA \sqcap \neg NA$ , and replacing each occurrence of  $\top$  in  $(O)$  with  $NA \sqcup \neg NA$ , where  $NA$  is a new atomic concept.

<sup>1</sup> The problem also exists for the 4-valued DL defined in [4] if  $\top$  and  $\perp$  are used arbitrarily.

**Example 2** (*Example 1 Continued*) By Definition 1,  $SF(\mathcal{T}) = \{NA \sqcup \neg NA \sqsubseteq NA \sqcap \neg NA\}$ . Obviously,  $SF(\mathcal{T})$  has a 4-valued model  $I = \langle \Delta^I, \cdot^I \rangle$ , where  $NA^I = \langle \Delta^I, \Delta^I \rangle$ .  $I$  is also a model of  $\{NA \sqcup \neg NA \mapsto NA \sqcap \neg NA\}$  and  $\{NA \sqcup \neg NA \rightarrow NA \sqcap \neg NA\}$ . That is, a 4-valued unsatisfiable ontology becomes 4-valued satisfiable.

Following proposition says the substitution doesn't impact the classical inconsistency.

**Proposition 1** For any ontology  $O$ , the following two claims hold:

- (1)  $SF(O)$  is 2-valued consistent if and only if  $O$  is 2-valued consistent.
- (2)  $SF(O)$  always has at least one 4-valued model.

Note that claim 2 in proposition 1 doesn't hold for 2-valued semantics, so we cannot expect to make an inconsistent ontology 2-valued satisfiable in the same way. Because of this proposition, we assume that all ontologies discussed in the rest have 4-valued models.

### 3 Resolution-based Reasoning with $\mathcal{ALC4}$

In this section, we basically follow [8, Chapter 4] to study a paraconsistent resolution for  $\mathcal{ALC4}$ , and indeed we have to assume that the reader is familiar with the KAON2-approach because of space restrictions.

We first note that resolution relies heavily on the *tertium non datur*, and thus does not lend itself easily to a paraconsistent setting. In particular, resolution cannot be based on the negation present in paraconsistent logics, as in this case  $A \vee B$  and  $\neg A \vee C$  does not imply  $B \vee C$ . We thus start by introducing a second kind of negation, called *total negation*, written  $\sim$ . To avoid confusion, we will refer to  $\neg$  as *paraconsistent negation*. Notice that we do not extend our four-valued DLs to have the total negation as a concept constructor. We rather use it only to provide a resolution-based decision procedure for four-valued DLs.

**Definition 2** Total negation  $\sim$  on  $\{\langle P, N \rangle \mid P, N \subseteq \Delta\}$  is defined by  $\sim\langle P, N \rangle = \langle \Delta \setminus P, \Delta \setminus N \rangle$ .

The intuition behind total negation is to reverse both the information of being true and of being false. By Definition 2, the double negation elimination law and Demogen Laws also hold for total negation  $\sim$ .

An important reason to propose the total negation is that it provides a way to reduce 4-valued entailment relation  $\models_4$  to 4-valued satisfiability, because for any ontology  $O$  and an axiom  $\alpha$ ,  $O \models_4 \alpha$  if and only if  $O \cup \{\sim\alpha\}$  is 4-unsatisfiable. Moreover, total negation can be used to obtain a representation of internal inclusion in terms of clauses, where equisatisfiability retains, because  $(C \sqsubseteq D)^I \in \{t, \ddot{t}\}$  if and only if  $(\forall x. (\sim C \sqcup D)(x))^I \in \{t, \ddot{t}\}$ . By these two points, we have following theorem shows that 4-entailment can be converted into 4-unsatisfiability.

**Theorem 2** Let  $O$  be a four-valued  $\mathcal{ALC4}$  ontology,  $C, D$  be concepts,  $I$  be an interpretation and  $\iota$  be a new individual not occurring in  $O$ . Then the following hold.

1.  $O \models_4 C(a)$  if and only if  $O \cup \{\sim C(a)\}$  is four-valued unsatisfiable.
2.  $O \models_4 C \mapsto D$ ,  $O \models_4 C \sqsubset D$ ,  $O \models_4 C \rightarrow D$  if and only if  $O \cup \{\sim(\neg C \sqcup D)(\iota)\}$ ,  $O \cup \{(C \sqcap \sim D)(\iota)\}$ , and  $O \cup \{(C \sqcap \sim D)(\iota), (\neg D \sqcap \sim C)(\iota)\}$  are four-valued unsatisfiable, respectively.

### 3.1 Translating $\mathcal{ALC4}$ into Clauses

To introduce clausal forms for  $\mathcal{ALC4}$  expressions, we first define an extended negation normal form for  $\mathcal{ALC4}$  called quasi-NNF. We are inspired by [9].

**Definition 3** A concept  $C$  is a quasi-atom, if it is an atomic concept, or in form  $\neg A$  where  $A$  is an atomic concept.  $C$  is a quasi-literal, if it is a quasi-atomic concept, or in form  $\sim L$  where  $L$  is a quasi-atomic concept.  $C$  is in quasi-NNF, if  $\sim$  occurs only in front of quasi-literals and  $\neg$  does not occur in front of  $\sim$ .

To give an example, let  $A$ ,  $B$ , and  $C$  be atomic concepts. Then  $(A \vee \sim \neg B) \sqcup \forall R.(\sim C)$  is in quasi-NNF. Based on the properties of  $\sim$ , it is easy to check that all  $\mathcal{ALC4}$  concepts can be transformed into equivalent expressions in quasi-NNF.

We next translate the concepts into predicate logic. This is done by the standard translation as e.g. spelled out in [8] in terms of the function  $\pi_y$  – we just have to provide for the total negation. We make one exception, namely for universal restriction, where we set  $\pi_y(\forall R.C, x) = \forall y.(\sim R(x, y) \sqcup C(y))$ . The obtained predicate logic formulae (with total negation) can now be translated into clauses in the standard way, i.e. by first casting them into Skolem form [10], which are adjusted for total negation in the straightforward way. To avoid the exponential blowup and to preserve the structure of formulae, we also can apply the structural transformation [11] to  $\mathcal{ALC4}$  as used in [8]. To do this, we define the paraconsistent Definitorial Form  $PDef(\cdot)$  of  $\mathcal{ALC4}$  concepts as follows. Note that total negation should be used.

$$PDef(C) = \begin{cases} \{C\} & \text{if } C \text{ is a literal concept,} \\ \{\sim Q \sqcup C|_p\} \cup PDef(C[Q]|_p) & \text{if } p \text{ is eligible for replacement in } C. \end{cases}$$

where  $C|_p$  to be the position  $p$  in concept  $C$ , as defined in [10, 8].

**Proposition 3** For an  $\mathcal{ALC4}$  concept  $C$  in quasi-NNF,  $\{\top \sqsubset C\}$  is four-valued satisfiable iff  $\{\top \sqsubset PDef(D_i) \mid D_i \in PDef(C)\}$  is.

Following the above transformations step by step, any  $\mathcal{ALC4}$  concept can be translated into a set of first order predicate logic clauses (with total negation) in polynomial size of the original concepts. We denote by  $Cls(C)$  the set of clauses which is obtained by the just mentioned transformation of  $C$ . These clauses are predicate logic formulae (with total negation). We give an example.

**Example 3** The concept  $\neg(\sim A \sqcap \exists R.(\forall S.C))$  is translated as follows.

$$\begin{aligned} & \neg(\sim A \sqcap \exists R.(\forall S.C)) \\ \text{in quasi-NNF: } & \sim \neg A \sqcup \forall R.(\exists S. \neg C) \\ \text{in } PDef: & \{\sim \neg A \sqcup \forall R.Q, \sim Q \sqcup \exists S. \neg C\} \\ \text{in predicate logic: } & \{\sim \neg A(x) \vee \sim R(x, y) \vee Q(y), \sim Q(x) \vee S(x, f(x)), \\ & \quad \sim Q(x) \vee \neg C(f(x))\} \end{aligned}$$

Based on the transformation described above, we finally can translate an  $\mathcal{ALC4}$  ontology  $O$  into a set of predicate logic clauses (with total negation)  $\Xi(O)$  which is the smallest set satisfying the following conditions:

- For each  $ABox$  axiom  $\alpha$  in  $ABox$ ,  $\text{Cls}(\alpha) \subseteq \Xi(O)$
- For each axiom  $C \mapsto D$ , each axiom  $C \sqsubseteq D$ , and each axiom  $C \rightarrow D$  in  $TBox$ ,  $\text{Cls}(\neg C \sqcup D) \subseteq \Xi(O)$ ,  $\text{Cls}(\sim C \sqcup D) \subseteq \Xi(O)$ , and  $\text{Cls}(\sim C \sqcup D, \sim \neg D \sqcup \neg C) \subseteq \Xi(O)$ , respectively.

**Theorem 4** *Let  $O$  be an  $\mathcal{ALC4}$  ontology.  $O$  is 4-valued satisfiable iff  $\Xi(O)$  is 4-valued satisfiable.*

### 3.2 Ordered Resolution with Selection Function $O_{4DL}$ for $\mathcal{ALC4}$

Given any fixed ordering  $\succ$  on ground quasi-atoms which is total and well-founded, we can obtain an *ordering on sets of clauses* in standard way as stated in [8]. By a slight abuse of notation, we use  $\succ$  also for  $\succ_L$  and  $\succ_C$  where the meaning is clear from the context. For example, if  $\neg A \succ A \succ B \succ \neg B \succ D$ , then  $[\sim]\neg A \succ [\sim]A \succ [\sim]B \succ [\sim]\neg[\sim]B \succ [\sim]D$  and  $\sim A \vee \neg B \prec \neg A \vee B \prec \sim \neg A \vee D$ .

By a *selection function* we mean a mapping  $S$  that assigns to each clause  $C$  a (possibly empty) multiset  $S(C)$  of literals with the prefix  $\sim$  in  $C$ . For example, both  $\{\sim \neg A\}$  and  $\{\sim \neg A, \sim D\}$  can be selected in clause  $\sim \neg A \vee \sim D \vee B \vee \neg C$ .

An ordered resolution step with selection function can now be described by the Inference Rule and Factorization Rule as follows, respectively:

$$\frac{C \vee A \quad D \vee \sim B}{C\sigma \vee D\sigma} \quad \text{and} \quad \frac{C \vee A \vee B}{(C \vee A)\sigma},$$

where

- $\sigma = MGU(A, B)$  is the most general unifier of the quasi-atoms  $A, B$ , and  $C, D$  are quasi-clauses.
- $A\sigma$  is strictly maximal in  $C\sigma \vee A\sigma$ , and no literal is selected in  $C\sigma \vee A\sigma$ ;
- $\sim B\sigma$  is either selected in  $D\sigma \vee \sim B\sigma$ , or it is maximal in  $D\sigma \vee \sim B\sigma$  and no literal is selected in  $D\sigma \vee \sim B\sigma$ .

**Theorem 5** (*Soundness and Completeness of  $O_{4DL}$* ) *Let  $N$  be an  $\mathcal{ALC4}$  knowledge base. Then  $\Xi(N) \vdash_{O_{4DL}} \square$  iff  $N$  is four-valued unsatisfiable.*

Although the inference rules are different from those of  $\mathcal{ALC}$ , we find that similar way of the selection of the literal ordering and selection function still provide us a decision procedure for  $\mathcal{ALC4}$ .

- The literal ordering  $\succ$  is defined such that  $R(x, f(x)) \succ \sim C(x)$  and  $D(f(x)) \succ \sim C(x)$ , for all function symbols  $f$ , and predicates  $R, C$ , and  $D$ .
- The selection function selects every binary literal which is preceded by  $\sim$ .

**Theorem 6** (*Decidability*) *For an  $\mathcal{ALC4}$  knowledge base  $KB$ , saturating  $\Xi(KB)$  by  $O_{4DL}$  decides satisfiability of  $KB$  and runs in time exponential in  $|KB|$ .*

#### 4 Translating $\mathcal{ALC4}$ to satisfiable Disjunctive Datalog

We should distinguish the total negation and paraconsistent negation during the translation, while there is only one kind of negation in [8]. As the translation from  $\mathcal{ALC}$  to disjunctive datalog, we can perform all inferences among nonground clauses first, after which we can simply delete all nonground clauses containing function symbols. The remaining clause set consists of clauses without function symbols [8].

**Definition 4** For an extensionally reduced  $\mathcal{ALC}$  ontology  $O$ , the function-free version of ontology  $O$  is defined as follows:

$$FF(O) = \lambda(\Gamma(O_T)) \cup \Gamma(O_A) \cup \{HU(a) \mid \text{for each individual } a \text{ occurring in } O\}$$

where  $\Gamma(O_T)$  is the set of clauses obtained by saturating  $\Xi(O_T)$  by  $O_{4DL}$  and then deleting all the clauses containing functions, and  $\Gamma(O_A)$  is the set of clauses obtained by saturating  $\Xi(O_A)$  by  $O_{4DL}$ . And for a clause  $C$ ,  $\lambda(C) = C \cup \{\sim HU(x) \mid \text{for each unsafe variable } x \text{ in } C\}$  and  $\lambda(\Sigma) = \{\lambda(C) \mid \text{for each clause } C \text{ in } \Sigma\}$ .

Note that the total negation is used in the  $\lambda$  operator. Next, we introduce following definition to simplify the statement of the translation process.

**Definition 5** All the literals in  $FF(O)$  are in one of the following cases:

- Pure positive literal which is just an atom (i.e., without neither paraconsistent negation nor total negation in front);
- Paraconsistent negative literal which is constructed by a pure positive literal with the paraconsistent negation occurring in its front.
- Total negative literal which is constructed by a pure positive literal or a paraconsistent negative literal with the total negation occurring in its front.

By this definition and Definition 3, positive quasi-literals include both pure positive literals and paraconsistent negative literals, while total negative literals are very negative quasi-literals. To give an example, let  $A(x)$  be an atom, then it is a pure positive literal,  $\neg A(x)$  is a paraconsistent negative literal, and  $\sim \neg A(x)$ ,  $\sim A(x)$  are total negative literals.

**Definition 6** The disjunctive datalog  $DD_4(O)$  corresponding to an ontology  $O$  is defined by moving in each clause from  $FF(O)$

- each pure positive liter into the rule head;
- each paraconsistent negative literal into the rule head as well and replacing it with a fresh atom simultaneously;
- each total negative literal into the rule body and replacing its quasi-atom part with a fresh atom simultaneously.

If  $KB$  is not extensionally reduced, then  $DD(KB) = DD(KB')$ , where  $KB'$  is an extensionally reduced knowledge base obtained from  $KB$  in the standard way [8].

From section 2.2, we can see that every ontology  $O$  can become 4-valued satisfiable after appropriate rewriting. That is, by 4-valued semantics, meaningful conclusions always can be derived from even a classically inconsistent ontology. The first claim of the following theorem shows that  $DD_4(O)$  is always consistent as well such that untrivial answers can be returned from it.

**Theorem 7** *Let  $KB$  be an  $\mathcal{ALC}4$  ontology. Then, the following claims hold:*

1.  $DD_4(\tilde{O})$  is always satisfiable for any ontology  $O$ , where  $\tilde{O}$  is the ontology defined in Definition 1.
2.  $O \models_4 \alpha$  if and only if  $DD_4(O) \models_c \alpha$ , where  $\alpha$  is of the form  $A(a)$  or  $R(a, b)$ , and  $A$  is an atomic concept.

## 5 Conclusions

In this paper, we work out for how to translate an  $\mathcal{ALC}4$  ontology to a consistent set of disjunctive datalog rules, such that meaningful consequences can be deduced from a possible inconsistent ontology.

Considering the algorithm described in this paper, it is rather apparent that all the benefits from the KAON2 system – like the ability to handle large ABoxes – can also be achieved by the paraconsistent version of KAON2. We will further study the similar paraconsistent approach for more complex DLs in the future work.

## References

1. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI. (2003) 355–362
2. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: IJCAI. (2005) 454–459
3. Patel-Schneider, P.F.: A four-valued semantics for terminological logics. Artificial Intelligence **38** (1989) 319–351
4. Straccia, U.: A sequent calculus for reasoning in four-valued description logics. In: TABLEAUX. (1997) 343–357
5. Straccia, U.: Foundations of a logic based approach to multimedia document retrieval. Department of Computer Science, University of Dortmund, Dortmund, Germany (June, 1999)
6. Ma, Y., Hitzler, P., Lin, Z.: Algorithms for paraconsistent reasoning with OWL. In: Proc. of ESWC. (2007) To appear.
7. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
8. Motik, B.: Reasoning in description logics using resolution and deductive databases. PhD thesis, University Karlsruhe, Germany (2006)
9. Kamide, N.: Foundations of paraconsistent resolution. Fundamenta Informaticae **71** (2006) 419–441
10. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd Edition. Texts in Computer Science. Springer (1996)
11. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comput. **2** (1986) 293–304

## Measuring Inconsistency for Description Logics Based on Paraconsistent Semantics <sup>\*</sup>

Yue Ma<sup>1,2</sup>, Guilin Qi<sup>2</sup>, Pascal Hitzler<sup>2</sup>, and Zuoquan Lin<sup>1</sup>

<sup>1</sup>Department of Information Science, Peking University, China

<sup>2</sup>AIFB, Universität Karlsruhe, Germany

{mayue,lz}@is.pku.edu.cn, {yum,gqi,hitzler}@aifb.uni-karlsruhe.de

**Abstract.** In this paper, we propose an approach for measuring inconsistency in inconsistent ontologies. We first define the degree of inconsistency of an inconsistent ontology using a four-valued semantics for the description logic  $\mathcal{ALC}$ . Then an ordering over inconsistent ontologies is given by considering their inconsistency degrees. Our measure of inconsistency can provide important information for inconsistency handling.

### 1 Introduction

Real knowledge bases and data for Semantic Web applications will rarely be perfect. They will be distributed and multi-authored. They will be engineered by more or less knowledgeable people and often be created automatically from raw data. They will be assembled from different sources and reused. Consequently, it is unreasonable to expect such realistic knowledge bases to be always logically consistent, and methods for the meaningful handling of such knowledge bases are being sought for.

Inconsistency has often been viewed as erroneous information in an ontology, but this is not necessarily the best perspective on the problem. The study of inconsistency handling in Artificial Intelligence indeed has a long tradition, and corresponding results are recently being transferred to description logics, which underly OWL.

There are mainly two classes of approaches to dealing with inconsistent ontologies. The first class of approaches is to circumvent the inconsistency problem by applying a non-standard reasoning method to obtain meaningful answers [1, 2] – i.e. to ignore the inconsistency in this manner. The second class of approaches to deal with logical contradictions is to resolve logical modeling errors whenever a logical problem is encountered [3, 4].

However, given an inconsistent ontology, it is not always clear which approach should be taken to deal with the inconsistency. Another problem is that when resolving inconsistency, there are often several alternative solutions and it would be helpful to have some extra information (such as an ordering on elements of the ontology) to decide which solution is the best one. It has been shown that analyzing inconsistency is

---

<sup>\*</sup> We acknowledge support by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project (grant 01 IMD01 B), by the EU under the IST project NeOn (IST-2006-027595, <http://www.neon-project.org/>), by the Deutsche Forschungsgemeinschaft (DFG) in the ReaSem project, and by China Scholarship Council.

helpful to decide how to act on inconsistency [5], i.e. whether to ignore it or to resolve it. Furthermore, measuring inconsistency in a *knowledge base* in classical logic can provide some context information which can be used to resolve inconsistency [6–8].

There are mainly two classes of inconsistency measures in classical logic. The first class of measures is defined by the number of formulas which are responsible for an inconsistency, i.e. a knowledge base in propositional logic is more inconsistent if more logical formulas are required to produce the inconsistency [9]. The second class considers the propositions in the language which are affected by the inconsistency. In this case, a knowledge base in propositional logic is more inconsistent if more propositional variables are affected by the inconsistency [6, 10]. The approaches belonging to the second class are often based on some paraconsistent semantics because we can still find models for inconsistent knowledge bases in paraconsistent logics.

Most of the work on measuring inconsistency is concerned with knowledge bases in propositional logic. In [13], the authors generalized the work on measuring inconsistency in quasi-classical logic to the first-order case. However, it is not clear how their approach can be implemented because there is no existing work on implementing first-order quasi-classical logic.

At the same time, there are potential applications for inconsistency measures for ontologies, as they provide evidence for reliability of ontologies when an inconsistency occurs. In a scenario, where ontologies are merged and used together, such evidence can be utilised to guide systems in order to arrive at meaningful system responses.

In this paper, we propose an approach for measuring inconsistency in inconsistent ontologies. We first define the degree of inconsistency of an inconsistent ontology using a four-valued semantics for description logic  $\mathcal{ALC}$ . By analyzing the degree of inconsistency of an ontology, we can either resolve inconsistency if the degree is high (e.g. greater than 0.7) or ignore it otherwise. After that, an ordering over inconsistent ontologies is given by considering their inconsistency degrees. We then can consider those ontology which are less inconsistent and more reliable.

This paper is organized as follows. We first provide some basic notions in four-valued description logic  $\mathcal{ALC}$  in Section 2. Our measure of inconsistency is then given in Section 3. Finally, we discuss related work and conclude the paper in Section 4.

## 2 Four-valued Models for $\mathcal{ALC}$

We assume that readers are familiar with the description logic  $\mathcal{ALC}$  [11]. An  $\mathcal{ALC}$  ontology is a pair  $(\mathcal{T}, \mathcal{A})$ , where  $\mathcal{T}$  is the set of class inclusions of the form  $C \sqsubseteq D$  and  $\mathcal{A}$  is the set of individual assertions in the forms  $C(a)$  and  $R(a, b)$ . We use  $\mathcal{L}_O$  to denote the set of all concepts, roles and individuals used in  $O$ , and assume the cardinality of  $\mathcal{L}_O$  is finite, which is acceptable for applications. We review the four-valued semantics for  $\mathcal{ALC}$  here, and see to [2] for details.

A four-valued interpretation (4-interpretation for short) is a pair  $I = (\Delta^I, \cdot^I)$  with  $\Delta^I$  as the domain, but  $\cdot^I$  is a mapping which assigns to each concept  $C$  a pair  $\langle P, N \rangle$  of subsets of  $\Delta^I$ , and each role  $R$  a pair  $\langle R_P, R_N \rangle$  of subsets of  $(\Delta^I)^2$ , such that the conditions in Table 1 are satisfied, where  $\text{proj}^+$  and  $\text{proj}^-$  are functions defined

**Table 1.** Semantics of  $\mathcal{ALC}4$  Concepts

Constructor Syntax	Semantics
$A$	$A^I = \langle P, N \rangle$ , where $P, N \subseteq \Delta^I$
$R$	$R^I = \langle R_P, R_N \rangle$ , where $R_P, R_N \subseteq \Delta^I \times \Delta^I$
$o$	$o^I \in \Delta^I$
$\top$	$\langle \Delta^I, \emptyset \rangle$
$\perp$	$\langle \emptyset, \Delta^I \rangle$
$C_1 \sqcap C_2$	$\langle \text{proj}^+(C_1^I) \cap \text{proj}^+(C_2^I), \text{proj}^-(C_1^I) \cup \text{proj}^-(C_2^I) \rangle$
$C_1 \sqcup C_2$	$\langle \text{proj}^+(C_1^I) \cup \text{proj}^+(C_2^I), \text{proj}^-(C_1^I) \cap \text{proj}^-(C_2^I) \rangle$
$\neg C$	$(\neg C)^I = \langle N, P \rangle$ , if $C^I = \langle P, N \rangle$
$\exists R.C$	$\langle \{x \mid \exists y, (x, y) \in \text{proj}^+(R^I) \text{ and } y \in \text{proj}^+(C^I)\}, \{x \mid \forall y, (x, y) \in \text{proj}^+(R^I) \text{ implies } y \in \text{proj}^-(C^I)\} \rangle$
$\forall R.C$	$\langle \{x \mid \forall y, (x, y) \in \text{proj}^+(R^I) \text{ implies } y \in \text{proj}^+(C^I)\}, \{x \mid \exists y, (x, y) \in \text{proj}^+(R^I) \text{ and } y \in \text{proj}^-(C^I)\} \rangle$

as follows:  $\text{proj}^+\langle P, N \rangle = P$ , and  $\text{proj}^+\langle R_P, R_N \rangle = R_P$ ;  $\text{proj}^-\langle P, N \rangle = N$ , and  $\text{proj}^-\langle R_P, R_N \rangle = R_N$ .

Intuitively, the first element  $P$  (e.g.  $\text{proj}^+(C^I)$ ) of the four-valued extension of a concept  $C$  is the set of elements known to belong to the extension of  $C$ , while the second element  $N$  (e.g.  $\text{proj}^-(C^I)$ ) is the set of elements known to be not contained in the extension of  $C$ . The intuition is similar for the semantics of roles. In order to reason with inconsistency, we are free of these constrains, thus forming four epistemic states of the individual assertions under an interpretation: (1) we know the individual is contained, (2) we know the individual is not contained, (3) we have contradictory information, namely that the individual is both contained in the concept and not contained in the concept, (4) we have no knowledge whether or not the individual is contained. Next, we use the four truth values  $\{t, f, \ddot{\top}, \ddot{\perp}\}$  from Belnap’s four-valued logic [12], which denote truth, falsity, contradiction, and incompleteness, respectively, to distinguish them.

**Definition 1** For instances  $a, b \in \Delta^I$ , concept name  $C$ , and role name  $R$ :

$$\begin{aligned}
 C^I(a) = t, & \text{ iff } a^I \in \text{proj}^+(C^I) \text{ and } a^I \notin \text{proj}^-(C^I), \\
 C^I(a) = f, & \text{ iff } a^I \notin \text{proj}^+(C^I) \text{ and } a^I \in \text{proj}^-(C^I), \\
 C^I(a) = \ddot{\top}, & \text{ iff } a^I \in \text{proj}^+(C^I) \text{ and } a^I \in \text{proj}^-(C^I), \\
 C^I(a) = \ddot{\perp}, & \text{ iff } a^I \notin \text{proj}^+(C^I) \text{ and } a^I \notin \text{proj}^-(C^I), \\
 R^I(a, b) = t, & \text{ iff } (a^I, b^I) \in \text{proj}^+(R^I) \text{ and } (a^I, b^I) \notin \text{proj}^-(R^I), \\
 R^I(a, b) = f, & \text{ iff } (a^I, b^I) \notin \text{proj}^+(R^I) \text{ and } (a^I, b^I) \in \text{proj}^-(R^I), \\
 R^I(a, b) = \ddot{\top}, & \text{ iff } (a^I, b^I) \in \text{proj}^+(R^I) \text{ and } (a^I, b^I) \in \text{proj}^-(R^I), \\
 R^I(a, b) = \ddot{\perp}, & \text{ iff } (a^I, b^I) \notin \text{proj}^+(R^I) \text{ and } (a^I, b^I) \notin \text{proj}^-(R^I).
 \end{aligned}$$

By the correspondence defined above, we can define a four-valued interpretation in terms of Table 1 or Definition 1. That is, the four-valued extension of a concept  $C$  can be defined either as a pair of subsets of a domain or by claiming the truth values for each  $C^I(a)$ , where  $a \in \Delta^I$ .

**Table 2.** Semantics of inclusion axioms in  $\mathcal{ALC}_4$

Axiom Name	Syntax	Semantics
inclusion	$C_1 \mapsto C_2$	$\Delta^I \setminus \text{proj}^-(C_1^I) \subseteq \text{proj}^+(C_2^I)$
concept assertion	$C(a)$	$a^I \in \text{proj}^+(C^I)$
role assertion	$R(a, b)$	$(a^I, b^I) \in \text{proj}^+(R^I)$

As to the semantics of inclusion axioms, it is formally defined in Table 2 (together with the semantics of concept assertions), which means that  $C \sqsubseteq D$  is true under an interpretation  $I$  if and only if for each individual which is not known to be not contained in the extension of  $C$ , it must be known to belong to the extension of  $D$ .

We say that a four-valued interpretation  $I$  satisfies (a model of) an ontology  $O$  iff  $I$  satisfies each assertion and each inclusion axiom in  $O$ . An ontology  $O$  is four-valued satisfiable (unsatisfiable) iff there exists (does not exist) a model for  $O$ . In this paper, we denote  $\mathcal{M}_4(O)$  as the set of four-valued models of an ontology  $O$ .

For an inconsistent ontology, it doesn't have classical two-valued models, but it may have four-valued models. For example,  $I = \langle \{a\}, \cdot^I \rangle$ , under which  $A^I = \langle \{a\}, \{a\} \rangle$  is a model of the ontology whose  $ABox = \{A(a), \neg A(a)\}$  and whose  $TBox$  is empty. However, an ontology does not always have four-valued model if top and bottom concepts are both allowed as concept constructors. Take  $\mathcal{T} = \{\top \sqsubseteq \perp\}$  for example. For any four-valued interpretation  $I$ ,  $\top^I = \langle \Delta^I, \emptyset \rangle$  and  $\perp^I = \langle \emptyset, \Delta^I \rangle$ .  $\mathcal{T}$  has no four-valued model since  $(\Delta^I \setminus \text{proj}^-(\top^I)) = \Delta^I \not\subseteq \text{proj}^+(\perp^I) = \emptyset$ , where  $\Delta^I \neq \emptyset$  for DL interpretations. So in this paper we only consider the inconsistency measure of an inconsistent ontology which has no  $\top$  and  $\perp$  as concept constructors. Note that this assumption is relatively mild, as  $\top$  can always be replaced by  $A \sqcup \neg A$ .

### 3 Inconsistency Measure

In this section, we use four-valued models of inconsistent  $\mathcal{ALC}$  ontologies to measure their inconsistency degrees.

**Definition 2** Let  $I$  be a four-valued model of an ontology  $O$  with domain  $\Delta^I$ , the inconsistency set of  $I$  for  $O$ , written  $\text{ConflictOnto}(I, O)$ , is defined as follows:

$$\text{ConflictOnto}(I, O) = \text{ConflictConcepts}(I, O) \cup \text{ConflictRoles}(I, O),$$

where  $\text{ConflictConcepts}(I, O) = \{A(a) \mid A^I(a) = \ddot{\top}, A \in \mathcal{L}_O, a \in \Delta^I\}$ , and  $\text{ConflictRoles}(I, O) = \{R(a_1, a_2) \mid R^I(a_1, a_2) = \ddot{\top}, R \in \mathcal{L}_O, a_1, a_2 \in \Delta^I\}$

Intuitively,  $\text{ConflictOnto}(I, O)$  stands for the set of conflicting atomic individual assertions. To define the degree of inconsistency, we still need following concepts.

**Definition 3** For the ontology  $O$  and a 4-valued interpretation  $I$ ,

$$\text{GroundOnto}(I, O) = \text{GroundConcepts}(I, O) \cup \text{GroundRoles}(I, O),$$

where  $\text{GroundConcepts}(I, O) = \{A(a) \mid a \in \Delta^I, A \in \mathcal{L}_O\}$  and  $\text{GroundRoles}(I, O) = \{R(a_1, a_2) \mid a_1, a_2 \in \Delta^I, R \in \mathcal{L}_O\}$

Intuitively,  $\text{GroundOnto}(I, O)$  is the collection of different atomic individual assertions.

In order to define the degree of inconsistency, we use an assumption that only interpretations with finite domains are considered in this paper. This is reasonable in practical cases because only finite individuals can be represented or would be used. This is also reasonable from the theoretical aspect because  $\mathcal{ALC}$  has finite model property — that is, if an ontology is consistent and within the expressivity of  $\mathcal{ALC}$ , then it has a classical model whose domain is finite.

**Definition 4** *The inconsistency degree of an ontology w.r.t. a model  $I \in \mathcal{M}_4(O)$ , denote  $\text{Inc}_I(O)$ , is a value in  $[0, 1]$  calculated in the following way:*

$$\text{Inc}_I(O) = \frac{|\text{ConflictOnto}(I, O)|}{|\text{GroundOnto}(I, O)|}$$

That is, The inconsistency degree of  $O$  w.r.t.  $I$  is the ratio of the number of conflicting atomic individual assertions divided by the amount of all possible atomic individual assertions of  $O$  w.r.t.  $I$ . It measures to what extent a given ontology contains inconsistency w.r.t.  $I$ .

**Example 5** *Consider Ontology  $O = (\mathcal{T}, \mathcal{A})$ , where  $\mathcal{T} = \{A \sqsubseteq B \sqcap \neg B\}$ ,  $\mathcal{A} = \{A(a)\}$ . A model of  $O$  is as follows:  $I_1 = (\Delta^{I_1}, \cdot^{I_1})$ , where  $\Delta^{I_1} = \{a\}$ ,  $A^{I_1}(a) = t$ , and  $B^{I_1}(a) = \perp$ . For this model,  $\text{GroundOnto}(I_1, O) = \{A(a), B(a)\}$ , and  $B(a)$  is the unique element in  $\text{ConflictOnto}(I_1, O)$ . Therefore,  $\text{Inc}_{I_1}(O) = \frac{1}{2}$ .*

In [13], it has been shown that for a fixed domain, not all the models need to be considered to define an inconsistency measure because some of them may overestimate the degree of inconsistency. Let us go back to Example 5.

**Example 6 (Example 5 Continued)** *Consider another model  $I_2$  of  $O$ :  $I_2 = (\Delta^{I_2}, \cdot^{I_2})$ , where  $\Delta^{I_2} = \{a\}$ ,  $A^{I_2}(a) = \perp$ ,  $B^{I_2}(a) = \perp$ .  $I_1$  and  $I_2$  share a same domain. Since  $|\text{ConflictOnto}(I_2, O)| = |\{B(a), A(a)\}| = 2$ , we have  $I_1 \leq_{\text{Incons}} I_2$  by Definition 7. This is because  $\cdot^{I_2}$  assigns contradiction to  $A(a)$ . However,  $A(a)$  is not necessary a conflicting axiom in four-valued semantics. Therefore, we conclude that  $\text{Inc}_{I_2}(O)$  overestimates the degree of inconsistency of  $O$ .*

We next define a partial ordering on  $\mathcal{M}_4(O)$  such that the minimal elements w.r.t. it are used to define the inconsistency measure for  $O$ .

**Definition 7 (Model ordering w.r.t. inconsistency)** *Let  $I_1$  and  $I_2$  be two four-valued models of ontology  $O$  such that  $|\Delta^{I_1}| = |\Delta^{I_2}|$ , we say the inconsistency of  $I_1$  is less than or equal to  $I_2$ , written  $I_1 \leq_{\text{Incons}} I_2$ , if and only if  $\text{Inc}_{I_1}(O) \leq \text{Inc}_{I_2}(O)$ .*

The condition  $|\Delta^{I_1}| = |\Delta^{I_2}|$  in this definition just reflects the attitude that only models with the same cardinality of domain are comparative. As usual,  $I_1 <_{\text{Incons}} I_2$  denotes  $I_1 \leq_{\text{Incons}} I_2$  and  $I_2 \not\leq_{\text{Incons}} I_1$ , and  $I_1 \equiv_{\text{Incons}} I_2$  denotes  $I_1 \leq_{\text{Incons}} I_2$  and  $I_2 \leq_{\text{Incons}} I_1$ .  $I_1 \leq_{\text{Incons}} I_2$  means that  $I_1$  is more consistent than  $I_2$ .

The model ordering w.r.t. inconsistency is used to define the preferred models.

**Definition 8** Let  $O$  be a DL-based ontology and  $n(n \geq 1)$  be a given cardinality, the preferred models w.r.t  $\leq_{Incons}$  of size  $n$ , written  $PreferModel_n(O)$ , are defined as follows:

$$PreferModel_n(O) = \{I \mid |\Delta^I| = n; \forall I' \in \mathcal{M4}(O), |\Delta^{I'}| = n \text{ implies } I \leq_{Incons} I'\}$$

That is,  $PreferModel_n(O)$  is the set of models of size  $n$  which are minimal w.r.t  $\leq_{Incons}$ .

The following theorem says that the cardinality of a domain is critical for measuring the inconsistency degree of an ontology, while the element differences can be ignored.

**Theorem 9** Let  $O$  be an ontology and  $n(\geq 1)$  be any given positive integer. Suppose  $I_1$  and  $I_2$  are two four-valued models of  $O$  such that  $|\Delta^{I_1}| = |\Delta^{I_2}| = n$ ,  $\{I_1, I_2\} \subseteq PreferModel_n(O)$ . The following equation always holds:

$$Inc_{I_1}(O) = Inc_{I_2}(O).$$

For simplicity of expression, we say an interpretation is *well-sized* if and only if the cardinality of its domain is equal to or greater than the number of individuals in  $O$ . Because of the unique name assumption of DL  $\mathcal{ALC}$ , an interpretation can be a model only if it is well-sized. Moreover, the following theorem asserts the existence of preferred models among the well-sized interpretations.

**Theorem 10** For any given  $\mathcal{ALC}$  ontology  $O$  without concepts  $\top$  and  $\perp$  in the language, the preferred models among well-sized interpretations always exist.

Above we consider the inconsistency degrees of an ontology w.r.t. its four-valued models, especially the preferred models. Now we define an integrated inconsistency degree of an ontology allowing for different domains.

**Definition 11** Given an ontology  $O$  and an arbitrary cardinality  $n(n \geq 1)$ , let  $I_n$  be an arbitrary model in  $PreferModels_n(O)$ . The inconsistency degree sequence of  $O$ , say  $OntoInc(O)$ , is defined as  $\langle r_1, r_2, \dots, r_n, \dots \rangle$ , where  $r_n = ModelInc(I_n, O)$  if  $I_n$  is well-sized. Otherwise, let  $r_n = *$ . We use  $*$  as a kind of null value.

From theorem 9 and 10, the following property holds obviously.

**Proposition 12** Assume  $O$  is an inconsistent ontology and  $OntoInc(O) = \langle r_1, r_2, \dots \rangle$ , and  $N$  is the number of individuals of  $O$ , then

$$r_i = \begin{cases} * & \text{if } 0 < i < N, \\ r_i \neq * \text{ and } r_i > 0 & \text{if } i \geq N. \end{cases}$$

This proposition shows that for an ontology, its inconsistency measure cannot be a meaningless sequence — that is, each element is the null value  $*$ . Moreover, the non-zero values in the sequence starts just from the position which equals to the number of individuals in the ontology, and remains greater than zero in the latter positions of the sequence. As for an example, we first measure an extreme inconsistent ontology.

**Example 13**  $O = \{C \sqcup \neg C \sqsubseteq C \sqcap \neg C\}$ .  $O$  is inconsistent. Obviously, for any four-valued model  $I = \langle \Delta^I, \cdot^I \rangle$  of  $O$ ,  $C$  is assigned to  $\langle \Delta^I, \Delta^I \rangle$ , so  $OntoInc(O) = \{1, 1, \dots\}$ .

After the inconsistency degree is defined for each ontology, we can use it to compare two ontologies one of which is less inconsistent. The ordering over inconsistent ontologies is defined as follows:

**Definition 14** *Given two ontologies  $O_1$  and  $O_2$ . Suppose  $\text{OntoInc}(O_1) = \langle r_1, r_2, \dots \rangle$  and  $\text{OntoInc}(O_2) = \langle r'_1, r'_2, \dots \rangle$ . We say  $O_1$  is less inconsistent than  $O_2$ , written  $O_1 \preceq_{\text{Incons}} O_2$ , iff*

$$r_n \leq r'_n, \text{ for all } n \geq \max\{k, k'\},$$

where  $k = \min\{i : r_i \neq *\}$ ,  $k' = \min\{j : r'_j \neq *\}$ .

According to proposition 12,  $\preceq_{\text{Incons}}$  is well-defined. In Definition 14, we compare the values from the position at which both sequences have non-null values because the null value  $*$  cannot reflect useful information about the inconsistency of the ontology.

To compare two ontologies with respect to the ordering  $\preceq_{\text{Incons}}$ , by Definition 14, we have to compare two infinite sequences, which is practically very hard. When designing an algorithm to compare two ontologies, we can set a termination condition in order to guarantee that an answer will be obtained. Suppose time (resource) is used up and  $\langle r_1, \dots, r_n \rangle$  and  $\langle r'_1, \dots, r'_m \rangle$  are the already obtained partial sequences of  $\text{OntoInc}(O)$  and  $\text{OntoInc}(O')$ , respectively. Then we can say that  $O$  is approximatively less inconsistent than  $O'$ , denoted by  $O \lesssim_{\text{Incons}} O'$ , if and only if  $r_i \leq r'_i$  for all  $1 \leq i \leq \min\{m, n\}$ .

**Example 15** (Example 5 continued) *Each preferred model  $I$  of  $O$  must satisfy that (1) it assigns one and only one individual assertion in  $\{B(a), A(a)\}$  to contradictory truth value  $\ddot{\top}$  — that is,  $B^I(a) = \ddot{\top}$  and  $A(a) = t$ , or  $B^I(a) = t$  and  $A(a) = \ddot{\top}$ ; (2) it assigns other grounded assertions to truth values among the set  $\{t, f, \perp, \ddot{\perp}\}$ . So  $|\text{ConflictOnto}(I, O)| = 1$ . Consequently,  $\text{OntoInc}(O) = \{\frac{1}{2}, \frac{1}{4}, \dots, \frac{1}{2n}, \dots\}$ .*

*Suppose  $O_1 = \{A \sqsubseteq B \sqcap \neg B, A \sqsubseteq C, A(a)\}$ . In its preferred models, the individual assertions related to  $C$  are not involved with contradictory truth value, so  $\text{OntoInc}(O_1) = \{\frac{1}{3}, \frac{1}{6}, \dots, \frac{1}{3n}, \dots\}$ . By definition 14,  $O_1 \prec_{\text{Incons}} O$ , which means that  $O_1$  has less inconsistency percent than  $O$  does.*

## 4 Related Work and Conclusion

To the best of our knowledge, this paper is the first work which can distinguish description logic based ontologies in many levels considering their different inconsistency degrees instead of only 0/1. In this paper, we mainly spells out for  $\mathcal{ALC}$ . The extension to more expressive languages is direct by extending four-valued semantics for them.

Our work is closely related to the work of inconsistency measuring given in [13], where Quasi-Classical models (QC logic [14]) are used as the underlying semantics. In this paper, we use four-valued models for description logics as the underlying semantics. This is because QC logic needs to translate each formula in the theory into prenex conjunctive normal form (PCNF). We claim that this is not practical, especially for a large ontology, because it may be quite time consuming and users probably do not like their ontologies to be modified syntactically. In this paper, we can see that four-valued models also provide us with a novel way to measure inconsistent degrees of ontologies.

It is also apparent that the inconsistency measure defined by our approach can be used to compute each axiom’s contribution to inconsistency of a whole ontology by adapting the method proposed in [8], thereby providing important information for resolving inconsistency in an ontology.

In [13], every set of formulae definitely has at least one QC model because neither the constant predicate  $t$  (tautology) nor the constant predicate  $f$  (false) is contained in the language. However, corresponding to  $t$  and  $f$ , the top concept  $\top$  and bottom concept  $\perp$  are two basic concept constructors for  $\mathcal{ALC}$ . This requirement leads to the possible nonexistence of four-valued models of an  $\mathcal{ALC}$  ontology. Due to the space limitation, we presume that the ontologies do not use  $\top$  and  $\perp$  as concept constructors. The discussion for an arbitrary inconsistent ontology will be left as future work.

For practical implementations, we suggest that a termination condition be used. In the further work, we will work on some guidance on selecting such a termination condition and on how to measure the trade-off between early termination and accuracy.

For the implementation of our approach, we are currently working on the algorithm, which will be presented in a future paper.

## References

1. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: Proc. of 19th International Joint Conference on Artificial Intelligence(IJCAI’05). Morgan Kaufmann (2005) 254–259
2. Ma, Y., Hitzler, P., Lin, Z.: Algorithms for paraconsistent reasoning with OWL. In: Proceedings of the 4th International Semantic Web Conference, ESWC07, Innsbruck, Austria, June 2007. (2007) To appear.
3. Schlobach, S.: Diagnosing terminologies. In: Proc. of AAAI’05. (2005) 670–675
4. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Proc. of WWW’05. (2005) 633–640
5. Hunter, A.: How to act on inconsistent news: Ignore, resolve, or reject. *Data Knowl. Eng.* **57** (2006) 221–239
6. Hunter, A.: Measuring inconsistency in knowledge via quasi-classical models. In: AAAI/IAAI. (2002) 68–73
7. Hunter, A., Konieczny, S.: Approaches to measuring inconsistent information. In Bertossi, L.E., Hunter, A., Schaub, T., eds.: *Inconsistency Tolerance*. Volume 3300 of *Lecture Notes in Computer Science.*, Springer (2005) 191–236
8. Hunter, A., Konieczny, S.: Shapley inconsistency values. In Doherty, P., Mylopoulos, J., Welty, C.A., eds.: *KR, AAAI Press* (2006) 249–259
9. Knight, K.M.: Measuring inconsistency. *Journal of Philosophical Logic* **31** (2001) 77–98
10. Oller, C.A.: Measuring coherence using LP-models. *J. Applied Logic* **2** (2004) 451–455
11. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press (2003)
12. Belnap, N.D.: A useful four-valued logic. *Modern uses of multiple-valued logics* (1977) 7–73
13. Grant, J., Hunter, A.: Measuring inconsistency in knowledgebases. *J. Intell. Inf. Syst.* **27** (2006) 159–184
14. Hunter, A.: A semantic tableau version of first-order quasi-classical logic. In Benferhat, S., Besnard, P., eds.: *ECSQARU*. Volume 2143 of *Lecture Notes in Computer Science.*, Springer (2001) 544–555

## Action Based ABox Update: an Example from the Chemical Compound Formulation

Alessandro Mosca, Matteo Palmonari

Department of Computer Science, Systems and Communication (DISCo)

University of Milano-Bicocca

via Bicocca degli Arcimboldi, 8

20126 - Milan (Italy)

{alessandro.mosca, matteo.palmonari}@disco.unimib.it

**Abstract.** The Chemical Formulation Problem (Compounding Problem) consists in modifying the chemical formulation of a compound in order to obtain a new compound showing a set of desired performances. This paper presents a computational model for the Compounding Problem based on the interpretation of transformation actions on the compound structures as ABox updates. Possible transformations of the compounds are defined as transitions of a Labelled Transition System (LTS), and the Compounding Problem has been defined as a Heuristic Search Problem. In order to reduce the complexity of the search space the paper presents an approach based on: (i) the representation of the ontological constraints on the compounds' structure in a TBox, and the representation of compound formulations as ABoxes. (ii) The representation of the transitions of a LTS as a set of actions producing updates on ABoxes; (iii) the definition of an algorithm to check ABox consistency with respect to a specific axiom schema in order to prune inconsistent branches of the search space.

### 1 Introduction

The Chemical Compound Formulation Problem (Compounding Problem) consists in modifying the chemical formulation of a *chemical compound* in order to obtain a new compound showing a set of desired performances. Reasoning on the structural and behavioral change dynamics of chemical compounds is a very hard combinatorial problem, even when a small number of chemical elements are taken into account. In fact, although there are a number of mathematical methods for computational chemistry based on quantitative and micro-level physical representations exist, their computational cost grows largely with the complexity of the investigated substances [1]. For this reasons it is challenging to overcome the computational intractability of the quantitative, mathematical, compound representations, exploring a higher-level qualitative approach to the Compounding Problem. This approach is based on the elicitation of the knowledge involved in the practice of experts working in the field. According to a mereological point of view, a compound is a blend of atomic chemical ingredients in various proportions (its “formulation”), where each ingredient is chosen according to its different chemical and physical properties. This perspective on chemical compounds has been supported by repeated observations of expert practices, and can be naturally supported by an ontological representation based on DL; DL formulas provide a tool to specify both the

ontological boundaries of the domain (TBox) and the representations of specific compounds (ABox). Within this perspective, the structural representation of compounds and their components need to be connected with their representation in terms of performances and behavior, on the basis of which a compound is chosen.

The computational model is defined as a product of  $n$  possible labelled transition systems (LTSs), each representing the domain entities at a given level of representation, and a set of morphisms between the LTSs. The *states* of each LTS are ABoxes representing compound formulations, while *transitions* are compound transformation actions. *Morphisms* establish relations between the different levels of representation in order to associate, e.g. a set of performance parameters to a structural representation of the chemical compound. Three representational dimensions are particularly relevant for a number of different industrial compounds (e.g. drugs, rubber compounds, and so on): (i) the *structural dimension* - a list of ingredients together with their quantity; (ii) the *behavioral dimension* - a description of the outcomes of artificial lab tests, and (iii) the *teleological dimension* - a description of the outcomes of tests conducted in the final application environment.

An instance of a Compounding Problem consists in acting on a specific LTS in order to obtain a compound fulfilling the requirements specified at other levels of the representation. Given the morphisms between the three levels introduced above, our goal is to produce changes on the structural level in order to meet the requirements specified at the behavioral level and at the teleological level. This problem can be easily interpreted as a Planning Problem within the Heuristic Search paradigm [2]: the structural representation of the given compound is the initial state, transitions represent possible actions and the goal test is defined on the behavioral and teleological representation associated to compounds; however, the great number of actions that is possible to take on every compound drastically boosts the branching factor compromising the feasibility of the algorithmic approach.

The computational effort that is needed to explore this structure can be significantly reduced by pruning the branches producing non admissible states, according to ontological criteria. A compound formulation can be considered *ontologically consistent* if it does not violate domain specific constraints imposed on its mereological composition. Our approach consists therefore in: (i) representing the ontological constraints on the compounds' structure in a TBox and the compounds structure in a ABox; (ii) representing the structural-level LTS as a set of actions producing updates on ABoxes; (iii) formally define the notion of admissible action according to notion of ABox satisfiability; (v) define an algorithm to check ABox satisfiability with respect to a specific axiom schema in order to prune inconsistent branches of the search space.

The present computational model, developed in the context of a project made in collaboration with the Business Unit Truck of Pirelli Tires [3], led to the design of a system for the automatically resolution of chemical formulation problems in the industrial domain of rubber compounds for tires [4]. In this paper we focus on the DL-related aspects of our approach, and in particular on the interpretation of compound transformations as ABox updates, and on the algorithm for checking consistency of the updates; while for further details about the implementation of the search algorithm we refer to [4]. The next section presents the axioms defining the ontological integrity conditions

for a chemical rubber compound formulation at the structural level. Section 3 further describes the heuristic search space based on DL descriptions. Section 2 describes the interpretation of the LTS transitions as ABox updates, and introduces the algorithm for the ABox updates consistency checking. Concluding remarks end the paper.

## 2 TBox and ABox in the compounding knowledge scenario

For a comprehensive description of the TBox we refer to [5, 6]. In the following, we present some core aspects of our ontological representation in the domain of tread rubber compounding. The following formulas are expressed in the *SIQ* DL under Unique Name Assumption. Let  $\prec$  be a primitive role standing for “is a functional part of”;  $\prec$  is a “composed” part-of relation in the sense of [7] (i.e. a part-of relation that is both *integral* and *functional*). In particular,  $\prec$  is *not reflexive*, *not symmetric*, and *not transitive*. It is useful to introduce the inverse role “has functional part” ( $\succ$ ) as  $\succ \doteq \prec^{-}$ . The axioms in the structural-level TBox guarantee that if a model exists, then the model describes the structure of a compound devoted to the production of tire in the industrial field of interest.

$$\begin{aligned} \text{TreadCompound} \equiv & \text{Compound} \sqcap (= 1 \succ .\text{PolymericMatrix}) \\ & \sqcap (= 1 \succ .\text{Vulcanization}) \\ & \sqcap (= 1 \succ .\text{ProcessAid}) \sqcap (= 1 \succ .\text{Antidegradant}) \\ & \sqcap (= 1 \succ .\text{ReinforcingFiller}) \\ & \sqcap ((\geq 0 \succ .\text{Softener}) \sqcap (\leq 1 \succ .\text{Softener})) \end{aligned}$$

It is standard that a rubber compound devoted to tread tire production is made of at least five essential systems [8]: (1) the *PolymericMatrix*; (2) the *Vulcanization* system; (3) the *ProcessAid*; (4) the *Antidegradant*; (5) the *ReinforcingFiller*<sup>1</sup>.

$$\begin{aligned} \text{PolymericMatrix} \equiv & \text{System} \sqcap (= 100 \succ .(\text{NaturalRubber} \sqcup \text{ButadieneRubber})) \\ \text{Vulcanization} \equiv & \text{System} \sqcap ((\geq 1 \succ .\text{Sulphur}) \sqcap (\leq n \succ .\text{Sulphur})) \\ & \sqcap ((\geq 1 \succ .(\text{GroundElement} \sqcap \text{hasFamilyName.Accellerant})) \\ & \sqcap (\leq m \succ .(\text{GroundElement} \sqcap \text{hasFamilyName.Accellerant}))) \\ & \sqcap ((\geq 2 \succ .\text{ZincOxide}) \sqcap (\leq p \succ .\text{ZincOxide})) \\ & \sqcap ((\geq 2 \succ .\text{StearicAcid}) \sqcap (\leq p \succ .\text{StearicAcid})) \end{aligned}$$

The *PolymericMatrix* is a system having 100 parts as a blend of natural and synthetic rubber or, alternatively, 100 parts of natural or synthetic rubber alone. Parts of the *Vulcanization* system are the Sulphur, the Oxide Zinc and the Stearic Acid in a predefined *quantity*. A vulcanization system contains a given quantity of a element in the family of the Accellerant.

$$\begin{aligned} \text{ButadieneRubber} \equiv & \text{GroundElement} \sqcap (= 1 \prec .\text{PolymericMatrix}) \\ & \sqcap \text{hasStructure} .(\text{Cis} \sqcup \text{Trans}) \sqcap \text{hasFamilyName} .\text{Polymer} \\ & \sqcap \text{hasMolecularWeight} .\text{NumericalValue} \end{aligned}$$

<sup>1</sup> Since the syntax of *SIQ* does not admit individuals in the TBox; the  $m, n, p$  symbols of the following formulas are interpreted as placeholders for appropriate integers according to specific compounding domains.

ButadieneRubber is a ground element and an exclusive part of the polymeric matrix system. Butadiene rubber, that is member of the family of Polymers, is characterized by having a specific configuration (*Cis* or a *Trans*), and a *molecular weight*.

Descriptions of specific compounds are represented by means of ABox assertions. Concept atoms in the ABoxes are all *defined names* of the TBox. Structural-level representation of compounds is completely represented by the mereological role assertions in the ABox, i.e. the assertions in the ABox concerning the  $\succ$  and  $\prec$  roles between the domain individuals. Moreover, the ABox specifies the quantity of all the ingredients available for a given compounding process<sup>2</sup>.

### 3 The heuristic search space

Within a Heuristic Search paradigm, the compound formulations (ABoxes) represent the *states*, while the *transitions* are interpreted as admissible transformation actions of these formulations. The state space is defined as a product of LTSs.

**Definition 1 (Labelled Transition System).** A labelled transition system  $\Gamma$  is a structure  $\langle S, i, \Lambda, \rightarrow \rangle$ , where (i)  $S$  is a set of states, with initial state  $i$ , (ii)  $\Lambda$  is a set of labels, and (iii)  $\rightarrow \subseteq S \times \Lambda \times S$  is a ternary relation of labeled transitions. If  $p, q \in S$  and  $\alpha \in \Lambda$ , then  $(p, \alpha, q) \in \rightarrow$ .

If the states of the LTSs are represented by sets of ABox assertions, the transitions between these states (*quantity increase* and *reduction*, and *substitution* actions) that define compound transformation actions can be viewed as ABox update operations. Let  $\theta$  be a transformation action whose effect is to *increase* the quantity of a given ingredient<sup>3</sup>.

**Definition 2 (Increase Transformation Action).** Given a LTS  $S = \langle S, i, \Lambda, \rightarrow \rangle$ , and a state  $s \in S$ , let  $\mathcal{A}_s$  be the ABox associated to  $s$ . Let  $\mathcal{A}_s$  contain a set of assertions of the form  $\succ (sys^i, ing^j)$ , where  $sys^i$  and  $ing^j$  are individuals respectively of the classes  $i$  and  $j$ , with  $i \sqsubseteq System$  and  $j \sqsubseteq GroundElement$ , and  $j$  is a base symbol in the definition of the concept  $i$ . A transition  $\theta_{i,j} \in \{\rightarrow\}$  is an increase transformation action for a given ingredient iff its application  $\theta_{i,j}\mathcal{A}_s$  returns an updated ABox  $\mathcal{A}_{s'}$  =  $\mathcal{A}_s \cup \succ (sys^i, ing'^j)$ , where  $ing^j \neq ing'^j$ .

Once the levels of representation have been chosen and the respective LTSs have been defined, the problem solving knowledge can be formally represented by means of two *morphisms*, mapping states to states and transitions to transitions of the different systems. Morphisms codify expert causal knowledge linking structural transformations on the compound to behavioral and teleological ones; they forecast the behavioral and teleological effects of a structural transformation (e.g. a quantity increase of Silica worsens the abrasive and resistance behaviors).

<sup>2</sup> Ingredients can be intuitively considered as chemical “bricks” with a proper name, where a brick is a fixed quantity of a given chemical substance.

<sup>3</sup> The transformation actions of quantity reduction and substitution of chemicals have been defined as ABox updates in a similar way.

A morphism  $\Gamma \rightarrow \Gamma'$  between transition systems can be introduced as a pair  $(\sigma, \lambda)$ , where  $\sigma$  is a function on states, preserving initial states, and  $\lambda$  is a partial function  $\lambda$  on the transition labels. The morphism maps a transition of  $\Gamma$  to a transition of  $\Gamma'$ : if  $(p, \alpha, q)$  is a transition in  $\Gamma$  then  $(\sigma(p), \lambda(\alpha), \sigma(q))$  is a transition in  $\Gamma'$  provided that  $\lambda(\alpha)$  is defined. Provided the suitable constraints on the transitions by means the introduced morphisms, the definition of the product of  $n$  labelled transition systems is as usual [9]. Suppose that the states  $s = \langle c, l, h \rangle$  and  $s' = \langle c', l', h' \rangle$  are elements of a product of three LTSs (where,  $c$  stands for *compound structure*,  $l$  for *low-level behaviors* and  $h$  for *high-level performances*). Let  $\tau = (\sigma, \lambda), \tau' = (\sigma', \lambda')$  are compounding morphisms such that  $\sigma(c) = l$  and  $\sigma'(l) = h$ ; these morphisms represent that a compound structure  $c$  is associated to specific compound behaviors  $l$  and performances  $h$ . Morphisms are necessarily given as inputs of the compounding problem. Then, the application of the transition  $\lambda_i$  to  $s$  (creating a new state  $s'$ ) is generated by the application of a structural transformation action  $\lambda_C: C \rightarrow C$  (mapping compound structures to compound structures) such that  $\lambda_C(c) = c'$ . But the application of  $\lambda_C$  leads to a partial state  $\langle c', l, h \rangle$  that is not well defined. In order to obtain a well defined state  $s'$ , representing a feasible solution of the compounding problem, morphisms are exploited applying the transformations associated to  $\lambda_C$  to obtain the other involved dimensions ( $l'$  and  $h'$ ). Formally, the notion of “well defined transition” is introduced as follows:

**Definition 3 (Well Defined Transition).** *Given the compounding morphisms  $\tau, \tau'$ , two states  $s = \langle c, l, h \rangle$  and  $s' = \langle c', l', h' \rangle$  in the product, and a structural transformation action  $\lambda_C$ . The triple  $(s, \lambda_i, s')$  is a well defined transition, written  $(s, \lambda_i, s') \in \rightarrow$ , iff  $\lambda_C(c) = c', \tau(c') = l'$  and  $\tau'(l') = h'$ . The state components  $l'$  and  $h'$  are obtained by mapping the transition  $\lambda_C$  to appropriate transitions  $\lambda_L$  and  $\lambda_H$ . In particular, if  $\tau(c) = l$ , and  $\lambda_C(c) = c'$ , then there exists a state  $l'$  such that  $\lambda_L(l) = l'$ , for some  $\lambda_L = \tau(\lambda_C)$ , with  $\tau(c') = l'$ .*

## 4 The compounding actions as ABox updates

Let us start with an example of update concerning the reinforcing filler system. Let  $\mathcal{T}$  be a TBox containing only the following terminological axiom (stating that a reinforcing filler system must contain *at least* one and *at most* two parts of carbon black):

$$\text{ReinforcingFiller} \sqsubseteq \text{System} \sqcap ((\geq 1 \succ .\text{CarbonBlack}) \sqcap (\leq 2 \succ .\text{CarbonBlack}))$$

Let  $s$  be a state of the LTS associated to  $\mathcal{T}$ . The following set of assertions  $\mathcal{A}$  is a fragment (that takes into account the composition of the reinforcing filler system) of the complete description of a compound formulation:  $\{\text{ReinforcingFiller}(\text{rf0a01}), \text{CarbonBlack}(\text{cb0a01}), \text{CarbonBlack}(\text{cb0a02}), \text{CarbonBlack}(\text{cb0a03}), \succ (\text{rf0a01}, \text{cb0a01}), \succ (\text{rf0a01}, \text{cb0a02})\}$ .

Let  $\theta$  be a transformation action whose effect is to *increase* the quantity of the carbon black contained in the reinforcing filler system. Considered as an ABox update, the application of  $\theta$  to  $\mathcal{A}$ , written  $\theta\mathcal{A}$ , returns the ABox  $\mathcal{A}'$  where  $\mathcal{A}' = \mathcal{A} \cup \{\succ (\text{rf0a01}, \text{cb0a03})\}$ . Let  $\mathcal{M}$  be an interpretation that satisfies  $\mathcal{A}$  w.r.t.

$\mathcal{T}$  ( $\mathcal{M}$  is said to be a model of  $\mathcal{A}$  w.r.t.  $\mathcal{T}$ ). In order to compute the consistency of the updated ABox  $\mathcal{A}'$ , the existence of a model  $\mathcal{M}'$  of  $\mathcal{A}'$  w.r.t. to  $\mathcal{T}$  has to be checked. Given the semantics of the cardinality constraint in  $\mathcal{T}$ , the example shows that a similar model does not exist: the insertion of  $\succ (rf0a01, cb0a03)$  produces the violation of the *at most* cardinality constraint in  $\mathcal{T}$ .

The above example can be generalized in order to define an algorithmic procedure that check consistency of an ABox update w.r.t. to an invariant TBox for compounding. The algorithm is aimed at checking consistency of updates that consist in insertion and deletion of mereological individual assertions (i.e. the assertions involving  $\prec$  and  $\succ$  roles). This assumption covers all the relevant cases of ABox updates of our model for chemical compound formulation. In order to achieve the generalization, we define the notions of *compounding TBox* and of *compounding ABox*.

A TBox  $\mathcal{T}$  is said to be a *compounding TBox* iff it contains a set of axioms with the following syntactic form:

$$\begin{aligned}
 C_{COMP} &\equiv \text{Compound} \sqcap (= 1 \succ .C_{SYS_1}) \\
 &\dots \\
 &\sqcap (= 1 \succ .C_{SYS_n}) \\
 &\sqcap ((\geq 0 \succ .C_{SYS_{n+1}}) \sqcap (\leq 1 \succ .C_{SYS_{n+1}})) \\
 &\dots \\
 &\sqcap ((\geq 0 \succ .C_{SYS_{n+m}}) \sqcap (\leq 1 \succ .C_{SYS_{n+m}})) \\
 C_{SYS} &\equiv \text{System} \sqcap ((\geq min_1 \succ .C_{ING_1}) \sqcap (\leq max_1 \succ .C_{ING_1})) \\
 &\dots \\
 &((\geq min_n \succ .C_{ING_n}) \sqcap (\leq max_n \succ .C_{ING_n})) \\
 C_{ING} &\equiv \text{GroundElement} \sqcap (= 1 \prec .C_{SYS_j}) \\
 &\sqcap (f_1.C_1) \sqcap \dots \sqcap (f_s.C_s)
 \end{aligned}$$

In the above formula schema,  $C_{COMP}, C_{SYS}, C_{ING_1}, \dots, C_{ING_n}$  are concept names for compounds, systems and ingredients respectively, and  $min_j, max_j$  are variable for integers with  $min_j \leq max_j$  for  $1 \leq j \leq n$ . The scheme can be instantiated in a number of different cardinality constraint axioms, fixing the quantity  $min_j$ , and  $max_j$  of a specific set of ingredient  $C_{ING_1}, \dots, C_{ING_n}$  a given functional system  $C_{SYS_i}$  must contain.

For all the systems  $C_{SYS_i}$  that are relevant for a given compounding problem, and all the available chemicals  $C_{ING_i}$ , let  $\mathcal{A}$  be an ABox containing assertions of the following forms:

$$\begin{aligned}
 C_{COMP}(\text{compound}), C_{SYS_i}(\text{sys}), C_{ING_1}(\text{ing}_{1_1}), \dots, C_{ING_1}(\text{ing}_{1_j}) \\
 \dots \\
 C_{ING_n}(\text{ing}_{n_1}), \dots, C_{ING_n}(\text{ing}_{n_j})
 \end{aligned}$$

where *sys* and *ing* are variable names for individuals. The assertions in  $\mathcal{A}$  define *what are* the available systems and chemicals that could be parts of the compound. Given a specific compounding domain, we assume that all the ABoxes involved in the process contain an identical set of such assertions. During the compounding process, the system can take a quantity increase for an ingredient in the compound, or a substitution of an

ingredient with a different one of the same family, but it cannot generate new chemicals from scratch. Therefore, the ABoxes differ only with respect to the compound formulations they describe. An ABox  $\mathcal{A}$  is said to be a *compounding ABox* iff it contains a suitable set of axioms with the above syntactic form, plus a set of axioms like:  $\succ(\text{compound}, \text{sys})$ , and  $\succ(\text{sys}, \text{ing})$  for all systems and ingredients that participate to a given compound.

Given a compounding TBox  $\mathcal{T}$  and a compounding ABox  $\mathcal{A}$ , and an interpretation  $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$  that is a model of  $\mathcal{A}$  w.r.t.  $\mathcal{T}$ , the algorithm computes if the application of an increase transformation action  $\theta$  generates an updated ABox  $\mathcal{A}'$  that is consistent w.r.t.  $\mathcal{T}$ <sup>4</sup>.

**INPUT:** (i) a TBox  $\mathcal{T}$ , an ABox  $\mathcal{A}$ , an interpretation  $\mathcal{I}$  such that  $\mathcal{I} \models \mathcal{A}$  and  $\mathcal{I} \models \mathcal{T}$ ; (ii) an Increase Transformation Action  $\theta$ .

**OUTPUT:** a satisfiable knowledge base  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}' \rangle$ , where  $\mathcal{A}' = \theta\mathcal{A}$ , or **ERROR**.

1. **If**  $\theta$  is an *Increase Transformation Action* and  $\theta\mathcal{A} = \mathcal{A}'$ ,  $\mathcal{A}' = \mathcal{A} \cup \{\succ(\text{sys}, \text{ing}_{j_n})\}$ , for some system  $\text{sys}$  and ingredient  $\text{ing}_{j_n}$  of the family  $j$ , with  $\{C_{SYS}(\text{sys}), ING_j(\text{ing}_j)\} \subset \mathcal{A}$  **then**
2. **for each**  $m$ ,  $\alpha := \succ(\text{sys}, \text{ing}_{j_m})$ , with  $\alpha \subset \mathcal{A}$ ,  $n \neq m$  and  $\beta := C_{SYS} \sqsubseteq \text{System} \sqcap ((\geq \text{min}_j \succ .C_{ING_j}) \sqcap (\leq \text{max}_j \succ .C_{ING_j}))$ , with  $\beta \subset \mathcal{T}$
3. **if**  $\text{countOccurrence}(\alpha, \mathcal{A}) = a$  and  $a \geq \text{min}_j$  and  $a \leq \text{max}_j$  **then**  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}' \rangle$  is satisfiable **else ERROR**.

## 5 Concluding remarks and future work

The number of results about reasoning techniques and algorithms for ontology updating is still poor, and the update of ontological KBs remains a promising and challenging field of research [10–12]. Our work exploited the notion of model of an ABox, once the intensional level of the knowledge base has been specified. This means that, given an ABox, the consistency check is always performed with respect to a TBox that we assumed to be invariant. Even if the result of [11] for  $\mathcal{ALC}$ , is naturally inherited by  $\mathcal{SILQ}$ , we tried to define an algorithmic procedure to perform reasoning, e.g. the consistency check for updated ABoxes w.r.t. a TBox, outside the logic itself.

In the context of the P-Truck Project, a specific experimental campaign has been devised and encouraging results have been obtained from the application of several search algorithms (namely  $A^*$ ,  $IDA^*$ , Iterative Expansion and Branch and Bound) to a state space defined and implemented according to the present knowledge model. The algorithm for the consistency check of the ABox updates, has produced a significant reduction of the expansion rate of the space, and an automatic system has been developed and tested on a significant number of prototypical chemical compounding problems (e.g. the problem of increasing the Tread Tear Resistance, the problem of increasing the Rolling Resistance, together with a reduction of the Wet Handling, and the maintenance of all the remaining performances) [13]. Future developments of our work are aimed at a complete generalization of the introduced computational model, in order to define an ABox updating algorithm for arbitrary DL-based mereological KBs.

<sup>4</sup> We proceeded in a similarly way for reduction and substitution transformation actions.

## References

1. Young, D.: Computational Chemistry : A Practical Guide for Applying Techniques to Real World Problems. Wiley-Interscience (2001)
2. Bonet, B., Geffner, H.: Planning as heuristic search. *Artificial Intelligence* **129** (2001) 5–33
3. Bandini, S., Manzoni, S., Sartori, F.: Knowledge maintenance and sharing in the KM context: the case of P-Truck. In Cappelli, A., Turini, F., eds.: *AI\*IA 2003: Advances in Artificial Intelligence*, Proceedings of 8th Congress of the Italian Association for Artificial Intelligence, Pisa, September 2003. Volume 2829 of LNAI., Berlin, Springer-Verlag (2003) 499–510
4. Mosca, A.: A theoretical and computational inquiry into the Compounding Problem. Ph.D. thesis, Department of Computer Science, Systems, and Communication - University of Milano-Bicocca, Italy (2005)
5. Bandini, S., Mosca, A., Palmonari, M.: Mereological knowledge representation for the chemical formulation. In: *Second Formal Ontologies Meet Industry Workshop (FOMI 2006)*, December 14-15, Trento (Italy). (2006) Electronically published.
6. Bandini, S., Mosca, A., Palmonari, M.: Model-based chemical compound formulation. In: *Model-based Reasoning in Medicine and Science (Mbr-2006)*, Guangzhou (Canton), P. R. China. (2006) To appear.
7. Sattler, U.: Description logics for the representation of aggregated objects. In Horn, W., ed.: *Proceedings of the 14th European Conference on Artificial Intelligence*, IOS Press, Amsterdam (2000)
8. Hoffmann, W.: *Rubber Technology Handbook*. Oxford University Press, New York (1989)
9. MacLane, S.: *Categories for Working Mathematicians*. Springer Verlag, New York (1971)
10. Giacomo, G.D., Lenzerini, M., Poggi, A., Rosati, R.: On the update of description logic ontologies at the instance level. In: *AAAI*, AAAI Press (2006)
11. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. In Doherty, P., Mylopoulos, J., Welty, C., eds.: *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, AAAI Press (2006) 46–56
12. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*. CEUR-WS (2005)
13. Bandini, S., Mosca, A., Vanneschi, L.: Towards the use of genetic algorithms for the chemical formulation problem. In Manzoni, S., Palmonari, M., Sartori, F., eds.: *Proceedings of the 9th Congress of the Italian Association for Artificial Intelligence (AI\*IA 2005)*, Workshop on Evolutionary Computation (GSICE 2005), Milano, Centro Copie Bicocca (2005) ISBN 88-900910-0-2.

## A Hypertableau Calculus for $\mathcal{SHIQ}$

Boris Motik, Rob Shearer, and Ian Horrocks

University of Manchester, UK

**Abstract.** We present a novel reasoning calculus for the Description Logic  $\mathcal{SHIQ}$ . In order to reduce the nondeterminism due to general inclusion axioms, we base our calculus on hypertableau and hyperresolution calculi, which we extend with a blocking condition to ensure termination. To prevent the calculus from generating large models, we introduce “*anywhere pairwise blocking*”. Our preliminary implementation shows significant performance improvements on several well-known ontologies. To the best of our knowledge, our reasoner is currently the only one that can classify the original version of the GALEN terminology.

### 1 Introduction

Modern Description Logic reasoners, such as Pellet [10], FaCT++ [15], and RACER [5], are typically based on tableau calculi [1, Chapter 2], which demonstrate (un)satisfiability of a knowledge base  $\mathcal{K}$  via a constructive search for an abstraction of a model of  $\mathcal{K}$ . Despite numerous optimizations of the tableau procedure, ontologies are still encountered in practice that cannot be handled by existing systems. This is mainly because many different models might need to be examined, and each model might be very large [1, Chapter 3]. The former problem is due to *or-branching*: given a disjunctive assertion  $C \sqcup D(s)$ , a tableau algorithm nondeterministically guesses that either  $C(s)$  or  $D(s)$  holds. To show unsatisfiability of  $\mathcal{K}$ , *every* possible guess must lead to a contradiction: if assuming  $C(s)$  leads to a contradiction, the algorithm must backtrack and assume  $D(s)$ . This can clearly result in exponential behavior. GCIs—axioms of the form  $C \sqsubseteq D$ —are the main source of disjunctions: to ensure that  $C \sqsubseteq D$  holds, a tableau algorithm adds a disjunction  $\neg C \sqcup D(s)$  to each individual  $s$  in the model. Various *absorption* optimizations [1, Chapter 9][7, 14] reduce the high degree of nondeterminism in such a procedure; however, they often fail to eliminate all sources of nondeterminism. This may be the case even for ontologies that can be translated into Horn clauses (such as GALEN, NCI, and SNOMED), for which reasoning without any nondeterminism should be possible in principle.

The size of the model being constructed is determined by *and-branching*—the expansion of a model due to existential quantifiers. Apart from memory consumption problems, and-branching can increase or-branching by increasing the number of individuals to which GCIs are applied.

In this paper, we present a reasoning calculus that addresses both sources of complexity. We focus on the DL  $\mathcal{SHIQ}$ ; however, our calculus should be applicable to most DLs with known tableau algorithms. A  $\mathcal{SHIQ}$  knowledge

base is first preprocessed into *DL-clauses*—universally quantified implications containing DL concepts and roles as predicates. The main inference rule for DL-clauses is hyperresolution: an atom from the head of a DL clause is derived *only if* all atoms from the clause body have been derived. On Horn clauses, this calculus is deterministic, which eliminates all or-branching. Our algorithm can be viewed as a hybrid of resolution and tableau, and is related to the hypertableau [2] and hyperresolution [12] calculi.

Hyperresolution decides many first-order fragments (see, e.g., [4, 3] for an overview). Unlike most of these fragments, *SHIQ* allows for cyclic GCIs of the form  $C \sqsubseteq \exists R.C$ , on which hyperresolution can generate infinite paths of successors. Therefore, to ensure termination, we use the pairwise blocking technique from [6] to detect cyclic computations. Due to hyper-inferences, the soundness and correctness proofs from [6] do not carry over to our calculus. In fact, certain simpler blocking conditions for weaker DLs cannot be applied in a straightforward manner in our setting. To limit and-branching, we extend the blocking condition from [6] to *anywhere pairwise blocking*: an individual can be blocked by an individual that is not necessarily an ancestor. This significantly reduces the sizes of the constructed models.

We have implemented our calculus in a new reasoner. Even with a relatively naïve implementation, our system outperforms existing reasoners on several real-world ontologies. For example, the deterministic treatment of GCIs significantly reduces the classification time for the NCI ontology. Furthermore, the pairwise anywhere blocking strategy seems to be very effective in limiting model sizes. To the best of our knowledge, our reasoner is currently the only one that can classify the original version of the GALEN terminology.

## 2 Algorithm Overview

To see how GCIs can increase or-branching and thus cause performance problems, consider the following knowledge base  $\mathcal{K}_1$ :

$$(1) \quad \begin{aligned} \mathcal{T}_1 &= \{\exists R.A \sqsubseteq A\} \\ \mathcal{A}_1 &= \{\neg A(a_0), R(a_0, b_1), R(b_1, a_1), \dots, R(a_{n-1}, b_n), R(b_n, a_n), A(a_n)\} \end{aligned}$$

To satisfy the GCI, a tableau algorithm derives  $(\forall R.\neg A \sqcup A)(a_i)$ ,  $0 \leq i \leq n$  and  $(\forall R.\neg A \sqcup A)(b_j)$ ,  $1 \leq j \leq n$ . Assuming that  $a_i$  are processed before  $b_j$ , the algorithm derives  $\forall R.\neg A(a_i)$ ,  $0 \leq i \leq n$  and  $\neg A(b_i)$ ,  $1 \leq i \leq n$ , after which it derives  $\forall R.\neg A(b_i)$ ,  $1 \leq i \leq n-1$  and  $\neg A(a_i)$ ,  $1 \leq i \leq n$ . The ABox now contains a contradiction on  $a_n$ , so the algorithm flips its guess on  $b_{n-1}$  to  $A(b_{n-1})$ . This generates a contradiction on  $b_{n-1}$ , so the algorithm backtracks from all guesses for  $b_i$ . Next, the guess on  $a_n$  is changed to  $A(a_n)$  and the work for all  $b_i$  is repeated. This also leads to a contradiction, so the algorithm must revise its guess for  $a_{n-1}$ ; but then, two guesses are again possible for  $a_n$ . In general, after revising a guess for  $a_i$ , all possibilities for  $a_j$ ,  $i < j \leq n$ , must be reexamined, which results in exponential behavior. Note that none of the standard backtracking optimizations [1, Chapter 9] help us avoid this problem. Namely, the problem arises

because the order in which the individuals are processed makes the guesses on  $a_i$  independent from the guesses on  $a_j$ ,  $i \neq j$ . It is difficult to estimate in advance which order is optimal; in fact, the processing order is typically determined by implementation side-effects (such as the data structures used to store  $\mathcal{K}$ ).

The GCI  $\exists R.A \sqsubseteq A$  is not inherently nondeterministic: it is equivalent to the Horn clause  $\forall x, y : [R(x, y) \wedge A(y) \rightarrow A(x)]$ . By hyperresolution, we derive the facts  $A(b_n), A(a_{n-1}), \dots, A(a_0)$ , and eventually we drive a contradiction on  $a_0$ . These inferences are deterministic, so we can conclude that  $\mathcal{K}_1$  is unsatisfiable without any backtracking. This example suggests that the way tableau algorithms handle GCIs can be “unnecessarily” nondeterministic.

*Absorption* [1, Chapter 9] reduces the nondeterminism introduced by GCIs. If possible, it rewrites GCIs as  $B \sqsubseteq C$  with  $B$  an atomic concept; then, during reasoning, it derives  $C(s)$  only if the ABox contains  $B(s)$ . This localizes the applicability of the rewritten GCIs. Absorption has been extended to *binary absorption* [7], which rewrites a GCI to  $B_1 \sqcap B_2 \sqsubseteq C$ , and to *role absorption* [14], which rewrites a GCI to  $\exists R.\top \sqsubseteq C$ . Note, however, that the axiom  $\exists R.A \sqsubseteq A$  cannot be absorbed directly. It can be absorbed if it is rewritten as  $A \sqsubseteq \forall R^-.A$ . In practice, it is often unclear in advance which combination of transformation and absorption techniques will yield the best results. Therefore, implemented absorption algorithms are guided primarily by heuristics.

Our algorithm can be seen as a generalization of absorption. It first translates GCIs into *DL-clauses*—universally quantified implications of the form  $\bigwedge U_i \rightarrow \bigvee V_j$ , where  $U_i$  are of the form  $R(x, y)$  or  $A(x)$ , and  $V_j$  are of the form  $R(x, y)$ ,  $A(x)$ ,  $\exists R.C(x)$ ,  $\geq n R.C(x)$ , or  $x \approx y$ . DL-clauses are used in *hyperresolution* inferences, which derive some  $V_j$ , but only if all  $U_i$  are matched to assertions in the ABox. This calculus is quite different from the standard DL tableau calculi. For example, it has no *choose*-rule for qualified number restrictions [13], and it can handle implications such as  $R(x, y) \rightarrow B(x) \vee A(y)$  (obtained from  $\exists R.\neg A \sqsubseteq B$ ) that contain several universally quantified variables.

It is easy to see that and-branching can cause the introduction of infinitely many new individuals. Consider the following (satisfiable) knowledge base  $\mathcal{K}_2$ :

$$(2) \quad \mathcal{T}_2 = \left\{ \begin{array}{l} A_1 \sqsubseteq \geq 2 S.A_2, \dots, A_{n-1} \sqsubseteq \geq 2 S.A_n, A_n \sqsubseteq A_1, \\ A_i \sqsubseteq (B_1 \sqcup C_1) \sqcap \dots \sqcap (B_m \sqcup C_m) \text{ for } 1 \leq i \leq n \end{array} \right\} \quad \mathcal{A}_2 = \{A_1(a)\}$$

To check satisfiability of  $\mathcal{K}_2$ , a tableau algorithm builds a binary tree with each node labeled with some  $A_i$  and an element of  $\Pi = \{B_1, C_1\} \times \dots \times \{B_m, C_m\}$ . A naïve algorithm would try to construct an infinite tree, so tableau algorithms employ *blocking* [6]: if a node  $a$  is labeled with the same concepts as some ancestor  $a'$  of  $a$ , then the existential quantifiers for  $a$  are not expanded. This ensures termination; however, the number of elements in  $\Pi$  is exponential, so, with “unlucky” guesses, the tree can be exponential in depth and doubly exponential in total. In the best case, the algorithm can, for example, choose  $B_j$  rather than  $C_j$  for each  $1 \leq j \leq m$ . It then constructs a polynomially deep binary tree and thus runs in exponential time.

To curb and-branching, we extend pairwise blocking [6] to *anywhere pairwise blocking*, in which an individual can be blocked not only by an ancestor, but by

any individual satisfying certain ordering requirements. This reduces the worst-case complexity of the algorithm by an exponential factor; for example, on  $\mathcal{K}_2$ , after we exhaust all members of  $\Pi$ , all subsequently created individuals will be blocked. Such blocking can sometimes also improve the best-case complexity; for example, on  $\mathcal{K}_2$  our algorithm can create a polynomial path and then use the individuals from that path to block their siblings.

### 3 The Satisfiability Checking Algorithm

Our algorithm consists of two phases: preprocessing and inferencing.

#### 3.1 Preprocessing

The goal of the preprocessing phase is to transform a  $\mathcal{SHIQ}$  knowledge base into a *normalized* ABox (in which all concept assertions are of the form  $B(s)$  or  $\geq n R.B(s)$  and all role assertions involve only atomic roles), and a collection of *DL-clauses*, which we denote as  $\Xi(\mathcal{K})$  in the rest of this paper. We omit the details of the transformation due to lack of space; the complete algorithm is described in [9] and illustrated here by example.

Our calculus does not deal with transitive roles, so we transform the  $\mathcal{SHIQ}$  knowledge base into an equisatisfiable  $\mathcal{ALCHIQ}$  knowledge base using the well-known encoding described in [8, Section 5.2]. The next problem is that concepts in  $\mathcal{ALCHIQ}$  axioms can occur under implicit negation. We make negation explicit by moving all concepts to the right-hand side of the implication and putting all concepts into negation-normal form. For example, the axiom

$$(3) \quad \exists R.(C \sqcap D) \sqsubseteq \exists S.(E \sqcup F)$$

is rewritten as follows:

$$(4) \quad \top \sqsubseteq \forall R.(\neg C \sqcup \neg D) \sqcup \exists S.(E \sqcup F)$$

It is well known that naïve clausification of  $\mathcal{ALCHIQ}$  axioms would result in exponential blowup. We instead apply a variant of the well-known structural transformation [11], which replaces complex concepts with new names and introduces new axioms to define these names. Our goal, however, is to obtain Horn DL-clauses whenever possible. As discussed in [9], if we are not careful the structural transformation can destroy the Horn-ness of an axiom. Therefore, we modify the transformation to replace a complex concept  $C$  with either  $A$  or  $\neg A$ , where  $A$  is a fresh atomic name. The polarity of the replacement concept is chosen such that the axiom that defines the replacement will not introduce additional nondeterminism into the final clause set; this condition is detected by analyzing the structure of literals within the replaced concept. Applying our structural transformation to (4) gives us the following axioms:

$$(5) \quad \top \sqsubseteq \forall R.\neg Q_1 \sqcup \exists S.Q_2$$

$$(6) \quad \neg Q_1 \sqsubseteq \neg C \sqcup \neg D$$

$$(7) \quad Q_2 \sqsubseteq E \sqcup F$$

Without complex subexpressions or implicit negation, the transformation to DL-clauses is straightforward. Universal restrictions are rewritten using their first-order-logic interpretations; e.g.  $\forall R.C$  is rewritten as  $\neg R(x, y) \vee C(y)$ . Negated atoms are then moved to the antecedent of the DL-clause, and positive atoms are moved to the consequent. The DL-clauses derived from (3) are as follows:

$$\begin{aligned}
 (8) \quad & R(x, y) \wedge Q_1(y) \rightarrow \exists S.Q_2(x) \\
 (9) \quad & C(x) \wedge D(x) \rightarrow Q_1(x) \\
 (10) \quad & Q_2(x) \rightarrow E(x) \vee F(x)
 \end{aligned}$$

### 3.2 The Hypertableau Calculus for DL-Clauses

We now present our hypertableau calculus for deciding satisfiability of  $\mathcal{A} \cup \Xi(\mathcal{K})$ .

**Definition 1. *Unnamed Individuals.*** For a set of named individuals  $N_I$ , the set of all individuals  $N_X$  is inductively defined as  $N_I \subseteq N_X$  and, if  $x \in N_X$ , then  $x.i \in N_X$  for each integer  $i$ . The individuals in  $N_X \setminus N_I$  are unnamed. An individual  $x.i$  is a successor of  $x$ , and  $x$  is a predecessor of  $x.i$ ; descendant and ancestor are the transitive closures of successor and predecessor, respectively.

**Pairwise Anywhere Blocking.** A concept is blocking-relevant if it is of the form  $A$ ,  $\geq n R.A$ , or  $\geq n R.\neg A$ , for  $A$  an atomic concept. The label of an individual  $s$  and of an individual pair  $\langle s, t \rangle$  in an ABox  $\mathcal{A}$  are defined as follows:

$$\begin{aligned}
 \mathcal{L}_{\mathcal{A}}(s) &= \{C \mid C(s) \in \mathcal{A} \text{ and } C \text{ is a blocking-relevant concept}\} \\
 \mathcal{L}_{\mathcal{A}}(s, t) &= \{R \mid R(s, t) \in \mathcal{A}\}
 \end{aligned}$$

Let  $\prec$  be a strict ordering (i.e., a transitive and irreflexive relation) on  $N_X$  containing the ancestor relation—that is, if  $s'$  is an ancestor of  $s$ , then  $s' \prec s$ . By induction on  $\prec$ , we assign to each individual  $s$  in  $\mathcal{A}$  a status as follows:

- $s$  is directly blocked by an individual  $s'$  iff both  $s$  and  $s'$  are unnamed,  $s'$  is not blocked,  $s' \prec s$ ,  $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(s')$ ,  $\mathcal{L}_{\mathcal{A}}(t) = \mathcal{L}_{\mathcal{A}}(t')$ ,  $\mathcal{L}_{\mathcal{A}}(s, t) = \mathcal{L}_{\mathcal{A}}(s', t')$ , and  $\mathcal{L}_{\mathcal{A}}(t, s) = \mathcal{L}_{\mathcal{A}}(t', s')$ , for  $t$  and  $t'$  the predecessors of  $s$  and  $s'$ , resp.
- $s$  is indirectly blocked iff its predecessor is blocked.
- $s$  is blocked iff it is either directly or indirectly blocked.

**Pruning.** The ABox  $\text{prune}_{\mathcal{A}}(s)$  is obtained from  $\mathcal{A}$  by removing all assertions of the form  $R(t, t.i)$ ,  $R(t.i, t)$ ,  $C(t.i)$ ,  $u \approx t.i$ , and  $u \not\approx t.i$ , where  $t$  is either  $s$  or some descendant of  $s$ ,  $i$  is an integer, and  $u$  is an arbitrary individual.

**Merging.** The ABox  $\text{merge}_{\mathcal{A}}(s \rightarrow t)$  is obtained from  $\text{prune}_{\mathcal{A}}(s)$  by replacing the individual  $s$  with the individual  $t$  in all assertions.

**Derivation Rules.** Table 1 specifies derivation rules that, given an ABox  $\mathcal{A}$  and a set of DL-clauses  $\Xi(\mathcal{K})$ , derive the ABoxes  $\mathcal{A}_1, \dots, \mathcal{A}_n$ . In the Hyp-rule,  $\sigma$  is a mapping from  $N_V$  to the individuals occurring in  $\mathcal{A}$ , and  $\sigma(U)$  is the atom obtained from  $U$  by replacing each variable  $x$  with  $\sigma(x)$ .

**Derivation.** For a normalized  $\mathcal{ALCHIQ}$  knowledge base  $\mathcal{K} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ , a derivation is a pair  $(T, \lambda)$  where  $T$  is a finitely branching tree and  $\lambda$  is a function

**Table 1.** Derivation Rules of the Tableau Calculus

<i>Hyp</i> -rule	If 1. $U_1 \wedge \dots \wedge U_m \rightarrow V_1 \vee \dots \vee V_n \in \Xi(\mathcal{K})$ , 2. a mapping $\sigma : N_V \rightarrow N_{\mathcal{A}}$ exists, for $N_{\mathcal{A}}$ the set of individuals in $\mathcal{A}$ , 3. $\sigma(U_i) \in \mathcal{A}$ for each $1 \leq i \leq m$ , 4. $\sigma(V_j) \notin \mathcal{A}$ for each $1 \leq j \leq n$ , then if $n = 0$ , then $\mathcal{A}_1 = \mathcal{A} \cup \{\perp\}$ , otherwise $\mathcal{A}_j := \mathcal{A} \cup \{\sigma(V_j)\}$ for $1 \leq j \leq n$ .
$\geq$ -rule	If 1. $\geq n R.C(s) \in \mathcal{A}$ , 2. $s$ is not blocked in $\mathcal{A}$ , and 3. there are no individuals $u_1, \dots, u_n$ such that $\{\text{ar}(R, s, u_i), C(u_i) \mid 1 \leq i \leq n\} \cup \{u_i \not\approx u_j \mid 1 \leq i < j \leq n\} \subseteq \mathcal{A}$ , then $\mathcal{A}_1 := \mathcal{A} \cup \{\text{ar}(R, s, t_i), C(t_i) \mid 1 \leq i \leq n\} \cup \{t_i \not\approx t_j \mid 1 \leq i < j \leq n\}$ where $t_1, \dots, t_n$ are fresh pairwise distinct successors of $s$ .
$\approx$ -rule	If 1. $s \approx t \in \mathcal{A}$ and 2. $s \neq t$ then $\mathcal{A}_1 := \text{merge}_{\mathcal{A}}(s \rightarrow t)$ if $t$ is named or if $s$ is a descendant of $t$ , $\mathcal{A}_1 := \text{merge}_{\mathcal{A}}(t \rightarrow s)$ otherwise.
$\perp$ -rule	If 1. $s \not\approx s \in \mathcal{A}$ or $\{A(s), \neg A(s)\} \subseteq \mathcal{A}$ and 2. $\perp \notin \mathcal{A}$ then $\mathcal{A}_1 := \mathcal{A} \cup \{\perp\}$ .

$$\text{ar}(R, s, t) = \begin{cases} R(s, t) & \text{if } R \text{ is an atomic role} \\ S(t, s) & \text{if } R \text{ is an inverse role and } R = S^- \end{cases}$$

that labels the nodes of  $T$  with ABoxes such that (i)  $\lambda(\epsilon) = \mathcal{A}$  for  $\epsilon$  the root of the tree, and (ii) for each node  $t$ , if one or more derivation rules are applicable to  $\lambda(t)$  and  $\Xi(\mathcal{K})$ , then  $t$  has children  $t_1, \dots, t_n$  such that  $\lambda(t_1), \dots, \lambda(t_n)$  are the result of applying one (arbitrarily chosen) applicable rule to  $\lambda(t)$  and  $\Xi(\mathcal{K})$ .

**Clash.** An ABox  $\mathcal{A}$  contains a clash iff  $\perp \in \mathcal{A}$ ; otherwise,  $\mathcal{A}$  is clash-free.

In [6], the successor relation is encoded using role arcs, which point only from predecessors to successors. Since our ABoxes contain only atomic roles, role arcs can point in both directions, so we encode the successor relation in the individuals. The ordering  $\prec$  ensures that there are no cyclic blocks, so all successors of nonblocked individuals have been constructed. *Ancestor pairwise blocking* from [6] is obtained if  $\prec$  is exactly the descendant relation.

Pruning prevents infinite loops of merge-create rule applications—the so-called “yo-yo” effect. Intuitively, merging ensures that no individual “inherits” successors through merging. In [6], the successors are not physically removed, but are marked as “not present” by setting their edge labels to  $\emptyset$ . This has exactly the same effect as pruning.

The relationship between our new calculus and knowledge base satisfiability is given by the following theorem:

**Theorem 1.** A SHIQ knowledge base  $\mathcal{K}$  is satisfiable if and only if each derivation from  $\mathcal{K}' = \Xi(\mathcal{K})$  contains a leaf node  $t$  such that  $\lambda(t)$  is clash-free; furthermore, the construction of each such derivation terminates.

**Table 2.** Results of Performance Evaluation

Ontology	HT	HT-anc	Pellet	FaCT++	Racer
NCI	8 s	9 s	44 min	32 s	36 s
GALEN original	44 s	—	—	—	—
GALEN simplified	7 s	104 s	—	859 s	—

## 4 Implementation

Based on the calculus from Section 3, we have implemented a prototype DL reasoner.<sup>1</sup> Currently, it can only handle Horn DL-clauses—our main goal was to show that significant performance improvements can be gained by exploiting the deterministic nature of many ontologies.

To classify a knowledge base  $\mathcal{K}$ , we run our algorithm on  $\mathcal{K}_i = \mathcal{K} \cup \{C_i(a_i)\}$  for each concept  $C_i$ , obtaining an ABox  $\mathcal{A}_i$ . If  $D(a_i) \in \mathcal{A}_i$  and  $D(a_i)$  was derived without making any nondeterministic choices, then  $\mathcal{K} \models C_i \sqsubseteq D$ . Since our test ontologies are translated to Horn DL-clauses on which our algorithm is deterministic,  $D(a_i) \in \mathcal{A}_i$  iff  $\mathcal{K} \models C_i \sqsubseteq D$ . Thus, we can classify  $\mathcal{K}$  with a linear number of calls to our algorithm. This optimization is also applicable in standard tableau calculi; the nondeterministic handling of GCIs, however, diminishes its value. We also developed an optimization of anywhere blocking which caches the signatures of unblocked nodes in completed models and uses them as blocking candidates in new models; full details can be found in [9].

Table 2 shows the times that our reasoner, Pellet 1.3, FaCT++ 1.1.4, and Racer 1.9.0 take to classify our test ontologies. To isolate the improvements due to each of the two innovations of our algorithm, we evaluated our system with anywhere blocking (denoted as HT), as well as with ancestor blocking [6] (denoted as HT-anc). All ontologies are available from our reasoner’s Web page.

NCI is a relatively large (about 23000 atomic concepts) but simple ontology. FaCT++ and RACER can classify NCI in a short time mainly due to an optimization which eliminates many unnecessary tests, and the fact that all axioms in NCI are definitional so they are handled efficiently by absorption. We conjecture that Pellet is slower by two orders of magnitude because it does not use these optimizations, so it must deal with disjunctions.

GALEN has often been used as a benchmark for DL reasoning. The original version of GALEN contains about 2700 atomic concepts and many GCIs similar to (2). Most GCIs cannot be absorbed without any residual nondeterminism. Thus, the ontology is hard because it requires the generation of large models with many nondeterministic choices. Hence, GALEN has been simplified by removing 273 axioms, and this simplified version of GALEN has commonly been used for performance testing. As Table 2 shows, only HT can classify the original version of GALEN. In particular, anywhere blocking prevents our reasoner from generating the same fragments of a model in different branches.

<sup>1</sup> <http://www.cs.man.ac.uk/~bmotik/Hermit/>

## 5 Conclusion

In this paper, we presented a novel reasoning algorithm for DLs that combines hyper-inferences to reduce the nondeterminism due to GCIs with anywhere blocking to reduce the sizes of generated models. In future, we shall extend our reasoner to handle disjunction and conduct a more comprehensive performance evaluation. Furthermore, we shall investigate the possibilities of optimizing the blocking condition and heuristically guiding the model construction to further reduce the sizes of the models created. Finally, we shall try to extend our approach to the DLs *SHOIQ* and *SROIQ*, which provide the logical underpinning of the Semantic Web ontology languages.

## References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. JELIA '96*, pages 1–17, Évora, Portugal, September 30–October 3 1996.
3. C. Fermüller, T. Tammet, N. Zamov, and A. Leitsch. *Resolution Methods for the Decision Problem*, volume 679 of *LNAI*. Springer, 1993.
4. C. G. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution Decision Procedures. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 25, pages 1791–1849. Elsevier Science, 2001.
5. V. Haarslev and R. Möller. RACER System Description. In *Proc. IJCAR 2001*, pages 701–706, Siena, Italy, June 18–23 2001.
6. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic *SHIQ*. In *Proc. CADE-17*, pages 482–496, Pittsburgh, USA, 2000.
7. A. K. Hudek and G. Weddell. Binary Absorption in Tableaux-Based Reasoning for Description Logics. In *Proc. DL 2006*, Windermere, UK, May 30–June 1 2006.
8. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe, Germany, 2006.
9. Boris Motik, Rob Shearer, and Ian Horrocks. Optimized Reasoning in Description Logics using Hypertableaux. In *Proc. of the 16th Int. Conf. on Automated Deduction (CADE-16)*, July 17–20 2007. To appear.
10. B. Parsia and E. Sirin. Pellet: An OWL-DL Reasoner. Poster, In Proc. ISWC 2004, Hiroshima, Japan, November 7–11, 2004.
11. D. A. Plaisted and S. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Logic and Computation*, 2(3):293–304, 1986.
12. A. Robinson. Automatic Deduction with Hyper-Resolution. *Int. Journal of Computer Mathematics*, 1:227–234, 1965.
13. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.
14. D. Tsarkov and I. Horrocks. Efficient Reasoning with Range and Domain Constraints. In *Proc. DL 2004*, Whistler, BC, Canada, June 6–8 2004.
15. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. IJCAR 2006*, pages 292–297, Seattle, WA, USA, 2006.

## Expressive Querying over Fuzzy DL-Lite Ontologies<sup>(\*)</sup>

Jeff Z. Pan<sup>1</sup>, Giorgos Stamou<sup>2</sup>, Giorgos Stoilos<sup>2</sup>, and Edward Thomas<sup>1</sup>

<sup>1</sup> Department of Computing Science, University of Aberdeen, AB24 3UE, UK

<sup>2</sup> Department of Computer Science, National & Technical Univ. of Athens, Zographou 15780, GR

**Abstract.** Fuzzy Description Logics (f-DLs) have been proposed as formalisms capable of capturing and reasoning about imprecise and vague knowledge. The last years, research in Description Logics, and also in f-DLs, is largely focused on the development of languages where complexity of query answering is as efficient as query answering in data bases. One such example is the DL-Lite language and its fuzzy extension f-DL-Lite. In the current paper we present various a variety of query languages by which we can query a fuzzy DL-Lite knowledge base. Then, we present a prototype implementation for querying f-DL-Lite ontologies.

### 1 Introduction

Recently there have been quite a few work on DL-based fuzzy ontology languages [11, 9, 10, 8, 12], which have been proposed as formalisms capable of capturing and reasoning about imprecise and vague knowledge. In particular, Straccia [12] extended the DL-Lite ontology language [2], which enables highly efficient query answering procedures, to fuzzy DL-Lite. He showed that conjunctive query answering in f-DL-Lite is quite similar to query answering in crisp DL-Lite, although some technical details like top- $k$  answering and a modification on the knowledge base consistency algorithm of crisp DL-Lite need to be considered.

In this paper, we propose two novel query languages, which provide one with different ways on querying fuzzy DL-Lite ontologies. More precisely, we allow the users to specify threshold queries and general fuzzy queries. Comparing with Straccia’s query language, the threshold query language is flexible as it allows one to specify a threshold for each query atom (such as “tell me e-shops that are popular [with degrees at least 0.8] and sell good books [with degrees at least 0.9]”), while the general fuzzy query language is a general form of Straccia’s query language. Furthermore, we present algorithms for answering these queries and report implementations as well as preliminary but encouraging evaluation based on the ONTOSEARCH2, which is a query engine for both DL-Lite and fuzzy DL-Lite.

<sup>(\*)</sup> This extended abstract is accompanied with an online technical report (<http://www.ontosearch.org/TR/f-DL-Lite.pdf>), which contains more details including algorithms and the proofs.

## 2 f-DL-Lite

In the current section we will briefly introduce the fuzzy DL-Lite (which we call f-DL-Lite) language [12], which extends DL-Lite core with fuzzy assertions of the forms  $B(\mathbf{a}) \geq n, R(\mathbf{a}, \mathbf{b}) \geq n$ , where  $B$  is basic class,  $R$  is a property,  $\mathbf{a}$  and  $\mathbf{b}$  are individuals and  $n$  is a real number in the range  $[0, 1]$ . We assume that the reader is familiar with DL-Lite; in a different case [1, 2] are the standard sources. We only remind the form of *conjunctive queries* which we will use in the following. A conjunctive query (CQ)  $q$  is of the form

$$q(X) \leftarrow \exists Y. conj(X, Y) \tag{1}$$

where  $q(X)$  is called the head,  $conj(X, Y)$  is called the body,  $X$  are called the distinguished variables,  $Y$  are existentially quantified variables called the non-distinguished variables, and  $conj(X, Y)$  is a conjunction of atoms of the form  $A(v), R(v_1, v_2)$ , where  $A, R$  are respectively *named* classes and *named* properties,  $v, v_1$  and  $v_2$  are *individual* variables in  $X$  and  $Y$  or individual names in  $\mathcal{O}$ . The semantics of f-DL-Lite ontologies is defined in terms of *fuzzy interpretations* [11]. A fuzzy interpretation is a pair  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  where the domain  $\Delta^{\mathcal{I}}$  is a non-empty set of objects and  $\cdot^{\mathcal{I}}$  is a fuzzy interpretation function, which maps:

- an individual  $\mathbf{a}$  to an element of  $\mathbf{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ ,
- a named class  $A$  to a membership function  $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0, 1]$ , and
- a named property  $R$  to a membership function  $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]$ .

Using the fuzzy set theoretic operations [6], fuzzy interpretations can be extended to interpret f-DL-Lite class and property descriptions. Following Straccia [12], we use the Lukasiewicz negation,  $c(\mathbf{a})=1-\mathbf{a}$  and the Gödel t-norm for interpreting conjunctions,  $t(\mathbf{a}, \mathbf{b}) = \min(\mathbf{a}, \mathbf{b})$ . The semantics of f-DL-Lite class and property descriptions, and f-DL-Lite axioms are depicted in Table 1. Given the above semantics, it is obvious that crisp assertions  $B(\mathbf{a}), R(\mathbf{a}, \mathbf{b})$  are special forms of fuzzy assertions where  $n = 1$ .

Syntax	Semantics
$\exists R$	$(\exists R)^{\mathcal{I}}(o_1) = \sup_{o_2 \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(o_1, o_2)\}$
$\neg B$	$(\neg B)^{\mathcal{I}}(o) = 1 - B^{\mathcal{I}}(o)$
$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}}(o) = t(C_1^{\mathcal{I}}(o), C_2^{\mathcal{I}}(o))$
$R^-$	$(R^-)^{\mathcal{I}}(o_2, o_1) = R^{\mathcal{I}}(o_1, o_2)$
$B \sqsubseteq C$	$\forall o \in \Delta^{\mathcal{I}}, B^{\mathcal{I}}(o) \leq C^{\mathcal{I}}(o)$
$\text{Func}(R)$	$\forall o_1 \in \Delta^{\mathcal{I}}, \#\{o_2 \mid R^{\mathcal{I}}(o_1, o_2) > 0\} = 1$
$B(\mathbf{a}) \geq n$	$B^{\mathcal{I}}(\mathbf{a}^{\mathcal{I}}) \geq n$
$R(\mathbf{a}, \mathbf{b}) \geq n$	$R^{\mathcal{I}}(\mathbf{a}^{\mathcal{I}}, \mathbf{b}^{\mathcal{I}}) \geq n$

**Table 1.** Semantics of f-DL-Lite class and property descriptions, and f-DL-Lite axioms

### 3 Querying f-DL-Lite Ontologies

In this section, we introduce two query languages for f-DL-Lite ontologies. The first language extends conjunctive queries with thresholds for atoms in queries. This is motivated by the extension of the instance checking problem which considers a single fuzzy assertion. The second language is a general fuzzy query language, which is a general form of query languages, such as the fuzzy threshold query language and the query language proposed in [12].

#### 3.1 Two New Query Languages

*Threshold Queries* As noted in [1] in DL-Lite the instance checking problem is a special case of conjunctive queries. Since f-DL-Lite extends DL-Lite with fuzzy assertions, it would be natural for a query language to allow users to specify fuzzy assertions for atoms in queries. Thus, we define *conjunctive threshold queries* (CTQ) which extend atoms  $A(v), R(v_1, v_2)$  in conjunctive queries of the form (1) into the following forms  $A(v) \geq t_1, R(v_1, v_2) \geq t_2$ , where  $t_1, t_2 \in (0, 1]$  are thresholds.

*Example 1.* We can query models who are tall with a degree no less than 0.7 and light with a degree no less than 0.8 with the following conjunctive threshold query:

$$q(v) \leftarrow \text{Model}(v) \geq 1, \text{Tall}(v) \geq 0.7, \text{Light}(v) \geq 0.8.$$

It is obvious that threshold queries are more flexible than queries of the form (1) in that users can specify different thresholds for different atoms in their queries.

Formally, given an f-DL-Lite ontology  $\mathcal{O}$ , a conjunctive threshold query  $q_T$  and an evaluation  $[X \mapsto S]$ , we say  $\mathcal{O}$  entails  $q_T$  (denoted as  $\mathcal{O} \models_T q_T$ ) if every interpretation  $\mathcal{I}$  of  $\mathcal{O}$  satisfies the following condition: for each atom  $A(v) \geq t_1$  ( $R(v_1, v_2) \geq t_2$ ) of  $q_T$ , we have  $A^{\mathcal{I}}(v)_{X \mapsto S} \geq t_1$  (resp.  $R^{\mathcal{I}}(v_1, v_2)_{X \mapsto S} \geq t_2$ ). In this case,  $S$  is called a *solution* of  $q_T$ . A disjunctive threshold query (DTQ) is a set of conjunctive threshold queries sharing the same head.

*General Fuzzy Queries* Since f-DL-Lite associates assertions with degrees of truth, another useful feature for its query language is to associate degrees of truth with answers in answer sets of queries over f-DL-Lite ontologies. In threshold queries, an evaluation  $[X \mapsto S]$  either satisfies the query entailment or not; hence, answers of such queries are crisp. In this subsection, we introduce general fuzzy queries which allow fuzzy answers. Syntactically, general fuzzy conjunctive queries (GFCQ) extend the atoms  $A(v), R(v_1, v_2)$  of conjunctive queries of the form (1) into ones with the following form  $A(v) : k_1, R(v_1, v_2) : k_2$ , where  $k_1, k_2 \in (0, 1]$  are degrees. These syntactic extensions are similar with the ones proposed for fuzzy-SWRL in [8]. Thus, the existential quantifier is interpreted as sup, while we leave the semantics of the conjunction ( $G$ ) and that of the degree-associated atoms ( $a$ ) open. To simplify the presentation of the semantics, we use a unified representation  $atom_i(\bar{v})$  for atoms in general fuzzy conjunctive queries.

Given an f-DL-Lite ontology  $\mathcal{O}$ , an interpretation  $\mathcal{I}$  of  $\mathcal{O}$ , a general fuzzy conjunctive query  $q_F$  and an evaluation  $[X \mapsto S]$ , the degree of truth of  $q_F$  under  $\mathcal{I}$  is  $d = \sup_{S' \in \Delta^{\mathcal{I}} \times \dots \times \Delta^{\mathcal{I}}} \{G_{i=1}^n a(k_i, atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})\}$ , where  $k_i$  ( $1 \leq i \leq n$ ) and  $atom_i$  are as mentioned above,  $G$  is the semantic function for conjunctions and  $a$  is the semantic function for degree-associated atoms.  $S : d$  is called a *candidate solution* of  $q_F$ . When  $d > 0$ ,  $S : d$  is called a *solution* of  $q_F$ . Furthermore, the semantic functions should satisfy the following condition: If  $atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']} = 0$  for all possible  $S'$ ,  $d = 0$ .

A general fuzzy disjunctive query (GFDQ) is a set of general fuzzy conjunctive queries sharing the same head. The disjunction is interpreted as the s-norm ( $u$ ) of disjuncts. In what follows, we give some example of the semantic functions for conjunctions and degree-associated atoms.

1. Fuzzy threshold queries: If we use t-norms ( $t$ ) as the semantic function for conjunctions and  $R$ -implications ( $\omega_t$ ) as the semantic function for degree-associated atoms, we get fuzzy threshold queries, in which the degree of truth of  $q_F$  under  $\mathcal{I}$  is  $d = \sup_{S' \in \Delta^{\mathcal{I}} \times \dots \times \Delta^{\mathcal{I}}} \{t_{i=1}^n \omega_t(k_i, atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})\}$ .

Given some  $S'$ , if for all atoms we have  $atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']} \geq k_i$ , since  $\omega_t(x, y) = 1$  when  $y \geq x$  [6], we have  $d = 1$ ; this corresponds to threshold queries introduced earlier.

2. Straccia’s query language [12]: It is a special case of fuzzy threshold query language, where all  $k_i = 1$ . Since  $\omega_t(1, y) = y$  [6], the degree of truth of  $q_F$  under  $\mathcal{I}$  is  $d = \sup_{S' \in \Delta^{\mathcal{I}} \times \dots \times \Delta^{\mathcal{I}}} \{t_{i=1}^n atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})\}$ .

3. Fuzzy aggregation queries: if we use fuzzy aggregation functions [6], such as  $G(x) = \sum_{i=1}^n x_i$ , for conjunctions and  $a(k_i, y) = \frac{k_i}{\sum_{i=1}^n k_i} * y$  as the semantic function for degree-associated atoms, we get fuzzy aggregation queries, in which the degree of truth of  $q_F$  under  $\mathcal{I}$  is  $d = \sup_{S' \in \Delta^{\mathcal{I}} \times \dots \times \Delta^{\mathcal{I}}} \frac{\sum_{i=1}^n k_i * atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})}{\sum_{i=1}^n k_i}$ .

4. Fuzzy weighted queries: If we use generalised weighted t-norms as the semantic function for conjunction, we get fuzzy weighted queries, in which the degree of truth of  $q_F$  under  $\mathcal{I}$  is  $d = \sup_{S' \in \Delta^{\mathcal{I}} \times \dots \times \Delta^{\mathcal{I}}} \{\min_{i=1}^n u(\bar{k} - k_i t(\bar{k}, atom_i^{\mathcal{I}}(\bar{v})_{[X \mapsto S, Y \mapsto S']})\}$ , where  $\bar{k} = \max_{i=1}^n k_i$  and  $u$  is a t-conorm (fuzzy union).

The main idea of this type of queries is that they provide an aggregation type of operation, on the other hand an entry with a low value for a low-weighted criterion should not be critically penalized. Moreover, lowering the weight of a criterion in the query should not lead to a decrease of the relevance score, which should mainly be determined by the high-weighted criteria. For more details see [3].

### 3.2 Query Answering

This sub-section provides algorithms to answer the two kinds of queries (presented in the previous sub-section) over f-DL-Lite ontologies.

Algorithms for answering queries in f-DL-Lite mainly consist of four steps (like the algorithm for crisp DL-Lite [1]): (i) normalisation of the set  $\mathcal{T}$  of the class axioms of  $\mathcal{O}$  by the procedure  $\text{Normalise}(\mathcal{T})$ , which returns the normalised set  $\mathcal{T}'$  of class axioms; (ii) normalisation and storage of the set  $\mathcal{A}$  of individual axioms in  $\mathcal{O}$  by the procedure  $\text{Store}(\mathcal{A})$  that normalise  $\mathcal{A}$  and returns the relational database  $\text{DB}(\mathcal{A})$  of  $\mathcal{A}$ , as well as checking the consistency of  $\mathcal{O}$  by the procedure  $\text{Consistency}(\mathcal{O}, \mathcal{T}')$ ; (iii) reformulation of the input query  $q$  against the normalised set  $\mathcal{T}$  of the class axioms by the procedure  $\text{PerfectRef}(q, \mathcal{T}')$ , which returns a set  $Q$  of (conjunctive) queries; (iv) transformation of the set  $Q$  of (conjunctive) queries into SQL queries by the procedure  $\text{SQL}(Q)$ , as well as the evaluation of  $\text{SQL}(Q)$  by the procedure  $\text{Eval}(\text{SQL}(Q), \text{DB}(\mathcal{A}))$ .

*Answering Threshold Queries* Given an f-DL-Lite ontology  $\mathcal{O}$ , a conjunctive threshold query  $q_T$ , the procedure  $\text{Answer}_T(\mathcal{O}, q_T)$  computes the solutions of  $q_T$  w.r.t.  $\mathcal{O}$ , following the above steps (i) - (iv).

Algorithm	A-1:	$\text{Answer}_T(\mathcal{O}, q_T)$	Algorithm	A-2: $\text{SQL}_T(Q)$
1:	$\mathcal{T} = \text{Class-Axioms}(\mathcal{O})$		1:	$QS := \emptyset$
2:	$\mathcal{T}' = \text{Normalise}(\mathcal{T})$ //normalisation of class axioms		2:	for every query $q$ in $Q$ do
3:	$\mathcal{A} = \text{Individual-Axioms}(\mathcal{O})$		3:	sc:=Select-Clause( $q$ ) //construct the select-clause of $q$
4:	$\text{DB}(\mathcal{A}) = \text{Store}(\mathcal{A})$ //normalisation and storage of individual axioms		4:	fc:=From-Clause( $q$ ) //construct the from-clause of $q$
5:	if $\text{Consistency}(\mathcal{O}, \mathcal{T}') = \text{false}$ then		5:	wc1:=WC-Binding( $q$ ) //construct the part of the where-clause about binding
6:	return inconsistent // $\mathcal{O}$ is inconsistent		6:	wc2:=WC-Threshold( $q$ ) //construct the part of the where-clause that relates to thresholds
7:	end if		7:	$QS := QS \cup \text{Construct-SQL}(sc, fc, wc1, wc2)$
8:	return $\text{Eval}(\text{SQL}_T(\text{PerfectRef}_T(q_T, \mathcal{T}')), \text{DB}(\mathcal{A}))$		8:	end for
			9:	return $QS$

**Theorem 1.** Let  $\mathcal{O}$  be an f-DL-Lite ontology,  $q_T$  a conjunctive threshold query and  $S$  a tuple of constants.  $S$  is a solution of  $q_T$  w.r.t.  $\mathcal{O}$  iff  $S \in \text{Answer}_T(\mathcal{O}, q_T)$ .

See the online TR<sup>(\*)</sup> for detailed explanations of the algorithms A-1 and A-2, as well as the proof of Theorem 1.

*Answering General Fuzzy Queries* Similarly, given an f-DL-Lite ontology  $\mathcal{O}$ , a general fuzzy conjunctive query  $q_F$ , the procedure  $\text{Answer}_F(\mathcal{O}, q_F)$  computes the solutions of  $q_F$  w.r.t.  $\mathcal{O}$ .

**Algorithm A-3:**  $\text{Answer}_F(\mathcal{O}, q_F, a, G)$

- 1:  $T = \text{Class-Axioms}(\mathcal{O})$
- 2:  $T' = \text{Normalise}(T)$  //normalisation of class axioms
- 3:  $\mathcal{A} = \text{Individual-Axioms}(\mathcal{O})$
- 4:  $\text{DB}(\mathcal{A}) = \text{Store}(\mathcal{A})$  //normalisation and storage of individual axioms
- 5: **if**  $\text{Consistency}(\mathcal{O}, T') = \text{false}$  **then**
- 6:   **return** *inconsistent* //  $\mathcal{O}$  is inconsistent
- 7: **end if**
- 8:  $q = \text{Remove-Degrees}(q_F)$  //  $q$  is transformed from  $q_F$  by removing the degrees from  $q_F$

9: **return**  $\text{Cal}(q_F, \text{EvalSQL}(\text{PerfectRef}(q, T')), \text{DB}(\mathcal{A}), a, G)$

**Algorithm A-4:**  $\text{Cal}(q_F, SS, a, G)$

- 1:  $ANS := \emptyset$
- 2: **for** every tuple  $S \in SS$  **do**
- 3:    $ANS := ANS \cup \text{Cal-Soln}(q_F, S, a, G)$  //Calculate the solution  $S : d$  based on the semantic functions  $a$  and  $G$
- 4: **end for**
- 5: **return**  $ANS$

**Theorem 2.** *Let  $\mathcal{O}$  be an  $f$ -DL-Lite ontology,  $q_F$  a general fuzzy conjunctive query and  $S : d$  a pair of a tuple of constants together with a truth degree,  $a$  a semantic function for conjunctions and  $G$  a semantic function for degree-associated atoms.  $S : d$  is a solution of  $q_F$  w.r.t.  $\mathcal{O}$  iff  $(S : d) \in \text{Answer}_F(\mathcal{O}, q_F, a, G)$ .*

See the online TR<sup>(\*)</sup> for detailed explanations of the algorithms A-3 and A-4, as well as the proof of Theorem 2.

## 4 Implementation and Evaluation

The fuzzy DL-Lite knowledge base was implemented by extending the DL-Lite knowledge base system used in ONTOSEARCH2 [7]. In ONTOSEARCH2 we slightly extend SPARQL to make fuzzy queries. In this section, we mainly focus on the evaluations of the performance. Please refer to the online TR<sup>(\*)</sup> for more details of the fuzzy extension of SPARQL.

We evaluated the performance of our system by modifying the Lehigh University Benchmark [5] to include fuzzy concepts, and restricted the semantic complexity of the underlying ontology to that of DL-Lite. This allowed us to create data sets of arbitrary size. Comparative benchmarking was performed with ONTOSEARCH2, this is a non-fuzzy implementation of DL-Lite which allowed us to determine the overhead of a fuzzy query compared to a normal DL-Lite query. We added two fuzzy concepts to the ontology, “Busy” and “Famous”. The first of these was determined by the number of courses taught or taken by a member of staff or student, the second is determined by the number of papers published. The values are calculated using the s-shaped curve functions  $k_f(n)$  to calculate the fuzzy value for fame given  $n$  papers published, and  $k_b(n)$  to calculate the fuzzy value for busyness given  $n$  courses taken:

$$k_f(n) = \frac{2}{1 + \exp(-0.1n)} - 1 \quad k_b(n) = \frac{2}{1 + \exp(-0.4n)} - 1$$

For GFCQs we used a summation operation for  $a$ , a multiplication operation for  $G$  giving a semantics of weights.

To test the system we created dl-lite datasets containing 1, 10 and 50 universities, and processed these to include the fuzzy concepts described above. Two queries were created, the first simple instance retrieval of all “Famous” members of staff. In CTQ form this had a threshold of  $\geq 0.5$  in GFCQ form the query returned all staff members in order of Fame, and in DL-Lite form, this query simply returned all members of staff. The second query found all busy students which were taught by famous members of staff. The CTQ returned all students with a busyness  $\geq 0.5$  who were taught by staff with fame  $\geq 0.5$ , the GFCQ returned a list of students sorted by a weight function based on their business and the fame of any members of staff who taught them, the DL-Lite query simply returned a list of all students who were taught by any member of staff. The results are shown in 2.

**Table 2.** Results of the fuzzy Lehigh University Benchmark queries

Query	$T[1]$ (ms)	$T[10]$ (ms)	$T[50]$ (ms)
CTQ-1	179	536	1061
GFCQ-1	220	683	1887
DL-Lite-1	152	422	891
CTQ-2	532	845	2922
GFCQ-2	520	973	3654
DL-Lite-2	494	892	2523

The performance of the fuzzy reasoner is in all cases close to the performance of the crisp case reasoner for query answering. With small data sets, it has almost identical performance, particularly on more complex queries. As more data must be evaluated, the performance drops slightly.

## 5 Conclusion and Outlook

DL-based fuzzy ontology languages have attracted much attention the last years. That is mainly due to the fact that compared to other fuzzy formalisms, fuzzy ontology languages provide an expressive and yet efficient way to perform reasoning over a fuzzy knowledge. In this paper, we report on evaluations of efficient conjunctive query answering over fuzzy DL-Lite ontologies.

Although there have been quite a few work on fuzzy SQL, such as [4], the closest work to ours is Straccia’s work on f-DL-Lite [12], since DL-Lite itself goes beyond relational databases. Our paper builds on Straccia’s work [12], to further propose two new expressive query languages accompanied with query answering algorithms over f-DL-Lite not proposed in [12]. The first one is a simple generalization of the instance checking (entailment) problem for fuzzy DLs, while the second one consists of a very expressive fuzzy query language

which goes beyond traditional conjunctive queries used in [12]. Actually this query language can work as a framework and by giving different semantics to its parts we can create different fuzzy query languages. Finally, our preliminary evaluations indicate that the performance of the fuzzy query engine is at least in many cases close to the performance of the crisp query engine.

### Acknowledgements

This research was partially supported by (1) the FP6 Network of Excellence EC project Knowledge Web (IST-2004-507842), (2) the Advanced Knowledge Technologies (AKT6) project which is sponsored by the UK Engineering and Physical Sciences Council under grant number GR/N15764/01 and (3) the FP6 EC project X-Media (FP6-26978). Giorgos Stoilos was also partially supported by project PENED 03ED475 2003, which is cofinanced 75% of public expenditure through EC - European Social Fund, 25% of public expenditure through Ministry of Development - General Secretariat of Research and Technology and through private sector, under measure 8.3 of OPERATIONAL PROGRAMME “COMPETITIVENESS” in the 3rd Community Support Programme.

### References

1. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, , and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, 2005.
2. D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical Reasoning for Rich DLs. In *Proc. of the DL2004 Workshop*, 2004.
3. A. Chortaras, Giorgos Stamou, and Andreas Stafylopatis. Adaptation of weighted fuzzy programs. In *Proc. of the International Conference on Artificial Neural Networks ICANN 2006*, pages 45–54. Springer, 2006.
4. J. Galindo, M. armen Aranda, J.L. Caro, A. Guevara, and A. Aguayo. Applying fuzzy databases and fsql to the management of rural accommodation. *Tourism Management*, 23:623–629(7), 2002.
5. Y. Guo, Z. Pan, and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. *Journal of Web Semantics*, 3(2):158–182, 2005.
6. G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, 1995.
7. J. Z. Pan, E. Thomas, and D. Sleeman. ONTOSEARCH2: Searching and Querying Web Ontologies. In *Proc. of WWW/Internet 2006*, pages 211–218, 2006.
8. J.Z. Pan, G. Stoilos, G. Stamou, V. Tzouvaras, and I. Horrocks. f-SWRL: A fuzzy extension of SWRL. *Journal on Data Semantics, special issue on Emergent Semantics*, 4090:28–46, 2006.
9. G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrocks. The fuzzy description logic f-*SHIN*. In *Proc. of the International Workshop on Uncertainty Reasoning for the Semantic Web*, 2005.
10. G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrocks. Fuzzy OWL: Uncertainty and the semantic web. In *Proc. of the International Workshop on OWL: Experiences and Directions*, 2005.
11. U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.
12. U. Straccia. Answering vague queries in fuzzy DL-Lite. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, (IPMU-06)*, pages 2238–2245, 2006.

## A Possibilistic Extension of Description Logics

Guilin Qi<sup>1</sup>, Jeff Z. Pan<sup>2</sup>, and Qiu Ji<sup>1</sup>

<sup>1</sup> Institute AIFB, University of Karlsruhe, Germany  
 {gqi,qiji}@aifb.uni-karlsruhe.de

<sup>2</sup> Department of Computing Science, The University of Aberdeen  
 Aberdeen AB24 3FX  
 jpan@csd.abdn.ac.uk

**Abstract.** Possibilistic logic provides a convenient tool for dealing with inconsistency and handling uncertainty. In this paper, we propose possibilistic description logics (DLs) as an extension of description logics. We give semantics and syntax of possibilistic description logics. Two kinds of inference services are considered in our logics and algorithms are provided for them. These algorithms are implemented using KAON2 reasoner.

### 1 Introduction

Dealing with uncertainty in the Semantic Web has been recognized as an important problem in the recent decades. Two important classes of languages for representing uncertainty are probabilistic logic and possibilistic logic. Arguably, another important class of language for representing uncertainty is fuzzy set theory or fuzzy logic. Many approaches have been proposed to extend description logics with probabilistic reasoning, such as approaches reported in [14, 12, 10]. These approaches can be classified according to ontology languages, the supported forms of probabilistic knowledge and the underlying probabilistic reasoning formalism. The work on fuzzy extension of ontology languages has also received a lot of attention (e.g., [16]). By contrast, there is relatively few work on combining possibilistic logic and description logic.

Possibilistic logic [5] or possibility theory offers a convenient tool for handling uncertain or prioritized formulas and coping with inconsistency. It is very powerful to represent partial or incomplete knowledge [4]. There are two different kinds of possibility theory: one is qualitative and the other is quantitative. Qualitative possibility theory is closely related to default theories and belief revision [7, 3] while quantitative possibility can be related to probability theory and can be viewed as a special case of belief function [8].

The application of possibilistic logic to deal with uncertainty in the Semantic Web is first studied in [13] and is then discussed in [6]. When we obtain an ontology using ontology learning techniques, the axioms of the ontology are often attached with confidence degrees and the learned ontology may be inconsistent [11]. In this case, possibilistic logic provides a flexible framework to interpret the confidence values and to reason with the inconsistent ontology under uncertainty.

However, there exist problems which need further discussion. First, there is no formal definition of the semantics of possibilistic description logics. The semantic extension of possibilistic description logic is not trivial because we need negation of axioms to define the *necessity measure* from a *possibility distribution*. Second, there is no implementation of possibilistic inference in description logics.

In this paper, we discuss possibilistic extension of description logics. Both syntax and semantics of possibilistic description logics are provided in Section 3. The inference services in possibilistic description logics are also given. After that, we provide algorithms for implementing reasoning problems in Section 4. Finally, we conclude this paper in Section 5.

We assume that the reader is familiar with description logics and refer to the description logic handbook [1] for more details.

## 2 Possibilistic Logic

Possibilistic logic [5] is a weighted logic where each classical logic formula is associated with a number in  $(0, 1]$ . Semantically, the most basic and important notion is *possibility distribution*  $\pi: \Omega \rightarrow [0, 1]$ , where  $\Omega$  is the set of all classical interpretations.  $\pi(\omega)$  represents the degree of compatibility of interpretation  $\omega$  with available beliefs. From *possibility distribution*  $\pi$ , two measures can be determined, one is the possibility degree of formula  $\phi$ , defined as  $\Pi(\phi) = \max\{\pi(\omega) : \omega \models \phi\}$ , the other is the necessity or certainty degree of formula  $\phi$ , defined as  $N(\phi) = 1 - \Pi(\neg\phi)$ .

At syntactical level, a *possibilistic formula* is a pair  $(\phi, \alpha)$  consisting of a classical logic formula  $\phi$  and a degree  $\alpha$  expressing certainty or priority<sup>1</sup>. A possibilistic knowledge base is the set of possibilistic formulas of the form  $B = \{(\phi_i, \alpha_i) : i = 1, \dots, n\}$ . The classical base associated with  $B$  is denoted as  $B^*$ , namely  $B^* = \{\phi_i | (\phi_i, \alpha_i) \in B\}$ . A possibilistic knowledge base is consistent iff its classical base is consistent.

Given a possibilistic knowledge base  $B$  and  $\alpha \in (0, 1]$ , the  $\alpha$ -cut (strict  $\alpha$ -cut) of  $B$  is  $B_{\geq \alpha} = \{(\phi, \beta) \in B \text{ and } \beta \geq \alpha\}$  ( $B_{> \alpha} = \{(\phi, \beta) \in B \text{ and } \beta > \alpha\}$ ). The *inconsistency degree* of  $B$ , denoted  $Inc(B)$ , is defined as  $Inc(B) = \max\{\alpha_i : B_{\geq \alpha_i} \text{ is inconsistent}\}$ .

There are two possible definitions of inference in possibilistic logic.

**Definition 1.** Let  $B$  be a possibilistic knowledge base.

- A formula  $\phi$  is said to be a plausible consequence of  $B$ , denoted by  $B \vdash_P \phi$ , iff  $B_{> Inc(B)} \vdash \phi$ .
- A formula  $\phi$  is said to be a possibilistic consequence of  $B$  to degree  $\alpha$ , denoted by  $B \vdash_{\pi}(\phi, \alpha)$ , iff the following conditions hold: (1)  $B_{\geq \alpha}$  is consistent, (2)  $B_{\geq \alpha} \vdash \phi$ , (3)  $\forall \beta > \alpha, B_{\geq \beta} \not\vdash \phi$ .

<sup>1</sup> In possibilistic logic, the weight of a possibilistic formula  $(\phi, a)$  can be also considered as possibility degree of the formula. However, in most applications of possibilistic logic, we often consider the weight as certainty degree.

### 3 Possibilistic Description Logics

In this section, we define the semantics and syntax of possibilistic DLs and inference problems of it. We do not specify the underlying DL language, which can be any (decidable) description logic.

#### 3.1 Syntax

The syntax of possibilistic DL is based on the syntax of classical DL. A *possibilistic axiom* is a pair  $(\phi, \alpha)$  consisting of an axiom  $\phi$  and a weight  $\alpha \in (0, 1]$ . A *possibilistic RBox* (resp., *TBox*, *ABox*) is a finite set of possibilistic axioms  $(\phi, \alpha)$ , where  $\phi$  is an RBox (resp., TBox, ABox) axiom. A possibilistic DL knowledge base  $\mathcal{B} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$  consists of a possibilistic RBox  $\mathcal{R}$ , a possibilistic TBox  $\mathcal{T}$  and a possibilistic ABox  $\mathcal{A}$ . We use  $\mathcal{R}^*$  to denote the classical DL axioms associated with  $\mathcal{R}$ , i.e.,  $\mathcal{R}^* = \{\phi_i : (\phi_i, \alpha_i) \in \mathcal{R}\}$  ( $\mathcal{T}^*$  and  $\mathcal{A}^*$  can be defined similarly). The classical base  $\mathcal{B}^*$  of a possibilistic DL knowledge base is  $\mathcal{B}^* = (\mathcal{R}^*, \mathcal{T}^*, \mathcal{A}^*)$ . A possibilistic DL knowledge base  $\mathcal{B}$  is said to be inconsistent if and only if its classical base  $\mathcal{B}^*$  is inconsistent.

Given a possibilistic DL knowledge base  $\mathcal{B} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$  and  $\alpha \in (0, 1]$ , the  $\alpha$ -cut of  $\mathcal{R}$  is  $\mathcal{R}_{\geq \alpha} = \{\phi \in \mathcal{B}^* \mid (\phi, \beta) \in \mathcal{R} \text{ and } \beta \geq \alpha\}$  (the  $\alpha$ -cut of  $\mathcal{T}$  and  $\mathcal{A}$ , denoted as  $\mathcal{T}_{\geq \alpha}$  and  $\mathcal{A}_{\geq \alpha}$ , can be defined similarly). The strict  $\alpha$ -cut of  $\mathcal{R}$  (resp.,  $\mathcal{T}$ ,  $\mathcal{A}$ ) can be defined similarly as the strict cut in possibilistic logic. The  $\alpha$ -cut (resp., strict  $\alpha$ -cut) of  $\mathcal{B}$  is  $\mathcal{B}_{\geq \alpha} = (\mathcal{R}_{\geq \alpha}, \mathcal{T}_{\geq \alpha}, \mathcal{A}_{\geq \alpha})$  (resp.,  $\mathcal{B}_{> \alpha} = (\mathcal{R}_{> \alpha}, \mathcal{T}_{> \alpha}, \mathcal{A}_{> \alpha})$ ). The *inconsistency degree* of  $\mathcal{B}$ , denoted  $Inc(\mathcal{B})$ , is defined as  $Inc(\mathcal{B}) = \max\{\alpha_i : \mathcal{B}_{\geq \alpha_i} \text{ is inconsistent}\}$ .

We use the following example as a running example throughout this paper.

*Example 1.* Suppose we have a possibilistic DL knowledge base  $\mathcal{B} = (\mathcal{R}, \mathcal{T}, \mathcal{A})$ , where  $\mathcal{R} = \emptyset$ ,  $\mathcal{T} = \{(Bird \sqsubseteq Fly, 0.8), (HasWing \sqsubseteq Bird, 0.95)\}$  and  $\mathcal{A} = \{(Bird(chirpy), 1), (HasWing(tweety), 1), (\neg Fly(tweety), 1)\}$ . The TBox  $\mathcal{T}$  states that it is rather certain that birds can fly and it is almost certain that something with wing is a bird. The ABox  $\mathcal{A}$  states that it is certain that tweety has wing and it cannot fly, and chirpy is a bird. Let  $\alpha = 0.8$ . We then have  $\mathcal{B}_{\geq 0.8} = (\mathcal{R}_{\geq 0.8}, \mathcal{T}_{\geq 0.8}, \mathcal{A}_{\geq 0.8})$ , where  $\mathcal{R}_{\geq 0.8} = \emptyset$ ,  $\mathcal{T}_{\geq 0.8} = \{Bird \sqsubseteq Fly, HasWing \sqsubseteq Bird\}$  and  $\mathcal{A}_{\geq 0.8} = \{HasWing(tweety), \neg Fly(tweety), Bird(chirpy)\}$ . It is clear that  $\mathcal{B}_{\geq \alpha}$  is inconsistent. Now let  $\alpha = 0.95$ . Then  $\mathcal{B}_{\geq \alpha} = (\mathcal{R}_{\geq 0.95}, \mathcal{T}_{\geq 0.95}, \mathcal{A}_{\geq 0.95})$ , where  $\mathcal{R}_{\geq 0.95} = \emptyset$ ,  $\mathcal{T}_{\geq 0.95} = \{HasWing \sqsubseteq Bird\}$  and  $\mathcal{A}_{\geq 0.95} = \{HasWing(tweety), \neg Fly(tweety), Bird(chirpy)\}$ . So  $\mathcal{B}_{\geq \alpha}$  is consistent. Therefore,  $Inc(\mathcal{B}) = 0.8$ .

#### 3.2 Semantics

The semantics of possibilistic DL is defined by a *possibility distribution*  $\pi$  over the set  $\mathbf{I}$  of all classical description logic interpretations, i.e.,  $\pi : \mathbf{I} \rightarrow [0, 1]$ .  $\pi(I)$  represents the degree of compatibility of interpretation  $I$  with available information. For two interpretations  $I_1$  and  $I_2$ ,  $\pi(I_1) > \pi(I_2)$  means that  $I_1$  is preferred to  $I_2$  according to the available information. Given a possibility

distribution  $\pi$ , we can define the possibility measure  $\Pi$  and necessity measure  $N$  as follows:  $\Pi(\phi) = \max\{\pi(I) : I \in \mathbf{I}, I \models \phi\}$  and  $N(\phi) = 1 - \Pi(\neg\phi)$ , where  $\neg\phi$  is the consistency negation defined in [9]<sup>2</sup>. Given two possibility distributions  $\pi$  and  $\pi'$ , we say that  $\pi$  is more specific (or more informative) than  $\pi'$  iff  $\pi(I) \leq \pi'(I)$  for all  $I \in \Omega$ . A possibility distribution  $\pi$  satisfies a possibilistic axiom  $(\phi, \alpha)$ , denoted  $\pi \models (\phi, \alpha)$ , iff  $N(\phi) \geq \alpha$ . It satisfies a possibilistic DL knowledge base  $\mathcal{B}$ , denoted  $\pi \models \mathcal{B}$ , iff it satisfies all the possibilistic axioms in  $\mathcal{B}$ .

Given a possibilistic DL knowledge base  $\mathcal{B} = \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ , we can define a possibility distribution from it as follows: for all  $I \in \mathbf{I}$ ,

$$\pi_{\mathcal{B}}(I) = \begin{cases} 1 & \text{if } \forall \phi_i \in \mathcal{R}^* \cup \mathcal{T}^* \cup \mathcal{A}^*, I \models \phi_i, \\ 1 - \max\{\alpha_i \mid I \not\models \phi_i, (\phi_i, \alpha_i) \in \mathcal{R} \cup \mathcal{T} \cup \mathcal{A}\} & \text{otherwise.} \end{cases} \quad (1)$$

As in possibilistic logic, we can also show that the possibility distribution defined by Equation 1 is most specific possibility distribution satisfying  $\mathcal{B}$ . Let us consider Example 1 again.  $I = \langle \Delta^I, \circ^I \rangle$  is an interpretation, where  $\Delta^I = \{tweety, chirpy\}$  and  $Bird^I = \{tweety, chirpy\}$ ,  $Fly^I = \{chirpy\}$ , and  $HasWing^I = \{tweety\}$ . It is clear that  $I$  satisfies all the axioms except  $Bird \sqsubseteq Fly$  (whose weight is 0.8), so  $\pi_{\mathcal{B}}(I) = 0.2$ .

We have the following theorem which says that consistency of a possibilistic DL knowledge bases can be equivalently defined by the possibility distribution associated with it.

**Theorem 1.** *Let  $\mathcal{B}$  be a possibilistic DL knowledge base and  $\pi_{\mathcal{B}}$  be the possibility distribution obtained by Equation 1. Then  $\mathcal{B}$  is consistent if and only if  $\pi_{\mathcal{B}} \models \mathcal{B}$ , where  $\pi_{\mathcal{B}}$  is the possibility distribution defined by Equation 1.*

The proof of is clear by considering Condition (i) of the consistency negation.

Similar to possibilistic logic, we have the following result.

**Proposition 1.** *Let  $\mathcal{B}$  be a possibilistic DL knowledge base and  $\pi_{\mathcal{B}}$  be the possibility distribution obtained by Equation 1. Then  $Inc(\mathcal{B}) = 1 - \max_{I \in \mathbf{I}} \pi_{\mathcal{B}}(I)$ .*

Proposition 1 shows that the inconsistency degree of a possibilistic DL knowledge base can be equivalently defined by the possibility distribution.

### 3.3 Inference in possibilistic DLs

We consider the following inference services in possibilistic DLs.

- Instance checking: an individual  $a$  is a *plausible* instance of a concept  $C$  with respect to a possibilistic DL knowledge base  $\mathcal{B}$ , written  $\mathcal{B} \models_P C(a)$ , if  $\mathcal{B}_{>Inc(\mathcal{B})} \models C(a)$ .

<sup>2</sup> There are two kinds of negations defined in [9]: consistency negation and coherence negation. An axioms  $\psi$  is said to be a consistency-negation of an axiom  $\phi$ , written  $\neg\phi$ , iff it satisfies the following two conditions: (i)  $\{\phi, \psi\}$  is inconsistent and (ii) there exists no other  $\psi'$  such that  $\psi'$  satisfies condition (i) and  $Cn(\{\psi'\}) \subset Cn(\{\psi\})$ .

- Subsumption: a concept  $C$  is *plausible* subsumed by a concept  $D$  with respect to a possibilistic DL knowledge base  $\mathcal{B}$ , written  $\mathcal{B} \models_P C \sqsubseteq D$ , if  $\mathcal{B}_{>Inc(\mathcal{B})} \models C \sqsubseteq D$ .
- Instance checking with necessity degree: an individual  $a$  is an instance of a concept  $C$  to degree  $\alpha$  with respect to  $\mathcal{B}$ , written  $\mathcal{B} \models_\pi (C(a), \alpha)$ , if the following conditions hold: (1)  $\mathcal{B}_{\geq \alpha}$  is consistent, (2)  $\mathcal{B}_{\geq \alpha} \models C(a)$ , (3) for all  $\beta > \alpha$ ,  $\mathcal{B}_{\geq \beta} \not\models C(a)$ .
- Subsumption with necessity degree: a concept  $C$  is subsumed by a concept  $D$  to a degree  $\alpha$  with respect to a possibilistic DL knowledge base  $\mathcal{B}$ , written  $\mathcal{B} \models_\pi (C \sqsubseteq D, \alpha)$ , if the following conditions hold: (1)  $\mathcal{B}_{\geq \alpha} \models C \sqsubseteq D$ , (2)  $\mathcal{B}_{\geq \alpha} \models C \sqsubseteq D$ , (3) for all  $\beta > \alpha$ ,  $\mathcal{B}_{\geq \beta} \not\models C \sqsubseteq D$ .

We illustrate the inference services by reconsidering Example 1.

*Example 2.* (Example 1 continued) According to Example 1, we have  $Inc(\mathcal{B}) = 0.8$  and  $\mathcal{B}_{>0.8} = (\mathcal{R}_{>0.8}, \mathcal{T}_{>0.8}, \mathcal{A}_{>0.8})$ , where  $\mathcal{R}_{>0.8} = \emptyset$ ,  $\mathcal{T}_{>0.8} = \{HasWing \sqsubseteq Bird\}$  and  $\mathcal{A}_{>0.8} = \{HasWing(tweety), \neg Fly(tweety), Bird(chirpy)\}$ . Since  $\mathcal{B}_{>0.8} \models Bird(tweety)$ , we can infer that *tweety* is plausible to be a bird from  $\mathcal{B}$ . Furthermore, since  $\mathcal{B}_{\geq 0.95} \models Bird(tweety)$  and  $\mathcal{B}_{\geq 1} \not\models Bird(tweety)$ , we have  $\mathcal{B} \models_\pi (Bird(tweety), 0.95)$ . That is, we are almost certain that *tweety* is a bird.

#### 4 Algorithms for Inference in Possibilistic DLs

In this section, we give algorithms for implementing possibilistic inference in possibilistic DLs and analyze the computational complexity of the algorithms.

Algorithm 1 computes the inconsistency degree of a possibilistic DL knowledge base using a binary search. The function *Asc* takes a finite set of numbers in  $(0, 1]$  as input and returns a vector which contains those distinct numbers in the set in an ascending order. For example,  $Asc(0.2, 0.3, 0.3, 0.1) = (0.1, 0.2, 0.3)$ . Let  $W = (\beta_1, \dots, \beta_n)$  is a vector consisting of  $n$  distinct numbers, then  $W(i)$  denotes  $\beta_i$ . If the returned inconsistency degree is 0, that is  $W(-1) = 0$ , it shows the ontology to be queried is consistent.

Since Algorithm 1 is based on binary search, to compute the inconsistency degree, it is easy to check that the algorithm requires  $\lceil \log_2 n \rceil + 1$  satisfiability checks using a DL reasoner in the worst case.

Algorithm 2 returns the necessity degree of an axiom inferred from a possibilistic DL knowledge base *w.r.t* the possibilistic inference. We compute the inconsistency degree of the input ontology. If the axiom is a plausible consequence of a possibilistic DL knowledge base, then we compute its necessity degree using a binary search (see the first “if” condition). Otherwise, its necessity degree is 0, i.e., the default value given to  $w$ . Note that our algorithm is different from the algorithm given in [15] for computing the necessity of a formula in possibilistic logic (this algorithm needs to compute the negation of a formula, which is computationally hard in DLs according to [9]). We consider only subsumption checking here. However, the algorithm can be easily extended to reduce instance checking as well.

---

**Algorithm 1:** Compute the inconsistency degree

---

**Data:**  $\mathcal{B} = \langle \mathcal{T}, \mathcal{A} \rangle$ , where  $\mathcal{T} \cup \mathcal{A} = \{(\phi_i, \alpha_i) : \alpha_i \in (0, 1], i = 1, \dots, n\}$ , where  $n$  is the number of axioms in the testing ontology  $\mathcal{B}$ ;

**Result:** The inconsistency degree  $d$

```

begin
   $b := 0$  //  $b$  is the begin pointer of the binary search
   $m := 0$  //  $m$  is the middle pointer of the binary search
   $d := 0.0$  // The initial value of inconsistency degree  $d$  is set to be 0.0
   $W = Asc(\alpha_1, \dots, \alpha_n)$ 
   $W(-1) = 0.0$  // The special element  $-1$  of  $W$  is set to be 0.0
   $e := |W| - 1$  //  $e$  is the end pointer of the binary search
  if  $\mathcal{B}_{\geq W(0)}$  is consistent then
     $d := 0.0$ 
  else
    while  $b \leq e$  do
      if  $b = e$  then
         $d := b$ 
       $m := \lceil (b + e) / 2 \rceil$ 
      if  $\mathcal{B}_{\geq W(m)}$  is consistent then
         $e := m - 1$ 
      else
         $b := m + 1$ 
     $d := W(b)$ 
  end

```

---

**Proposition 2.** Let  $\mathcal{B}$  be a possibilistic DL knowledge base and  $\phi$  be a DL axiom. Deciding whether  $\mathcal{B} \models_P \phi$  requires  $\lceil \log_2 n \rceil + 1$  satisfiability check using a DL reasoner, where  $n$  is the number of distinct certainty degrees in  $\mathcal{B}$ . Furthermore, deciding whether  $\mathcal{B} \models_\pi (\phi, \alpha)$  requires at most  $\lceil \log_2 n \rceil + \lceil \log_2 n - l \rceil + 1$  satisfiability check using a DL reasoner, where  $n$  is the number of distinct certainty degrees in  $\mathcal{B}$  and  $l$  is the inconsistency degree of  $\mathcal{B}$ .

## 5 Conclusions and Future Work

We gave a possibilistic extension of description logics in this paper. Two kinds of inference services were considered: one is a plausible consequence relation and the other is a possibilistic consequence relation. Algorithms were given to check the inference services and we implemented the algorithms in Java using KAON2<sup>3</sup> as the basic reference service. The source codes and some ontologies used for testing can be downloaded from

<http://radon.ontoware.org/incoquery.zip>

To represent the weight for each axiom, we use an annotation property “Rating” to associate one value with the class defined. Thus the axioms starting

<sup>3</sup> <http://kaon2.semanticweb.org/>

---

**Algorithm 2:** Possibilistic inference with certainty degrees

---

**Data:**  $\mathcal{B} = \langle \mathcal{T}, \mathcal{A} \rangle$ , where  $\mathcal{T} \cup \mathcal{A} = \{(\phi_i, \alpha_i) : \alpha_i \in (0, 1], i = 1, \dots, n\}$ ; a DL axiom  $\phi$ .

**Result:** The certainty degree  $w$  associated with a query  $\phi$

```

begin
   $m := 0$ 
   $w := 0.0$  // The initial certainty degree of  $\phi$  is set to be 0.0
   $W = Asc(\alpha_1, \dots, \alpha_n)$ 
   $W(-1) = 0.0$ 
   $e := |W| - 1$ 
  compute  $l$  such that  $W(l) = Inc(\mathcal{B})$  //  $Inc(\mathcal{B})$  is computed by Algorithm 1
   $b := l + 1$ 
  if  $\mathcal{B}_{\geq W(b)} \models \phi$  then
    while  $b \leq e$  do
      if  $b = e$  then
         $\perp$  return  $b$ 
       $m := \lceil (b + e) / 2 \rceil$ 
      if  $\mathcal{B}_{\geq W(m)} \not\models \phi$  then
         $\perp$   $e := m - 1$ 
      else
         $\perp$   $b := m + 1$ 
     $w := W(b)$ 
  end

```

---

with this class also have the same value as their weights. Take the following code as an example. The axioms  $Messaging \sqsubseteq \neg Kerberos$  and  $Messaging \sqsubseteq \neg GeneralReliabilityUsernamePolicy$  will have the same weight 0.345.

```

<owl:Class rdf:about="#Messaging">
  <Rating rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.345
</Rating>
  <owl:disjointWith rdf:resource="#Kerberos"/>
  <owl:disjointWith rdf:resource="#GeneralReliabilityUsernamePolicy"/>
</owl:Class>

```

The advantage of this way to represent confidence values is that confidence values and the ontology can be kept in the same owl file. So far, we only support some simple queries like instance checking  $A(a)$  and subsumption  $A \sqsubseteq B$ , where  $a$  is an instance and  $A, B$  are concepts.

Possibilistic inference has been criticized for the “drowning problem”, i.e., all the axioms whose necessity degrees which are less than or equal to the inconsistency degree of the possibilistic DL knowledge base do not contribute to the inference. Several variants of possibilistic inference have been proposed in classical logic to solve the drowning problem [2]. We plan to implement these approaches in our future work.

*Acknowledgements* Research presented in this paper was partially supported by the European Commission NeOn (IST-2006-027595, <http://www.neon-project.org/>), the Knowledge Web projects (IST-2004-507842, <http://knowledgeweb.semanticweb.org/>), and the X-Media project ([www.x-media-project.org](http://www.x-media-project.org)) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

## References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, implementation and application*. Cambridge University Press, 2003.
2. Salem Benferhat, Claudette Cayrol, Didier Dubois, Jérôme Lang, and Henri Prade. Inconsistency management and prioritized syntax-based entailment. In *Proc. of IJCAI'93*, pages 640–647. Morgan Kaufmann, 1993.
3. Salem Benferhat, Didier Dubois, and Henri Prade. Representing default rules in possibilistic logic. In *Proc. of KR'92*, pages 673–684, 1992.
4. Salem Benferhat, Sylvain Lagrue, and Odile Papini. Reasoning with partially ordered information in a possibilistic logic framework. *Fuzzy Sets and Systems*, 144(1):25–41, 2004.
5. Didier Dubois, Jérôme Lang, and Henri Prade. Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 439–513. Oxford University Press, 1994.
6. Didier Dubois, Jérôme Mengin, and Henri Prade. Possibilistic uncertainty and fuzzy features in description logic: A preliminary discussion. In *Capturing Intelligence: Fuzzy Logic and the Semantic WEb*, pages 101–113. Elsevier, 2006.
7. Didier Dubois and Henri Prade. Epistemic entrenchment and possibilistic logic. *Artif. Intell.*, 50(2):223–239, 1991.
8. Didier Dubois and Henri Prade. Possibility theory: qualitative and quantitative aspects. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 169–226, 1998.
9. Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *Proc. of AAAI'06*, 2006.
10. Rosalba Giugno and Thomas Lukasiewicz. P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In *Proc. of JELIA'02*, pages 86–97, 2002.
11. Peter Haase and Johanna Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In *Proc. of URSW'05*, pages 45–55, 2005.
12. Jochen Heinsohn. Probabilistic description logics. In *Proc. of UAI'94*, pages 311–318, 1994.
13. Bernhard Hollunder. An alternative proof method for possibilistic logic and its application to terminological logics. *Int. J. Approx. Reasoning*, 12(2):85–109, 1995.
14. Manfred Jaeger. Probabilistic reasoning in terminological logics. In *Proc. of KR'94*, pages 305–316, 1994.
15. Jérôme Lang. Possibilistic logic: complexity and algorithms. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 179–220. 2000.
16. Umberto Straccia. Reasoning within fuzzy description logics. *J. Artif. Intell. Res.*, 14:137–166, 2001.

## DL-based alternating-offers protocol for automated multi-issue bilateral negotiation

Azzurra Ragone<sup>1</sup>, Tommaso Di Noia<sup>1</sup>, Eugenio Di Sciascio<sup>1</sup>, Francesco M. Donini<sup>2</sup>

<sup>1</sup> SisInfLab, Politecnico di Bari, Bari, Italy  
{a.ragone,t.dinoia,disciascio}@poliba.it

<sup>2</sup> Università della Tuscia, Viterbo, Italy  
donini@unitus.it

**Abstract.** We present a novel approach to logic-based bilateral negotiation in e-commerce systems. We use Description Logics to describe both requests/offers submitted by buyers and sellers, and relations among issues as axioms in a TBox. Moreover, exploiting concept contraction in DLs, we are able to handle *conflicting* information both in goods and services descriptions. We ground the approach in a P2P e-marketplace framework, and introduce a *logic-based* alternating-offers protocol. In such a protocol we exploit both knowledge representation tools and utility theory to find the most suitable agreements.

### 1 Introduction

We study automated bilateral negotiation in peer-to-peer (P2P) e-marketplaces, where buyers and sellers may want to submit articulate advertisements to find best available counterparts, and price is obviously not the single issue to negotiate on. In such frameworks, using a logic formalism it is possible to recognize that an advertisement for a *Notebook equipped with a Linux operating system* actually fulfills a buyer's request for a *PC having a Unix operating system*. Or, conversely, that a buyer's request for a *Notebook with Wi-Fi adapter* is in conflicting with a seller's supply for a *Notebook with Wired Adapter*. To manage automated bilateral negotiation in such a framework we introduce a novel *logic-based* alternating-offers protocol. The protocol merges both Description Logics formalism and reasoning services, and utility theory, to find the most suitable agreements. To this aim it takes into account existing logical relations between issues in requests and offers and related utilities of agents, expressed through logical formulas.

The roadmap to the remainder of this paper is as follows: next section presents an outline of the whole approach. Then we move on the DL we adopt and related inference services. We show the modeling of advertisements and then the protocol is presented and discussed. Related work and discussion close the paper.

### 2 Negotiation Scenario

In order to outline the negotiation mechanism we define: the *negotiation protocol*, the *negotiation strategy*[6], the *utility function* of the agents [7]. The assumptions characterizing our negotiation mechanism are:

**one-to-many:** the negotiation is a one-to-many negotiation, since the buyer’s agent will negotiate simultaneously with others  $m$  different agents – each of them representing a seller, whose offer has been previously stored in the system.

**rationality:** agents are *rational*s, they behave according to their preferences and try to maximize their utilities [7, p.19] doing in each step the minimum possible concession, *i.e.*, the concession involving the minimum utility loss, see protocol Section 5.

**incomplete information:** each agent knows its utility function and ignores the opponent disagreement thresholds and utility functions.

**conflict deal:** disagreement is better than an agreement iff the agent’s utility over such an agreement is smaller than disagreement thresholds<sup>3</sup> set by the agent before negotiation starts. Therefore when the agent’s utility deriving from accepting an agreement (or going on with the negotiation) and opting out it is the same, it will prefer not to opt out [7]. The protocol we propose is inspired by Rubinstein’s alternating-offers one [9]. In that setting an agent starts making an offer to its opponent, who can either accept, make a counter-offer or exit the negotiation. If a counter-offer is made, the negotiation goes on until one of the agent accepts an offer or exits the negotiation. In some cases there is a negotiation *deadline*; if the deadline is reached before one agent has accepted an offer, the negotiation ends in a conflict deal. Our protocol anyway is quite different from that of Rubinstein; actually we consider *multi-issue negotiation*: buyer and seller do not negotiate on a single item or on a single bundle of items, but on many issues, which are related with each other through an ontology; such issues may also characterize a more complex item (*e.g.*, in the computer domain a notebook equipped with Wi-Fi adapter and DVD recorder). Differently from many alternating-offers protocols we do not consider a time deadline.

The protocol is sorted out by a finite set of steps<sup>4</sup>: the negotiation always terminates because either the agreement has been reached or because one agent opts out. The agent who moves first is selected randomly for each negotiation. At each step the agent who moves has two choices: *concede* or *opt out*, while the other one *stands still*. Agents are forced to concede until a *logical compatibility* is reached between the initial request and the initial supply, *i.e.*, until the inconsistency sources are eliminated in both the demand and the supply. At each step, amongst all the allowed concessions that satisfy the concession criteria enforced by the protocol, the agent should choose the concession that gives the highest utility to himself (and then the concession less decreasing its utility): the *minimal concession*. Therefore a concession should be *minimal* w.r.t. the utility loss paid by the agent who makes the concession [4]. The negotiation ends either if a logical compatibility is reached (*the negotiation succeeds*) or if one agent opts out (*the negotiation ends in a conflict deal*). For what concerns **strategy**, the main target of the agent is to reach the compatibility, because only through compatibility it is possible to reach an agreement. If it is its turn to move, an agent can choose to concede or opt

<sup>3</sup> “*disagreement thresholds*, also called disagreement payoffs, or reservation values, [...] are the minimum utility that each agent requires to pursue a deal”[8].

<sup>4</sup> In the following, for the sake of clarity, we always describe an interaction between only two opposite agents; although notice that multiple negotiations can be performed at the same time, among *one* agent and *many* candidate partners.

out: if the utility of the agent at that step is smaller than its disagreement threshold, then the agent opts out and the negotiation ends immediately. Otherwise, it will do a concession. We define an agent’s utility function over all possible outcomes [7]:

$$u^p : \{A \cup \{Opt\}\} \rightarrow \mathfrak{R}$$

where  $p \in \{\beta, \sigma\}$ — $\beta$  and  $\sigma$  stand for buyer and seller respectively—  $A$  is the set of all possible agreements,  $Opt$  stands for Opt out.

### 3 Description Logics for negotiation

Here we refer to  $\mathcal{AL}(D)$ . Besides concepts and roles,  $\mathcal{AL}(D)$  allows one to express quantitative properties on objects such as year of building, length, weight and many others by means of *concrete domains*. For the scope of the framework we propose in this paper, it is sufficient to introduce only unary predicates  $=_x(\cdot)$  and  $>_x(\cdot)$  where  $x \in D$ <sup>5</sup>. Without loss of generality we assume that concrete domains we deal with are admissible [1]. In order to model the domain knowledge and represents relationships among elements, an ontology  $\mathcal{O}$  is used in the form:

$$CN_1 \sqsubseteq CN_2 \qquad CN_1 \sqsubseteq \neg CN_2$$

Formulas representing demands  $D$  and supplies  $S$ , are expressed as generic formulas  $\exists R \sqcap \forall R.C$ , so an example advertisement can be formalized as in the following formula:

$$PC \sqcap \neg \text{Notebook} \sqcap (\text{ram} \geq 1024) \sqcap (\text{hdd} \leq 160) \sqcap \exists \text{hasOS} \sqcap \forall \text{hasOS.Linux} \sqcap \exists \text{monitor} \sqcap \forall \text{monitor} . (\text{LCDmonitor} \sqcap (\text{inch} \geq 17))$$

Notice that for what concerns numerical properties, also range expressions are allowed in the form  $(f \geq n) \sqcap (f \leq n)$ . Even though subsumption and satisfiability are very useful reasoning tasks for matchmaking in e-commerce scenarios [3], there are typical problems related to negotiation that need non-standard reasoning services. For instance, suppose you have the buyer’s agent  $\beta$  with her demand represented by the concept  $D$  and the seller’s agent  $\sigma$  with his supply represented by  $S$ . In case  $S \sqcap D \sqsubseteq_{\mathcal{O}} \perp$  holds, how to suggest to  $\beta$  what in  $D$  is in conflict with  $S$  and conversely to  $\sigma$  what in  $S$  is conflict with  $D$ ? The above question is very common, among others, in negotiation scenarios where you need to know “what is wrong” between  $D$  and  $S$  and negotiate on it. In order to give an answer to the previous question and provide explanations, concept contraction[3] can be exploited.

**Concept Contraction** . Given two concepts  $C_1$  and  $C_2$  and an ontology  $\mathcal{O}$ , where both  $C_1 \sqcap C_2 \sqsubseteq_{\mathcal{O}} \perp$  holds, find two concepts  $K$  (for Keep) and  $G$  (for Give up) such that both  $C_1 \equiv K \sqcap G$  and  $K \sqcap C_2 \not\sqsubseteq_{\mathcal{O}} \perp$ .

In other words  $K$  represents a contraction of  $C_1$  which is satisfiable with  $C_2$ , whilst  $G$  represents the reason why  $C_1$  and  $C_2$  are not compatible with each other. With concept contraction, conflicting information both in  $\beta$ ’s request w.r.t.  $\sigma$ ’supply can be computed

<sup>5</sup> Hereafter, for the sake of clarity we will use an infix notation instead of a prefix one to deal with predicates over concrete domains e.g.,  $(f \leq n) = \neg >_n(f)$ .

and vice versa. Actually, for concept contraction minimality criteria have to be introduced. Following the Principle of Informational Economy [5], for  $G$  we have to give up as little information as possible. In [2, 3] some minimality criteria were introduced and analyzed. In particular, if the adopted DL admits a normal form with conjunctions of concepts as  $\mathcal{AL}(D)$ ,  $G_{\exists}$  minimal irreducible solutions can be defined.

Let  $C_1$  and  $C_2$  be two concepts such that  $C_1 \sqcap C_2 \sqsubseteq_{\mathcal{O}} \perp$ . For the corresponding Concept Contraction problem  $\mathcal{Q}$ , we say the solution  $\langle G_{irr}, K_{irr} \rangle$  problem is  $G$ -irreducible if the following conditions hold:

1.  $G_{irr} = \prod_{i=1..n} G_i$  where  $G_i$  is in the form  $\exists R$  iff  $C_2 \sqsubseteq \forall R.\perp$ ; [ $G_{\exists}$  minimal condition]
2.  $K \sqcap G_{\bar{i}} \sqcap C_2 \sqsubseteq_{\mathcal{O}} \perp$ , for any  $G_{\bar{i}}$ ,  $\bar{i} = 1 \dots n$ ;
3. if  $\langle \bar{G}, \bar{K} \rangle$  is another solution to  $\mathcal{Q}$  satisfying condition 1, then  $\bar{G} \sqsubseteq G_{irr}$ .

#### 4 Dealing with Incomplete Information

Information about supply/demand descriptions can be, in our setting, incomplete. This may happen not only because some information may be unavailable, but also because some details have been considered irrelevant by either the seller or the buyer when they submitted their advertisements. Some user may find tiresome to specify a lot of characteristics related *e.g.*, to the brand or more technical characteristics of the product the user can be unaware of. The most common approach to this problem is avoiding incompleteness by forcing the user to fill long and tedious forms. There are several ways to deal with incomplete information and the choice may influence a negotiation.

Under an *open-world assumption* we have two possible choices. First, we can keep incomplete information as *missing* information: we do not know *e.g.*, if the buyer is not interested in a particular characteristic or he simply has forgotten to specify it. In this case the system has to contact to buyer/seller to further refine her/his description. Asking the users to refine their descriptions before the negotiation process starts it seems quite unrealistic, because of the amount of descriptions that can be stored in the system itself. It appears more feasible to leave this phase after the negotiation process has been performed with the counterparts in the e-marketplace, and only a small amount of supplies/demands have been retrieved. For instance, the ones with the highest utility product [9].

Once buyer and seller have refined their descriptions it is possible to start a new negotiation (the so-called *post-negotiation* phase) where only the *updated* information is negotiated.

On the other way, still in the open-world assumption setting, a second possible choice can be to assume incomplete information as an *any-would-fit* assertion (don't care), so the system should cope with this incompleteness as is. Therefore also this information will be presented in the final agreement.

#### 5 A logic-based alternating-offers protocol

In this Section we show how to use DLs and a non-standard reasoning service, namely Concept Contraction, to model an alternating-offer protocol taking into account the

semantics of request and offers as well as the domain knowledge modeled within an ontology.

For the sake of clarity and without loss of generality, from now on we consider that the agent entering the marketplace is the buyer  $\beta$  and her potential partners are the sellers' agents  $\sigma$ .

First of all, the buyer's demand  $D$  is normalized considering the equivalence  $\forall R.(C \sqcap D) \equiv \forall R.C \sqcap \forall R.D$  as a rewrite rule from left to right.

After the normalization  $D$  is then a conjunction of elements in the form

$$D = \prod_k \exists R_k \sqcap \prod_i C_i \quad (1)$$

where  $C_i \in \{CN, \neg CN, p(f), \neg p(f), \forall R.C\}$ . As an example consider the concept in Section 3. After the normalization it is then rewritten as  $PC \sqcap \neg \text{Notebook} \sqcap (\text{ram} \geq 1024) \sqcap (\text{hdd} \leq 160) \sqcap \exists \text{hasOS} \sqcap \forall \text{hasOS.linux} \sqcap \exists \text{monitor} \sqcap \forall \text{monitor.LCDmonitor} \sqcap \forall \text{monitor.}(\text{inch} \geq 17)$

In the normalized form  $\exists R_k$  and  $C_i$  represent issues on which the user is willing to negotiate on. The buyer is able to express her utilities on single issues or on bundles of them. For instance, w.r.t. the previous request the buyer may set utility values on a single issue  $PC$  as well as on the whole formula  $(\text{ram} \geq 1024) \sqcap \forall \text{monitor.LCDmonitor}$  (bundle of issue). We indicate these concepts with  $P_k$  — for **P**references.

Now, for each  $P_k$  the buyer  $\beta$  expresses a utility value  $u^\beta(P_k)$  such that  $\sum_i u^\beta(P_k) = 1$ . As usual, both agents' utilities are normalized to 1 to eliminate outliers, and make them comparable. Since we assumed that utilities are additive, the global utility is just a sum of the utilities related to preferences.

$$u^\beta = \sum_k u^\beta(P_k)$$

After single item's utilities have been elicited,  $\beta$  set the **disagreement threshold**  $t_\beta$  (see Section 2). The same for the seller.

### 5.1 The Protocol

Summing up, before the real negotiation starts (step 0) we have a demand  $D$  and a supply  $S$  such that

$$D = \prod_k \exists R_k \sqcap \prod_i C_i \quad S = \prod_l \exists R_l \sqcap \prod_j C_j$$

Based on  $C_i$  and  $C_j$ , the buyer and seller respectively, formulate their preferences  $P_k$  (for the buyer) and  $P_h$  (for the seller) and for each of them set a utility value such that:

$$\sum_k u^\beta(P_k) = 1 \quad \sum_h u^\sigma(P_h) = 1$$

Finally, both for  $\beta$  and  $\sigma$  we have the corresponding **disagreement thresholds** and utility functions  $t_\beta, u^\beta$  and  $t_\sigma, u^\sigma$ .

If  $D \sqcap S \sqsubseteq_{\mathcal{O}} \perp$  then the demand and the supply are in conflict with each other and  $\beta$  and  $\sigma$  need to negotiate on conflicting information if they want to reach an agreement. The negotiation will follow the alternating offers protocol as described in Section 2. At each step, either  $\beta$  or  $\sigma$  gives up a portion of its conflicting information choosing the item with the minimum utility. At the beginning, both  $\beta$  and  $\sigma$  need to know what are the conflicting information. Notice that both agents  $\beta$  and  $\sigma$  know  $D$  and  $S$ , but they have no information neither on counterpart utilities nor preferences. Both  $\beta$  and  $\sigma$  solve two Concept Contraction problems, computing a  $G_{\exists}$  minimal irreducible solution, and rewrite  $D$  and  $S$  as:

$$D = G_0^\beta \sqcap K_0^\beta \qquad S = G_0^\sigma \sqcap K_0^\sigma$$

In the above rewriting  $G_0^\beta$  and  $G_0^\sigma$  represent respectively the reason why  $D$  is in conflict with  $S$  and the reason why  $S$  is in conflict with  $D$ . At a first glance it would seem  $\beta$  needs only  $\langle G_0^\beta, K_0^\beta \rangle$  and  $\sigma$  needs  $\langle G_0^\sigma, K_0^\sigma \rangle$ . We will see later that  $\beta$  needs also information on  $\sigma$  in order to check its fairness during negotiation steps.

Since we compute  $G$ -irreducible solutions we can normalize  $G_0^\beta$  and  $G_0^\sigma$ , following the same procedure for  $D$  and  $S$ , as:

$$G_0^\beta = G_{(0,1)}^\beta \dots \sqcap G_{(0,n)}^\beta = \prod_{i=1}^n G_{(0,i)}^\beta \qquad G_0^\sigma = G_{(0,1)}^\sigma \dots \sqcap G_{(0,m)}^\sigma = \prod_{j=1}^m G_{(0,j)}^\sigma$$

In the previous formulas, indexes  $(0, i)$  and  $(0, j)$  represent the  $i$ -th and  $j$ -th conjunctive element in  $G^\beta$  and  $G^\sigma$  at round 0.

Due to the logic adopted for  $D$ ,  $S$  and  $\mathcal{O}$  we have that: **for each  $G_{(0,i)}^\beta$  there always exists a  $C_i$  in the normalized version of  $D$ —as represented in equation (1)—such that  $G_{(0,i)}^\beta = C_i$ .** The same relation holds between each  $G_{(0,j)}^\sigma$  and  $C_j$  in the normalized form of  $S$ . Hence, some of  $P_k$  and  $P_h$  can be partially rewritten in terms of  $G_{(0,i)}^\beta$  and  $G_{(0,j)}^\sigma$  respectively. Since the information in  $G_0^\beta$  and  $G_0^\sigma$  are the reason why an agreement is not possible, then either  $\beta$  or  $\sigma$  will start conceding one of  $G_{(0,i)}^\beta$  or  $G_{(0,j)}^\sigma$  reducing their global utility of  $u(G_{(0,i)}^\beta)$  or  $u(G_{(0,j)}^\sigma)$  respectively.

Suppose  $\beta$  starts the negotiation and gives up  $G_{(0,2)}^\beta = C_5$  with  $P_3 \sqsubseteq_{\mathcal{O}} G_{(0,2)}^\beta$ . Then it reformulates its request as  $D_1 = \prod_k \exists R_k \sqcap \prod_{i=1..4,6..} C_i$  and sends it to  $\sigma$ . Notice that since  $P_3 \sqsubseteq_{\mathcal{O}} G_{(0,2)}^\beta$ , the global utility of  $\beta$  decreases to  $u_1^\beta = \sum_{k=1..2,4..} u(P_k)$ .

Now,  $\sigma$  is able to validate if  $\beta$  really changed its request to reach an agreement and did not lie. To do so,  $\sigma$  computes  $\langle G_1^\beta, K_1^\beta \rangle$  solving a concept contraction problem w.r.t. the new demand  $D_1$  and checks if  $G_0^\beta \sqsubseteq_{\mathcal{O}} G_1^\beta$ . In case of positive answer, then  $\sigma$  knows that  $\beta$  did not lie and it continues the negotiation process. Otherwise it may decide to leave the negotiation (conflict deal) or ask  $\beta$  to reformulate its counteroffer.

If the negotiation continues,  $\sigma$  computes its conflicting information w.r.t. to  $D_1$  and rewrites  $S$  as  $S = G_1^\sigma \sqcap K_1^\sigma$  where  $G_1^\sigma = \prod_{j=1}^m G_{(1,j)}^\sigma$ : Again, for each  $G_{(1,j)}^\sigma$  there exists a  $C_j$  in the normalized version of  $S$ . Hence, if  $\sigma$  decides to concede  $G_{(1,j)}^\sigma$ , its global utility decreases proportionally to the utility of  $P_h$  to which  $G_{(1,j)}^\sigma$  belongs to.

Similarly to  $\sigma$  in step 0,  $\beta$  computes  $\langle G_1^\sigma, K_1^\beta \rangle$  and checks if  $G_0^\sigma \sqsubseteq_O G_1^\sigma$  in order to check if  $\sigma$  lied.

The process ends when one of the following two conditions holds:

1. the global utility of an agent is lower than its **disagreement threshold**. In this case the negotiation terminates with a conflict deal.
2. there is nothing more to negotiate on and the global utility of each agent is greater than its disagreement threshold. In this case the negotiation terminates with an agreement. **The agreement  $A$  is computed** simply as  $A = D_{last} \sqcap S_{last}$ , where  $D_{last}$  and  $S_{last}$  are the request and the offer in the last step.

## 5.2 Minimum Concession

Since users can express an utility value also on bundles, whenever they concede an issue as the **minimum concession** (in term of minimum global utility decrease), the set of all the bundles in which the issue is present has to be taken into account. They choose based on the utility of the whole set.

For instance, consider the buyer set as preferences the following ones:

$$\begin{array}{ll}
 P_1 = \forall R.CN_1 & u^\beta(P_1) = 0.1 \\
 P_2 = (f \leq 200) & u^\beta(P_2) = 0.4 \\
 P_3 = \forall R.CN_1 \sqcap \forall R.CN_2 & u^\beta(P_3) = 0.5
 \end{array}$$

and at the n-th step the conflicting information is:

$$G_n^\beta = \forall R.CN_1 \sqcap (f \leq 200)$$

Hence,  $\beta$  can concede whether  $\forall R.CN_1$  or  $(f \leq 200)$ . If it concedes  $\forall R.CN_1$  then its global utility decreases of  $u^\beta(P_1) + u^\beta(P_3) = 0.6$ , while conceding  $(f \leq 200)$  its utility decreases of only  $u^\beta(P_2) = 0.4$ . In this case the **minimum concession** is  $(f \leq 200)$ .

## 5.3 The Algorithm

Here we define the behavior of agents during a generic n-th round of the negotiation process. For the sake of conciseness, we present only the algorithm related to  $\beta$ 's behavior. The behavior of  $\sigma$  is dual w.r.t. to the one of  $\beta$ .

**1-4** If there is nothing in conflict between the old  $D_{n-1}$  and just-arrived  $S_n$ , then there nothing more to negotiate on and the agreement is reached and computed. Notice that while computing the final agreement we use the "any-would-fit" approach to deal with *incomplete information* (see Section 4).

**5-11** If  $\beta$  discovers that  $\sigma$  lied on its concession, then  $\beta$  decides to exit the negotiation and terminates with a conflict deal. If we want  $\beta$  ask  $\sigma$  to concede again it is straightforward to change the protocol to deal with such a behavior.

**13-15** If after the minimum concession, the utility of  $\beta$  is less than its **disagreement threshold**, then the negotiation ends with a conflict deal.

```

1 if  $D_{n-1} \sqcap S_n \not\subseteq_{\mathcal{O}} \perp$  then
2   agreement  $A$  reached;
3   return  $A = D_{n-1} \sqcap S_n$ ;
4 end
5 if  $n > 0$  then
6   compute  $\langle G_n^\sigma, K_n^\sigma \rangle$  from  $D_{n-1}$  and  $S_n$ ;
7   if  $G_{n-1}^\sigma \not\subseteq_{\mathcal{O}} G_n^\sigma$  then
8      $\sigma$  lied;
9     conflict deal: exit;
10  end
11 end
12 compute minimum concession  $G_{(n-1,i)}^\beta$ ;
13 if  $u_{n-1}^\beta < t^\beta$  then
14   conflict deal: exit;
15 end
16 formulate  $D_n$  deleting  $G_{(n-1,i)}^\beta$  from  $D_{n-1}$ ;
17 send  $D_n$  to  $\sigma$ ;

```

**Algorithm 1:** The behavior of  $\beta$  at step  $n$

## 6 Conclusion

We have motivated and illustrated a logic-based approach to bilateral negotiation in P2P e-marketplaces, and proposed a *DL-based* alternating-offers protocol exploiting Description Logics and related inference services and utility theory to find the most suitable agreements. Work is ongoing on various directions, namely: extending the DL adopted, finding a “cheap” way to ensure that the reached agreement is Pareto-efficient, and carry out large scale experiments with real advertisements.

## References

1. F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *proc. of IJCAI-91*, pages 452–457, 1991.
2. S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Concept Abduction and Contraction in Description Logics. In *Proc. of DL’03*, volume 81 of *CEUR Workshop Proceedings*, September 2003.
3. S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an E-Marketplace. *Electronic Commerce Research and Applications*, 4(4):345–361, 2005.
4. U. Endriss. Monotonic concession protocols for multilateral negotiation. In *AAMAS*. ACM, 2006.
5. P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books, MIT Press, Cambridge, MA, 1988.
6. J.S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press., 1994.
7. S. Kraus. *Strategic Negotiation in Multiagent Environments*. The MIT Press.
8. A. Ragone, T. Di Noia, E. Di Sciascio, and F. Donini. A logic-based framework to compute pareto agreements in one-shot bilateral negotiation. In *Proc. of ECAI’06*, pages 230–234, 2006.
9. A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50:97–109, 1982.

## On conjunctive query answering in $\mathcal{EL}$

Riccardo Rosati

Dipartimento di Informatica e Sistemistica  
 Università di Roma “La Sapienza”  
 Via Salaria 113, 00198 Roma, Italy  
 rosati@dis.uniroma1.it

### 1 Introduction

In this paper we study conjunctive query answering in the description logics of the  $\mathcal{EL}$  family [2, 3, 7, 6, 5], in particular we consider the DLs  $\mathcal{EL}$ ,  $\mathcal{ELH}$ ,  $\mathcal{EL}^+$ , and  $\mathcal{EL}^{++}$ . The  $\mathcal{EL}$  family has been recently defined in order to identify DLs both having interesting expressive abilities and allowing for tractable reasoning. While the standard reasoning tasks (like concept subsumption and instance checking) have been analyzed in the past for such logics, almost no result is known about answering conjunctive queries in the logics of the  $\mathcal{EL}$  family, with the exception of the lower complexity bounds which are immediate consequence of the results in [8] (and of the characterization of instance checking in [7, 3]).

More specifically, we present the following results:

1. we define a query-rewriting-based technique for answering unions of conjunctive queries in  $\mathcal{EL}$ . More precisely, we present an algorithm based on the idea of reducing query answering in  $\mathcal{EL}$  to answering recursive Datalog queries. We also show that this technique can be easily extended to deal with  $\mathcal{ELH}$  KBs;
2. based on the above technique, we prove that answering unions of conjunctive queries in  $\mathcal{EL}$  and  $\mathcal{ELH}$  is PTIME-complete with respect to both data complexity (i.e., with respect to the size of the ABox) and knowledge base complexity (i.e., with respect to the size of the knowledge base) and is NP-complete with respect to combined complexity (i.e., with respect to the size of both the knowledge base and the query);
3. conversely, we prove that answering conjunctive queries is undecidable in both  $\mathcal{EL}^+$  and  $\mathcal{EL}^{++}$ .

As an immediate consequence of the above results, it turns out that if, besides the standard reasoning tasks in DL, also conjunctive query answering is of interest, then  $\mathcal{EL}$  and  $\mathcal{ELH}$  still exhibit a nice computational behaviour, since they allow for tractable query answering, while the two extensions  $\mathcal{EL}^+$  and  $\mathcal{EL}^{++}$  do not show the same behaviour, since conjunctive query answering is undecidable in such DLs. Consequently,  $\mathcal{EL}^+$  and  $\mathcal{EL}^{++}$  do not appear well-suited for applications requiring the full power of conjunctive queries.

### 2 Preliminaries

In this section we briefly recall the logics in the  $\mathcal{EL}$  family, in particular the DLs  $\mathcal{EL}$ ,  $\mathcal{EL}^+$ , and  $\mathcal{EL}^{++}$ , and introduce query answering over knowledge bases expressed in such description logics.

**$\mathcal{EL}$  and its extensions.**  $\mathcal{EL}$  [2] is the DL whose abstract syntax for concept expressions is the following:  $C ::= A \mid \exists R.C \mid C_1 \sqcap C_2 \mid \top$ , where  $A$  is a concept name,  $R$  is a role name, and the TBox is a set of concept inclusion assertions of the form  $C_1 \sqsubseteq C_2$ .  $\mathcal{ELH}$  [7, 6] extends  $\mathcal{EL}$  by also allowing in the TBox simple role inclusion assertions of the form  $R_1 \sqsubseteq R_2$ , where  $R_1$  and  $R_2$  are role names.  $\mathcal{EL}^+$  [5] extends  $\mathcal{EL}$  by also allowing in the TBox role inclusion assertions of the form  $R_1 \circ \dots \circ R_n \sqsubseteq R_{n+1}$ , where each  $R_i$  is a role name. Finally,  $\mathcal{EL}^{++}$  [3] extends  $\mathcal{EL}^+$  by allowing the new concept expressions  $\perp$ ,  $\{a\}$  and the concrete domain constructor  $p(f_1, \dots, f_n)$ .

As usual in DLs, a knowledge base (KB)  $\mathcal{K}$  is a pair  $\langle \mathcal{T}, \mathcal{A} \rangle$  where the TBox is a set of concept inclusions and role inclusions, and the ABox  $\mathcal{A}$  is a set of instance assertions of the form  $A(a)$ ,  $R(a, b)$  where  $A$  is a concept name,  $R$  is a role name, and  $a, b$  are constant (individual) names. Notice that in all the DLs considered,  $\mathcal{T}$  may contain cyclic concept inclusions (GCIs) (as well as cyclic role inclusions in  $\mathcal{ELH}$ ,  $\mathcal{EL}^+$  and  $\mathcal{EL}^{++}$ ).

The semantics of concept and role constructs is well-known [5]. The semantics of a KB is defined as usual, based on the interpretation of concept and role expressions [4]. We point out that we do not impose the unique name assumption (UNA) on constant names: however, our results also hold under the UNA.

As shown in [3],  $\mathcal{EL}^+$  TBoxes admit a normal form, i.e., we can assume without loss of generality that every concept inclusion in the TBox is in one of the following forms:  $A_1 \sqsubseteq A_2$ ,  $A_1 \sqcap A_2 \sqsubseteq A_3$ ,  $A_1 \sqsubseteq \exists R.A_2$ ,  $\exists R.A_1 \sqsubseteq A_2$ , where each  $A_1, A_2, A_3$  is either a concept name or the concept  $\top$ ,  $R$  is a role name, and every role inclusion is of the form  $R_1 \sqsubseteq R_2$  or  $R_1 \circ R_2 \sqsubseteq R_3$ , where  $R_1, R_2, R_3$  are role names.

**Unions of conjunctive queries.** We now briefly recall conjunctive queries and unions of conjunctive queries. To simplify the notation in the next sections, we use a Datalog-like notation for such queries.

A Datalog rule is an expression of the form  $\alpha :- body$ , in which the head  $\alpha$  is an atom (i.e., an expression of the form  $p(t_1, \dots, t_n)$  in which each  $t_i$  is either a constant or a variable) and  $body$  is a set of atoms, such that each variable occurring in  $\alpha$  also occurs in some atom in  $body$ .

A conjunctive query (CQ) over a DL-KB  $\mathcal{K}$  is a Datalog rule using a special predicate name  $p_q$  (i.e.,  $p_q$  does not belong to the set of concept and role names occurring in  $\mathcal{K}$ ) in the head of the rule, and whose body is a set of atoms whose predicates are concept and role names occurring in  $\mathcal{K}$  (notice that the predicate  $p_q$  cannot occur in the body of the rule). The arity of  $q$  is defined as the arity of  $p_q$ . A Boolean CQ is a CQ whose arity is zero. For a CQ  $q$ , we denote by  $body(q)$  the body of the Datalog rule corresponding to  $q$ . A union of conjunctive queries (UCQ)  $Q$  over  $\mathcal{K}$  is a set of CQs of the same arity which use the same predicate  $p_Q$  in the head of every rule.

For ease of exposition, and without loss of generality, from now on we only consider Boolean queries, and the *query entailment* problem. It is well-known that *query answering* of an arbitrary (non-Boolean) query can be reduced to query entailment.

The semantics of (Boolean) UCQs over DL-KBs is the usual one (see, e.g., [8]).

In the following, we study complexity of query entailment over DL-KBs. In particular, we consider *data complexity*, i.e., the complexity with respect to the size of the ABox, *KB complexity*, i.e., the complexity with respect to the size of both the ABox and the TBox, and *combined complexity*, i.e., the complexity with respect to the size of both the KB and the query.

### 3 Answering unions of conjunctive queries in $\mathcal{EL}$ and $\mathcal{EL}\mathcal{H}$

We now present an algorithm for answering unions of conjunctive queries posed to  $\mathcal{EL}$ -KBs. We start by introducing the auxiliary procedures *Unify*, *Roll-up*, *Normalize*, *Rename*, *Rules*, and *0-Rules*.

**The procedure *Unify*.** Given a UCQ  $Q$ ,  $Unify(Q)$  returns a UCQ obtained by adding to  $Q$  all the possible unifications of terms for every conjunctive query  $q$  in  $Q$ .

**The procedure *Roll-up*.** Given a UCQ  $Q'$ ,  $Roll-up(Q')$  returns a rewriting of the query  $Q'$  obtained by expressing subtrees in the query through  $\mathcal{EL}$  concept expressions. Formally, we define  $Roll-up(Q') = \bigcup_{q \in Q'} Roll-up(q)$  where  $Roll-up(q)$  returns the CQ obtained from the CQ  $q$  by exhaustively applying the following rewriting rules to the atoms in  $body(q)$ :

1. if variable  $y$  only occurs in a binary atom of the form  $R(t, y)$ , then replace  $R(t, y)$  with the unary atom  $\exists R.\top(t)$
2. if variable  $y$  only occurs in unary atoms of the form  $C_1(y), \dots, C_n(y)$ , then replace the above atoms with the 0-ary atom  $(C_1 \sqcap \dots \sqcap C_n)^0$ ;
3. if variable  $y$  only occurs in unary atoms of the form  $C_1(y), \dots, C_n(y)$  and in a single binary atom  $R(t, y)$ , then replace all the above atoms in which  $y$  occurs with the unary atom  $(\exists R.C_1 \sqcap \dots \sqcap C_n)(t)$ ;
4. if  $y$  is a variable which only occurs in an atom of the form  $R(y, z)$  where  $z$  is a variable different from  $y$ , and there is another atom of the form  $R(t, z)$  in  $body(q)$ , then delete  $R(y, z)$ .

Notice that the query returned by *Roll-up* is not exactly a UCQ according to the definition given in Section 2, since arbitrary concept expressions  $C$  may occur as (both unary and 0-ary) predicate symbols in the body of the CQs of the returned query. So we call such a query an *extended UCQ*.

**The procedure *Normalize*.** Given an  $\mathcal{EL}$  TBox  $\mathcal{T}$  and an extended UCQ  $Q'$ ,  $Normalize(\mathcal{T}, Q')$  returns an  $\mathcal{EL}$  TBox  $\mathcal{T}'$  in normal form which: (i) is a conservative extension of  $\mathcal{T}$ ; (ii) defines all concept expressions occurring in  $Q'$  and  $\mathcal{T}'$ ; (iii) is closed with respect to the entailed concept inclusions. More precisely,  $\mathcal{T}'$  is such that:

- for every concept expression  $C$  such that  $\mathcal{T}'$  contains a concept inclusion of the form  $C \sqsubseteq D$  or  $D \sqsubseteq C$ , there exists a concept name  $C'$  such that  $\mathcal{T}' \models C' \equiv C$ ;
- for every concept expression  $C$  such that  $Q'$  contains either a unary atom of the form  $C(t)$  or a 0-ary atom of the form  $(C)^0$ , there exists a concept name  $C'$  such that  $\mathcal{T}' \models C' \equiv C$ .
- $\mathcal{T}'$  is closed with respect to the entailment of simple concept inclusions, i.e., for every pair of distinct concept names  $A_1, A_2$  occurring in  $\mathcal{T}'$ , if  $\mathcal{T}' \models A_1 \sqsubseteq A_2$  then  $A_1 \sqsubseteq A_2 \in \mathcal{T}'$ .

From the existence of a linear normalization procedure for  $\mathcal{EL}$  KBs and from the results on entailment of concept inclusions in  $\mathcal{EL}$  shown in [7], it follows that it is possible to compute a TBox  $\mathcal{T}'$  satisfying the above conditions in polynomial time with respect to the size of  $\mathcal{T}$  and  $Q'$ .

**The procedure *Rename*.** Given an extended UCQ  $Q'$  and a normalized  $\mathcal{EL}$  TBox  $\mathcal{T}'$ ,  $Rename(Q', \mathcal{T}')$  returns the UCQ obtained from  $Q'$  by replacing each complex concept

expression  $C$  (i.e., such that  $C$  is not a concept name) occurring in  $Q'$  with the corresponding concept name  $C'$  in  $\mathcal{T}'$  (i.e., the concept name  $C'$  such that  $\mathcal{T}' \models C' \equiv C$ ). Since the presence of complex concept expressions is eliminated from the query returned by  $Rename(Q', \mathcal{T}')$ , such a query corresponds to a set of ordinary Datalog rules.

**The procedure Rules.** Given a normalized  $\mathcal{EL}$  TBox  $\mathcal{T}'$ ,  $Rules(\mathcal{T}')$  returns the set of Datalog rules corresponding to  $\mathcal{T}'$ . More precisely,  $Rules(\mathcal{T}')$  is the following set of Datalog rules:

- the rule  $A_2(x) :- A_1(x)$  for each concept inclusion  $A_1 \sqsubseteq A_2$  in  $\mathcal{T}'$ , where  $A_1, A_2$  are concept names;
- the rule  $A_3(x) :- A_1(x), A_2(x)$  for each  $A_1 \sqcap A_2 \sqsubseteq A_3$  in  $\mathcal{T}'$ , where  $A_1, A_2, A_3$  are concept names;
- the rule  $A_2(x) :- R(x, y), A_1(y)$  for each  $\exists R.A_1 \sqsubseteq A_2$  in  $\mathcal{T}'$ , where  $A_1, A_2$  are concept names.

Notice that concept inclusions of the form  $A_1 \sqsubseteq \exists R.A_2$  are not actually considered in the computation of  $Rules(\mathcal{T}')$ .

**The procedure 0-Rules.** Finally, to correctly handle 0-ary atoms in the query, we have to define entailment of inclusions between 0-ary predicates with respect to the TBox  $\mathcal{T}'$ . In particular, for every pair of concept names  $A_1, A_2$  occurring in  $\mathcal{T}'$ , we want to decide whether the first-order existential sentence  $\exists x.A_1(x) \rightarrow \exists y.A_2(y)$  is satisfied by every model of  $\mathcal{T}'$ . Actually, entailment of such sentences can be decided in a way very similar to entailment of concept inclusions. More precisely, we define inductively the following relation  $\vdash_{\mathcal{T}'}^{\exists}$ , between concept names occurring in  $\mathcal{T}'$ :

- $A \vdash_{\mathcal{T}'}^{\exists} A$  for every concept name  $A$  occurring in  $\mathcal{T}'$ ;
- if  $A_1 \vdash_{\mathcal{T}'}^{\exists} A_2$  and  $A_2 \sqsubseteq A_3 \in \mathcal{T}'$  then  $A_1 \vdash_{\mathcal{T}'}^{\exists} A_3$ ;
- if  $A_1 \vdash_{\mathcal{T}'}^{\exists} A_2$  and  $A_2 \sqsubseteq \exists R.A_3 \in \mathcal{T}'$ , then  $A_1 \vdash_{\mathcal{T}'}^{\exists} A_3$ .

Based on the fact that  $\mathcal{T}'$  is closed with respect to entailment of inclusions between atomic concepts, it can be shown that, for every pair of concept names  $A_1, A_2$  occurring in  $\mathcal{T}'$ , the sentence  $\exists x.A_1(x) \rightarrow \exists y.A_2(y)$  is satisfied by every model of  $\mathcal{T}'$  iff  $A_1 \vdash_{\mathcal{T}'}^{\exists} A_2$ .

Then, based on the relation  $\vdash_{\mathcal{T}'}^{\exists}$ , we define the procedure  $0\text{-Rules}(\mathcal{T}')$ , which returns the following set of Datalog rules:

- $A_2^0 :- A_1^0$  for each pair of concept names  $A_1, A_2$  such that  $A_1 \vdash_{\mathcal{T}'}^{\exists} A_2$ ;
- $A^0 :- A(x)$  for each concept name  $A$ .

**The algorithm computeRewriting.** We are now ready to define the algorithm `computeRewriting` which, given an  $\mathcal{EL}$  TBox  $\mathcal{T}$  and a Boolean UCQ  $Q$ , computes a Datalog program  $\mathcal{P}$  by making use of the procedures previously defined.

**Algorithm** `computeRewriting`( $Q, \mathcal{T}$ )

**Input:** Boolean union of conjunctive queries  $Q$ ,  $\mathcal{EL}$  TBox  $\mathcal{T}$

**Output:** Datalog program  $\mathcal{P}$

$Q' := Unify(Q)$ ;  
 $Q' := Roll\text{-up}(Q')$ ;  
 $\mathcal{T}' := Normalize(\mathcal{T}, Q')$ ;  
 $Q' := Rename(Q', \mathcal{T}')$ ;  
 $\mathcal{P} := Q' \cup Rules(\mathcal{T}') \cup 0\text{-Rules}(\mathcal{T}')$ ;  
**return**  $\mathcal{P}$

**The algorithm** `computeQueryEntailment`. The Datalog program  $\mathcal{P}$  computed by `computeRewriting(Q, T)` can be used to decide entailment of the query  $Q$  with respect to every  $\mathcal{EL}$ -KB, as shown by the algorithm `computeQueryEntailment(Q, K)` defined below. In the following,  $\top(\mathcal{A})$  denotes the set of facts  $\{\top(a) \mid a \text{ is a constant occurring in } \mathcal{A}\}$ , while  $MM(\mathcal{P})$  denotes the minimal model of a Datalog program  $\mathcal{P}$ .

**Algorithm** `computeQueryEntailment(Q, K)`

**Input:** Boolean UCQ  $Q$  (with head predicate  $p_Q$ ),  $\mathcal{EL}$ -KB  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

**Output:** *true* if  $\mathcal{K} \models Q$ , *false* otherwise

$\mathcal{P} := \text{computeRewriting}(Q, \mathcal{T})$ ;

**if**  $MM(\mathcal{P} \cup \mathcal{A} \cup \top(\mathcal{A})) \models p_Q$

**then return true else return false**

In practice, the above algorithm simply evaluates the Datalog program  $\mathcal{P}$  over the ABox  $\mathcal{A}$  (remember that  $\mathcal{A}$  is a set of ground atoms, hence  $\mathcal{P} \cup \mathcal{A}$  is a Datalog program) in order to decide whether  $Q$  is entailed by  $\mathcal{K}$ . The addition of the facts  $\top(\mathcal{A})$  is necessary in order to correctly handle the presence of the concept  $\top$  in the query (more precisely, in the evaluation of the Datalog program we consider  $\top$  as a concept name, i.e., an EDB predicate).

**Correctness.** We now show correctness of the algorithm `computeQueryEntailment`.

**Theorem 1.** *Let  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  be an  $\mathcal{EL}$ -KB and let  $Q$  be a UCQ. Then,  $\mathcal{K} \models Q$  iff `computeQueryEntailment(Q, K)` returns *true*.*

*Proof (sketch).* The proof of soundness of the technique is immediate. The proof of completeness is based on the construction of a canonical model for a normalized  $\mathcal{EL}$ -KB  $\mathcal{K}$  through the definition of the *chase* of  $\mathcal{K}$ . The chase of  $\mathcal{K}$  (denoted by  $\text{chase}(\mathcal{K})$ ) is a function which returns a generally infinite ABox and is inductively defined starting from the initial ABox  $\mathcal{A}$  and adding facts to  $\mathcal{A}$  based on the following *chase rules*:

- *chase-rule-1:* if  $A(a) \in \text{chase}(\mathcal{K})$  and  $A \sqsubseteq B \in \mathcal{T}$  and  $B(a) \notin \text{chase}(\mathcal{K})$  then add  $B(a)$  to  $\text{chase}(\mathcal{K})$ ;
- *chase-rule-2:* if  $A_1(a) \in \text{chase}(\mathcal{K})$  and  $A_2(a) \in \text{chase}(\mathcal{K})$  and  $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$  and  $B(a) \notin \text{chase}(\mathcal{K})$  then  $B(a) \in \text{chase}(\mathcal{K})$ ;
- *chase-rule-3:* if  $R(a, b) \in \text{chase}(\mathcal{K})$  and  $A(a) \in \text{chase}(\mathcal{K})$  and  $\exists R.A \sqsubseteq B \in \mathcal{T}$  and  $B(a) \notin \text{chase}(\mathcal{K})$  then  $B(a) \in \text{chase}(\mathcal{K})$ ;
- *chase-rule-4:* if  $A(a) \in \text{chase}(\mathcal{K})$  and  $A \sqsubseteq \exists R.B \in \mathcal{T}$  and there is no  $b$  such that both  $R(a, b) \in \text{chase}(\mathcal{K})$  and  $B(b) \in \text{chase}(\mathcal{K})$  then add  $R(a, n)$  and  $B(n)$  to  $\text{chase}(\mathcal{K})$ , where  $n$  is a constant that does not occur already in  $\text{chase}(\mathcal{K})$ ;
- *chase-rule-5:* if  $a$  is a constant occurring in  $\text{chase}(\mathcal{K})$  and  $\top(a) \notin \text{chase}(\mathcal{K})$  then add  $\top(a)$  to  $\text{chase}(\mathcal{K})$ .

The chase of  $\mathcal{K}$  is a generally infinite ABox which is isomorphic to a *canonical model* of  $\mathcal{K}$ , denoted by  $\mathcal{I}_{\text{chase}(\mathcal{K})}$ . Such a model can be used to compute entailment of UCQs in  $\mathcal{K}$ , which is formally stated by the following property:

**Lemma 1.** *For every Boolean UCQ  $Q$ ,  $\mathcal{K} \models Q$  iff  $\mathcal{I}_{\text{chase}(\mathcal{K})} \models Q$ .*

Then, we use the chase to prove completeness of our algorithm. Let us consider the first part of the algorithm `computeRewriting`, which ends with the execution of `Rename(Q', T')`. With respect to this part of the rewriting, we prove the following:

**Lemma 2.** *Let  $Q''$  be the UCQ returned by  $\text{Rename}(Q', T')$  in the algorithm  $\text{computeRewriting}(Q, T)$ . If  $\mathcal{I}_{\text{chase}(\mathcal{K})} \models Q''$  then there exists a CQ  $q \in Q''$  and a homomorphism  $H$  of  $\text{body}(q)$  in  $\text{chase}(\mathcal{K})$  such that, for every variable  $x$  occurring in  $q$ ,  $H(x)$  is a constant occurring in  $\mathcal{A}$ .*

Although the above lemma might seem rather obscure, it implies the following crucial property: answering the query computed by  $\text{Rename}(Q', T')$  can actually be done by first “grounding” the query (considering all the instantiations of the variables with constants occurring in  $\mathcal{A}$ ) and then considering each atom in a separate way. So, the above lemma shows that the first part of the rewriting reduces entailment of a UCQ to entailment of single atoms.

Then, we consider the second part of the rewriting, i.e., the set of Datalog rules generated by  $\text{Rules}(T')$  and  $0\text{-Rules}(T')$ . Here, we use the chase of  $\mathcal{K}$  to prove that the Datalog program  $\mathcal{P}'$  returned by  $\text{Rules}(T')$  constitutes a correct encoding of the entailment of unary and binary atoms (which correspond to standard instance checking problems), in the sense that the minimal model of  $\mathcal{P}' \cup \mathcal{A} \cup \top(\mathcal{A})$  contains all ground unary atoms  $A(a)$  such that  $\mathcal{K} \models A(a)$  and all ground binary atoms  $R(a, b)$  such that  $\mathcal{K} \models R(a, b)$ ; similarly, we prove that the Datalog program returned by  $0\text{-Rules}(T')$  constitutes a correct encoding of the entailment of 0-ary atoms. Formally:

**Lemma 3.** *Let  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  be a normalized  $\mathcal{EL}$ -KB, let  $\mathcal{P}'$  be the Datalog program returned by  $\text{Rules}(T)$ , and let  $\alpha$  be either an atom of the form  $A(a)$  where  $A$  is a concept name or an atom of the form  $R(a, b)$  where  $R$  is a role name ( $a, b$  are constants occurring in  $\mathcal{A}$ ). If  $\mathcal{K} \models A(a)$  then  $\text{MM}(\mathcal{P}' \cup \mathcal{A} \cup \top(\mathcal{A})) \models \alpha$ .*

**Lemma 4.** *Let  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$  be a normalized  $\mathcal{EL}$ -KB, let  $\mathcal{P}'$  be the Datalog program returned by  $0\text{-Rules}(T)$ , and let  $A$  be a concept name occurring in  $\mathcal{T}$ . If the sentence  $\exists x.A(x)$  is satisfied by every model for  $\mathcal{K}$ , then  $\text{MM}(\mathcal{P}' \cup \mathcal{A} \cup \top(\mathcal{A})) \models A^0$ .*

From the above properties of the two parts of the rewriting, the thesis follows.  $\square$

**Complexity results.** Based on the above algorithm, we can characterize the computational properties of entailment of UCQs in  $\mathcal{EL}$ .

**Theorem 2.** *Entailment of UCQs in  $\mathcal{EL}$  is: (i) PTIME-complete with respect to data complexity; (ii) PTIME-complete with respect to KB complexity; (iii) NP-complete with respect to combined complexity.*

*Proof (sketch).* PTIME-hardness with respect to data complexity has been proved in [8], while NP-hardness with respect to combined complexity follows from NP-hardness of simple database evaluation of a CQ [1]. Membership in PTIME with respect to KB complexity follows from the fact that the procedures *Normalize*, *Rename*, *Rules*, and *0-Rules* run in time polynomial with respect to their input, which implies that the algorithm  $\text{computeRewriting}$  runs in time polynomial with respect to the size of  $\mathcal{T}$ . Now, since the Datalog program  $\mathcal{P}$  returned by  $\text{computeRewriting}(Q, T)$  has size polynomial with respect to  $\mathcal{T}$ , and the number of variables used in each rule of  $\mathcal{P}$  does not depend on  $\mathcal{K}$ , it follows that the minimal model of  $\mathcal{P} \cup \mathcal{A} \cup \top(\mathcal{A})$  can be computed in time polynomial in the size of  $\mathcal{K}$ , thus the algorithm  $\text{computeQueryEntailment}(Q, \mathcal{K})$  also runs in time polynomial in the size of  $\mathcal{K}$ . Finally, membership in NP with respect to combined complexity follows from the fact that all the procedures executed by

computeRewriting run in time polynomial with respect to their input, with the only exception of the procedure *Unify*, which runs in exponential time with respect to the size of  $Q$ . However, if we consider a nondeterministic version of such a procedure, which returns just one CQ  $q'$  obtained by choosing one CQ  $q$  in  $Q$  and one substitution which is applied to  $q$ , then the whole algorithm computeRewriting runs in time polynomial with respect to the size of both  $Q$  and  $\mathcal{T}$ . Then, in a way analogous to the above proof of PTIME-membership for KB complexity, it follows that this nondeterministic version of the algorithm computeQueryEntailment( $Q, \mathcal{K}$ ) also runs in time polynomial in the size of  $Q$  and  $\mathcal{K}$ , which implies the thesis.  $\square$

We remark that the above characterization with respect to data complexity was already stated in [12].

**Extension to  $\mathcal{ELH}$ .** Finally, the above technique for deciding entailment of UCQs can be easily extended in order to deal with  $\mathcal{ELH}$ -KBs. The algorithms computeRewriting and computeQueryEntailment are actually the same as before, the only differences concern the procedures *Roll-up*, *Normalize*, and *Rules*. Specifically: (i) the procedure *Roll-up* must take into account the presence of role assertions, since such inclusions allow for additional eliminations of redundant binary atoms. More precisely, we add to the previous definition of *Roll-up* the following rule: if  $R_1$  and  $R_2$  are distinct role names,  $R_1(t_1, t_2)$  and  $R_2(t_1, t_2)$  occur in  $body(q)$ , and  $\mathcal{T} \models R_1 \sqsubseteq R_2$ , then delete  $R_2(t_1, t_2)$ ; (ii) the procedure *Normalize*( $\mathcal{T}, Q'$ ) must be modified in order to account for the presence of simple role inclusions in the TBox. A procedure for deciding entailment of concept and role inclusions in  $\mathcal{ELH}$  TBoxes has been presented in [7]: such a procedure shows that entailment of such inclusions can still be computed in polynomial time; (iii) the procedure *Rules* also adds a Datalog rule for each role inclusion in the  $\mathcal{ELH}$  TBox. More precisely, in the case of an  $\mathcal{ELH}$  TBox, the previous definition of the set of rules returned by *Rules*( $\mathcal{T}'$ ) is modified by adding the following condition: add the rule  $R_2(x, y) :- R_1(x, y)$  for each role inclusion  $R_1 \sqsubseteq R_2$  in  $\mathcal{T}'$ .

The above extension demonstrates that the computational characterization of entailment of UCQs provided by Theorem 2 extends to the case of  $\mathcal{ELH}$ -KB.

#### 4 Undecidability of conjunctive query answering in $\mathcal{EL}^+$

We now show that the nice computational properties of answering conjunctive queries in  $\mathcal{EL}$ , shown in the previous section, do not extend to  $\mathcal{EL}^+$  and  $\mathcal{EL}^{++}$ , since answering conjunctive queries in such DLs is undecidable.

**Theorem 3.** *Entailment of conjunctive queries in  $\mathcal{EL}^+$  is undecidable.*

*Proof (sketch).* We reduce the emptiness problem for intersection of context-free languages, which is known to be undecidable [9], to conjunctive query entailment in  $\mathcal{EL}^+$ . Consider two context-free grammars  $G_1 = \langle NT_1, Term, S_1, P_1 \rangle$ ,  $G_2 = \langle NT_2, Term, S_2, P_2 \rangle$ , where  $NT_1$  is the alphabet of nonterminal symbols of  $G_1$ ,  $NT_2$  is the alphabet of nonterminal symbols of  $G_2$  (which is disjoint from  $NT_1$ ),  $Term$  is the alphabet of terminal symbols (which is the same for  $G_1$  and  $G_2$  and is disjoint from  $NT_1 \cup NT_2$ ),  $S_1$  is the axiom of  $G_1$ ,  $S_2$  is the axiom of  $G_2$ ,  $P_1$  is the set of production rules of  $G_1$  and  $P_2$  is the set of production rules of  $G_2$ . W.l.o.g., we assume that no production rule has an empty right-hand side. Now consider the  $\mathcal{EL}^+$  KB  $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ , which uses  $Term \cup NT_1 \cup NT_2$  as the set of role names plus the concept name  $C$ . The TBox  $\mathcal{T}$  is composed of: (i) the role inclusion assertions encoding the production rules

in  $P_1$ : e.g., if  $P_1$  contains the production rule  $X \rightarrow UVW$ , we add to  $\mathcal{T}$  the role inclusion  $U \circ V \circ W \sqsubseteq X$ ; (ii) the role inclusion assertions encoding the production rules in  $P_2$ ; (iii) the role inclusion assertion  $C \sqsubseteq \exists T_i.C$  for every  $T_i \in \text{Term}$ . Moreover, the ABox  $\mathcal{A}$  contains the only assertion  $C(a)$ . Finally, consider the Boolean conjunctive query  $q$  of the form  $p_q :- S_1(a, y), S_2(a, y)$ , where  $S_1$  is the axiom of grammar  $G_1$  and  $S_2$  is the axiom of grammar  $G_2$ .

We prove that the language  $\mathcal{L}(G_1) \cap \mathcal{L}(G_2)$  is non-empty iff  $\langle \mathcal{T}, \mathcal{A} \rangle \models q$ . To this aim, we make use of three auxiliary properties. Such properties make use of the notion of chase of an  $\mathcal{EL}^+$ -KB, which extends in a straightforward way the chase for  $\mathcal{EL}$ , introduced in the proof of Theorem 1, by adding a chase rule for role inclusions.

**Lemma 5.** *Let  $x$  and  $y$  be two terms in  $\text{chase}(\mathcal{K})$ . There is at most one path of terminal symbols between  $x$  and  $y$  in  $\text{chase}(\mathcal{K})$ , i.e., a sequence  $T_1(z_1, z_2), \dots, T_k(z_k, z_{k+1})$  with  $z_1 = x, z_{k+1} = y$ , and s.t. each  $T_i(z_i, z_{i+1}) \in \text{chase}(\mathcal{K})$  and each  $T_i \in \text{Term}$ .*

**Lemma 6.** *Let  $x$  and  $y$  be two terms in  $\text{chase}(\mathcal{K})$ . Let  $\pi$  be the path of terminal symbols between  $x$  and  $y$  in  $\text{chase}(\mathcal{K})$ . Then, for every nonterminal symbol  $N \in NT_1$  (resp., for every  $N \in NT_2$ ),  $N(x, y) \in \text{chase}(\mathcal{K})$  iff  $N \Rightarrow_{G_1}^* \pi$  (resp., iff  $N \Rightarrow_{G_2}^* \pi$ ).*

**Lemma 7.** *For every word  $T_1 \dots T_k$  in  $\text{Term}^*$ , there exists a pair  $x, y$  such that there exists the path of terminal symbols  $T_1 \dots T_k$  between  $x$  and  $y$  in  $\text{chase}(\mathcal{K})$ .*

From the above properties, the thesis easily follows.  $\square$

Obviously, the above theorem also implies undecidability of conjunctive query entailment (and thus of conjunctive query answering) in  $\mathcal{EL}^{++}$ .

We point out that the above theorem has been independently proved by other authors [11, 10].

**Acknowledgments.** This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU, by project HYPER, funded by IBM through a SUR Award grant, and by MIUR FIRB project TOCAI.IT.

## References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
2. F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. of IJCAI 2003*, pages 325–330, 2003.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the  $\mathcal{EL}$  envelope. In *Proc. of IJCAI 2005*, pages 364–369, 2005.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*, 2003.
5. F. Baader, C. Lutz, and B. Suntisrivaraporn. Efficient reasoning in  $\mathcal{EL}^+$ . In *Proc. of DL 2006*. <http://ceur-ws.org/Vol-189/>, 2006.
6. S. Brandt. On subsumption and instance problem in ELH w.r.t. general TBoxes. In *Proc. of DL 2004*. <http://ceur-ws.org/Vol-104/>, 2004.
7. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In *Proc. of ECAI 2004*, pages 298–302, 2004.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, pages 260–270, 2006.
9. M. A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley, 1978.
10. A. Krisnadhi and C. Lutz. Data complexity in the EL family of DLs. In *Proc. DL 2007*.
11. M. Krötzsch and S. Rudolph. Conjunctive queries for EL with role composition. In *Proc. DL 2007*.
12. R. Rosati. The limits of querying ontologies. In *Proc. ICDT 2007, LNCS 4353*, pages 164–178. Springer, 2007.

# Combining Two Formalism for Reasoning about Concepts<sup>\*</sup>

(extended abstract)

N.V. Shilov, N.O. Garanina, and I.S. Anureev

A.P. Ershov Institute of Informatics Systems,  
Lavren'ev av., 6, Novosibirsk 630090, Russia,  
{shilov, garanina, anureev}@iis.nsk.su

**Abstract.** There are two major formalisms that are developed around concepts. The first one is Formal Concept Analysis (FCA) by R. Wille and B. Ganter. Roughly speaking, FCA is an extension of algebraic Lattice Theory for knowledge representation. The second formalism, Description Logic (DL), goes back to the universal terminological logic by P.F. Patel-Schneider. It is closely related to modal and program logics. DL is widely used for ontology research, design, and implementation. Since both formalism use concepts and are used for closely related purposes, it is very natural to compare and combine them.

In this paper we introduce and study variants of DL extended by three constructs motivated by FCA. Intentional semantics of two of the new constructs are new modalities that correspond to ‘intent’ and ‘extent’ (two major algebraic constructions of FCA). The third new construct is a connective that is designed to express the ‘formal concept’ property. If  $\mathcal{L}$  is a variant of DL then we call  $\mathcal{L}$  extended by these new constructs by  $\mathcal{L}$  for FCA and denote this logic by  $\mathcal{L}/\text{FCA}$ .

We compare expressive powers of  $\mathcal{L}/\text{FCA}$  and  $\mathcal{L}(\neg, -)$  – another variant of  $\mathcal{L}$  extended by role complement  $\neg$  and inverse  $-$  simultaneously. We demonstrate that  $\mathcal{L}/\text{FCA}$  can be expressed in  $\mathcal{L}(\neg, -)$ . It implies that for the basic description logic  $\mathcal{ALC}$ ,  $\mathcal{ALC}/\text{FCA}$  is decidable.

## 1 Basic Description Logics

Description logics [2] has originated from the universal terminological logic [9]. There exists many variants of description logics, but we will define only some of them in this section<sup>1</sup>.

**Definition 1.** *Syntax of every description logic is constructed from disjoint alphabets of concept, role, and object symbols  $CS$ ,  $RS$ , and  $OS$ , respectively. The*

<sup>\*</sup> This research is supported in parts by joint grant RFBR 05-01-04003-a - DFG project COMO, GZ: 436 RUS 113/829/0-1, by grant RFBR 06-01-00464-a, and Integration Grant n.14 Siberia Branch, Russian Academy of Science.

<sup>1</sup> We give detailed definition of description logics for avoiding ambiguities, since there exists some difference in syntax notation between different research groups.

sets of concept terms (or concepts)  $CT$  and role terms (or roles)  $RT$  are defined by induction. The usual definition admits the following clauses<sup>2</sup>.

- (Concept terms)
  - the top concept  $\top$  and the bottom concept  $\perp$  are concept terms;
  - any concept symbol is a concept term;
  - for any concepts  $X$  and  $Y$  their union ( $X \sqcup Y$ ) and intersection ( $X \sqcap Y$ ) are concept terms;
  - for any concept  $X$  its complement ( $\neg X$ ) is a concept term;
  - for any role  $R$  and any concept  $X$  the universal ( $\forall R. X$ ) and existential ( $\exists R. X$ ) restrictions are concept terms;
- (Role terms)
  - the top role  $\nabla$  and the bottom role  $\triangle$  are role terms;
  - any role symbol is a role term;
  - for any roles  $R$  and  $S$  their union ( $R \sqcup S$ ), intersection ( $R \sqcap S$ ), and composition ( $R \circ S$ ) are terms;
  - for any role  $R$  its complement ( $\neg R$ ), inverse ( $R^{-}$ ), and transitive closure ( $R^{+}$ ) are role terms.

Concept and role terms altogether form the set of terminological expressions.

**Definition 2.**

- For any concepts  $X$  and  $Y$ , any roles  $R$  and  $S$  the following expressions are called terminological sentences:  $X \sqsubseteq Y$ ,  $X \doteq Y$ ,  $R \sqsubseteq S$ , and  $R \doteq S$ . A  $TBox$  is a set of terminological sentences.
- For any concept  $X$ , any role  $R$ , and any object symbols  $a$  and  $b$  the following expressions are called assertional sentences: a concept assertion  $a : X$  and a role assertion  $(a, b) : R$ . An  $ABox$  is a set of assertional sentences.
- A knowledge base is a finite set of terminological and assertional sentences. Every knowledge base consists of an appropriate  $TBox$  and  $ABox$ .

**Definition 3.** Semantics of any description logic is defined in Kripke-like terminological interpretations. Every terminological interpretation is a pair  $(D, I)$ , where  $D$  is a set (that is called domain) and  $I$  is a mapping (that is called interpretation function). This function maps object symbols to elements of  $D$ , concept symbols to subsets of  $D$ , role symbols to binary relations on  $D$ :  $I = I_{OS} \cup I_{CS} \cup I_{RS}$ , where  $I_{OS} : OS \rightarrow D$ ,  $I_{CS} : CS \rightarrow 2^D$ , and  $I_{RS} : RS \rightarrow 2^{D \times D}$ . The unique name assumption holds for this function:  $I(a) \neq I(b)$  for all different object symbols  $a$  and  $b$ . The interpretation function can be extended to all terminological expressions as follows.

- (Concept semantics)
  - $I(\top) = D$  and  $I(\perp) = \emptyset$ ;
  - $I(X \sqcup Y) = I(X) \cup I(Y)$  and  $I(X \sqcap Y) = I(X) \cap I(Y)$ ;
  - $I(\neg X) = D \setminus I(X)$ ;

<sup>2</sup> We omit some constructs that can be treated as derived features.

- $I(\forall R. X) = \{s \in D : \forall t \in D(\text{if } (s, t) \in I(R) \text{ then } t \in I(X))\}$ ,
- $I(\exists R. X) = \{s \in D : \exists t \in D((s, t) \in I(R) \text{ and } t \in I(X))\}$ ;
- (Role semantics)
  - $I(\nabla) = D^2$  and  $I(\Delta) = \emptyset$ ;
  - $I(R \sqcup S) = I(R) \cup I(S)$ ,  $I(R \sqcap S) = I(R) \cap I(S)$ ,  $I(R \circ S) = I(R) \circ I(S)$  (righthand side ‘ $\circ$ ’ is composition of binary relations);
  - $I(\neg R) = D^2 \setminus I(R)$ ,  $I(R^-) = (I(R))^-$ , and  $I(R^+) = (I(R))^+$  (righthand side ‘ $+$ ’ is transitive closure of binary relations).

**Definition 4.** *Semantics of sentences is defined in terminological interpretations in terms of satisfiability relation as follows:*

- $(D, I) \models a : X$  iff  $I(a) \in I(X)$ ;
- $(D, I) \models X \sqsubseteq Y$  iff  $I(X) \subseteq I(Y)$ ;
- $(D, I) \models X \doteq Y$  iff  $I(X) = I(Y)$ ;
- $(D, I) \models (a, b) : R$  iff  $(I(a), I(b)) \in I(R)$ ;
- $(D, I) \models R \sqsubseteq S$  iff  $I(R) \subseteq I(S)$ ;
- $(D, I) \models R \doteq S$  iff  $I(R) = I(S)$ .

*This satisfiability relation can be extended on knowledge bases in a natural way: for any knowledge base  $KBase$ ,  $(D, I) \models KBase$  iff  $(D, I) \models \phi$  for every sentence  $\phi \in KBase$ . In the case of  $(D, I) \models KBase$ , the terminological interpretation  $(D, I)$  is said to be a (terminological) model for the knowledge base  $KBase$ . Let us say that a knowledge base  $KBase$  entails a sentence  $\psi$  (and write<sup>3</sup>  $Kbase \models \psi$ ) iff  $(D, I) \models \psi$  for every model  $(D, I)$  for  $KBase$ .*

**Definition 5.** *A concept  $X$  is said to be coherent (or satisfiable) with respect to a knowledge base  $KBase$  iff there exists a terminological model  $(D, I)$  for  $KBase$  such that  $I(X)$  is not empty. A knowledge base  $KBase$  is said to be satisfiable iff the top concept  $\top$  is coherent with respect to  $KBase$ .*

Satisfiability problem is to check for input knowledge base  $KBase$  whether it is satisfiable or not. It is well known that the problem is undecidable [2] for description logic that admits all syntax constructs that are enumerated in the definition 1. This undecidability boundary drives many researchers to study of description logics with decidable satisfiability problem. An important role in these studies belongs to a fragment that is called Attribute Language with Complements ( $\mathcal{ALC}$ ) [11]. In simple words,  $\mathcal{ALC}$  adopts role symbols as the only role terms, concept symbols – as elementary concept terms, and permits ‘boolean’ constructs ‘ $\neg$ ’, ‘ $\sqcup$ ’, ‘ $\sqcap$ ’, universal and existential (but non-limited) restrictions ‘ $\forall$ ’ and ‘ $\exists$ ’ as the only concept constructs. The formal definition follows.

**Definition 6.**  *$\mathcal{ALC}$  is a fragment of DL that comprises concepts that are defined by the following context-free grammar:*

$$C_{\mathcal{ALC}} ::= CS \mid \top \mid \perp \mid (\neg C_{\mathcal{ALC}}) \mid (C_{\mathcal{ALC}} \sqcup C_{\mathcal{ALC}}) \mid (C_{\mathcal{ALC}} \sqcap C_{\mathcal{ALC}}) \mid (\forall RS. C_{\mathcal{ALC}}) \mid (\exists RS. C_{\mathcal{ALC}})$$

<sup>3</sup> Let us write ‘ $\models \psi$ ’ instead of ‘ $\emptyset \models \psi$ ’ when knowledge base is empty.

where metavariables  $CS$  and  $RS$  represent any concept and role symbols, respectively. Semantics of  $\mathcal{ALC}$  is defined in the standard way in accordance with Definition 3.

Many description logics can be defined as extensions of  $\mathcal{ALC}$  by concept and/or role constructs. For example, the website [13] uses the following approach: for any collection of concept and/or role constructs  $C\&R$ , let  $\mathcal{ALC}(C\&R)$  be a ‘closure’ of  $\mathcal{ALC}$  that admits all concept and/or role constructs in  $C\&R$ . Formal definitions follows.

**Definition 7.** A variant of DL is a description logic  $\mathcal{L}$  with syntax that

- contains all concept and role symbols  $CS$  and  $RS$ ,
- is closed under concept constructs ‘ $\neg$ ’, ‘ $\sqcup$ ’, ‘ $\sqcap$ ’, ‘ $\forall$ ’ and ‘ $\exists$ ’.

From the viewpoint of the above definition,  $\mathcal{ALC}$  is the smallest variant<sup>4</sup> of DL.

**Definition 8.** Let  $\mathcal{L}$  be a variant of DL and  $C\&R$  be a collection of concept and/or role constructs. Then let  $\mathcal{L}(C\&R)$  be the smallest variant of DL that includes  $\mathcal{L}$  and is closed under all constructs in  $C\&R$ .

For instance,  $\mathcal{ALC}(\neg, -)$  is an extension of  $\mathcal{ALC}$  where any role symbol can be negated and/or inverted. This variant of DL has decidable satisfiability problem [11, 7].

## 2 Integrating FCA operations to DL

Basic Formal Concept Analysis (FCA) definitions below follow monograph [3].

**Definition 9.** A formal context is a triple  $(O, A, B)$  where  $O$  and  $A$  are sets of ‘objects’ and ‘attributes’ respectively, and  $B \subseteq O \times A$  is a binary relation connecting objects and attributes. Let us say that a formal context  $(O, A, B)$  is homogeneous<sup>5</sup> iff  $O = A$ , i.e. the set of objects coincide with the set of attributes.

For example, for every terminological interpretation  $(D, I)$  and every role  $r$  one can define a formal context  $(D, D, I(r))$ . It implies that every terminological interpretation  $(D, I)$  defines a family of homogeneous formal contexts  $(D, D, I(R))$  indexed by role symbols  $R \in RS$  or by role terms  $R \in RT$ .

Vise verse, there is a number of ways how to define a terminological interpretation for given formal contexts. For example, if we have a family of formal contexts  $(O_j, A_j, B_j)$  indexed by elements of some set  $J$ , then we can adopt the set of indices  $J$  as the alphabet role symbols  $RS$ , a set of symbols  $\{o_j, a_j : j \in J\}$  as the alphabet of concept symbols  $CS$ , and define a terminological interpretation  $(D, I)$  where

<sup>4</sup> Of course, ‘smaller’ description logics can be defined and examined by means of stronger syntax restrictions.

<sup>5</sup> ‘Homogeneous’ is our own non-standard FCA term.

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
01: Fun95							x																			
02: God93																										
03: God95		x																								
04: God98		x	x				x																			
05: Huc99		x																								
06: Huc02																	x			x						
07: Kro94																										
08: Kui00											x		x	x	x	x			x							x
09: Leb99		x																								
10: Lin95							x																			
11: Lin97		x					x							x												
12: Sah97			x									x		x												
13: Sif97											x	x		x												
14: Sne96							x				x															
15: Sne98C	x						x			x		x		x	x											
16: Sne98R		x					x				x		x	x	x											
17: Sne99	x	x		x			x			x		x	x	x	x	x										
18: Sne00S							x				x		x	x					x	x						x
19: Sne00U		x		x			x				x			x	x	x										
20: Str99														x	x	x										
21: Til03S											x								x							
22: Til03T																										
23: Ton99							x				x		x	x												
24: Tone01							x				x		x	x						x						
25: Van98											x		x				x	x								

Table 1. Context Citations

- $D = \cup_{j \in J} (O_j \cup A_j)$ ,
- $I(j) = B_j \subseteq (O_j \times A_j) \subseteq D \times D$  for every  $j \in J$ ,
- $I(a_j) = A_j \subseteq D$  and  $I(o_j) = O_j \subseteq D$  for every  $j \in J$ .

Table 1 represents an example of a homogeneous formal context *Citations* from [10]. It represents citations between papers in some collection. In this particular case the set of objects (rows) and the set of attributes (columns) are both equal to [1..25].

Two basic algebraic operations for formal contexts are upper and lower derivations. These operations are used in the definition of a notion of a formal concept, its extent and intent.

**Definition 10.** Let  $(O, A, B)$  be a formal context. For every set of objects  $X \subseteq O$  its upper derivation  $X^\uparrow$  is the following set of attributes

$$\{t \in A : \text{for every } s \in O, \text{ if } s \in X \text{ then } (s, t) \in B\},$$

i.e. the collection of all attributes that are satisfied by all objects in  $X$  simultaneously. For every set of attributes  $Y \subseteq A$  its lower derivation  $Y^\downarrow$  is the following set of objects

$$\{s \in O : \text{for every } t \in A, \text{ if } t \in Y \text{ then } (s, t) \in B\},$$

i.e. the collection of all objects that satisfy to all objects in  $Y$  simultaneously. A formal concept is a pair  $(Ex, In)$  such that  $Ex \subseteq O$ ,  $In \subseteq A$ , and  $Ex^\uparrow = In$ ,  $In^\downarrow = Ex$ ; components  $Ex$  and  $In$  of the formal concept  $(Ex, In)$  are called its extent and its intent respectively.

For example,  $\{3, 4\}^\uparrow = \{2\}$  and  $\{2\}^\downarrow = \{3, 4, 5, 9, 16, 17, 19\}$  in the context *Citations*. Pair  $(\{1, 4, 10, 11, 14, 15, 16, 17, 18, 19, 23, 24\}, \{7\})$  is an example of a formal concept in the formal context *Citations*.

**Definition 11.** Let  $\mathcal{L}$  be a variant of DL. Then let  $\mathcal{L}/FCA$  be a variant of DL that is the closure of  $\mathcal{L}$  with respect to two new formula constructors for the upper and lower derivatives. Syntax of these two constructs is as follows: for every role term  $R$  and every concept term  $X$  let  $(X^{\uparrow R})$  and  $(X^{\downarrow R})$  be concept terms too. They are read as ‘upper derivative of  $X$  with respect to  $R$ ’ and, respectively, as ‘lower derivative of  $X$  with respect to  $R$ ’. For every terminological interpretation  $(D, I)$ ,

- $I(X^{\uparrow R}) = \{t : \text{for every } s \in D, \text{ if } s \in I(X) \text{ then } (s, t) \in I(R)\}$ , i.e. the upper derivation of  $I(X)$  in a homogenous formal context  $(D, D, I(R))$ ;
- $I(X^{\downarrow R}) = \{s : \text{for every } t \in D, \text{ if } t \in I(X) \text{ then } (s, t) \in I(R)\}$ , i.e. the lower derivation of  $I(X)$  in a homogenous formal context  $(D, D, I(R))$ .

In particular,  $\mathcal{ALC}/FCA$  is an extension of  $\mathcal{ALC}$  where both derivative constructors are allowed.

**Proposition 1.**

1. Let  $\mathcal{L}$  be a variant of DL. For every role and concept terms within  $\mathcal{L}$  the following concepts of  $\mathcal{L}/FCA$  and  $\mathcal{L}(\neg, -)$  are equivalent (i.e. have equal semantics in every terminological interpretation):
  - (a)  $X^{\uparrow R}$  and  $\forall \neg R^-. \neg X$ ,
  - (b)  $X^{\downarrow R}$  and  $\forall \neg R. \neg X$ .
2.  $\mathcal{L}/FCA$  can be expressed in  $\mathcal{L}(\neg, -)$  with linear complexity, i.e. every concept  $X$  in  $\mathcal{L}/FCA$  is equivalent to some concept  $Y$  in  $\mathcal{L}(\neg, -)$  that can be constructed in linear time.

**Proposition 2.** For every terminological interpretation  $(D, I)$ , every concept terms  $X$  and  $Y$  of DL, and every role term  $R$  of DL the following holds:

a pair of sets  $(I(X), I(Y))$  is a formal concept  
in the homogeneous formal context  $(D, D, I(R))$



$(D, I)$  is a terminological model  
for the following two terminological sentences  
 $X^{\uparrow R} \doteq Y$  and  $Y^{\downarrow R} \doteq X$ .

Proposition 2 makes sense to the following definition.

**Definition 12.**

For any concept terms  $X$  and  $Y$ , for any role term  $R$ , let  $(X, Y)FC(R)$  be a ‘terminological sentence’ that is a shorthand (a notation or an abbreviation) for the following pair of standard terminological sentences  $X^{\uparrow R} \doteq Y$  and  $Y^{\downarrow R} \doteq X$ . This notation is read as ‘ $(X, Y)$  is a Formal Concept with respect to  $R$ ’.

The above propositions and the decidability of the satisfiability problem for  $\mathcal{ALC}(\neg, -)$  [7] together imply the next corollary.

**Corollary 1.** The satisfiability problem for  $\mathcal{ALC}/FCA$  (including terminological sentences for formal concepts) is decidable.

### 3 Concluding Remarks

The primary target of our research was to make explicit relations between two formalisms for reasoning about concepts. The first formalism, Formal Concept Analysis (FCA), is of algebraic nature. The second one, Description Logic (DL), is of logical nature. We have demonstrated in the present paper that FCA can be ‘absorbed’ by DL at least from viewpoint of ‘abstract’ expressive power. It implies that any collection set-theoretic (in)equalities written in terms of uninterpreted symbols for individual objects and attributes, for sets of objects and attributes, for formal contexts and concepts, with aid of set-theoretic operations, FCA operations for upper and lower derivative, intent and extent operations, can be easily translated to a description logic knowledge base, so that the base is satisfiable iff there is a formal context where all these (in)equalities realize simultaneously. Since the satisfiability problem is decidable for many Description Logics, the realization problem for collections of (in)equalities of this kind can be done (as a rule) automatically (i.e. by some algorithm).

At the same time in the present paper we give a partial answer to a question from [12], whether a variant of Propositional Dynamic Logic (PDL) extended by upper and lower derivations for atomic programs is decidable. PDL [5,6] has been introduced by M.J. Fischer and R.E. Ladner as an extension of the classical propositional logic and propositional modal logic K for reasoning about partial correctness of structured nondeterministic programs. Many variants of PDL have been studied extensively especially from the viewpoint of decidability and axiomatizability. In particular, recently C. Lutz and D. Walther [8] have proved that PDL with complement of atomic programs is decidable in exponential time (while it is well known that in general case PDL with complement is undecidable).

Paper [12] has introduced and studied PDL/FCA – a variant of PDL extended by modalities inspired by Formal Concept Analyses (FCA). Formal semantics of these modalities is upper and, respectively, lower derivations. (Please refer [12] for discussion on utility of these modalities for program specification and verification.)

Paper [12] has proved that PDL/FCA is more expressive than PDL, and has interpreted a fragment of PDL/FCA without upper derivation in PDL with complement. It implies decidability of PDL extended by extent of atomic programs with exponential upper bound (since PDL with complement of atomic programs is decidable [8]). It remains an open question whether PDL/FCA (without any restriction for upper and lower derivations) is decidable and what is the expressive power of this logic (with respect to PDL with complement). But now (due to Corollary 1) we can claim that a fragment of PDL/FCA with atomic programs is decidable (since this fragment is equal to  $\mathcal{ALC}/FCA$ ).

There is a number of research papers on combination of Formal Concept Analysis with Description Logic for better knowledge processing<sup>6</sup>. But there are

<sup>6</sup> For instance please refer a recent paper [1], but a survey of this topic is out of scope of the present paper.

few papers on comparison and integration of both formalism in one. We can point just a single one [10] related to this topic. The cited paper has attempted to develop in the framework of FCA an algebraic operation inspired by the universal restriction of DL and to demonstrate an utility of it for analysis of relational data. In accordance with [10], the attempt has resulted in a so-called ‘Relational Concept Analysis’ that had been implemented in an open platform Galicia for lattices.

**Acknowledgement:** We would like to thank Prof. Karl Erich Wolff for fruitful discussions of the research and for the comments on the draft of this paper. We also would like to thanks an anonymous referee for pointing a paper [4], where a construct corresponding to the lower derivation operator has been considered before our research in a framework of description logics.

## References

1. Baader F., Ganter B., Sattler U., and Sertkaya B. *Completing Description Logic Knowledge Bases using Formal Concept Analysis*. In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07). AAAI Press, 2007.
2. Baader F., Calvanese D., Nardi D., McGuinness D., and Patel-Schneider P., editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
3. Ganter B., Wille R. *Formal Concept Analysis*. Mathematical Foundations. Springer Verlag, 1996.
4. Givan R., McAllester D., and Shalaby S. *Natural language based inference procedures applied to schubert’s steamroller*. In AAAI-91, p. 915–920. Morgan Kaufmann Publishers.
5. Fischer M.J. and Ladner R.E. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194-211, 1979.
6. Harel D., Kozen D., Tiuryn J. *Dynamic Logic (Foundations of Computing)*. MIT Press, 2000.
7. Hustadt U, Schmidt R.A., and Georgieva L. *A Survey of decidable first-order fragments and description logics*. *J. Relational Methods in Computer Science*, Vol. 1, 2004, pp. 251-276.
8. Lutz C. and Walther D. PDL with Negation of Atomic Programs. *Journal of Applied Non-Classical Logic*, 15(2):189-214, 2005.
9. Patel-Schneider P. F. *Decidable, Logic-Based Knowledge Representation*. PhD thesis, Univ. Toronto, 1987.
10. Rouane A.H., Huchard M., Napoli A., Valtchev P. *A proposal for combining formal concept analysis and description logics for mining relational data*. *Lecture Notes in Artificial Intelligence*, Vol. 4390, 2007, pp. 51-65.
11. Schmidt-Schauß M., and Smolka G. *Attributive concept descriptions with complements*. *J. Artificial Intelligence*, Vol. 48, 1991, pp. 1-26.
12. Shilov N.V., Garanina N.O., Anureev I.S. *Combining Propositional Dynamic Logic with Formal Concept Analysis*. *Concurrency, Specification and Programming CS&P’2006*. Volume 2: Specification. Humboldt-Universität zu Berlin, Informatik-Bericht Nr.206, 2006, p.152-161.
13. Zolin E. Complexity of reasoning in Description Logics. Web-resource available at URL <http://www.cs.man.ac.uk/~ezolin/logic/complexity.html>.

## DLMedia: an Ontology Mediated Multimedia Information Retrieval System

Umberto Straccia and Giulio Visco

ISTI-CNR  
Pisa, ITALY,  
straccia@isti.cnr.it

**Abstract.** We outline DLMedia, an ontology mediated multimedia information retrieval system, which combines logic-based retrieval with multimedia feature-based similarity retrieval. An ontology layer may be used to define (in terms of a DLR-Lite like description logic) the relevant abstract concepts and relations of the application domain, while a content-based multimedia retrieval system is used for feature-based retrieval.

### 1 Introduction

*Multimedia Information Retrieval* (MIR) concerns the retrieval of those multimedia objects of a collection that are relevant to a user information need.

Here we outline *DLMedia*, an ontology mediated Multimedia Information Retrieval (MIR) system, which combines logic-based retrieval with multimedia feature-based similarity retrieval. An ontology layer may be used to define (in terms of a DLR-Lite like description logic) the relevant abstract concepts and relations of the application domain, while a content-based multimedia retrieval system is used for feature-based retrieval.

### 2 The Logic-based MIR Model in DLMedia

Overall, DLMedia follows the *Logic-based Multimedia Information Retrieval* (LMIR) model described in [9] (see [9] for an overview on LMIR literature. A recent work is also *e.g.* [6]). Let us first roughly present (parts of) the LMIR model of [9]. In doing this, we rely on Figure 1. The model has two layers addressing the multidimensional aspect of multimedia objects  $o \in \mathbb{O}$  (*e.g.* objects  $o_1$  and  $o_2$  in Figure 1): that is, their *form* and their *semantics* (or *meaning*). The form of a multimedia object is a collective name for all its *media dependent*, typically automatically extracted features, like text index term weights (object of type text), colour distribution, shape, texture, spatial relationships (object of type image), mosaiced video-frame sequences and time relationships (object of type video). On the other hand, the semantics (or meaning) of a multimedia object is a collective name for those features that pertain to the slice of the real world being *represented*, which exists independently of the existence of a object referring to it. Unlike form, the semantics of a multimedia object is thus *media independent* (typically, constructed manually perhaps with the assistance of some automatic tool). Therefore, we have two layers, the *object form layer* and the *object semantics layer*. The former represents media dependent features of the objects, while the latter describes the semantic

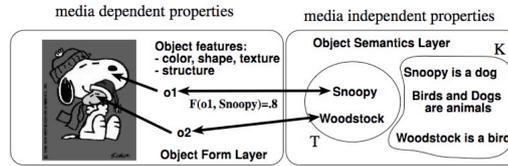


Fig. 1. LMIR model layers and objects

properties of the slice of world the objects are about. The semantic entities (e.g., Snoopy, Woodstock), which objects can be about are called *semantic index terms* ( $t \in \mathbb{T}$ ). The mapping of objects  $o \in \mathbb{O}$  to semantic entities  $t \in \mathbb{T}$  (e.g., “object  $o_1$  is about Snoopy”) is called *semantic annotation*. According to the fuzzy information retrieval model (e.g. [2]), semantic annotation can be formalized as a membership function  $F: \mathbb{O} \times \mathbb{T} \rightarrow [0, 1]$  describing the *correlation* between multimedia objects and semantic index terms. The value  $F(o, t)$  indicates to which degree the multimedia object  $o$  deals with the semantic index term  $t$ . Depending on the context, the function  $F$  may be computed automatically (e.g., for text we may have [4], for images we may have an automated image annotation (classification) tool, as e.g. [5]).

Corresponding to the two dimensions of a document just introduced, there are three categories of retrieval: one for each dimension (*form-based retrieval* and *semantics-based retrieval*) and one concerning the combination of both of them. The retrieval of information based on form addresses, of course, the syntactical properties of documents. For instance, form-based retrieval methods automatically create the document representations to be used in retrieval by extracting low-level features from documents, such as the number of occurrences of words in text, or color distributions in images. To the contrary, semantics-based retrieval methods rely on a symbolic representation of the meaning of documents, that is descriptions formulated in some suitable formal language. Typically, meaning representations are constructed manually, perhaps with the assistance of some automatic tool.

A data model for MIR not only needs both dimensions to be taken into account, but also requires that each of them be tackled by means of the tools most appropriate to it, and that these two sets of tools be integrated in a principled way. DLMedia’s data model is based on *logic* in the sense that retrieval can be defined in terms of logical entailment as defined in the next section.

### 3 The DLMedia query and representation language

For computational reasons the particular logic DLMedia adopt is based on a DLR-Lite [3] like Description Logic (DL) [1]. The DL will be used in order to both define the relevant abstract concepts and relations of the application domain, as well as to describe the information need of a user.

Our DL is enriched with build-in predicates allowing to address all three categories of retrieval (form-based, semantic-based and their combination). To support query answering, the DLMedia system has a DLR-Lite like reasoning component and a (feature-based)

multimedia retrieval component. In the latter case, we rely on our multimedia retrieval system MILOS<sup>1</sup>.

In order to support reasoning about form and content, DLMedia provides a logical query and representation language, which closely resembles a fuzzy variant DLR-Lite [3, 12, 11] with fuzzy concrete domains [10].

The concrete predicates that we allow are not only relational predicates such as ( $x \leq 1500$ ) (e.g.  $x$  is less or equal than 1500), but also similarity predicates such as ( $x \text{ simTxt } \textit{logic, image, retrieval}$ ), which given a piece of text  $x$  returns the system’s degree (in  $[0, 1]$ ) of being  $x$  about the keywords ‘logic, image, retrieval’.

A *fuzzy concrete domain* (or simply *fuzzy domain*) is a pair  $\langle \Delta_D, \Phi_D \rangle$ , where  $\Delta_D$  is an interpretation domain and  $\Phi_D$  is the set of *fuzzy domain predicates*  $d$  with a predefined arity  $n$  and an interpretation  $d^D: \Delta_D^n \rightarrow [0, 1]$ . An *axiom* is of the form ( $m \geq 1$ )

$$Rl_1 \sqcap \dots \sqcap Rl_m \sqsubseteq Rr,$$

where  $Rl$  is a so-called *left-hand relation* and  $Rr$  is a *right-hand relation* with following syntax ( $l \geq 1$ ):

$$\begin{aligned} Rr &\longrightarrow A \mid \exists[i_1, \dots, i_k]R \\ Rl &\longrightarrow A \mid \exists[i_1, \dots, i_k]R \mid \exists[i_1, \dots, i_k]R.(Cond_1 \sqcap \dots \sqcap Cond_l) \\ Cond &\longrightarrow ([i] \leq v) \mid ([i] < v) \mid ([i] \geq v) \mid ([i] > v) \mid ([i] = v) \mid ([i] \neq v) \mid \\ &\quad ([i] \text{ simTxt } k_1, \dots, k_n) \mid ([i] \text{ simImg URN}) \end{aligned}$$

where  $A$  is an atomic concept,  $R$  is an  $n$ -ary relation with  $1 \leq i_1, i_2, \dots, i_k \leq n$ ,  $1 \leq i \leq n$  and  $v$  is a value of the concrete interpretation domain of the appropriate type. Informally,  $\exists[i_1, \dots, i_k]R$  is the projection of the relation  $R$  on the columns  $i_1, \dots, i_k$  (the order of the indexes matters). Hence,  $\exists[i_1, \dots, i_k]R$  has arity  $k$ .  $\exists[i_1, \dots, i_k]R.(Cond_1 \sqcap \dots \sqcap Cond_l)$  further restricts the projection  $\exists[i_1, \dots, i_k]R$  according to the conditions specified in  $Cond_i$ . For instance,  $([i] \leq v)$  specifies that the values of the  $i$ -th column have to be less or equal than the value  $v$ ,  $([i] \text{ simTxt } k_1 \dots k_n)$  evaluates the degree of being the text of the  $i$ -th column similar to the list of keywords  $k_1 \dots k_n$ , while  $([i] \text{ simImg URN})$  returns the system’s degree of being the image identified by the  $i$ -th column similar to the object  $o$  identified by the URN (*Uniform Resource Name*<sup>2</sup>). We further assume that all  $Rl_i$  and  $Rr$  in  $Rl_1 \sqcap \dots \sqcap Rl_m \sqsubseteq Rr$  have the same arity. For instance assume we have a relation  $Person(\textit{name, age, father\_name, mother\_name, gender})$  then the following are axioms:

$$\begin{aligned} \exists[1, 2]Person &\sqsubseteq \exists[1, 2]hasAge \\ &\quad // \text{ constrains relation } hasAge(\textit{name, age}) \\ \exists[3, 1]Person &\sqsubseteq \exists[1, 2]hasChild \\ &\quad // \text{ constrains relation } hasChild(\textit{father\_name, name}) \\ \exists[4, 1]Person &\sqsubseteq \exists[1, 2]hasChild \\ &\quad // \text{ constrains relation } hasChild(\textit{mother\_name, name}) \\ \exists[3, 1]Person. &(( [2] \geq 18) \sqcap ([5] = \textit{female}')) \sqsubseteq \exists[1, 2]hasAdultDaughter \\ &\quad // \text{ constrains relation } hasAdultDaughter(\textit{father\_name, name}) \end{aligned}$$

Note that in the last axiom, we require that the age is greater or equal than 18 and the gender is female. On the other hand examples axioms involving similarity predicates are,

$$\exists[1]ImageDescr.([2] \text{ simImg urn1}) \sqsubseteq Child \tag{1}$$

<sup>1</sup> <http://milos.isti.cnr.it/>

<sup>2</sup> [http://en.wikipedia.org/wiki/Uniform\\_Resource\\_Name](http://en.wikipedia.org/wiki/Uniform_Resource_Name)

$$\exists[1]Title.([2] simText'lion') \sqsubseteq Lion \quad (2)$$

where *urn1* identifies the image in Figure 2. The former axiom (axiom 1) assumes that we have an *ImageDescr* relation, whose first column is the application specific image identifier and the second column contains the image URN. Then, this axiom (informally) states that an image similar to the image depicted in Figure 2 is about a *Child* (to a system computed degree in  $[0, 1]$ ).

Similarly, in axiom (2) we assume that an image is annotated with a metadata format, e.g. MPEG-7, the attribute *Title* is seen as a binary relation, whose first column is the identifier of the metadata record, and the second column contains the title (piece of text) of the annotated image. Then, this axiom (informally) states that an image whose metadata record contains an attribute *Title* which is about 'lion' is about a *Lion*. The following example



**Fig. 2.** Service Model

$$\begin{aligned} \exists[1]F &\sqsubseteq MultiMediaObject \\ \exists[2]F &\sqsubseteq SemanticIndexTerm \\ \exists[1,2]F &\sqsubseteq \exists[1,2]IsAbout \end{aligned}$$

gives some constraints on the semantic annotation function  $F$ .

From a semantics point of view, DLMedia is based on fuzzy logic, both because the the LMIR annotation model it is based on the fuzzy information retrieval model, as well as each instance of atoms and relations may have a score, and, thus we have to define how these scores are *combined* using the logical connectives of the language.

Given a fuzzy concrete domain  $\langle \Delta_D, \Phi_D \rangle$ , an *interpretation*  $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$  consists of a *fixed infinite domain*  $\Delta$ , containing  $\Delta_D$ , and an *interpretation function*  $\cdot^{\mathcal{I}}$  that maps every atom  $A$  to a function  $A^{\mathcal{I}}: \Delta \rightarrow [0, 1]$  and maps an  $n$ -ary predicate  $R$  to a function  $R^{\mathcal{I}}: \Delta^n \rightarrow [0, 1]$  and constants to elements of  $\Delta$  such that  $a^{\mathcal{I}} \neq b^{\mathcal{I}}$  if  $a \neq b$  (unique name assumption). We assume to have one object for each constant, denoting exactly that object. In other words, we have standard names, and we do not distinguish between the alphabet of constants and the objects in  $\Delta$ . Furthermore, we assume that the relations have a typed signature and the interpretations have to agree on the relation's type. For instance, the second argument of the *Title* relation (see axiom 2) is of type *String* and any interpretation function requires that the second argument of  $Title^{\mathcal{I}}$  is of type *String*. To the easy of presentation, we omit the formalization of this aspect and leave it at the intuitive level. In the following, we use  $\mathbf{c}$  to denote an  $n$ -tuple of constants, and  $\mathbf{c}[i_1, \dots, i_k]$  to denote the  $i_1, \dots, i_k$ -th components of  $\mathbf{c}$ . For instance,  $(a, b, c, d)[3, 1, 4]$  is  $(c, a, d)$ . Let  $t$  be a so-called T-norm, which is a function used to combine the truth of “conjunctive” expressions.<sup>3</sup> Then,  $\cdot^{\mathcal{I}}$  has to satisfy, for all  $\mathbf{c} \in \Delta^k$  and  $n$ -ary relation  $R$ :

$$\begin{aligned} (\exists[i_1, \dots, i_k]R)^{\mathcal{I}}(\mathbf{c}) &= \sup_{\mathbf{c}' \in \Delta^n, \mathbf{c}'[i_1, \dots, i_k] = \mathbf{c}} R^{\mathcal{I}}(\mathbf{c}') \\ (\exists[i_1, \dots, i_k]R.(Cond_1 \sqcap \dots \sqcap Cond_i))^{\mathcal{I}}(\mathbf{c}) &= \\ &= \sup_{\mathbf{c}' \in \Delta^n, \mathbf{c}'[i_1, \dots, i_k] = \mathbf{c}} t(R^{\mathcal{I}}(\mathbf{c}'), Cond_1^{\mathcal{I}}(\mathbf{c}'), \dots, Cond_i^{\mathcal{I}}(\mathbf{c}')) \end{aligned}$$

<sup>3</sup>  $t$  has to be symmetric, associative, monotone in its arguments and  $t(x, 1) = x$ . Examples of t-norms are:  $\min(x, y)$ ,  $x \cdot y$ ,  $\max(x + y - 1, 0)$ .

with  $([i] \leq v)^{\mathcal{I}}(\mathbf{c}') = 1$  if  $\mathbf{c}'[i] \leq v$ , and  $([i] \leq v)^{\mathcal{I}}(\mathbf{c}') = 0$  otherwise (and similarly for the other comparison operators), while

$$\begin{aligned} ([i] \text{ simTtxt}'k_1, \dots, k'_n)^{\mathcal{I}}(\mathbf{c}') &= \text{simTtxt}^{\mathcal{D}}(\mathbf{c}'[i], k_1, \dots, k'_n) \in [0, 1] \\ ([i] \text{ simImgURN})^{\mathcal{I}}(\mathbf{c}') &= \text{simImg}^{\mathcal{D}}(\mathbf{c}'[i], \text{URN}) \in [0, 1]. \end{aligned}$$

It is pretty clear that many other concrete predicates can be added as well.

Then,  $\mathcal{I} \models Rl_1 \sqcap \dots \sqcap Rl_m \sqsubseteq Rr$  iff for all  $\mathbf{c} \in \Delta^n$ ,  $t(Rl_1^{\mathcal{I}}(\mathbf{c}), \dots, Rl_m^{\mathcal{I}}(\mathbf{c})) \leq Rr^{\mathcal{I}}(\mathbf{c})$ , where we assume that the arity of  $Rr$  and all  $Rl_i$  is  $n$ .

Concerning queries, a *query* consists of a conjunctive query of the form

$$q(\mathbf{x}) \leftarrow R_1(\mathbf{z}_1) \wedge \dots \wedge R_l(\mathbf{z}_l),$$

where  $q$  is an  $n$ -ary predicate, every  $R_i$  is an  $n_i$ -ary predicate,  $\mathbf{x}$  is a vector of variables, and every  $\mathbf{z}_i$  is a vector of constants, or variables. We call  $q(\mathbf{x})$  its *head* and  $R_1(\mathbf{z}_1) \wedge \dots \wedge R_l(\mathbf{z}_l)$  its *body*.  $R_i(\mathbf{z}_i)$  may also be a concrete unary predicate of the form  $(z \leq v)$ ,  $(z < v)$ ,  $(z \geq v)$ ,  $(z > v)$ ,  $(z = v)$ ,  $(z \neq v)$ ,  $(z \text{ simTtxt}'k_1, \dots, k'_n)$ ,  $(z \text{ simImgURN})$ , where  $z$  is a variable,  $v$  is a value of the appropriate concrete domain,  $k_i$  is a keyword and  $\text{URN}$  is an URN. Example queries are:

```
q(x) ← Child(x)
      // find objects about a child (strictly speaking, find instances of Child)

q(x) ← CreatorName(x, y) ∧ (y = 'paolo'), Title(x, z), (z simTtxt'tour')
      // find images made by Paolo whose title is about 'tour'

q(x) ← ImageDescr(x, y) ∧ (y simImg urn2)
      // find images similar to a given image identified by urn2

q(x) ← ImageObject(x) ∧ isAbout(x, y1) ∧ Car(y1) ∧ isAbout(x, y2) ∧ Racing(y2)
      // find image objects about cars racing
```

From a semantics point of view, an interpretation  $\mathcal{I}$  is a *model* of a rule  $r$  of form  $q(\mathbf{x}) \leftarrow \phi(\mathbf{x}, \mathbf{y})$ , where  $\phi(\mathbf{x}, \mathbf{y})$  is  $R_1(\mathbf{z}_1) \wedge \dots \wedge R_l(\mathbf{z}_l)$ , denoted  $\mathcal{I} \models r$ , iff for all  $\mathbf{c} \in \Delta^n$ :

$$q^{\mathcal{I}}(\mathbf{c}) \geq \sup_{\mathbf{c}' \in \Delta \times \dots \times \Delta} \phi^{\mathcal{I}}(\mathbf{c}, \mathbf{c}'),$$

where  $\phi^{\mathcal{I}}(\mathbf{c}, \mathbf{c}')$  is obtained from  $\phi(\mathbf{c}, \mathbf{c}')$  by replacing every  $R_i$  by  $R_i^{\mathcal{I}}$ , and the T-norm  $t$  is used to combine all the truth degrees  $R_i^{\mathcal{I}}(\mathbf{c}')$  in  $\phi^{\mathcal{I}}(\mathbf{c}, \mathbf{c}')$ .

Finally, in DL-Media, we may also have so-called set of facts, *i.e.* a finite set of instances of relations, *i.e.* a set of expressions of the form

$$\langle R(c_1, \dots, c_n), s \rangle,$$

where  $R$  is an  $n$ -ary predicate, every  $c_i$  is a constant and  $s$  is the degree of truth (score) of the fact. If  $s$  is omitted, as *e.g.* in traditional databases, then the truth degree 1 is assumed.  $\mathcal{I} \models \langle R(c_1, \dots, c_n), s \rangle$  iff  $R^{\mathcal{I}}(c_1, \dots, c_n) \geq s$ .

For instance, related to Figure 1, we may have the facts

$$\begin{aligned} \langle F(o1, \text{snoopy}), 0.8 \rangle \quad \langle F(o2, \text{woodstock}), 0.6 \rangle \\ \text{Dog}(\text{snoopy}) \quad \text{Bird}(\text{woodstock}). \end{aligned}$$

A DLMedia *multimedia base*  $\mathcal{K} = \langle \mathcal{F}, \mathcal{O} \rangle$  consists of a *facts component*  $\mathcal{F}$ , and a *axioms component*  $\mathcal{O}$ .  $\mathcal{I} \models \mathcal{K}$  iff  $\mathcal{I}$  is a model of each component of  $\mathcal{K}$ . We say  $\mathcal{K}$  *entails*  $R(\mathbf{c})$  to degree  $s$ , denoted  $\mathcal{K} \models \langle R(\mathbf{c}), s \rangle$ , iff for each model  $\mathcal{I}$  of  $\mathcal{K}$ , it is true that  $R^{\mathcal{I}}(\mathbf{c}) \geq s$ . The *greatest lower bound* of  $R(\mathbf{c})$  relative to  $\mathcal{K}$  is  $glb(\mathcal{K}, R(\mathbf{c})) = \sup\{s \mid \mathcal{K} \models \langle R(\mathbf{c}), s \rangle\}$ .

The basic inference problem that is of interest in DLMedia is the top- $k$  retrieval problem, formulated as follows. Given a multimedia base  $\mathcal{K}$  and a query with head  $q(\mathbf{x})$ , retrieve  $k$  tuples  $\langle \mathbf{c}, s \rangle$  that instantiate the query predicate  $q$  with maximal score, and rank them in decreasing order relative to the score  $s$ , denoted

$$ans_k(\mathcal{K}, q) = \text{Top}_k\{\langle \mathbf{c}, s \rangle \mid s = glb(\mathcal{K}, q(\mathbf{c}))\}.$$

From a reasoning point of view, the DLMedia system extends the DL-Lite/DLR-Lite reasoning method [3] to the fuzzy case. The algorithm is a straightforward extension of the one described in [12, 11]). Roughly, given a query  $q(\mathbf{x}) \leftarrow R_1(\mathbf{z}_1) \wedge \dots \wedge R_l(\mathbf{z}_l)$ ,

1. by considering  $\mathcal{O}$  only, the user query  $q$  is *reformulated* into a set of conjunctive queries  $r(q, \mathcal{O})$ . Informally, the basic idea is that the reformulation procedure closely resembles a top-down resolution procedure for logic programming, where each axiom is seen as a logic programming rule. For instance, given the query  $q(x) \leftarrow A(x)$  and suppose that  $\mathcal{O}$  contains the axioms  $B_1 \sqsubseteq A$  and  $B_2 \sqsubseteq A$ , then we can reformulate the query into two queries  $q(x) \leftarrow B_1(x)$  and  $q(x) \leftarrow B_2(x)$ , exactly as it happens for top-down resolution methods in logic programming;
2. the reformulated queries in  $r(q, \mathcal{O})$  are *evaluated* over  $\mathcal{F}$  only (which is solved by accessing a top- $k$  database engine [7] and a multimedia retrieval system), producing the requested top- $k$  answer set  $ans_k(\mathcal{K}, q)$  by applying the *Disjunctive Threshold Algorithm* (DTA, see [12] for the details). For instance, for the previous query, the answers will be the top- $k$  answers of the union of the answers produced by all three queries.

#### 4 DLMedia at work

A preliminary prototype of the DLMedia system has been implemented. The architecture is pretty similar to the QuOnto system<sup>4</sup>. The main interface is shown in Figure 3.

In the upper pane, the currently loaded ontology component  $\mathcal{O}$  is shown. Below it and to the right, the current query is shown (“find a child”, we also do not report here the concrete syntax of the DLMedia DL).

So far, in DLMedia, given a query, it will be transformed, using the ontology, into several queries (according to the query reformulation step described above) and then the conjunctive queries are transformed into appropriate queries (this component is called wrapper) in order to be submitted to the underlying database and multimedia engine. To support the query rewriting phase, DLMedia allows also to write *schema mapping* rules, which map *e.g.* a relation name  $R$  into the concrete name of a relational table of the underlying database. The currently supported wrappers are for (of course other wrappers can be plugged in as well.)

- the relational database system Postgres;<sup>5</sup>

<sup>4</sup> <http://www.dis.uniroma1.it/~quonto/>.

<sup>5</sup> <http://www.postgresql.org/>

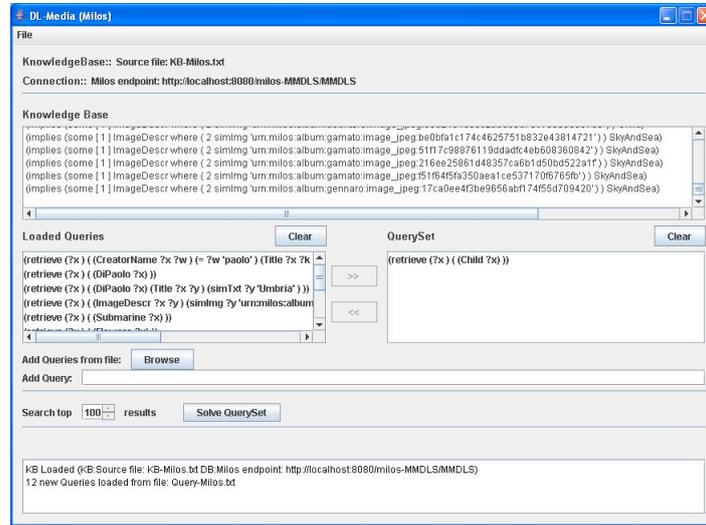


Fig. 3. DLMedia main interface.

- the relational database system with text similarity MySQL;<sup>6</sup> and
- our multimedia retrieval system Milos, which supports XML data.

For instance, the execution of the toy query shown in Figure 3 (“find a child”) produces the ranked list of images shown in Figure 4.

## 5 Conclusions

In this work, we have outlined the DLMedia system, *i.e.* an ontology mediated multimedia retrieval system. Main features (so far) of DLMedia are that: (i) it uses a DLR-Lite(D) like language as query and ontology representation language; (ii) it supports queries about the form and content of multimedia data; and (iii) is scalable -though we did not address it here, query answering in DLMedia is LogSpace-complete in data complexity. The data complexity of DLMedia directly depends by the data complexity of the underlying database and multimedia retrieval engines.

## References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
2. G. Bordogna, P Carrara, and G. Pasi. Query term weights as constraints in fuzzy information retrieval. *Information Processing and Management*, 27(1):15–26, 1991.

<sup>6</sup> <http://www.mysql.org/>

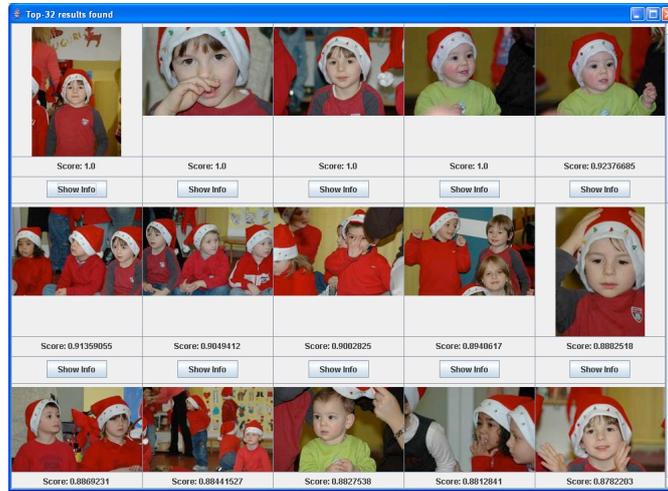


Fig. 4. DLMedia results pane.

3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 260–270, 2006.
4. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *The 12th Int. World Wide Web Conference*, pages 178–186, 2003.
5. Th. Gevers and A.W.M. Smeulders. Content-based image retrieval: An overview. In *Emerging Topics in Computer Vision*. Prentice Hall, 2004.
6. S. Hammiche, S. Benbernou, and A. Vakali. A logic based approach for the multimedia data representation and retrieval. In *7th IEEE Int. Symp. on Multimedia*, pages 241–248. IEEE Computer Society, 2005.
7. C. Li, K. C. C. Chang, I. F. Ilyas, and S. Song. RankSQL: query algebra and optimization for relational top-k queries. In *Proc. of the 2005 ACM SIGMOD Int. Conf. on Management of Data*, pages 131–142, New York, NY, USA, 2005. ACM Press.
8. C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*. King’s College Publications, 2003.
9. C. Meghini, F. Sebastiani, and U. Straccia. A model of multimedia information retrieval. *Journal of the ACM*, 48(5):909–970, 2001.
10. U. Straccia. Description logics with fuzzy concrete domains. In *21st Conf. on Uncertainty in Artificial Intelligence*, pages 559–567, 2005. AUAI Press.
11. U. Straccia. Answering vague queries in fuzzy dl-lite. In *Proc. of the 11th Int. Con. on Information Processing and Management of Uncertainty in Knowledge-Based Systems, (IPMU-06)*, pages 2238–2245. E.D.K., Paris, 2006.
12. U. Straccia. Towards top-k query answering in description logics: the case of DL-Lite. In *Proc. of the 10th European Conf. on Logics in Artificial Intelligence*, pages 439–451, 2006. Springer Verlag.

## Approximate Subsumption in $\mathcal{ALCQ}$

Heiner Stuckenschmidt

University of Mannheim, Germany  
 A5, 6 68159 Mannheim  
 heiner@informatik.uni-mannheim.de

**Abstract.** We present a non-standard interpretation for concept expressions in  $\mathcal{ALCQ}$  that defines approximate notions of subsumption based on approximating a subset of the concept and role names. We present the non-standard semantics, and the corresponding notion of approximate subsumption, discuss its formal properties and show that it can be computed by syntactic manipulations of concept expressions.

### 1 Introduction

Description Logics are becoming more and more popular as a formalism for representing and reasoning about conceptual knowledge in different areas such as databases and semantic web technologies. In particular, subsumption reasoning for expressive ontologies has been used to compute matches between conceptual descriptions in the context of different real world tasks including information integration, product and service matching and data retrieval. In practical situations, however, it often turns out that logical reasoning is inadequate in many cases, because it does not leave any room for *partial matches*.

Recently, there are some efforts that try to address this problem by combining description logics with numerical techniques for uncertain reasoning in OWL, in particular with techniques for probabilistic [1] and fuzzy reasoning [2]. These approaches are able to compute partial matches by assigning an assessment of the degree of matching to the subsumption relation. This degree of matching normally is a real number or an interval between zero and one and therefore allows some ordering of the solutions. Although, in principle this is a solution to the problem of computing the best partial match but defining an interpreting numerical assessments of uncertainty is a difficult problem. Further, the reduction to a single numerical assessment of the mismatch does not allow different users to discriminate between different kinds of mismatches.

In this paper, we propose a notion of approximate subsumption that supports the computation of partial matches between complex concept expressions without relying on a single number to represent the degree of mismatch. Instead, *the approach describes the degree of matching in terms of a subset of the aspects of the request that are met by the solution*. This approach allows the user to decide whether to accept a partial match based on whether important aspects are missed or not. In order to implement this approach we borrow from the area of approximate deduction. In particular, we extend the notion of S-Interpretations of propositional logic proposed in [3] to description logics

and use the result notion of a non-standard interpretation of concept expressions to define an approximate subsumption operator that computes subsumption with respect to a particular subset of the vocabulary used.

## 2 Approximation based on Sub-Vocabularies

In propositional logic, the vocabulary of a formula consists of a set of propositional letters. A formula consists of a Boolean expression over these letters. A classical interpretation  $I$  assigns to each letter either the value *true* or *false*. The semantics of negation now implies that a letter and its negation cannot have the same truth value, in particular, for all propositional letters  $p$  one of the following :

$$\begin{aligned} I(p \wedge \neg p) &= false \\ I(p \vee \neg p) &= true \end{aligned} \tag{1}$$

Checking satisfiability of a formula relies on showing that there is no assignment of truth values that satisfies this condition and makes the whole formula true. A possible way for approximating satisfiability testing for propositional logic is now to restrict the condition above to a subset of the propositional letters. This subset is denoted as  $S$  and the corresponding interpretation is called an S-interpretation of the formula [3].

Depending on how the letters not in  $S$  are treated, an S-Interpretation is sound or complete with respect to the classical interpretation. One kind of non-standard interpretation called S-3 Interpretation assigns both, a letter and its negation to *true*.

$$I(p \wedge \neg p) = true, p \notin S \tag{2}$$

When applying this interpretation to the satisfiability problem, we observe that formulas that were unsatisfiable before now become satisfiable. This means that the resulting calculus is sound, but incomplete, because some results that could be proven using the principle of proof by refutation can not be proven any more, because the conjunction of the knowledge base with the negation of the result to be proven becomes satisfiable under the new interpretation. The counterpart of S-3 interpretation are S-1 Interpretations that assign *false* to both a letters and their negation if the letters are not in the set  $S$ .

$$I(p \vee \neg p) = false, p \notin S \tag{3}$$

Following the same argument as above, S-1 Interpretations define a complete but unsound calculus for propositional logic. In both cases, the advantage of the approach is that we can decide which parts of the problem to approximate by selecting an appropriate set of letters  $S$ . Therefore the approach provides a potential solution to the problem of partial matching described above.

The idea of our approach is now to apply the underlying idea of S-Interpretations to the Description Logic  $\mathcal{ALCQ}$  which covers most of the expressive power of OWL in order to support approximate subsumption reasoning where parts of the vocabulary are interpreted in the classical way and other parts are approximated. In fact, Cadoli and Schaerf do propose an extension of S-Interpretations to Description logics, but they define  $S$  not in terms of a subset of the vocabulary, but in terms of the structure of the concept expression [4]. In [5] it has been shown that this way of applying S-Interpretations to description logics does not produce satisfying results on real data. In this paper, we therefore propose an alternative way of defining S-Interpretations for description logics which is closer to the notion of S-Interpretations in propositional logic. The idea is to interpret description logics as an extension of propositional logic,

where class names correspond to propositional letters<sup>1</sup>. As for propositional logic, we select a subset of the class names that is interpreted in the classical way and approximate class names not in this set. In particular, a classical interpretation  $(\Delta^{\mathcal{I}}, \mathcal{I})$  of class names requires that a concept name and its negation form a disjoint partition of the domain:

$$\begin{aligned} C^{\mathcal{I}} \cap (\neg C)^{\mathcal{I}} &= \emptyset \\ C^{\mathcal{I}} \cup (\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \end{aligned} \tag{4}$$

We can now define approximations for description logics by relaxing these requirements for a subset of the concept names. The corresponding S-3 and S-1 Interpretations are very similar to the ones for propositional logic. In particular, for S-3 Interpretations we have.

$$C^{\mathcal{I}} \cap (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}}, C \notin S \tag{5}$$

This means that both,  $C$  and  $\neg C$  are mapped to  $\Delta^{\mathcal{I}}$  by the interpretation. As a consequence, the concept name  $C$  cannot cause a clash in a tableaux proof and therefore, constraints that force a certain value to be of type  $C$  will be ignored in a subsumption proof. The resulting subsumption operator is sound, but incomplete. For S-1 Interpretations, we have

$$C^{\mathcal{I}} \cup (\neg C)^{\mathcal{I}} = \emptyset, C \notin S \tag{6}$$

which means that both  $C$  and  $\neg C$  are mapped to the empty set. In a tableaux proof, all attempts to construct a model that involves a variable of type  $C$  will fail. The corresponding subsumption operator is complete, but unsound with respect to classical subsumption.

While approximation based on concept names is a straightforward application of the notion of S-1 and S-3 interpretations, things become more complicated if we want to extend the approach to relation names. In Description Logics relations are used to formulate constraints that apply to all members of a certain class. The most general formulation of these constraints is in terms of qualified number restrictions. Qualified number restrictions have the following form  $(\leq nr.C)$  or  $(\geq nr.C)$  where  $n$  is a positive natural number (including zero),  $r$  is the name of a binary relation and  $C$  is a concept expression. In a tableaux these qualified number restrictions are a second potential source of inconsistency besides the negation operator. In particular, we have

$$(\leq nr.C)^{\mathcal{I}} \cap (\geq mr.C)^{\mathcal{I}} = \emptyset \text{ for all } n < m$$

on the other hand, we have

$$(\leq nr.C)^{\mathcal{I}} \cup (\geq mr.C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \text{ for all } n \geq m$$

We can use this analogy to extend the notion of S-1 and S-3 interpretations to qualified number restrictions in the following way. For S-3 Interpretations we define that

$$(\leq nr.C)^{\mathcal{I}} \cup (\geq mr.C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \text{ for all } r \notin S \tag{7}$$

In particular, we weaken the condition for the expression to become the universal concept by making it independent of the values for  $m$  and  $n$ . Further, we claim that the conjunction of qualified number expressions can never be the empty concept, i.e.

$$(\leq nr.C)^{\mathcal{I}} \cap (\geq mr.C)^{\mathcal{I}} \neq \emptyset \text{ for all } r \notin S \tag{8}$$

<sup>1</sup> In fact, a description logic that just contains the Boolean operators is equivalent to propositional logic.

This leaves us with a weaker interpretation, because inconsistencies arising from the relations not in the set  $S$  cannot be detected. For  $S$ -1 interpretations, we make analogous claim by demanding that the union of two qualified number restrictions can never be the universal concept

$$(\leq n r.C)^{\mathcal{I}} \cup (\geq m r.C)^{\mathcal{I}} \neq \Delta^{\mathcal{I}} \text{ for all } r \notin S \quad (9)$$

Further, we strengthen the interpretation by claiming that the intersection of the two qualified number restrictions on the same relation and concept is always inconsistent

$$(\leq n r.C)^{\mathcal{I}} \cap (\geq m r.C)^{\mathcal{I}} = \emptyset \text{ for all } r \notin S \quad (10)$$

This gives us a stronger version of the semantics, because any two assertions using this relation in combination with the same concept expression  $C$  leads to an inconsistency<sup>2</sup>. The result is a complete but unsound subsumption operator. This unsound approximation operator is exactly what we need for specifying the notion of a partial match, because it forces a match on the constraints involving class names from  $S$  and treats constraints involving classes not in  $S$  as optional. Using subsumption operators with different sets  $S$ , we can focus on different aspects of the matching task and also rank results based on the number of requirements met. In the following, we will therefore concentrate on complete, but unsound approximations of subsumption reasoning for concept expressions based on the idea described above. In particular, we will formally specify non-standard interpretations and define a family of approximate subsumption operators that can be used to compute partial matches.

### 3 Non-Standard Semantics

In the following, we introduce a non-standard interpretation for concept expressions in the logic  $\mathcal{ALCQ}$ . A limited vocabulary is a subset  $S \subseteq \mathcal{V}$  of the concept and relation names occurring in a concept expression. Our aim is to define approximate reasoning in Description Logics based on such a subset of the vocabulary. For this purpose, we define an upper and a lower approximation of an interpretation  $\mathcal{I}$  with respect to a set  $S$  referred to as  $\mathcal{I}_S^+$  and  $\mathcal{I}_S^-$  respectively. We call  $\mathcal{I}_S^+$  an upper approximation and  $\mathcal{I}_S^-$  a lower approximation of  $\mathcal{I}$  with respect to  $S$ .

**Definition 1 (Lower Approximation).** A lower approximation of an interpretation  $\mathcal{I}$  with respect to  $S$  is a non standard interpretation  $(\Delta^{\mathcal{I}}, \mathcal{I}_S^-)$  such that:

$$A^{\mathcal{I}_S^-} = \begin{cases} A^{\mathcal{I}} & A \in S \\ \emptyset & \text{otherwise} \end{cases} \quad (11)$$

$$(\neg C)^{\mathcal{I}_S^-} = \Delta^{\mathcal{I}} - C^{\mathcal{I}_S^+} \quad (12)$$

$$(C \sqcap D)^{\mathcal{I}_S^-} = C^{\mathcal{I}_S^-} \cap D^{\mathcal{I}_S^-} \quad (13)$$

$$(C \sqcup D)^{\mathcal{I}_S^-} = C^{\mathcal{I}_S^-} \cup D^{\mathcal{I}_S^-} \quad (14)$$

$$(\geq n r.C)^{\mathcal{I}_S^-} = \begin{cases} \{x \mid \#\{y.(x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^-}\} \geq n\} & r \in S \\ \{x \mid \#\{y.(x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^-}\} \geq \infty\} & \text{otherwise} \end{cases} \quad (15)$$

$$(\leq n r.C)^{\mathcal{I}_S^-} = \begin{cases} \{x \mid \#\{y \mid (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^+}\} \leq n\} & r \in S \\ \{x \mid \#\{y \mid (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^+}\} \leq 0\} & \text{otherwise} \end{cases} \quad (16)$$

<sup>2</sup> As we will see later, it is sufficient if the two restrictions use concept expressions that are logically equivalent

where  $(\Delta^{\mathcal{I}}, \mathcal{I}_S^+)$  is an upper approximation as defined in definition 2

**Definition 2 (Upper Approximation).** An upper approximation of an interpretation  $\mathcal{I}$  with respect to  $S$  is a non standard interpretation  $(\Delta^{\mathcal{I}}, \mathcal{I}_S^+)$  such that:

$$A^{\mathcal{I}_S^+} = \begin{cases} A^{\mathcal{I}} & A \in S \\ \Delta^{\mathcal{I}} & \text{otherwise} \end{cases} \quad (17)$$

$$(-C)^{\mathcal{I}_S^+} = \Delta^{\mathcal{I}} - C^{\mathcal{I}_S^-} \quad (18)$$

$$(C \cap D)^{\mathcal{I}_S^+} = C^{\mathcal{I}_S^+} \cap D^{\mathcal{I}_S^+} \quad (19)$$

$$(C \cup D)^{\mathcal{I}_S^+} = C^{\mathcal{I}_S^+} \cup D^{\mathcal{I}_S^+} \quad (20)$$

$$(\geq n r.C)^{\mathcal{I}_S^+} = \begin{cases} \{x | \#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^+}\} \geq n\} & r \in S \\ \{x | \#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^+}\} > 0\} & \text{otherwise} \end{cases} \quad (21)$$

$$(\leq n r.C)^{\mathcal{I}_S^+} = \begin{cases} \{x | \#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^-}\} \leq n\} & r \in S \\ \{x | \#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^-}\} < \infty\} & \text{otherwise} \end{cases} \quad (22)$$

where  $(\Delta^{\mathcal{I}}, \mathcal{I}_S^-)$  is a lower approximation as defined in definition 1

A nice property of this definition is that it ensures the existence of a negation normal form that can be computed using the same transformation rules as usually.

**Corollary 1 (Negation Normal Form).** For every concept expression  $C$  there is an expression  $nnf(C)$  in negation normal form such that  $nnf(C)^{\mathcal{I}_S^-} = C^{\mathcal{I}_S^-}$  and  $nnf(C)^{\mathcal{I}_S^+} = C^{\mathcal{I}_S^+}$

Another useful property of the non standard interpretation is that it makes concept expressions strictly more general for upper and strictly more specific for lower approximations. This property which we call monotonicity is important in order to be able to guarantee formal properties of approximation methods defined based on this interpretation. Therefore the following theorem describes a central property of approximation in description logics.

**Lemma 1 (Monotonicity).** Given a non-standard interpretation as defined above, the following equation holds for all concept expressions  $C$ :

$$C^{\mathcal{I}_S^-} \subseteq C^{\mathcal{I}} \subseteq C^{\mathcal{I}_S^+} \quad (23)$$

We can generalize the theorem by observing that the standard interpretation is an extreme case of the non-standard interpretation with  $S = \mathcal{V}$ . In particular, the general version of monotonicity says that for upper approximations removing names from the set  $S$  will make concepts expressions strictly more general. Conversely, for lower approximations concept expressions become less general when we remove concept or relation names from the set  $S$ . The corresponding general property is defined in the following theorem:

**Lemma 2 (Generalized Monotonicity).** Given a non-standard interpretation as defined above and two sub-vocabularies  $S_1$  and  $S_2$  with  $S_1 \subseteq S_2$ , the following equations hold for all concept expressions  $C$ :

$$C^{\mathcal{I}_{S_1}^-} \supseteq C^{\mathcal{I}_{S_2}^-} \quad C^{\mathcal{I}_{S_1}^+} \subseteq C^{\mathcal{I}_{S_2}^+} \quad (24)$$

The generalized monotonicity property is interesting, because it allows us to successively compute more precise upper and lower approximations of a concept by adding names to the set  $S$ . This is convenient in cases where users provide a preference order over the vocabulary indicating the relative importance of different aspects of a concept. In this case, use the preference relation provided by the user to determine a sequence of approximations to be used in the matching process.

#### 4 An Approximate Subsumption Operator

Up to now, we have only considered interpretations as such. As our aim is to develop approximate notions of subsumption as a basis for approximate matching, we now have to define the notion of approximate subsumption based on the non-standard interpretation defined above. It turns out, that this can be done in a straightforward way using the standard definition of the subsumption operator as:

$$\forall \mathcal{I} : \mathcal{I} \models C \sqsubseteq D \Leftrightarrow (C \sqcap \neg D)^{\mathcal{I}} = \emptyset$$

The idea is now to use this definition and replace the standard interpretation  $\mathcal{I}$  by a the lower approximation  $\mathcal{I}_S^-$  with respect to a certain sub-vocabulary  $S$ . Based on the choice of  $S$ , this defines different subsumption operators with certain formal properties that will be discussed in the following.

**Definition 3 (Approximate Subsumption).** *Let  $S \subseteq \mathcal{V}$  be a subset of the concept names and  $(\Delta^{\mathcal{I}}, \mathcal{I}_S^-)$  a lower approximation, then the corresponding approximate subsumption relation  $\sqsubseteq_S$  is defined as follows*

$$\forall \mathcal{I} : \mathcal{I} \models (C \sqsubseteq_S D) \Leftrightarrow_{def} (C \sqcap \neg D)^{\mathcal{I}_S^-} = \emptyset \tag{25}$$

We say that  $C$  is subsumed by  $D$  with respect to sub-vocabulary  $S$ .

The monotonicity of the non-standard interpretation has an impact on the formal properties of the approximate subsumption operator. In particular, we can establish a relation between the subset of the vocabulary considered and the strength of the subsumption operator. The more concepts we exclude from the set  $S$  the weaker the subsumption operator as well as the matches we can compute get. This implies that if we can prove subsumption with respect to a particular set  $S$  the subsumption relation also holds for all subsets of  $S$ . Conversely, if we fail to prove subsumption with respect to a set  $S$ , we can be sure that the subsumption relation does also not hold with respect to any superset of  $S$ . These properties are stated formally in the following theorem.

**Theorem 1 (Properties of Approximate Subsumption).** *Let  $f$  be a lower approximation, then the following equation holds:*

$$\left( C \sqsubseteq_{S_2} D \right) \Rightarrow \left( C \sqsubseteq_{S_1} D \right) \text{ for } S_1 \subseteq S_2 \tag{26}$$

$$\left( C \not\sqsubseteq_{S_1} D \right) \Rightarrow \left( C \not\sqsubseteq_{S_2} D \right) \text{ for } S_1 \subseteq S_2 \tag{27}$$

These properties allow us to develop approximation strategies by successively selecting smaller subsets of concepts to be considered for matching and trying to compute the corresponding subsumption relation until we succeed.

## 5 Computing Approximate Subsumption

A nice feature of our approach is that it can actually be implemented by simply performing syntactic modifications on concept expressions. In particular, in order to check whether a statement  $C \sqsubseteq_{\mathcal{S}} D$  holds, we take the expression  $(C \sqcap \neg D)^-$  and transform it into a concept expression that simulates the non-standard interpretation. For the lower approximation, the corresponding transformation  $(.)^-$  is defined as follows

$$\begin{aligned}
 (A)^- &\rightarrow \perp \text{ if } A \in \mathcal{S} \\
 (\neg A)^- &\rightarrow \perp \text{ if } A \in \mathcal{S} \\
 (\neg C)^- &\rightarrow \neg(C)^+ \\
 (C \sqcap D)^- &\rightarrow (C)^- \sqcap (D)^- \\
 (C \sqcup D)^- &\rightarrow (C)^- \sqcup (D)^- \\
 (\leq n r.C)^- &\rightarrow (\leq 0 r.(C)^+) \text{ if } r \in \mathcal{S} \\
 (\leq n r.C)^- &\rightarrow (\leq n r.(C)^+) \text{ if } r \notin \mathcal{S} \\
 (\geq n r.C)^- &\rightarrow (\geq \max r.(C)^-) \text{ if } r \in \mathcal{S} \\
 (\geq n r.C)^- &\rightarrow (\geq n r.(C)^-) \text{ if } r \notin \mathcal{S}
 \end{aligned}$$

Here  $\max$  is an integer number that is larger than any other number occurring in any qualified number restriction in the concept expression. This is sufficient to model the interpretation that requires less than an infinite number of  $r$ -successors. Analogously, we define a transformation function  $(.)^+$  that creates a concept expression that simulates the upper approximation of a concept expression. This transformation is defined as follows:

$$\begin{aligned}
 (A)^+ &\rightarrow \top \text{ if } A \in \mathcal{S} \\
 (\neg A)^+ &\rightarrow \top \text{ if } A \in \mathcal{S} \\
 (\neg C)^+ &\rightarrow \neg(C)^- \\
 (C \sqcap D)^+ &\rightarrow (C)^+ \sqcap (D)^+ \\
 (C \sqcup D)^+ &\rightarrow (C)^+ \sqcup (D)^+ \\
 (\leq n r.C)^+ &\rightarrow (\leq \max - 1 r.(C)^-) \text{ if } r \in \mathcal{S} \\
 (\leq n r.C)^+ &\rightarrow (\leq n r.(C)^-) \text{ if } r \notin \mathcal{S} \\
 (\geq n r.C)^+ &\rightarrow (\geq 1 r.(C)^+) \text{ if } r \in \mathcal{S} \\
 (\geq n r.C)^+ &\rightarrow (\geq n r.(C)^+) \text{ if } r \notin \mathcal{S}
 \end{aligned}$$

We again use the number  $\max$  for modeling an infinite number of  $r$ -successors. Further, we have to use the condition  $\geq 1$  instead of  $< 0$  which is equivalent. It can be shown that these rewriting rules provide a way for computing approximate subsumption as stated by the following theorem.

**Theorem 2 (Syntactic approximation I).** *Let  $C$  and  $D$  be concept expressions in  $\mathcal{ALCQ}$ , then  $\mathcal{I} \models C \sqsubseteq_{\mathcal{S}} D$  if and only if  $(C \sqcap \neg D)^-$  is unsatisfiable.*

It turns out that the equivalence of a concept expression and its normal form and the symmetry of upper and lower approximation with respect to negation can be used to define an alternative way of computing approximate subsumption based on the syntactic manipulations shown above

**Theorem 3 (Syntactic approximation II).** *Let  $C$  and  $D$  be concept expressions in  $\mathcal{ALCQ}$ , then  $\mathcal{I} \models C \sqsubseteq_S D$  if and only if  $\mathcal{I} \models (C)^- \sqsubseteq (D)^+$*

This means that we have two rather straightforward ways of computing approximate subsumption using standard DL reasoners.

## 6 Discussion

We presented an approach for computing approximate subsumption between concept expressions in  $\mathcal{ALCQ}$  based on a subset of the vocabulary used in the expressions. The approach solves some of the problems of classical reasoning in description logics, in particular, the inability to accept imperfect matches between concepts without having to leave the realms of formal logic. As a side-effect, the subset of the vocabulary also provides us with a qualitative characterization of the mismatch between the expressions, which is clearly an advantage over numerical approaches for dealing with imperfect matches. An approach for partial matching in description logics that is more similar to ours is reported in [6]. This approach, however, cannot deal with disjunction and qualified number restrictions.

## References

1. Giugno, R., Lukasiewicz, T.: P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In: Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02). (2002)
2. Straccia, U.: Towards a fuzzy description logic for the semantic web (preliminary report). In: Proceedings of the 2nd European Semantic Web Conference ESWC-05. (2005) 167–181
3. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. *Artificial Intelligence* **74**(2) (1995) 249–310
4. Cadoli, M., Schaerf, M.: Approximation in concept description languages. In: Proceedings of the International Conference on Knowledge Representation and Reasoning. (1992) 330–341
5. Groot, P., Stuckenschmidt, H., Wache, H.: Approximating description logic classification for semantic web reasoning. In: Proceedings of the 2nd European Semantic Web Conference, Heraklion, Crete (2005)
6. Di Noia, T., Eugenio Di Sciascio, F.D., Mongiello, M.: A system for principled matchmaking in an electronic marketplace. In: Proceedings of the Twelfth International World Wide Web Conference. (2003) 321–330

## Speeding up Approximation with nicer Concepts

Anni-Yasmin Turhan\* and Yusri Bong

Theoretical Computer Science, TU Dresden, Germany  
 {turhan, bong}@tcs.inf.tu-dresden.de

**Abstract.** Concept approximation is an inference service for Description Logics that provides “translations” of concept descriptions from one DL to a less expressive DL. In [4] a method for optimizing the computation of  $\mathcal{ALC}$ - $\mathcal{AL}\mathcal{E}$ -approximations of  $\mathcal{ALC}$ -concept descriptions was introduced. The idea is to characterize a certain class of concept descriptions for which conjuncts can be approximated independently. In this paper we provide relaxed conditions for this class of  $\mathcal{ALC}$ -concept descriptions, extend this notion to number restrictions and report on a first implementation of this method for  $\mathcal{ALCN}$ - $\mathcal{AL}\mathcal{EN}$ -approximation.

### 1 Motivation

Approximation is a non-standard inference service in Description Logics (DLs) introduced in [3]. Approximating a concept description, defined in one DL, means to translate this concept description to another concept description, defined in a second, typically less expressive DL, such that both concepts are as closely related as possible with respect to subsumption. Like other non-standard inferences such as computing the least common subsumer (lcs) or matching of concepts, approximation has been introduced to support the construction and maintenance of DL knowledge bases.

The building of ontologies can be assisted by employing the bottom-up approach, where new concepts can be derived from a collection of already existing ones by computing their commonalities. This task is typically realized by computing the lcs. If the employed DL provides disjunction, the lcs is just the disjunction of the input concepts. Thus, a user inspecting this concept does not learn anything about their commonalities. By using concept approximation, however, one can make the commonalities explicit to some extent by first approximating the concepts in a sub-language which does not provide disjunction, and then computing the lcs of the approximations. Besides this approximation is used to compute semantically closely related versions of expressive knowledge bases to port knowledge bases between different systems or to integrate different knowledge bases. Moreover, users who are no DL experts can be supported in comprehending a knowledge base written in an expressive DL by offering them a simplified view of it obtained by approximation.

A worst case double exponential algorithm for approximating  $\mathcal{ALCN}$ - by  $\mathcal{AL}\mathcal{EN}$ -concept descriptions was presented in [2]. A first implementation of approximation presented in [3] revealed that run-times for concepts of moderate size

\* This work was partially supported by EU FET TONES (grant IST-2005-7603).

is already a couple of seconds. For an interactive application as the bottom-up construction faster run-times are desirable. In [4] the following approach to optimize computation of approximations was presented: Instead of approximating a concept  $C$  as a whole, a significant amount of time could be saved by splitting  $C$  into its conjuncts and approximating them separately. If, for instance,  $C$  consists of two conjuncts of size  $n$  then the approximation of  $C$  takes some  $a^{b^{2n}}$  steps while the conjunct-wise approach would just take  $2a^{b^n}$ . Unfortunately, splitting an arbitrary input concept at conjunctions leads to incorrect approximations. A class for which this conjunct-wise approximation still produces the correct result is the class of so-called *nice concepts*. Moreover, a characterization of nice concepts is also an important prerequisite for another way of optimizing computations of concept approximation, namely, non-naive use of caching. Say, we want to approximate  $C \sqcap D$ , where  $C$  and  $D$  are complex  $\mathcal{ALCN}$ -concept descriptions. Now, if  $C \sqcap D$  is nice and we have cached the approximation of  $C$ , we only need to compute the approximation of  $D$  and conjoin it with the cached result, instead of computing the whole approximation from scratch.

## 2 Preliminaries

We assume that the reader is familiar with the basic notions of DLs, see [1]. *Concept descriptions* are inductively defined based on a set of *concept constructors* starting with a set  $N_C$  of *concept names* and a set  $N_R$  of *role names*. The DL  $\mathcal{ALC}$  provides the constructors conjunction, existential and value restrictions as well as primitive negation, i.e., only concept names can be negated.  $\mathcal{ALC}$  extends  $\mathcal{ALC}$  by full negation and disjunction.  $\mathcal{ALCN}$  ( $\mathcal{ALEN}$ ) adds number restrictions to  $\mathcal{ALC}$  ( $\mathcal{ALC}$ ). For the syntax and model theoretic semantics of the mentioned concept constructors, see [1]. In addition to the usual definition, we require that TBoxes are *unfoldable*, i.e., their concept definitions are acyclic and unique. In order to approximate  $\mathcal{ALCN}$ -concept descriptions by  $\mathcal{ALEN}$ -concept descriptions, we need to compute the lcs in  $\mathcal{ALEN}$ .

**Definition 1 (lcs).** *Given  $\mathcal{ALEN}$ -concept descriptions  $C_1, \dots, C_n$  with  $n \geq 2$ , the  $\mathcal{ALEN}$ -concept description  $C$  is the least common subsumer (lcs) of  $C_1, \dots, C_n$  iff (i)  $C_i \sqsubseteq C$  for all  $1 \leq i \leq n$ , and (ii)  $C$  is the least concept description with this property, i.e., if  $C'$  satisfies  $C_i \sqsubseteq C'$  for all  $1 \leq i \leq n$ , then  $C \sqsubseteq C'$ .*

As already mentioned, in  $\mathcal{ALCN}$  the lcs trivially exists since  $lcs(C, D) \equiv C \sqcup D$ . To obtain a more meaningful concept description, we first approximate the  $\mathcal{ALCN}$ -concept descriptions and then compute their lcs.

**Definition 2 (approximation).** *Let  $\mathcal{L}_1$  and  $\mathcal{L}_2$  be two DLs, and let  $C$  be an  $\mathcal{L}_1$ - and  $D$  be an  $\mathcal{L}_2$ -concept description. Then,  $D$  is called an  $\mathcal{L}_1 - \mathcal{L}_2$ -approximation of  $C$  (written  $D = approx_{\mathcal{L}_2}(C)$ ) iff (i)  $C \sqsubseteq D$ , and (ii)  $D$  is minimal with this property, i.e.,  $C \sqsubseteq D'$  and  $D' \sqsubseteq D$  implies  $D' \equiv D$  for all  $\mathcal{L}_2$ -concept descriptions  $D'$ .*

Intuitively, an approximation of an  $\mathcal{ALCN}$ -concept description is an  $\mathcal{ALEN}$ -concept description that is more general than the input concept description but minimal w.r.t. subsumption.

### 2.1 Computation Algorithm for $\mathcal{ALCN}$ - $\mathcal{ALEN}$ -approximations

We sketch the computation algorithm for  $\mathcal{ALCN}$ - $\mathcal{ALEN}$ -approximations briefly—for its exact definition refer to [2]. In case an approximation of an  $\mathcal{ALCN}$ -concept  $C$  that uses names defined in a TBox is to be computed, then these names have to be first replaced by their definition. If  $C$  is equivalent to  $\top$  ( $\perp$ ), its approximation is  $\top$  ( $\perp$ ), otherwise the concept description is normalized. First, the concept description is transformed into negation normal form (NNF). Second, the obtained concept description is transformed into  $\mathcal{ALCN}$ -normal form ( $\mathcal{ALCN}$ -NF). In this step conjunctions are distributed over the disjunctions. In order to describe the disjuncts obtained by the  $\mathcal{ALCN}$ -NF, some notation is needed to access the different parts of a concept description  $C$ .

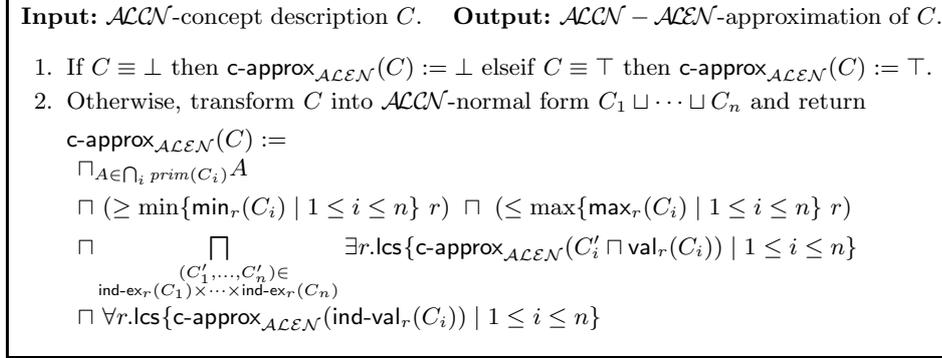
- $\text{prim}(C)$  denotes the set of all (negated) concept names and  $\perp$  occurring on the top-level of  $C$ ;
- $\text{val}_r(C) := C_1 \sqcap \dots \sqcap C_n$ , if value restrictions of the form  $\forall r.C_1, \dots, \forall r.C_n$  exist on the top-level of  $C$ ; otherwise,  $\text{val}_r(C) := \top$ ;
- $\text{ex}_r(C) := \{C' \mid \text{there exists } \exists r.C' \text{ on the top-level of } C\}$ ;
- $\text{min}_r(C) := \max\{k \mid C \sqsubseteq (\geq k r)\}$  (Note that  $\text{min}_r(C)$  is always finite.);
- $\text{max}_r(C) := \min\{k \mid C \sqsubseteq (\leq k r)\}$ ; if there exists no  $k$  with  $C \sqsubseteq (\leq k r)$ , then  $\text{max}_r(C) := \infty$ .

Now, an  $\mathcal{ALCN}$ -concept description  $C$  in  $\mathcal{ALCN}$ -NF is of the form  $C = C_1 \sqcup \dots \sqcup C_n$  with  $C_i :=$

$$\prod_{A \in \text{prim}(C_i)} A \sqcap \prod_{r \in N_R} \left( \prod_{C' \in \text{ex}_r(C_i)} \exists r.C' \sqcap \forall r.\text{val}_r(C_i) \sqcap (\geq \text{min}_r(C_i) r) \sqcap (\leq \text{max}_r(C_i) r) \right),$$

for all  $i = 1, \dots, n$ , where the concept descriptions  $\text{val}_r(C_i)$  and  $C'$  again are in  $\mathcal{ALCN}$ -normal form and  $C_i$  is removed from the disjunction in case  $C_i \equiv \perp$ .

Next, implicit information captured in the concept description is made explicit. Due to space limitations, we can only give an intuition in which combinations of concept constructors information is induced. For a thorough discussion refer to [6] or [2]. In case of the number restrictions appearing in the concept description ( $\geq \text{min}_r(C) r$ ) and ( $\leq \text{max}_r(C) r$ ) already make induced information explicit. At-least restrictions can be induced by incompatible existential restrictions, e.g.,  $\exists r.A \sqcap \exists r.\neg A$  induces ( $\geq 2 r$ ). At-most restrictions can be induced only by value restrictions equivalent to  $\forall r.\perp$  implying ( $\leq 0 r$ ). Vice versa, value restrictions can be induced by ( $\leq 0 r$ ). Furthermore, value restrictions can be implied, if the minimal number of  $r$ -successors required by either at-least or by incompatible existential restrictions coincide with  $\text{max}_r(C)$ . For example in  $(\leq 2 r) \sqcap (\exists r.A \sqcap B) \sqcap (\exists r.A \sqcap \neg B)$  the value restriction  $\forall r.A$  is induced. Induced value restrictions are denoted  $\text{ind-val}_r(C)$ .



**Fig. 1.** The computation algorithm for  $\mathcal{ALCN}$ - $\mathcal{ALCN}$ -approximation.

Induced existential restrictions are obtained if  $\|\text{ex}_r(C)\| > \max_r(C)$ . In this case the existential restrictions have to be merged. More precisely, such a merging must yield  $\max_r(C)$  existential restrictions s.t. the set  $\text{ex}_r(C)$  is partitioned and only consistent existential restrictions are obtained. Moreover,  $\text{val}_r(C)$  has to be propagated onto each existential restriction. The induced existential restrictions are obtained by computing the commonalities of all ways of obtaining valid mergings.

Figure 1 displays the computation algorithm for  $\mathcal{ALCN}$ - $\mathcal{ALCN}$ -approximation. In the computation of the approximation as well as in the computation of  $\text{ind-val}_r(C)$  and  $\text{ind-ex}_r(C)$  the lcs for  $\mathcal{ALCN}$  is used, which was introduced in [6].

### 3 Nice concepts for Approximation

In general, the computation of an approximation cannot be split at the conjunction because of possible interactions—in case of  $\mathcal{ALC}$ - $\mathcal{ALC}$ -approximation between existential and value restrictions on the one hand and inconsistencies induced by negation on the other. For example, the approximation  $\mathbf{c}\text{-approx}_{\mathcal{ALC}}(\exists r. \top \sqcap (\forall r. A \sqcup \exists r. A))$  yields  $\exists r. A$  while the conjunct-wise version  $\mathbf{c}\text{-approx}_{\mathcal{ALC}}(\exists r. \top) \sqcap \mathbf{c}\text{-approx}_{\mathcal{ALC}}(\forall r. A \sqcup \exists r. A)$  only produces  $\exists r. \top$ . In [4] those concept descriptions were called *nice* for which this strategy still produces the correct result.

**Definition 3 (nice concepts).** Let  $C := C_1 \sqcap \dots \sqcap C_n$  be a  $\mathcal{L}_1$ -concept description.  $C$  is nice if  $\text{approx}_{\mathcal{L}_2}(C) \equiv \text{approx}_{\mathcal{L}_2}(C_1) \sqcap \dots \sqcap \text{approx}_{\mathcal{L}_2}(C_n)$ .

For these concept descriptions interactions between conjuncts are excluded. Since the use of nice concept descriptions is to speed-up approximation, it is important that the conditions to distinguish these concept descriptions can be tested easily. Therefore the test for nice concept descriptions should be based on simple discrimination conditions. To this end the conditions for nice  $\mathcal{ALC}$ -concept descriptions given in [4] are sound, but not complete:

1. The restrictions are limited to one type per role-depth: on every role depth of a nice concept either no  $\forall$ -restrictions or no  $\exists$ -restrictions occur.

2. A concept name and its negation may not occur on the same role-depth of a nice concept.

It is shown in [4] that for  $\mathcal{ALC}$ -concept descriptions fulfilling these conditions conjunct-wise approximation is correct. The above conditions are intuitive and easy to test, but very strict. Consider the concept description  $(\exists r.\exists s.A \sqcup B) \sqcap (\exists t.\forall s.\neg A \sqcup C)$ , which violates both conditions. However, we get the correct result, if the conjuncts are approximated independently, since the relevant concept descriptions are nested in existential restrictions for different roles. In general, a concept description can still be approximated conjunct-wise, if the “interacting” concept descriptions are reachable via different role paths. Too strict conditions to distinguish nice concept descriptions would rule out too many concept descriptions that could actually be approximated conjunct-wise. In these cases the “expensive” approximation must be applied. In order to be able to classify more concept descriptions as nice, we devise relaxed conditions for nice concept descriptions.

### 3.1 Nice $\mathcal{ALCN}$ -concept descriptions

The conditions for nice  $\mathcal{ALCN}$ -concept descriptions have to take into account the information induced by number restrictions in combination with other concept constructors, i.e., the interactions we sketched in Section 2.1. For example, consider the  $\mathcal{ALCN}$ -concept description  $C = C_1 \sqcap C_2$ , where the conjuncts are:  $C_1 = (\exists s.A) \sqcap (\exists s.\neg A) \sqcup \perp$  and  $C_2 = (\leq 1 s) \sqcup B$ . Now, if we approximate  $C$  conjunct-wise we obtain  $\text{c-approx}_{\mathcal{ALCN}}(C_1) = (\exists s.A) \sqcap (\exists s.\neg A)$  and  $\text{c-approx}_{\mathcal{ALCN}}(C_2) = \top$ , yielding  $(\exists s.A) \sqcap (\exists s.\neg A) \sqcap \top$  as the combined result. Due to the incompatibilities between  $C_1$  and  $C_2$  for the information on the role  $s$ , the disjunction in  $C_2$  collapses to  $B$ . The approximation yields  $\text{c-approx}_{\mathcal{ALCN}}(C) = (\exists s.A) \sqcap (\exists s.\neg A) \sqcap B$ , which is more specific than the conjunct-wise approximation.

To devise syntactic conditions to detect nice  $\mathcal{ALCN}$ -concept descriptions, we introduce notation to access the numbers in number restrictions. For an  $\mathcal{ALCN}$ -concept description  $C$  and a role  $r$  let  $\text{at-least}_r(C)$  ( $\text{at-most}_r(C)$ ) denote the maximal (minimal) number appearing in at-least (at-most) restrictions on the top-level of  $C$  or, if the top-level of  $C$  has no at-least restriction, 0 (at-most restriction,  $\infty$ .) Next, we specify the notion of sub-concept descriptions accessible by a role path, that will be used in the conditions for nice concept descriptions.

**Definition 4.** Let a  $Qr$ -path (denoted  $\rho$ ) be defined as  $\rho = (Qr)^*$  for  $Q \in \{\exists, \forall\}$  and  $r \in N_R$  and let  $\lambda$  denote the empty  $Qr$ -path. Let  $C$  be an  $\mathcal{ALCN}$ -concept description and  $\rho$  be a  $Qr$ -path, then  $\text{sub}(C, \rho) :=$

- $\text{prim}(C) \sqcap \prod_{s \in N_R} (\leq \text{at-most}_s(C) s) \sqcap (\geq \text{at-least}_s(C) s)$ , if  $\rho = \lambda$ ,
- $\text{sub}(\text{val}_r(C), \rho')$ , if  $\rho = \forall r \cdot \rho'$  and  $\text{val}_r(C) \neq \top$ ,
- $\bigcup_{C' \in \text{ex}_r(C)} \text{sub}(C', \rho')$ , if  $\rho = \exists r \cdot \rho'$ ,
- $\emptyset$ , otherwise.

Based on role-paths we can give necessary conditions for nice  $\mathcal{ALCN}$ -concept descriptions, which additionally relax the necessary conditions given in [4] for nice  $\mathcal{ALC}$ -concept descriptions.

**Definition 5 (necessary conditions for nice  $\mathcal{ALCN}$ -concept descriptions).**

Let  $C$  be an  $\mathcal{ALCN}$ -concept description in NNF. Then  $C$  is nice, if for every  $Qr$ -path  $\rho$  with  $C_1, C_2 \in \text{sub}(C, \rho)$  and  $C'_1, C'_2$  denoting  $C_1, C_2$  in  $\mathcal{ALCN}$ -NF and all  $r \in N_R$  it holds

1.  $\|\{\exists \mid \text{at-least}_r(C_i) \neq 0, i \in \{1, 2\}\} \cup \{\exists \mid \text{ex}_r(C_1) \cup \text{ex}_r(C_2) \neq \emptyset\}\| + \|\{\forall \mid \text{at-most}_r(C_i) \neq \infty, i \in \{1, 2\}\} \cup \{\forall \mid \prod_{i \in \{1, 2\}} \text{val}_r(C_i) \neq \top\}\| \leq 1$ , and
2.  $\text{prim}(C_1) \cup \text{prim}(C_2)$  does not contain a concept name and its negation.

The Condition 1 rules out three constellations for sub-concept description accessible via the same  $Qr$ -path: (1) concept descriptions that induce role successors (either by existential or by at-least restrictions) and that have a possibly induced value restriction and (2) concept descriptions with contradicting number restrictions and (3) concept descriptions that require merging of existential restrictions. Condition 2 rules out concept descriptions that have a concept name and its negation accessible via the same  $Qr$ -path. To show that concept descriptions fulfilling the conditions from Definition 5 can be approximated conjunct-wise, we adapt the proof from [4] in [7]. The following lemma is central in the proof of the correctness of the characterization of nice concepts.

**Lemma 1.** For  $1 \leq i \leq 2$ , let  $C_i$  and  $D_i$  be  $\mathcal{ALCN}$ -concept descriptions such that  $C_1 \sqcap C_2 \sqcap D_1 \sqcap D_2$  is a nice concept description. Then it holds that  $\text{lcs}\{C_i \sqcap D_j \mid i, j \in \{1, 2\}\} \equiv \text{lcs}\{C_1, C_2\} \sqcap \text{lcs}\{D_1, D_2\}$ .

The following theorem states our claim that approximations of  $\mathcal{ALCN}$ -concept descriptions fulfilling the conditions from Definition 5, can be obtained by a conjunction of approximations of the conjuncts from the original concept.

**Theorem 1.** Let  $C \sqcap D$  be a nice  $\mathcal{ALCN}$ -concept description, then  $\text{c-approx}_{\mathcal{ALCN}}(C \sqcap D) \equiv \text{c-approx}_{\mathcal{ALCN}}(C) \sqcap \text{c-approx}_{\mathcal{ALCN}}(D)$ .

The claim is proved by induction over the sum of the nesting depths of  $\sqcap$  and  $\sqcup$  on every role level in  $C$  and  $D$ . For the induction step a case distinction is made depending on whether  $C$  or  $D$  are conjunctions or disjunctions. If at least one concept description is a disjunction the approximation is defined as the lcs of all  $\mathcal{ALCN}$ -normalized and approximated disjuncts (if one of the concept descriptions is a conjunction, it firstly has to be distributed over the disjunction). The main idea then is to use Lemma 1 to transform single lcs calls of a certain form into a conjunction of lcs calls which eventually leads to the conjunction of the approximations of  $C$  and  $D$ . The full proof can be found in [7].

Now, the actual  $\mathcal{ALCN}$ - $\mathcal{ALCN}$ -approximation algorithm can be adapted by first testing whether a concept description is nice and then performing conjunct-wise approximation for each conjunct and conjoin the results. The algorithm performs the nice test at the beginning of every recursive call, unless the current (sub-)concept description is already known to be nice.

## 4 Implementation

The implementation of approximation for nice  $\mathcal{ALCN}$ -concept descriptions requires little changes in the implementation of the approximation algorithm. The major part to implement is the tester for nice concept descriptions `nice-p`. Since this test has to be performed at the beginning of every approximation computation, the implementation must be very efficient.

The procedure `nice-p` in our implementation realizes the necessary conditions for nice  $\mathcal{ALCN}$ -concept descriptions from Definition 5. Our implementation of the nice test employs a couple of optimizations. Firstly, some steps are only taken on demand, such as unfolding and the transformation into NNF. Furthermore, `nice-p` stores information of a certain named concept, say  $C$ , in a so-called info-table, where the key of this info-table is a path  $\rho$  and the value is the sub( $C, \rho$ ). This enhances the checking of the conditions from Definition 5 “on-the-fly”. Suppose that while unfolding and transforming concept  $C$ , we encounter the concept name  $A$  inspecting Qr-path  $\rho$ . Then, we update the info-table of  $C$  by adding the concept name  $A$  to the value of the key  $\rho$ . Before we add  $A$ , we first check whether adding  $A$  violates Condition 2 in Definition 5. A similar procedure is carried out, if we encounter number, value and existential restrictions during the unfolding and transformation. Furthermore, our implementation does not only use dynamic programming to re-use results obtained during the current computation on whether a concept is already known to be nice or not, but caches these results. Based on this cache the already obtained information on whether a named concept is nice or not is re-used in subsequent runs of `nice-p`.

We extended the implementation of  $\mathcal{ALCN}$ - $\mathcal{ALEN}$ -approximation from our non-standard inference system SONIC [8] to the use of nice  $\mathcal{ALCN}$ -concepts in a naive way. This way the full potential of the conjunct-wise application of concepts is not yet used. One can in addition implement a caching strategy based on nice concepts in the following way: if we want to approximate a conjunction that is nice and we have computed the approximation of some of its conjuncts already, we only need to conjoin the cached values with the freshly computed approximations of the remaining conjuncts. The implementation of this technique is future work.

## 5 First tests

Although our implementation is not yet mature, we can already report on some experiences. Most importantly, it was unknown whether nice concepts do appear in TBoxes from real-world applications and if, how frequently. We tested our implementation of `nice-p` on the DICE TBox, which is a medical knowledge base from the intensive care domain, see [5]. The DICE TBox contains about 3500 concepts, of which 3249 have a (primitive) definition, and DICE is an unfoldable TBox. Originally, this TBox uses  $\mathcal{ALCQ}$  (and disjointness statements), we used a variant of it pruned down to  $\mathcal{ALCN}$  for our tests. It turned out that this knowledge base has 493 nice concepts satisfying the necessary conditions from

Definition 5. These are concepts not only with a nice sub-concept description, but which are nice “completely”. So a first result of our test is that nice  $\mathcal{ALCN}$  concepts do appear in knowledge bases from practical applications. In case of the DICE knowledge base about 14.3% of all named concepts with a definition are nice. This might not seem very much at first, but indicates that for many concepts significant parts of the unfolded concept descriptions are nice and can be approximated independently. The run-time we measured for each call of `nice-p` when testing all concepts in DICE, was 0.91s on the average. This run-time includes the time needed for unfolding and transforming the concept into NNF—steps that the approximation algorithm requires anyway.

We tested our implementation of conjunct-wise  $\mathcal{ALCN}$ - $\mathcal{AL\mathcal{E}N}$ -approximation on those 463 concepts from the DICE knowledge base that are nice. It turned out that the implementation of the ordinary approximation algorithm needed 1.89s per concept on the average, while the nice approximation needed 1.44s. This is a speed-up of about 24%. Taking into account that the time for nice approximation included the time for the `nice-p` test, one can say that it performed reasonably well.

In this paper we have extended necessary conditions for nice concepts to  $\mathcal{ALCN}$ -concepts. The notion of nice concepts is the basis for the two optimization techniques for concept approximation: non-naive caching and conjunct-wise approximation. A first test of our implementation of nice concepts showed that these kind of concepts do appear in knowledge bases from practical application and that conjunct-wise approximation results in a substantial performance gain.

## References

1. F. Baader and W. Nutt. *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter Basic Description Logics. Cambr. Univ. Press, 2003.
2. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximating  $\mathcal{ALCN}$ -concept descriptions. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, 2002.
3. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In *Proc. of the 8th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-02)*. Morgan Kaufm. Publ., 2002.
4. S. Brandt and A.-Y. Turhan. An approach for optimized approximation. In *Proc. of the 2002 Applications of Description Logic Workshop (ADL 2002)*, 2002. LTCS-Report 02-03 is an extended version, see <http://lat.inf.tu-dresden.de/research/reports.html>.
5. R. Cornet and A. Abu-Hanna. Using description logics for managing medical terminologies. In *A. I. in Medicine: 9th Conference on Artificial Intelligence in Medicine in Europe (AIME 2003)*. Springer, 2003.
6. R. Küsters and R. Molitor. Computing Least Common Subsumers in  $\mathcal{AL\mathcal{E}N}$ . In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI-01)*. Morgan Kaufman, 2001.
7. A.-Y. Turhan. *On the computation of common subsumers*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, 2007. forthcoming.
8. A.-Y. Turhan and C. Kissig. SONIC — Non-standard inferences go OILED. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-04)*. Springer, 2004.

## **$\mathcal{DL}clog$ : A Hybrid System Integrating Rules and Description Logics with Circumscription <sup>\*</sup>**

Fangkai Yang and Xiaoping Chen

Multi-Agent System Lab  
University of Science and Technology of China, Hefei, China

**Abstract.** In this paper, we propose  $\mathcal{DL}clog$ , a new hybrid formalism combining Description Logics and Logic Programming for Semantic Web serving as an extension to  $\mathcal{DL} + log$ [19]. Negative dl-atoms are allowed to occur in the bodies of the rules, and we extend NM-Semantics of  $\mathcal{DL} + log$  to evaluate dl-atoms with *circumscriptive models* of DL ontology in the sense of parallel circumscription rather than classical models. In this way, negative dl-atoms are treated in nonmonotonic way under Extended Closed World Assumption, and the formalism still remains faithful to NM-Semantics, DL and LP. Finally, we present decidability and complexity result for a restricted form of  $\mathcal{DL}clog$ .

### **1 Introduction**

The problem of adding rules to Description Logics is currently a hot research topic, due to the Semantic Web applications of integrating rule-based systems with ontologies. Practically, most research work[3, 16–19] in this area focuses on the integration of Description Logic with datalog rules or its non-monotonic extensions, and  $\mathcal{DL} + log$ [19] is a powerful result of a series of hybrid approaches: DL-log[16],  $r$ -hybrid KB[18],  $r^+$ -hybrid KB[17], which define integrated models on the basis of single models of classical theory.

However, there is a severe limitation in  $\mathcal{DL} + log$  that DL predicates cannot occur behind “not” in rules. This syntactical restriction makes it impossible to use rules to draw conclusions by the results currently undervivable from DL ontology, which cannot satisfies practical needs such as closed world reasoning and modeling exceptions and defaults for DL predicates[12]. To overcome this limitation requires to introduce negative dl-atoms in the body of the rules and interpret them with a nonclassical semantics, which is a nontrivial generalization of NM-Semantics of  $\mathcal{DL} + log$ . As DL adopts Open World Assumption(OWA), we must introduce a kind of closed world reasoning to DL ontology to interpret the unknown results as negation. Such closed world reasoning must be transparently integrated into the framework of NM-Semantics in order that the generalized semantics remains faithful to NM-Semantics, DL and LP. Finally, the definition of the semantics should be concise, model-theoretical and if possible, decidable.

<sup>\*</sup> This work is supported by NSFC (60275024) and 973 Programme (2003CB317000) of People’s Republic of China.

In this paper, a new hybrid formalism  $\mathcal{DL}log$  is presented to achieve this goal. We allow DL predicates occurring behind "not" in rules, and extend NM-Semantics of  $\mathcal{DL} + log$  to be Nonmonotonic Circumscriptive Semantics(NMC-Semantics). In NMC-Semantics, dl-atoms in rules are evaluated under the *circumscriptive models* of the DL ontology in the sense of McCarthy's parallel circumscription[10] rather than classical models as in NM-Semantics. Parallel circumscription is a general form of circumscription formalizing Closed World Assumption(CWA)[15], and is equivalent to Extended CWA[6] which avoids several anomalies in CWA. By parallel circumscription, "not" in negative dl-atoms are interpreted in a nonmonotonic manner closely similar to the treatment of "not" in Logic Programming(LP). As circumscriptive models serve as an intermediate models only used to evaluate dl-atoms, NMC-semantics remains faithful to DL, LP and NM-Semantics of  $\mathcal{DL} + log$ , in that users can switch  $\mathcal{DL}log$  KB to any of these formalisms by syntactical restriction. When  $\mathcal{DL}log$  is restricted to the form that ontologies are written in  $\mathcal{ALCI}O$  or  $\mathcal{ALC}QO$  and roles are not allowed to occur under "not" in rules, NMC-Satisfiability is  $NEXP^{NP}$ -complete.

The rest of the paper is organized as follows. Section 2 presents the motivation of our formalism. Section 3 presents the syntax and semantics of  $\mathcal{DL}log$ . Section 4 presents the decidability and complexity results. Related work is presented in Section 5 and Section 6 ends with conclusion and future work. We assume that the readers be familiar with McCarthy's parallel circumscription[10, 14].

## 2 Motivation

In this section, we focus on clarifying the motivation of our solution. We analyze the semantic characterizations of "not" in LP, show how negative dl-atoms can be interpreted in similar way, and finally, present the solution to capture this semantics. Note that we adopt Standard Names Assumption[18], so we can use the same symbol to denote a constant and its interpretation.

Simply applying NM-Semantics of  $\mathcal{DL} + log$  to evaluate negative dl-atoms by classical models is implausible, as in following example. We use " $\models_{NM}$ " to denote the satisfiability of NM-Semantics of  $\mathcal{DL} + log$ .

*Example 1.* For a KB  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$

$\mathcal{O}$	$\mathcal{P}$
$\neg seaside \equiv notSC$ $portCity(Barcelona)$	$seasideCity(x) \leftarrow portCity(x), O(x), not notSC(x)$ $O(Barcelona)$

We have two classical models of  $\mathcal{O}$ :  $\mathcal{I}_1$  and  $\mathcal{I}_2$ , where  $notSC(Barcelona) \in \mathcal{I}_1$ ,  $notSC(Barcelona) \notin \mathcal{I}_2$ . Evaluating with  $\mathcal{I}_1$ , the rule in  $\mathcal{P}$  is blocked due to negative dl-atoms, thus for the stable model  $\mathcal{J}_1$ ,  $seasideCity(Barcelona) \notin \mathcal{J}_1$ , while with  $\mathcal{I}_2$ , the rule functions, with  $seasideCity(Barcelona) \in \mathcal{J}_2$ . As a result,  $\mathcal{K} \not\models_{NM} seasideCity(Barcelona)$ . However, we hope to obtain  $seasideCity(Barcelona)$ : since  $notSC(Barcelona)$  is unknown to  $\mathcal{O}$ , the first rule should function.

The problem with the above example is that in the convention of nonmonotonic reasoning and LP, ground atom "not  $p(x)$ " is interpreted as "if  $\neg p(x)$  can be *consistently assumed*". Whether an assumption can be *consistently assumed* depends on two facts, which obviously cannot be achieved by NM-Semantics: (1) it is currently underivable and, (2) it is justified in the ultimate extension of the theory[9]. We hope to interpret  $notSC(Barcelona)$  in this form. In Example 1,  $notSC(Barcelona)$  is underivable from  $\mathcal{O}$ . When it comes to justification, we find that  $\neg notSC(Barcelona)$  cannot be justified by the stable models of the rules because  $notSC$  is not involved in the stable models. Alternatively, it should be justified by the fact that the extension of  $\mathcal{O}$  remains consistent after being completed with asserted assumptions, i.e.  $\mathcal{I}_2 \cup \{\neg notSC(Barcelona)\}$  is consistent. Therefore, we claim that  $\neg notSC(Barcelona)$  is *consistently assumed*. So the task we left with is to specify a proper way to complete the extension of the classical part. Naturally, CWA[15] is a candidate, in which a classical model is completed with the negation of underivable facts, but potential anomalies occur.

*Example 2.* For a  $\mathcal{DL}log$  KB  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$  as follows.

$\mathcal{O}$	$\mathcal{P}$
$M \equiv P \sqcup Q$	$r(x) \leftarrow O(x), not P(x)$
$M(a)$	$r(x) \leftarrow O(x), not Q(x)$
	$O(a)$

With the standard CWA, the completed extension is  $\mathcal{J} = \{\neg P(a), \neg Q(a), M(a)\}$  which is a contradiction:  $\mathcal{J}$  is no longer a model of  $\mathcal{O}$ . On the contrary, we would rather prefer to use two classical models  $\mathcal{J}_1$  and  $\mathcal{J}_2$  to evaluate dl-atoms, with  $\{\neg P(a), Q(a)\} \subset \mathcal{J}_1$  and  $\{P(a), \neg Q(a)\} \subset \mathcal{J}_2$ .

A similar proposal is to evaluate negative dl-atoms by whether it is entailed from DL ontology, but integrating "interaction via entailment" complicates the framework of "interaction via single model" of NM-Semantics, and makes the resulted semantics tedious and ill-defined. Finally, our decision is to use Extended CWA[7]. Its semantics is model-theoretically formalized by parallel circumscription[10], and a circumscriptive model is also a classical model. Note that in Example 2,  $\mathcal{J}_1$  and  $\mathcal{J}_2$  are the models of a circumscribed theory  $CIRC[\mathcal{O}; P, Q]$ .

### 3 $\mathcal{DL}log$ : Syntax and Semantics

Based on the motivation above, we present the syntax and semantics of  $\mathcal{DL}log$ .  
**Syntax**

We partition the alphabet of predicates  $\Sigma$  into three mutually disjoint sets  $\Sigma = \Sigma_C \cup \Sigma_R \cup \Sigma_D$ , where  $\Sigma_C$  is an alphabet of concepts names,  $\Sigma_R$  is an alphabet of role names and  $\Sigma_D$  is an alphabet of Datalog predicates. The syntax of  $\mathcal{DL}log$  KB is defined as follows.

**Definition 1.** A hybrid knowledge base  $\mathcal{K}$  is a pair  $(\mathcal{O}, \mathcal{P})$ , where  $\mathcal{O}$  is the a description logic ontology  $(\mathcal{T}, \mathcal{A})$  as its TBOX and ABOX, and  $\mathcal{P}$  is a finite set

of  $Datalog^{\neg, \vee}$  rules of following form:

$$\begin{aligned} \mathcal{R} : & H_1(X_1) \vee \dots \vee H_k(X_n) \leftarrow RB_1(Y_1), \dots, RB_m(Y_m), \\ & \text{not } RB_{m+1}(Y_{m+1}), \dots, \text{not } RB_s(Y_s), \\ & CB_1(Z_1), \dots, CB_n(Z_n), \\ & \text{not } CB_{n+1}(Z_{n+1}), \dots, \text{not } CB_t(Z_t). \end{aligned}$$

where  $H_i(X_i)$ ,  $RB_i(Y_i)$ ,  $CB_i(Z_i)$  are atoms,  $X_i$ ,  $Y_i$ , and  $Z_i$  are vectors of variables. Let  $\mathcal{C}$  denote a set of countably infinite constant names. And

- each  $H_i$  is either a DL predicate or a Datalog predicate.
- each  $RB_i$  is a Datalog predicate, and  $RB_i(Y_i)$  is called a rule-atom.
- each  $CB_j$  is a DL predicate,  $CB_j(Z_j)$  is called a dl-atom.
- (DL-safeness) every variable of  $\mathcal{R}$  must appear in at least one of the  $RB_i(Y_i)$  ( $1 \leq i \leq m$ ).

In this version of our work, we only consider  $\mathcal{DL}log$  with DL-safeness. Introducing weak safeness[19] will be included in our future work. Besides, we use  $M(\mathcal{P})$  to denote the set of DL predicates occurring under "not" in  $\mathcal{P}$ .

Let  $P$  and  $Z$  be two disjoint sets of predicates and  $A$  is the first order theory containing  $P$  and  $Z$ . Use  $CIRC[A; P; Z]$  to denote the parallel circumscription of  $A$ , in which the extensions of the predicates in  $P$  are minimized and the interpretations of the predicates in  $Z$  are fixed[10, 14]. We use  $CIRC[A; P]$  to denote the parallel circumscription  $CIRC[A; P; Z]$  with  $Z = \emptyset$ , indicating that all other predicates' interpretations can vary to support the minimization.

**Semantics**

Like  $\mathcal{DL} + log$ , in the definition of the semantics of  $\mathcal{DL}log$  we adopt *Standard Names Assumption*[18, 19]: every interpretation is defined over the same fixed, countably infinite domain  $\Delta$ , and the alphabet of constant  $\mathcal{C}$  is such that it is in the same one-to-one correspondence with  $\Delta$  in each interpretation. Under SNA, with a bit abuse of the notation, we can use the same symbol to denote both a constant and its semantic interpretation, and reformulate the notion of satisfaction in FOL accordingly. Such reformulation doesn't change any standard DLs' consequences. See [18, 12] for the details.

Given a set of constants  $\mathcal{C}$ , the *ground instantiation of  $\mathcal{P}$  with respect to  $\mathcal{C}$* , denoted by  $gr(\mathcal{P}, \mathcal{C})$ , is the program obtained from  $\mathcal{P}$  by replacing every rule  $\mathcal{R}$  in  $\mathcal{P}$  with the set of rules obtained by applying all possible substitutions of variables in  $\mathcal{R}$  with constants in  $\mathcal{C}$ .

Based on the motivation discussed in Section 2, we present the definition of semantics. Following is a generalization of the projection  $\Pi$  in  $\mathcal{DL} + log$  to the case of negative dl-atoms.

**Definition 2.** Given an interpretation  $\mathcal{I}$  over an alphabet of predicates  $\Sigma' \subset \Sigma$  and a ground program  $gr(\mathcal{P}, \mathcal{C})$  over the predicates in  $\Sigma$ , the projection of  $gr(\mathcal{P}, \mathcal{C})$  with respect to  $\mathcal{I}$ , denoted by  $\Pi(gr(\mathcal{P}, \mathcal{C}), \mathcal{I})$  is the program obtained from  $gr(\mathcal{P}, \mathcal{C})$  as follows. Let  $r(t)$  denote the literal (either unary or binary) in  $gr(\mathcal{P}, \mathcal{C})$ , for each rule  $\mathcal{R} \in gr(\mathcal{P}, \mathcal{C})$ ,

- delete  $\mathcal{R}$  if there exists an atom  $r(t)$  in the head of  $\mathcal{R}$  such that  $r \in \Sigma'$  and  $t \in r^{\mathcal{I}}$ ;
- delete each atom  $r(t)$  in the head of  $\mathcal{R}$  such that  $r \in \Sigma'$  and  $t \notin r^{\mathcal{I}}$ ;
- delete  $\mathcal{R}$  if there exists a positive literal  $r(t)$  in the body of  $\mathcal{R}$  such that  $r \in \Sigma'$  and  $t \notin r^{\mathcal{I}}$ ;
- delete each positive literal  $r(t)$  in the body of  $\mathcal{R}$  such that  $r \in \Sigma'$  and  $t \in r^{\mathcal{I}}$ ;
- delete  $\mathcal{R}$  if there exists a negative atom  $\text{not } r(t)$  in the body of  $\mathcal{R}$  such that  $r \in \Sigma'$  and  $t \in r^{\mathcal{I}}$ ;
- delete each negative atom  $\text{not } r(t)$  in the body of  $\mathcal{R}$  such that  $r \in \Sigma'$  and  $t \notin r^{\mathcal{I}}$ ;

Based on this definition, dl-atoms can be eliminated by projecting with an interpretation on  $\Sigma_R \cup \Sigma_C$ . Then one can obtain its stable model via *Gelfond-Lifschitz reduction* [7]. In following we define the Nonmonotonic Circumscriptive Semantics (NMC-Semantics) for  $\mathcal{DLclog}$ .

**Definition 3.** (*Nonmonotonic Circumscriptive Semantics, NMC-Semantics*) For a hybrid KB  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$  and  $\mathcal{C}$  the set of individuals explicitly stated in  $\mathcal{O}$ , let  $\mathcal{U}, \mathcal{V}, \mathcal{W}$  be sets of interpretations on language  $\Sigma_C \cup \Sigma_R, \Sigma_C \cup \Sigma_R$  and  $\Sigma_D$ , respectively. A structure  $\mathcal{M} = (\mathcal{U}, \mathcal{V}, \mathcal{W})$  is the Nonmonotonic Circumscriptive Model (NMC-Model) of  $\mathcal{K}$ , denoted as  $\mathcal{M} \models_{NMC} \mathcal{K}$ , if and only if

- for each  $\mathcal{I} \in \mathcal{U}, \mathcal{I} \models \mathcal{O}$ .
- for each  $\mathcal{I} \in \mathcal{V}, \mathcal{I} \models \text{CIRC}[\mathcal{O}; M(\mathcal{P})]$ .
- for each  $\mathcal{J} \in \mathcal{W}, \mathcal{J}$  is a stable model of  $\Pi(\text{gr}(\mathcal{P}, \mathcal{C}), \mathcal{I}_c)$  where  $\mathcal{I}_c \in \mathcal{V}$ .

We call  $\mathcal{U}, \mathcal{V}, \mathcal{W}$  the classical part, circumscriptive part, and stable part of the NMC-model.  $\mathcal{K}$  is NMC-satisfiable if and only if it has an NMC-model without any part as  $\emptyset$ .  $c$  denotes a tuple of constants. A ground atom  $p(c)$  is NMC-entailed by  $\mathcal{K}$ , denoted as  $\mathcal{K} \models_{NMC} p(c)$ , if and only if

- if  $p$  is a DL predicate, for each interpretation  $\mathcal{I} \in \mathcal{U}, \mathcal{I} \models p(c)$ .
- if  $p$  is a rule predicate, for each interpretation  $\mathcal{J} \in \mathcal{W}, \mathcal{J} \models p(c)$ .

We use an example to illustrate this treatment.

*Example 3.* We use the hybrid KB of Example 1, and analyze three cases.

1. Querying  $\text{seasideCity}(\text{Barcelona})$ . With NMC-Semantics, negative dl-atom "not notSC(x)" is satisfied in circumscriptive models containing  $\neg \text{notSC}(\text{Barcelona})$ , in which  $\text{notSC}$  is circumscribed. This is the only model used for evaluating dl-atoms. Thus we obtain  $\mathcal{K} \models_{NMC} \text{seasideCity}(\text{Barcelona})$ .
2. Query  $\text{notSC}(\text{Barcelona})$ . As in Example 1,  $\mathcal{O}$  have two models  $\mathcal{I}_1$  and  $\mathcal{I}_2$ . Thus we have  $\mathcal{O} \not\models_{NMC} \text{notSC}(\text{Barcelona})$  rather than  $\mathcal{O} \models_{NMC} \neg \text{notSC}(\text{Barcelona})$ . We can see that in NMC-Semantics, circumscriptive models don't affect reasoning with  $\mathcal{O}$ , which remains to be monotonic and classical.
3. Circumscriptive models can introduce nonmonotonicity to reasoning with rules by negative dl-atoms, Once we add "notSC(Barcelona)" into  $\mathcal{O}$ , we obtain  $\mathcal{K} \models_{NMC} \neg \text{seasideCity}(\text{Barcelona})$ .

NMC-Semantics is the generalization of  $\mathcal{DL}+log$ . When there are no negative dl-atoms, we have  $\mathcal{U} = \mathcal{V}$ , and  $\mathcal{DL}clog$  is reduced to  $\mathcal{DL} + log$  in both syntax and semantics. Obviously, we have the following result.

**Proposition 1.** *For a  $\mathcal{DL}clog$  KB  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ , when there are no negative dl-atoms in  $\mathcal{P}$ ,  $\mathcal{K}$  is NMC-satisfiable iff  $\mathcal{K}$  is NM-satisfiable in the sense of  $\mathcal{DL}+log$ .*

NMC-Semantics is also faithful to DL and LP, in that DL and LP are the restricted forms of  $\mathcal{DL}clog$ .

**Proposition 2.** *For a  $\mathcal{DL}clog$  KB  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ , (i) if  $\mathcal{P} = \emptyset$ ,  $\mathcal{K}$  is NMC-satisfiable iff  $\mathcal{O}$  is classically satisfiable. (ii) if  $\mathcal{O} = \emptyset$ ,  $\mathcal{K}$  is NMC-satisfiable iff  $\mathcal{P}$  has stable model(s).*

NMC-Semantics is nonmonotonic. Negative dl-atoms evaluated by circumscriptive models add nonmonotonicity features to rules. Consequently, reasoning with DL is monotonic and with OWA, while reasoning with rules is nonmonotonic and with CWA.

**Proposition 3.** *Given a  $\mathcal{DL}clog$  KB  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ ,  $\mathcal{O}'$ ,  $\mathcal{P}'$  are sets of DL formulae and rules such that  $\mathcal{O} \cup \mathcal{O}'$  and  $\mathcal{P} \cup \mathcal{P}'$  are consistent.  $\mathcal{K}' = (\mathcal{O} \cup \mathcal{O}', \mathcal{P} \cup \mathcal{P}')$ , and  $p(c)$  is a query such that  $\mathcal{K} \models_{NMC} p(c)$ . Then (i)  $\mathcal{K}' \models_{NMC} p(c)$  holds if  $p(c)$  is a DL query. (ii)  $\mathcal{K}' \models_{NMC} p(c)$  may not hold if  $p(c)$  is a rule query.*

## 4 Decidability and Complexity

In this section, we restrict  $\mathcal{DL}clog$  to the case that the DL ontology is written in  $\mathcal{ALCI}\mathcal{O}$  or  $\mathcal{ALCQ}\mathcal{O}$ , and roles are not allowed under "not" in rules. We leave the decidability of general  $\mathcal{DL}clog$  relative to NMC-Satisfiability as an open problem.

As the satisfiability of Datalog <sup>$\neg, \vee$</sup>  program  $\mathcal{P}$ , which is  $NEXP^{NP}$ -Complete in [4], can be trivially reduced the NMC-satisfiability of  $(\emptyset, \mathcal{P})$ , we have following proposition.

**Proposition 4.** *NMC-satisfiability of  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$  is  $NEXP^{NP}$ -hard.*

Furthermore, we remind of the exponential hierarchy  $NEXP \subseteq NP^{NEXP} \subseteq NEXP^{NP} \subseteq 2-EXP$ , and obtain following result.

**Proposition 5.** *For a restricted  $\mathcal{DL}clog$  KB  $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ , deciding satisfiability of NMC-Semantics of  $\mathcal{K}$  is  $NEXP^{NP}$ -complete.*

*Proof.* (Sketch) NMC-Satisfiability can be determined by calling three oracles in an non-deterministic framework (due to the consideration of space, the proof of correctness is omitted here.). For all the ground dl-atoms in  $gr(\mathcal{P}, \mathcal{C})$ , guess its partition  $(G_P, G_N)$ . (1) Check the satisfiability of following  $\mathcal{ALCI}\mathcal{O}$  or  $\mathcal{ALCQ}\mathcal{O}$  KB:  $\mathcal{O} \cup \{C(a) | C(a) \in G_P\} \cup \{\neg C(a) | C(a) \in G_N\} \cup \{\exists R.b(a) | R(a, b) \in G_P\} \cup \{\neg \exists R.b(a) | R(a, b) \in G_N\}$  in PSPACE or  $NEXP^{NP}$  [20]. (2) The dl-atoms in

$gr(\mathcal{P}, \mathcal{C})$  can be evaluated with  $(G_P, G_N)$  in polynomial time, and obtain a program  $\mathcal{P}_D$  without dl-atoms. Checking the existence of stable models for  $\mathcal{P}_D$  is in  $NEXP^{NP}$  [4]. (3) Let  $G_{NP} = \{r(t) | r \in M(\mathcal{P}) \wedge r(t) \in G_P\}$ ,  $G_{NN} = \{r(t) | r \in M(\mathcal{P}) \wedge r(t) \in G_N\}$  and  $\neg G_{NN} = \{\neg r(t) | r(t) \in G_{NN}\}$ , check whether there exists an interpretation  $\mathcal{I}_c$  such that  $\mathcal{I}_c \models CIRC[\mathcal{O}; M(\mathcal{P})]$  and  $G_{NP} \cup \neg G_{NN} \subseteq \mathcal{I}_c$  is  $NEXP^{NP}$  by being reduced to checking the satisfiability of  $\mathcal{ALC}\mathcal{IO}$  or  $\mathcal{ALC}\mathcal{QO}$  with counting formulae [1]. As it is a non-deterministic process, we obtain the upper bound of  $NEXP^{NP}$ . Together with Proposition 4, we finish the proof for completeness.

## 5 Related Work

$\mathcal{DL}clog$  follows from a series of "r-hybrid" work [16–19] which are hybrid approaches defining integrated models on the basis of single models of classical theory. We inherit the framework of these methods. By introducing negative dl-atoms and use parallel circumscription to evaluate them, we obtain the semantics with nonmonotonic features treating negative dl-atoms, while it remains faithful to previous work and to both DL and LP.

In dl-program [3], DL predicates in rules are treated as queries to the ontology, in which the evaluation of DL-atoms in rules are actually by entailment of DL ontology rather than models, as in our method. Thus, the semantics framework is different from ours. Besides,  $\mathcal{DL}clog$  cannot pass facts from rules to DL ontology.

CLP [8] is a method by introducing open domain to Answer Set Programming. It allows DL-predicates occurring under "not" in rules, but the models of rules must be organized in tree-like manner to obtain decidability. Finally, CLP can be used to simulate several expressive DLs. This is a homogenous method.

Recently, there are some "full-integration" methods proposed. [2] proposed to use First Order Autoepistemic Logic [5] as a host language to accommodate DL and rules. [12, 13] proposed to build a hybrid KB in the framework of the logic of MKNF. Both of the methods treat DL and LP in a uniform logic rather than integrating existing formalisms, and by introducing modal operators, dl-atoms are treated in precisely the same way as rule-atoms. Compared with these work, instead of extending language, our formalism is based on a hybrid, modular semantics integrating classical semantics, circumscription and stable model.

## 6 Conclusion and Future Work

In this paper, we present a hybrid formalism  $\mathcal{DL}clog$  as both semantic and syntactic extension of Rosati's  $\mathcal{DL} + log$  by allowing negative dl-atoms occurring in the body of the rules. To obtain the stable models of the rules, dl-atoms are evaluated by the circumscriptive models of the DL ontology in the sense of parallel circumscription. In this way, the negative dl-atoms are treated nonmonotonically and is closely similar to the treatment of "not" in LP. This formalism strengthens the nonmonotonic expressing and reasoning ability of  $\mathcal{DL} + log$ , and remains faithful to the NM-Semantics of  $\mathcal{DL} + log$ , DL and LP. Besides, when

roles do not occur under “not” in rules and ontologies are written in *ALCLO* and *ALCQO*, NMC-satisfiability is a  $\text{NEXP}^{\text{NP}}$ -complete problem.

As our future work, we will compare *DLclog* with full integration methods.

## References

1. Piero Bonatti, Carsten Lutz and Frank Wolter: Description Logics with Circumscription in Proc. of KR06. 2006.
2. J.Bruijn, T.Eiter, A.Pollerer, and H.Tompits: Embedding Non-Ground Logic Programs into Autoepistemic Logic for Knowledge-Base Combination, in Proc of IJCAI2007, 2007.
3. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. In: Proc. of KR04.
4. Eiter, T., Gottlob, G., and Manilla, H. 1997. Disjunctive Datalog, ACM Trans. on Database Systems 22(3):364-418. 1997
5. R.Moore. Semantical considerations on nonmonotonic logic. in AIJ,25(1):75-94, 1985.
6. Michael Gelfond, Halina Przymusinska and Teodor Przymusinski: On the Relationship between Circumscription and Negation as Failure, in AIJ38(1989) 75-94, 1989.
7. Michael Gelfond, Vladimir Lifschitz: Classical Negation in Logic Programs and Disjunctive Databases. in New Generation Computing 9:365-385. 1991.
8. Heymans, S., Nieuwenborgh, D. V., and Vermeir, D.: Semantic Web Reasoning with Conceptual Logic Programs. In Proc. of RuleML2004, 2004.
9. Fangzhen Lin and Yoav Shoham: A Logic of Knowledge and Justified Assumption, in AIJ57(1992),271-289.
10. McCarthy, J. 1986. Applications of circumscription in formalizing common sense knowledge. Artif. Intell. 28:89C 116.
11. Motik, B.,Sattler, U., Studer, R.: Query answering for OWL-DL with rules. In: Proceedings ISWC2004, 2004.
12. Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler: Can OWL and Logic Programming Live Together Happily Ever After? in Proc. of ISWC06, Springer-Verlag, 2006.
13. Boris Motik, Riccardo Rosati: A Faithful Integration of Description Logics with Logic Programming. in Proc. of IJCAI2007.
14. V.Lifschitz: Circumscription, In The Handbook of Logic in AI and Logic Programming. Oxford University Press. 298-352.
15. R.Reiter: On Closed-world Data Bases, in H.Gallaire and J.Minker(Eds.) Logic and Data Bases(Plenum Press, New York,1978)55-76.
16. R.Rosati: Towards expressive KR systems integrating Datalog and description logics: Preliminary report. In Proc. of DL99', 1999.
17. R.Rosati: Semantic and Computational Advantages of the Safe Integration of Ontologies and Rules. In Proc. PPSWR2005, LNCS3703, Springer Verlag, 2005.
18. Rosati, R.: On the decidability and complexity of integrating ontologies and rules. Journal of Web Semantics 3(1) (2005) 61-73 9.
19. Rosati, R.: *DL+log*: Tight integration of description logics and disjunctive datalog. In: KR2006. (2006)
20. Stephan Tobies. Complexity Results and Practical Algorithms for Logics in Knowledge Representation. PhD thesis, 2001.

## A Boolean Lattice Based Fuzzy Description Logic in Web Computing

Changli Zhang<sup>1</sup>, Jian Wu<sup>1</sup>, Zhengguo Hu<sup>1</sup>

<sup>1</sup> Department of Computer Science and Engineering,  
Northwestern Polytechnical University, Xi'an 710072, China  
[charlieii@163.com](mailto:charlieii@163.com)

**Abstract.** Contrasting to Description Logics (DLs), there are some inherent shortcomings in classical fuzzy Description Logics (FDLs). For example, they no longer satisfy the complementary laws. In this paper, to analyze these shortcomings derived from Zadeh semantics, an improved fuzzy set definition using Boolean lattices is approached, whereas, the analogous definition of Zadeh Fuzzy Set can only use a distributive lattice. Therefore, an improved FDL in Web Computing Environment, using Boolean lattices rather than the interval  $[0,1]$  to represent fuzziness, is constructed. Finally, some probable extensions of the chosen lattice are discussed.

**Keywords:** Fuzzy Description Logic, Zadeh Fuzzy Set, Web Computing.

### 1 Introduction

Web Computing are the ways we managing, organizing, sharing and exchanging web resources (e.g. data and services) on the Web by means of web technologies, e.g. Semantic Web and Web Services. Among these technologies, Description Logics (DLs) [1] provide a logical reconstruction of object-centric and frame-based knowledge representation languages as essential means to describe web resources and to infer on them; Furthermore, Fuzzy Description Logics (FDLs) [2] extend DLs with capabilities of representing and reasoning on fuzziness of Web resources.

In this paper, rather than going on advocating their advantages, we will talk about some shortcomings of classical FDLs and try to find a solution. We proceed as follows. In the following section, taking  $f\text{-}\mathcal{ALC}$  [2] as example, we focus on classical FDLs' shortcomings (e.g. unsatisfactory of the complementary laws) derived from their Zadeh semantics. Then, in Section 3, an improved FDL in Web Computing, using a proper Boolean lattice rather than the interval  $[0,1]$  to represent fuzziness, is constructed. In Section 4, some probable extensions of the chosen lattice are discussed. At last, Section 5 concludes and presents some topics for further research.

## 2 Analysis on Shortcomings of Classical FDLs

### 2.1 The Problems

As mentioned in [2], complementary laws never hold in classical FDLs. To make it clear, let's take  $f\text{-}\mathcal{A}\mathcal{L}\mathcal{C}$ , a simple FDL whose detail can be found in [2], as an illustration. In  $f\text{-}\mathcal{A}\mathcal{L}\mathcal{C}$ , a fuzzy interpretation  $\mathcal{I}$  satisfies some Zadeh style equations like: for all  $d \in \Delta^{\mathcal{I}}$ ,  $(\perp)^{\mathcal{I}}(d) = 0$ ,  $(\top)^{\mathcal{I}}(d) = 1$ ,  $(C \sqcap D)^{\mathcal{I}}(d) = \min(C^{\mathcal{I}}(d), D^{\mathcal{I}}(d))$ ,  $(C \sqcup D)^{\mathcal{I}}(d) = \max(C^{\mathcal{I}}(d), D^{\mathcal{I}}(d))$ , and  $(\neg C)^{\mathcal{I}}(d) = 1 - C^{\mathcal{I}}(d)$ . Obviously,  $(C \sqcap \neg C)^{\mathcal{I}}(d) = 0$  and  $(C \sqcup \neg C)^{\mathcal{I}}(d) = 1$  no longer hold. That is,  $C \sqcap \neg C \neq \perp$  and  $C \sqcup \neg C \neq \top$ . What's more, serious to say, this is not the only issue, there are more implicit problems in classical FDLs, as depict in following examples.

**Example 1.** Consider the fuzzy KB  $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A}_1)$ , where

$$\begin{aligned} \mathcal{T}_1 &= \{Fowl \sqsubseteq Animal, Reptile \sqsubseteq Animal, Fowl \sqcap Reptile \sqsubseteq \perp\}, \\ \mathcal{A}_1 &= \{archaeopteryx:Fowl = 0.6, archaeopteryx:Reptile = 0.3\}. \end{aligned}$$

Suppose  $\alpha_1 = archaeopteryx:Fowl \sqcap Reptile$  and  $\alpha_2 = archaeopteryx:Fowl \sqcup Reptile$ . Easy to see, the conclusion  $\alpha_1 = \min\{0.6, 0.3\} = 0.3$  conflicts with the fuzzy axiom  $Fowl \sqcap Reptile \sqsubseteq \perp$ . The other conclusion  $\alpha_2 = \max\{0.6, 0.3\} = 0.6$  conflicts with the more reasonable fuzzy value  $0.6 + 0.3 = 0.9$ . Therefore, conjunction and disjunction of fuzzy concepts have the probability of causing conflicts in fuzzy KBs which are originally consistent.

**Example 2.** Consider another fuzzy KB  $\mathcal{K}_2 = (\mathcal{T}_2, \mathcal{A}_2)$ , where  $\mathcal{T}_2 = \{C \sqcap D \sqsubseteq \perp\}$  and every model  $\mathcal{I}$  of  $\mathcal{K}_2$  should satisfy (i) for all  $d \in \Delta^{\mathcal{I}}$ ,  $C^{\mathcal{I}}(d) \leq D^{\mathcal{I}}(d)$ ; (ii) there always exists a  $d \in \Delta^{\mathcal{I}}$  so that  $C^{\mathcal{I}}(d) > 0$ . We can infer that  $C \sqsubseteq D$  and  $C \neq \perp$ ,  $D \neq \perp$ , which contradict the axiom  $C \sqcap D \sqsubseteq \perp$ . In conclusion, the inclusion of fuzzy concepts sometime also leads to conflicts in originally consistent fuzzy KBs.

Well then, where do these problems lie in? We can find the answer in the Zadeh semantics of classical FDLs.

### 2.2 The Reason of Classical FDLs' Shortcomings

Coincidentally, Zadeh fuzzy set, under which the semantics of classical FDLs are defined, encounters same criticisms and should be treated as the root of abovementioned problems. To find the reason, we sketch a more sophisticated fuzzy set using Boolean lattice rather than the interval  $[0, 1]$  to represent fuzziness.

**Definition 1 (Boolean Lattice Based Fuzzy Set).** Suppose  $X$  is the universe, and suppose  $(\mathcal{L}, \preceq)$  is a Boolean lattice with operators as  $\otimes$ ,  $\oplus$ ,  $\ominus$ , and identity elements as  $0$ ,  $1$ , respectively. Then a *Boolean lattice based fuzzy set* (e.g.  $C$  and  $D$ ) is a mapping from  $X$  to  $\mathcal{L}$ , satisfying: (i) for each  $x \in X$ ,  $\emptyset(x) = 0$ ,

$X(x) = 1$  and  $(\sim C)(x) = \ominus C(x)$ ,  $(C \cap D)(x) = C(x) \otimes D(x)$ ,  $(C \cup D)(x) = C(x) \oplus D(x)$ ;  
 (ii)  $C \subseteq D$  iff for all  $x \in X$ ,  $C(x) \preceq D(x)$ .

Here, for a fuzzy set  $C$ ,  $C(x)(x \in X)$  is a *Boolean lattice based membership degree* of  $x$  to  $C$ . What's more, we use a function  $\sigma: \mathcal{L} \rightarrow [0,1]$  mapping Boolean lattice based membership degree (e.g.  $C(x)$ ) to Zadeh membership degree (e.g.  $\mu_c(x) = \sigma(C(x))$ ). Obviously, the Boolean lattice based fuzzy set satisfies all the set rules like a crisp set ever do, and the reason lies in its ability of modeling disjunction and inclusion relationships between fuzzy sets. Analogous to *Definition 1*, we can redefine Zadeh fuzzy set using a certain distributive lattice.

**Definition 2 (Zadeh Fuzzy Set).** Suppose  $X$  is the universe, and suppose  $(\mathcal{L}', \preceq)$  is a lattice, all of whose elements are of format  $[0, a]$  ( $a \in [0,1]$ ) and satisfies:  $[0, a_1] \preceq [0, a_2]$  iff  $a_1 \leq a_2$ . Then a *Zadeh fuzzy set*  $C$  can be represented as a mapping  $C: X \rightarrow \mathcal{L}'$ , satisfying: for any  $x \in X$ ,  $C(x) = [0, \mu_c(x)]$ .

Isomorphic to interval  $[0,1]$ , the lattice  $\mathcal{L}'$  is no longer a Boolean lattice, because every element inside doesn't have its inverse element. What's more, since  $\mathcal{L}'$  is also a linear order, we cannot model disjunction relationships between fuzzy sets. Therefore, in classical FDLs, Zadeh semantics blurs the overlapping relationships (e.g. disjunction, inclusion) between fuzzy concepts, and thus loses the information of these relationships. Essentially, these are the reason of abovementioned problems in classical FDLs.

To make improvements, some papers introduces other semantics of fuzzy connectives (e.g. Lukasiewicz semantics) [4], others using lattice representing fuzziness [5,6]. According *Definition 1*, to represent fuzziness, it's more applicable to use Boolean lattice than the general ones. In the following section, a proper Boolean lattice in Web Computing environment is founded and is used to build a Boolean lattice based FDL named lf- $\mathcal{A}\mathcal{L}\mathcal{C}$ .

### 3 lf- $\mathcal{A}\mathcal{L}\mathcal{C}$ : a Boolean Lattice Based FDL in Web Computing

#### 3.1 Choosing of Boolean Lattice

Consider fuzzy KB  $\mathcal{K}_1$  in *Example 1*, how do we deem that an archaeopteryx is a fowl in a degree about 60%, and is a reptile in 30%. Commonly, this is done in a voting approach. Suppose we sampled a certain number of (e.g. 100) famous paleontologists. Because of their different backgrounds, different major and different research experiences, even different believes, it is reasonable that their viewpoints on whether archaeopteryx belongs to fowl or reptile will vary each other. And, the last thing to do is to collect and synthesize their different viewpoints into a more reasonable fuzzy result (e.g. the fuzzy assertions in  $\mathcal{K}_1$ ).

Currently, for the sake of the universality of web entities (i.e. almost every people has their agents on the Web delegate their words and actions<sup>1</sup>) and the connectedness among them, such a voting approach can be enforced in Web Computing environment. A decision-making entity (called decision-maker) can put forwards its questionable assertion (e.g. whether archaeopteryx belongs to fowl) to some chosen entities (called observers, accordingly) and receives their responses (i.e. their viewpoint). Suppose  $O = \{o_1, o_2, \dots, o_n\}$  ( $n \geq 1$ ) is the set of chosen observers, then we have a Boolean lattice  $(\mathcal{L}_o, \subseteq)$  as  $O$ 's power set. Moreover, suppose, each element of  $\mathcal{L}_o$  is the set of those observers who are for the questionable assertion, then  $\mathcal{L}_o$  can be a proper Boolean lattice for fuzziness representation in Web Computing environment.

### 3.2 Syntax, Semantics and Inference Problems of If- $\mathcal{ALC}$

Here, by contrast to f- $\mathcal{ALC}$ , If- $\mathcal{ALC}$  is built on Boolean lattice  $\mathcal{L}_o$ . To make it general, we rename  $(\mathcal{L}_o, \subseteq)$  as  $(\mathcal{L}, \preceq)$ , and rename its set operators as  $\otimes, \oplus, \ominus$ , it's identity elements  $\emptyset, O$  as  $o, 1$ , respectively. In this way, we can use another better Boolean lattice to substitute  $\mathcal{L}_o$  whenever necessary.

Consider three alphabets of symbols, *primitive concepts* (denoted  $A$ ), *primitive roles* (denoted  $R$ ), and *individuals* (denoted  $a$  and  $b$ )<sup>2</sup>. A *concept* (denoted  $C$  or  $D$ ) of If- $\mathcal{ALC}$  is build out according to the following syntax rules:

$$C, D \rightarrow \perp | \top | A | C \sqcap D | C \sqcup D | \neg C | \forall R.C | \exists R.C$$

A *Boolean Lattice based fuzzy interpretation* of If- $\mathcal{ALC}$  is now a pair  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ , where  $\Delta^{\mathcal{I}}$  is as for the crisp case, the *domain*, whereas  $\cdot^{\mathcal{I}}$  is an *Boolean lattice based interpretation function* mapping (i) individuals as for the crisp case, i.e.,  $a^{\mathcal{I}} \neq b^{\mathcal{I}}$  if  $a \neq b$ ; (ii) a fuzzy concept  $C$  into a Boolean Lattice based membership function  $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \mathcal{L}$ ; (iii) a fuzzy role  $R$  into a Boolean Lattice based membership function  $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow \mathcal{L}$ .

For each individual  $a$  (resp. each individual pair  $(a, b)$ ),  $C^{\mathcal{I}}(a^{\mathcal{I}})$  (resp.  $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}})$ ) represents those observers who is for the crisp assertion  $a : C$  (resp.  $(a, b) : R$ ) w.r.t. interpretation  $\mathcal{I}$ . Furthermore, we use  $\mu_C^{\mathcal{I}}(a^{\mathcal{I}})$  as the Zadeh membership degree of  $a$  belongs to  $C$  w.r.t.  $\mathcal{I}$ . Reasonably, we have

$$\mu_C^{\mathcal{I}}(a^{\mathcal{I}}) = \frac{|C^{\mathcal{I}}(a^{\mathcal{I}})|_3}{\log_2 |\mathcal{L}|} \tag{1}$$

<sup>1</sup> Of cause, we are somewhat overstated, but we trust such a situation is expectable.

<sup>2</sup> Metavariables may have a subscript or superscript.

<sup>3</sup> To a crisp set  $X$ ,  $|X|$  means the cardinality of  $X$ .

as the proportion between those observers who are for  $a:C$  and the overall observers. Similarly,  $\mu_R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}})$ , the Zadeh membership degree w.r.t.  $\mathcal{I}$ , should be

$$\mu_R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) = \frac{|R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}})|}{\log_2 |\mathcal{L}|} . \quad (2)$$

Additionally, the Boolean lattice based interpretation function  $\bullet^{\mathcal{I}}$  has to satisfies the following equations: for all  $d \in \Delta^{\mathcal{I}}$ ,  $(\perp)^{\mathcal{I}}(d) = 0$ ,  $(\top)^{\mathcal{I}}(d) = 1$ , and

$$\begin{aligned} (C \sqcap D)^{\mathcal{I}}(d) &= C^{\mathcal{I}}(d) \otimes D^{\mathcal{I}}(d) \\ (C \sqcup D)^{\mathcal{I}}(d) &= C^{\mathcal{I}}(d) \oplus D^{\mathcal{I}}(d) \\ (\neg C)^{\mathcal{I}}(d) &= \ominus C^{\mathcal{I}}(d) \\ (\forall R.C)^{\mathcal{I}}(d) &= \bigotimes_{d' \in \Delta^{\mathcal{I}}} (\ominus R^{\mathcal{I}}(d, d') \oplus C^{\mathcal{I}}(d')) \\ (\exists R.C)^{\mathcal{I}}(d) &= \bigoplus_{d' \in \Delta^{\mathcal{I}}} (R^{\mathcal{I}}(d, d') \otimes C^{\mathcal{I}}(d')) \end{aligned}$$

Note that for an individual  $a$ ,  $(\forall R.C)^{\mathcal{I}}(a^{\mathcal{I}})$  represents those observers who believe that either there's no other individual  $b$  satisfies the assertion  $(a,b):R$ , or every individual  $b$  satisfying  $(a,b):R$  also satisfies  $b:C$ . Accordingly,  $(\exists R.C)^{\mathcal{I}}(a^{\mathcal{I}})$  represents those observers who believe that there exists an individual  $b$  satisfies both  $(a,b):R$  and  $b:C$ . Obviously, lf- $\mathcal{ALC}$  satisfies all rules like the crisp case.

A *terminological axiom* is either a *fuzzy concept specialization* of the form  $A \sqsubseteq C$ , or a *fuzzy concept definition* of the form  $A \equiv C$ . An interpretation  $\mathcal{I}$  *satisfies*  $A \sqsubseteq C$  iff for all  $d \in \Delta^{\mathcal{I}}$ ,  $A^{\mathcal{I}}(d) \preceq C^{\mathcal{I}}(d)$ ; Similarly for  $A \equiv C$ . A *fuzzy assertion* is an expression of the form  $\langle \alpha \succeq c_1 \rangle$  or  $\langle \alpha' \preceq c_2 \rangle$ , where  $\alpha$  is an crisp assertion,  $c_1$  and  $c_2$  are values in  $\mathcal{L}$ , but  $\alpha'$  is an crisp assertion of the form  $a:C$  only. An interpretation  $\mathcal{I}$  *satisfies*  $\langle a:C \succeq c \rangle$  (resp.  $\langle (a,b):R \succeq c \rangle$ ) iff  $C^{\mathcal{I}}(a^{\mathcal{I}}) \succeq c$  (resp.  $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \succeq c$ ). Similarly for  $\preceq$ .

A *fuzzy Knowledge Base* (KB) is pair  $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ , where  $\mathcal{T}$  and  $\mathcal{A}$  are finite sets of fuzzy terminological axioms and fuzzy assertions, respectively. An interpretation  $\mathcal{I}$  *satisfies* (is a *model* of)  $\mathcal{T}$  (resp.  $\mathcal{A}$ ) iff  $\mathcal{I}$  satisfies each element in  $\mathcal{T}$  (resp.  $\mathcal{A}$ ), while  $\mathcal{I}$  *satisfies* (is a *model* of)  $\mathcal{K}$  iff  $\mathcal{I}$  satisfies both  $\mathcal{T}$  and  $\mathcal{A}$ . A fuzzy KB  $\mathcal{K}$  *fuzzy entails* a fuzzy assertion  $\langle \alpha \succeq c \rangle$  (denoted  $\mathcal{K} \models \langle \alpha \succeq c \rangle$ ) iff every model of  $\mathcal{K}$  also satisfies  $\langle \alpha \succeq c \rangle$  (similarly for  $\preceq$ ). What's more, to the inference problems, the extended Tableau algorithm with complexity of PSPACE-complete in [5] is also valid for lf- $\mathcal{ALC}$ , since lf- $\mathcal{ALC}$  can be treated as a special case of the lattice based FDLs in this paper.

**Example 3.** Suppose  $O = \{o_1, o_2, o_3\}$ , consider the fuzzy KB  $\mathcal{K}_3 = (\emptyset, \mathcal{A}_3)$ , where

$$\mathcal{A}_3 = \{\text{archaeopteryx:Fowl}=\{o_1, o_2\}, \text{archaeopteryx:Reptile}=\{o_3\}\} .$$

Then  $\mu_{Fowl}^{\mathcal{I}}(archaeopteryx^{\mathcal{I}}) = 2/3 \approx 0.6$  ,  $\mu_{Reptile}^{\mathcal{I}}(archaeopteryx^{\mathcal{I}}) = 1/3 \approx 0.3$  . And, suppose  $\alpha = archaeopteryx:\neg Reptile$  , we have  $\mathcal{K}_3 \models \langle \alpha = \{o_1, o_2\} \rangle$  .

#### 4 Extensions of the Boolean Lattice in If- $\mathcal{ALC}$

In Web Computing, to the decision-maker’s aspect, it’s reasonable to prefer one observer than another. Therefore, we can attach each observer a weight representing the decision-maker’s preference. For example, suppose we have  $n$  ( $n \geq 1$ ) observers  $O = \{o_1, o_2, \dots, o_n\}$  and a weight function  $\gamma : O \rightarrow [0,1]$  assigning each observer a proper weight, then, to a crisp assertion  $a : C$  , the Zadeh membership degree w.r.t. interpretation  $\mathcal{I}$  becomes

$$\mu_C^{\mathcal{I}}(a^{\mathcal{I}}) = \frac{\sum_{o \in C^{\mathcal{I}}(a^{\mathcal{I}})} \gamma(o)}{\sum_{o \in O} \gamma(o)} . \tag{3}$$

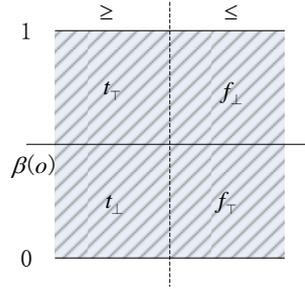
In addition, it’s hard to mandate observers to only provide unambiguous answers. In most cases, the observers can only give fuzzy viewpoint, e.g. an archaeopteryx is only about 60% possibilities to be a fowl. As a result, it is necessary to transform each observer’s fuzzy opinion (e.g.  $\langle \alpha \geq 0.7 \rangle$ ) in to a Boolean lattice format, although the original Zadeh style opinion has already lost the information of set overlapping relationships (as mentioned in Section 2). Here we using a simple four-valued Boolean lattice  $(\mathcal{L}_4, \leq_4)$  (with operators as  $\otimes_4, \oplus_4, \ominus_4$ ), customized from Belnap’s four-valued logic [7], to illustrate the transforming process.

Before talking about  $\mathcal{L}_4$ , we first introduce some components that compose the elements of it. Firstly, the elements of set  $\{t, f\}$  represent the attitude of each observer: for or against. Secondly, the elements of set  $\{\top, \perp\}$ <sup>4</sup> represent the confidence degree of the observer:  $\top$  means that he is sure on his attitude because he has enough information (i.e. proof), and  $\perp$  means he is unsure on his attitude because his information (i.e. proof) is relatively insufficient. Thus, we get elements in  $\mathcal{L}_4$  as  $t_{\top}, t_{\perp}, f_{\perp}$  and  $f_{\top}$ , with different means as surely trust, unsurely trust, unsurely distrust and surely distrust, respectively. What’s more,  $f_{\top} \leq_4 t_{\perp} \leq_4 t_{\top}$ ,  $f_{\perp} \leq_4 f_{\perp} \leq_4 t_{\perp}$ ,  $\ominus_4 t_{\top} =_4 f_{\perp}$  and  $\ominus_4 t_{\perp} =_4 f_{\top}$ .

Suppose the observers opinions on assertion  $\alpha$  are in formats of  $\langle \alpha \geq c_1 \rangle$  or  $\langle \alpha \leq c_2 \rangle$  where  $c_1, c_2 \in [0,1]$ . Here,  $\langle \alpha \geq c_1 \rangle$  (resp.  $\langle \alpha \leq c_2 \rangle$ ) means that the observer thinks the Zadeh membership degree behind  $\alpha$  is  $c_1$  (resp.  $c_2$ ) and he is for (resp. against)  $\alpha$ . Then, the process transforming such opinions to elements in  $\mathcal{L}_4$  is as following. Construct a function  $\beta : O \rightarrow [0,1]$  representing the dividing point between assurance and diffidence. It is necessary to point out that, for each observer

<sup>4</sup> The symbols here do not represent the top concept and bottom concept in FDLs.

$o \in O$ ,  $\beta(o)$  should be founded by negotiation between decision-maker and  $o$  before their interaction and remains unchanged throughout the whole interactive process. Therefore, for each  $o \in O$ , the mapping from his Zadeh style opinion to elements in  $\mathcal{L}_4$  is shown in Fig. 1.



**Fig. 1.** Mapping from Zadeh style opinion of observer  $o$  to elements of  $\mathcal{L}_4$

**Example 4.** Suppose for observer  $o \in O$ ,  $\beta(o) = 0.6$  and his opinion on assertions  $\alpha_1 = a : C$ ,  $\alpha_2 = a : D$  are  $\langle \alpha_1 \geq 0.4 \rangle$ ,  $\langle \alpha_2 \leq 0.8 \rangle$ , respectively. According to Fig. 1, we have  $\langle \alpha_1 =_4 t_{\perp} \rangle$ ,  $\langle \alpha_2 =_4 f_{\perp} \rangle$ .

To verify the soundness of the mapping in Fig. 1, we only need to verify the cases of conjunction, disjunction of  $t_{\perp}$  and  $f_{\perp}$ , since other rule are obviously holding. Consider Example 4 again, the opinions  $\langle \alpha_1 \geq 0.4 \rangle$  and  $\langle \alpha_2 \leq 0.8 \rangle$  mean that observer  $o$  thinks  $a$  belongs to  $C$  in 40% degree and belongs to  $D$  in 80%, but with different attitudes. Thus, the values of  $a : C \sqcap D$  and  $a : C \sqcup D$  should be  $\min(0.4, 0.8) = 0.4$  and  $\max(0.4, 0.8) = 0.8$ , respectively. Moreover, it's reasonable that  $o$  prefer to distrust the result of  $a : C \sqcap D$  and to trust  $a : C \sqcup D$ 's result. That is to say,  $a : C \sqcap D \leq 0.4$  and  $a : C \sqcup D \geq 0.8$ , which are in agreement with Boolean equalities  $t_{\perp} \otimes_4 f_{\perp} =_4 f_{\top}$ ,  $t_{\perp} \oplus_4 f_{\perp} =_4 t_{\top}$ .

Thus, Boolean lattice  $\mathcal{L}_o$  can be extended to a new one  $\mathcal{L}'_o = (\mathcal{L}_4)^n$  ( $n = |O|$ ), whose partial order  $\preceq$ , operators  $\otimes$ ,  $\oplus$ ,  $\ominus$ , and identity elements  $o$ ,  $1$  can be defined as: for any elements  $p = (q_1, q_2, \dots, q_n)$  and  $p' = (q'_1, q'_2, \dots, q'_n)$  in  $\mathcal{L}'_o$ , (i)  $p \preceq p'$  iff for each  $i$  ( $1 \leq i \leq n$ ),  $q_i \leq_4 q'_i$ ; (ii)  $\ominus p = (\ominus_4 q_1, \ominus_4 q_2, \dots, \ominus_4 q_n)$ ; (iii)  $p \otimes p' = (q_1 \otimes_4 q'_1, q_2 \otimes_4 q'_2, \dots, q_n \otimes_4 q'_n)$ ; (iv)  $p \oplus p' = (q_1 \oplus_4 q'_1, q_2 \oplus_4 q'_2, \dots, q_n \oplus_4 q'_n)$ ; (v)  $p = o$  iff for each  $i$  ( $1 \leq i \leq n$ ),  $q_i =_4 f_{\top}$ ; (vi)  $p = 1$  iff for each  $i$  ( $1 \leq i \leq n$ ),  $q_i =_4 t_{\top}$ .

If we use  $\mathcal{L}'_o$  as the Boolean lattice of lf- $\mathcal{ALC}$ , it's obvious that all the things still valid excepting Zadeh membership degree, which needs a new definition. Let's take the crisp assertion  $a : C$  as an example. Suppose for  $p \in \mathcal{L}'_o$  and  $q \in \mathcal{L}_4$ ,  $Ind(p, q)$  represent the indices set of  $\mathcal{L}_4$  values in  $p$  which are equal to  $q$ . Then, the Zadeh membership degree w.r.t. interpretation  $\mathcal{I}$  becomes (without weight function)

$$\mu_C^{\mathcal{I}}(a^{\mathcal{I}}) = \frac{\sum_{i \in \text{Ind}(C^{\mathcal{I}}(a^{\mathcal{I}}), t_{\tau})} 1 + \sum_{i \in \text{Ind}(C^{\mathcal{I}}(a^{\mathcal{I}}), t_{\perp}) \cup \text{Ind}(C^{\mathcal{I}}(a^{\mathcal{I}}), f_{\perp})} \beta(o_i)}{|O|}. \quad (4)$$

or (with weight function)

$$\mu_C^{\mathcal{I}}(a^{\mathcal{I}}) = \frac{\sum_{i \in \text{Ind}(C^{\mathcal{I}}(a^{\mathcal{I}}), t_{\tau})} \gamma(o_i) + \sum_{i \in \text{Ind}(C^{\mathcal{I}}(a^{\mathcal{I}}), t_{\perp}) \cup \text{Ind}(C^{\mathcal{I}}(a^{\mathcal{I}}), f_{\perp})} \beta(o_i) \gamma(o_i)}{\sum_{j=1}^{|O|} \gamma(o_j)}. \quad (5)$$

## 5 Conclusions

By analysis on the shortcoming of classical FDLs, we find a feasible improving method, i.e. using Boolean lattices rather than  $[0,1]$  to represent fuzziness. As a result, a proper Boolean lattice in Web Computing environment is constructed and is utilized to build an improved FDL, named If- $\mathcal{ALC}$ . And finally, some probable extensions of the chosen lattice are discussed. Still, there are many work deserves further research. For example, because of its limited expressiveness, it is necessary to extend both If- $\mathcal{ALC}$  and its Boolean lattice; even, the applications of If- $\mathcal{ALC}$  in Web Computing Environment are also necessary.

## References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.(Eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Straccia, U.: Reasoning within fuzzy description logics. J. of Artificial Intelligence Research, no. 14 (2001) 137-166
3. Zadeh, L. A.: Fuzzy sets. Information and Control, no. 3, vol. 8 (1965) 338-353
4. Lukasiewicz, T., Straccia, U.: Uncertainty and Vagueness in Description Logic Programs for the Semantic Web. INFOSYS Research Report, (2007)
5. Straccia, U.: Uncertainty in description logics: a lattice-based approach. In Proc. IPMU, (2004)
6. Straccia, U.: Uncertainty and Description Logic Programs over Lattices. In Sanchez, E., editor, Fuzzy Logic and the Semantic Web, Capturing Intelligence, chapter 7, Elsevier (2006) 115-133
7. Belnap, N. D.: A useful four-valued logic. Modern uses of multiple-valued logic, Reidel, Dordrecht, NL, (1977) 5-37

## Integrated Distributed Description Logics (extended abstract)

Antoine Zimmermann

INRIA Rhône-Alpes

**Abstract.** We propose a Description-Logics-based language that extends standard DL with distributed capabilities. More precisely, it offers the possibility to formally describe the semantic relations that exist between two ontologies in a networked knowledge-based system. Contrary to Distributed Description Logics [2], it is possible to compose correspondences ( $\approx$  bridge rules), while still being able to hide some of the discrepancies between ontologies. Moreover, when ontologies have no nominals, no A-Box axioms, and correspondences are restricted to cross-ontology subsumption, the satisfiability of a local ontology is not influenced by ontology alignments and other ontologies, *i.e.*, local deduction is invariant to the change of the outer system. Although we do not have a complete reasoning procedure, we provide inference rules and semantic properties, and a discussion on reasoning in this formalism.

### 1 Introduction

Description Logics (DL) have been widely used in knowledge-based systems and serve as the foundation for the accepted standard language of the semantic web, *viz.*, OWL [4]. However, in their basic form, DL are not so much appropriate when used in a strongly distributed environment like peer to peer systems, semantic web, or other networked, heterogeneous knowledge-based systems. In distributed environments, ontology engineers want to reuse third party ontologies or, even more, *parts of* existing ontologies, while maintaining consistency, at least in their local knowledge representation.

We offer an extension of the DL formalism (Integrated Distributed Description Logics, or IDDL) which comply with the requirements of a distributed knowledge representation. The main advantages of our approach, compared to others, are (1) its separation of local semantics (which is standard DL), and global semantics; and (2) it allows composition of ontology mappings.

First item means that it is conceptually in accordance with the notion of semantic integration: local knowledge and reasoning should not be disturbed when embedded in a network of ontologies. Several research initiative have been launched to define languages specifically adapted to these issues. Some of them are based on DL. Sect. 2 presents these formalisms, and compares them to our approach. Sect. 3 presents the syntax and semantics of our new formalism. In Sect. 4, we discuss reasoning in this formalism by listing the inference rules added to standard DL reasoning, and explain the particularities (advantages and drawbacks) of the approach.

## 2 Related work

In this section, we do not investigate distributed knowledge-based formalism in its generality. We focus on DL-related work. The use of DL as a basis for a semantic web representation language was envisaged early, and OWL supports “imports” of ontologies from a distant server. So, to a limited extent, OWL is a language for distributed architectures. However, since the “import” statement only copies the content of the identified ontology (using its URI as a URL), it does not so well comply with the specificities of an evolving, heterogeneous environment like the Web.

[2] defines a semantics for Distributed Description Logics (DDL), based on the same idea as DFOL [3]. It is built around the idea of contextualizing knowledge. More precisely, each ontology in a system relates other ontologies to itself in a directional way, enabling an ontology to “translate” others’ knowledge in its own context. It is directional because the context to which knowledge is transferred determines how things are interpreted. Technically, each local ontology has its own domain of interpretation, and a *domain relation* defines how information is translated from one ontology to the other. These relations are not necessarily symmetric. The main disadvantage, with respect to this semantics, is the impossibility to compose so-called “bridge rules”, *e.g.*, if local concept  $C_1$  of ontology  $O_1$  is seen (from  $O_1$ ’s context) as a subclass of foreign concept  $C_2$  in ontology  $O_2$ , and  $C_2$  is seen (from  $O_2$ ’s context) as a subclass of concept  $C_3$  in ontology  $O_3$ , it is not possible to deduce that  $C_1$  is a subclass of  $C_3$  (from  $O_1$ ’s point of view). So relations between two foreign ontologies are not really taken into account. Only the relations between foreign ontologies and the local ontology count in the interpretation of one given context.

Another possible approach, which has good features with respect to modularity of ontologies, is found in Package-based Description Logics (P-DL) [1]. In this DL-based formalism, local ontologies (or “package”) can import not just full ontologies but rather named concepts or roles from foreign ontologies. Each ontology is interpreted in a local domain, but instead of relating it to others, they simply overlap on the imported terms interpretation. So there is no difference between the interpretation of a concept from the importing and the imported ontologies. The biggest problem is that it obliges the whole network of ontologies to be quite homogeneous.

In [5], a more abstract formalism has been used to compare different approaches interpreting distributed systems. DDL corresponds to what authors called a *contextualized semantics*, and they prove that it does not comply with ontology alignment composition. P-DL would rather correspond to *simple semantics*, which is tied to homogeneous and consistent systems. Finally, they propose a third formalism that is conceptually well suited for heterogeneous ontology integration and complies with alignment composition. In the present paper, we instantiate it by giving it a concrete DL-based semantics.

## 3 Syntax and Semantics

A IDDL knowledge base contains two components: a family of local DL ontologies, and a family of ontology alignments. In Sect. 3.1, we define many DL constructors but the formalism is appropriate for any subset of them, *e.g.*,  $\mathcal{AL}$ ,  $\mathcal{ALC}$ ,  $\mathcal{ALCN}$ ,  $\mathcal{SHIQ}$ ,  $\mathcal{SHOIN}$ , *etc.*

### 3.1 DL Syntax and Semantics

IDDL ontologies have the same syntax and semantics as in standard DL. More precisely, a DL ontology is composed of concepts, roles and individuals, as well as axioms built out of these elements. A concept is either a primitive concept  $A$ , or, given concepts  $C$ ,  $D$ , role  $R$ , individuals  $a_1, \dots, a_k$ , and natural number  $n$ ,  $\perp$ ,  $\top$ ,  $C \sqcup D$ ,  $C \sqcap D$ ,  $\exists R.C$ ,  $\forall R.C$ ,  $\leq nR.C$ ,  $\geq nR.C$ ,  $\neg C$  or  $\{a_1, \dots, a_k\}$ . A role is either a primitive role  $P$ , or, given roles  $R$  and  $S$ ,  $R \sqcup S$ ,  $R \sqcap S$ ,  $\neg R$ ,  $R^-$ ,  $R \circ S$  and  $R^+$ .

Interpretations are pairs  $\langle \Delta^I, \cdot^I \rangle$ , where  $\Delta^I$  is a non-empty set (the domain of interpretation) and  $\cdot^I$  is the function of interpretation such that for all primitive concepts  $A$ ,  $A^I \subseteq \Delta^I$ , for all primitive roles  $P$ ,  $P^I \subseteq \Delta^I \times \Delta^I$ , and for all individuals  $a$ ,  $a^I \in \Delta^I$ . Interpretations of complex concepts and roles is inductively defined by  $\perp^I = \emptyset$ ,  $\top^I = \Delta^I$ ,  $(C \sqcup D)^I = C^I \cup D^I$ ,  $(C \sqcap D)^I = C^I \cap D^I$ ,  $(\exists R.C)^I = \{x | \exists y. y \in C^I \wedge \langle x, y \rangle \in R^I\}$ ,  $(\forall R.C)^I = \{x | \forall y. \langle x, y \rangle \in R^I \Rightarrow y \in C^I\}$ ,  $(\leq nR.C)^I = \{x | \#\{y \in C^I | \langle x, y \rangle \in R^I\} \leq n\}$ ,  $(\geq nR.C)^I = \{x | \#\{y \in C^I | \langle x, y \rangle \in R^I\} \geq n\}$ ,  $(\neg C)^I = \Delta^I \setminus C^I$ ,  $\{a_1, \dots, a_k\}^I = \{a_1^I, \dots, a_k^I\}$ ,  $(R \sqcup S)^I = R^I \cup S^I$ ,  $(R \sqcap S)^I = R^I \cap S^I$ ,  $(\neg R)^I = (\Delta^I \times \Delta^I) \setminus R^I$ ,  $(R^-)^I = \{\langle x, y \rangle | \langle y, x \rangle \in R^I\}$ ,  $(R \circ S)^I = \{\langle x, y \rangle | \exists z. \langle x, z \rangle \in R^I \wedge \langle z, y \rangle \in S^I\}$  and  $(R^+)^I$  is the reflexive-transitive closure of  $R^I$ .

Axioms are either subsumption  $C \sqsubseteq D$ , sub-role axioms  $R \sqsubseteq S$ , instance assertions  $C(a)$ , role assertions  $R(a, b)$  and individual identities  $a = b$ , where  $C$  and  $D$  are concepts,  $R$  and  $S$  are roles, and  $a$  and  $b$  are individuals. An interpretation  $I$  satisfies axiom  $C \sqsubseteq D$  iff  $C^I \subseteq D^I$ ; it satisfies  $R \sqsubseteq S$  iff  $R^I \subseteq S^I$ ; it satisfies  $C(a)$  iff  $a^I \in C^I$ ; it satisfies  $R(a, b)$  iff  $\langle a^I, b^I \rangle \in R^I$ ; and it satisfies  $a = b$  iff  $a^I = b^I$ . When  $I$  satisfies an axiom  $\alpha$ , it is denoted by  $I \models \alpha$ .

An ontology  $O$  is composed of a set of terms (primitive concepts/roles and individuals) called the signature of  $O$  and denoted by  $\text{Sig}(O)$ , and a set of axioms denoted by  $\text{Ax}(O)$ . An interpretation  $I$  is a model of an ontology  $O$  iff for all  $\alpha \in \text{Ax}(O)$ ,  $I \models \alpha$ . In this case, we write  $I \models O$ . The set of all models of an ontology  $O$  is denoted by  $\text{Mod}(O)$ . A semantic consequence of an ontology  $O$  is a formula  $\alpha$  such that for all  $I \in \text{Mod}(O)$ ,  $I \models \alpha$ .

### 3.2 Distributed Systems

A Distributed System (DS) is composed of a set of ontologies, interconnected by ontology alignments. An ontology alignment describes semantic relations between two ontologies.

**Syntax:** An ontology alignment is composed of a set of *correspondences*. A correspondence can be seen as an axiom that asserts a relation between concepts, roles or individuals of two distinct ontologies. They are homologous to bridge rules in DDL. We use a notation similar to DDL in order to identify in which ontology a concept, role or individual is defined. If a concept/role/individual  $E$  belongs to ontology  $i$ , then we write it  $i : E$ . The 6 possible types of correspondences between ontologies  $i$  and  $j$  are:

**Definition 1 (Correspondence).** A correspondence between two ontologies  $i$  and  $j$  is one of the following formula:

- $i:C \xleftrightarrow{\sqsubseteq} j:D$  is a cross-ontology concept subsumption;
- $i:R \xleftrightarrow{\sqsubseteq} j:S$  is a cross-ontology role subsumption;
- $i:C \xleftrightarrow{\perp} j:D$  is a cross-ontology concept disjunction;
- $i:R \xleftrightarrow{\perp} j:S$  is a cross-ontology role disjunction;
- $i:a \xleftrightarrow{\in} j:C$  is a cross-ontology membership;
- $i:a \xleftrightarrow{=} j:b$  is a cross-ontology identity.

An ontology alignment is simply a set of correspondences. Together with DL ontologies, they form the components of a Distributed System in IDDL.

**Definition 2 (Distributed System).** A Distributed System (*DS for short*), is a pair  $\langle \mathbf{O}, \mathbf{A} \rangle$  such that  $\mathbf{O}$  is a set of ontologies, and  $\mathbf{A} = (A_{ij})_{i,j \in \mathbf{O}}$  is a family of alignments relating ontologies of  $\mathbf{O}$ .<sup>1</sup>

**Semantics:** Distributed systems semantics depends on local semantics, but does not interfere with it. A standard DL ontology can be straightforwardly used in IDDL system. Informally, interpreting a IDDL system consists in assigning a standard DL interpretation to each local ontology, then correlating the domains of interpretation thanks to what we call an *equalizing function*.

**Definition 3 (Equalizing function).** Given a family of local interpretations  $\mathbf{I}$ , an equalizing function  $\epsilon$  is a family of functions indexed by  $\mathbf{I}$  such that for all  $I_i \in \mathbf{I}$ ,  $\epsilon_i : \Delta^{I_i} \rightarrow \Delta_\epsilon$  where  $\Delta_\epsilon$  is called the global domain of interpretation of  $\epsilon$ .

A distributed interpretation assigns a standard DL interpretation to each ontology in the system, as well as an equalizing function that correlate local knowledge into a global domain of interpretation.

**Definition 4 (Distributed interpretation).** Let  $S = \langle \mathbf{O}, \mathbf{A} \rangle$  be a DS. A distributed interpretation of  $S$  is a pair  $\langle \mathbf{I}, \epsilon \rangle$  where  $\mathbf{I}$  is a family of interpretations indexed by  $\mathbf{O}$ ,  $\epsilon$  is an equalizing function for  $\mathbf{I}$ , such that for all  $i \in \mathbf{O}$ ,  $I_i$  interprets  $i$  and  $\epsilon_i : \Delta^{I_i} \rightarrow \Delta_\epsilon$ .

While local satisfiability is the same as standard DL, correspondence satisfaction involves the equalizing function.

**Definition 5 (Satisfaction of a correspondence).** Let  $S$  be a DS, and  $i, j$  two ontologies of  $S$ . Let  $\mathcal{I} = \langle \mathbf{I}, \epsilon \rangle$  be a distributed interpretation. We define satisfaction of a correspondence  $c$  (denoted by  $\mathcal{I} \models_d c$ ) as follows:

- $\mathcal{I} \models_d i:C \xleftrightarrow{\sqsubseteq} j:D$  iff  $\epsilon_i(C^{I_i}) \subseteq \epsilon_j(D^{I_j})$ ;
- $\mathcal{I} \models_d i:R \xleftrightarrow{\sqsubseteq} j:S$  iff  $\epsilon_i(R^{I_i}) \subseteq \epsilon_j(S^{I_j})$ ;
- $\mathcal{I} \models_d i:C \xleftrightarrow{\perp} j:D$  iff  $\epsilon_i(C^{I_i}) \cap \epsilon_j(D^{I_j}) = \emptyset$ ;
- $\mathcal{I} \models_d i:R \xleftrightarrow{\perp} j:S$  iff  $\epsilon_i(R^{I_i}) \cap \epsilon_j(S^{I_j}) = \emptyset$ ;

<sup>1</sup> We consistently use bold face to denote a mathematical family of elements. So,  $\mathbf{O}$  denotes  $(O_i)_{i \in I}$  where  $I$  is a set of indices.

- $\mathcal{I} \models_d i:a \xleftrightarrow{\in} j:C$  iff  $\epsilon_i(a^{I_i}) \in \epsilon_j(C^{I_j})$ ;
- $\mathcal{I} \models_d i:a \xleftrightarrow{=} j:b$  iff  $\epsilon_i(a^{I_i}) = \epsilon_j(b^{I_j})$ .

Additionally, for all local formula  $i:\phi$ ,  $\mathcal{I} \models_d i:\phi$  iff  $I_i \models \phi$  (i.e., local satisfaction implies global satisfaction). A distributed interpretation  $\mathcal{I}$  satisfies an alignment  $A$  iff it satisfies all correspondences of  $A$  (denoted by  $\mathcal{I} \models_d A$ ) and it satisfies an ontology  $O_i$  iff it satisfies all axioms of  $O_i$  (denoted by  $\mathcal{I} \models_d O_i$ ). When all ontologies and all alignments are satisfied, the DS is satisfied by the distributed interpretation. In which case we call this interpretation a *model* of the system.

**Definition 6 (Model of a DS).** Let  $S = \langle \mathbf{O}, \mathbf{A} \rangle$  be a DS. A distributed interpretation  $\mathcal{I}$  is a model of  $S$  (denoted by  $\mathcal{I} \models_d S$ ), iff:

- for all  $O_i \in \mathbf{O}$ ,  $\mathcal{I} \models_d O_i$ ;
- for all  $A_{ij} \in \mathbf{A}$ ,  $\mathcal{I} \models_d A_{ij}$ .

The set of all models of a DS is denoted by  $\text{Mod}(S)$ . A formula  $\alpha$  is a consequence of a DS ( $S \models_d \alpha$ ) iff  $\forall \mathcal{M} \in \text{Mod}(S)$ ,  $\mathcal{M} \models_d \alpha$ . This model-theoretic semantics offers special challenge to the reasoning infrastructure, that we discuss in next section.

## 4 Reasoning in IDDL

Reasoning in IDDL is a tricky task because there are two levels of interpretation, which are separated yet interdependent. Local DL inferences are valid in IDDL, but correspondences add new inference rules. Also, we show in Sect. 4.2 how to transpose knowledge of a DS into a localized ontology. However, this process does not guarantee completeness in general.

### 4.1 Inference Rules

Correspondences and axioms from several ontologies are used to deduce new axioms or correspondences, as shown in Fig. 1.

It is easy to prove that these rules lead to correct reasoning<sup>2</sup> but completeness is still under investigation. Rule 14 is a special case because it requires the introduction of a new individual that does not appear in any other axioms. This is due to the fact that DL does not provide any means to assert that there exists an unnamed instance of a concept. Rules 7, 8, 9, 10 and 11 show that it is possible to compose correspondences. Rule 15 shows that alignments can produce inconsistencies, independently of ontologies. It must be remarked that several seemingly intuitive results are not true in IDDL. For instance,  $i:A \xleftrightarrow{=} j:\neg B$  does not imply, nor is implied by  $i:A \xleftrightarrow{\perp} j:B$ , because injectivity of the equalizing function is not required. Moreover, if  $i:A \xleftrightarrow{=} i:B$  (resp.  $i:a \xleftrightarrow{=} i:b$ , resp.  $i:a \xleftrightarrow{\in} i:A$ ), it is not possible to infer  $i:A \sqsubseteq B$  (resp.  $i:a = b$ , resp.  $i:A(a)$ ).

<sup>2</sup> See <http://www.inrialpes.fr/exmo/people/zimmer/DL2007Proof.pdf> for formal proofs.

$$\begin{array}{l}
 \frac{i: A \sqsubseteq B}{i: A \xleftrightarrow{\sqsubseteq} i: B} \quad (1) \\
 \frac{i: A(a)}{i: a \xleftrightarrow{\sqsubseteq} i: A} \quad (3) \\
 \frac{i: a \xleftrightarrow{\sqsupseteq} j: b}{j: b \xleftrightarrow{\sqsupseteq} i: a} \quad (5) \\
 \frac{i: A \xleftrightarrow{\sqsubseteq} j: B \quad j: B \xleftrightarrow{\sqsubseteq} k: C}{i: A \xleftrightarrow{\sqsubseteq} k: C} \quad (7) \\
 \frac{i: a \xleftrightarrow{\sqsubseteq} j: B \quad j: B \xleftrightarrow{\sqsubseteq} k: C}{i: a \xleftrightarrow{\sqsubseteq} k: C} \quad (9) \\
 \frac{i: A \xleftrightarrow{\sqsubseteq} j: B \quad j: B \xleftrightarrow{\perp} k: C}{i: A \xleftrightarrow{\perp} k: C} \quad (11) \\
 \frac{i: A \xleftrightarrow{\perp} j: B \quad i: a \xleftrightarrow{\sqsubseteq} j: B}{i: \neg A(a)} \quad (13) \\
 \frac{i: a \xleftrightarrow{\sqsubseteq} j: B \quad j: B \xleftrightarrow{\perp} j: B}{\square} \quad (15)
 \end{array}
 \qquad
 \begin{array}{l}
 \frac{i: a = b}{i: a \xleftrightarrow{\sqsupseteq} i: b} \quad (2) \\
 \frac{i: A \xleftrightarrow{\sqsubseteq} j: B \quad i: A' \xleftrightarrow{\sqsubseteq} j: B'}{i: A \sqcup A' \xleftrightarrow{\sqsubseteq} j: B \sqcup B'} \quad (4) \\
 \frac{i: A \xleftrightarrow{\perp} j: B}{j: B \xleftrightarrow{\perp} i: A} \quad (6) \\
 \frac{i: a \xleftrightarrow{\sqsupseteq} j: b \quad j: b \xleftrightarrow{\sqsupseteq} k: c}{i: a \xleftrightarrow{\sqsupseteq} k: c} \quad (8) \\
 \frac{i: a \xleftrightarrow{\sqsupseteq} j: b \quad j: b \xleftrightarrow{\sqsubseteq} k: C}{i: a \xleftrightarrow{\sqsubseteq} k: C} \quad (10) \\
 \frac{i: A \xleftrightarrow{\perp} j: B \quad i: A' \xleftrightarrow{\sqsubseteq} j: B'}{i: A \sqsubseteq \neg A'} \quad (12) \\
 \frac{i: a \xleftrightarrow{\sqsubseteq} j: B}{j: B(x) \quad i: a \xleftrightarrow{\sqsupseteq} j: x} \quad (14)
 \end{array}$$

Fig. 1. Inference rules in IDDL.

It is interesting to note that when the ontologies have no A-Box and do not use nominals, and when moreover correspondences are limited to cross-ontology subsumptions, it is not possible to deduce new local axioms with outer knowledge (only rules 1, 4 and 7 apply). Therefore, in this particular case, local reasoning is not disturbed by the surrounding DS. This is a particularly interesting feature when the network of ontologies is constantly evolving. Yet, it does not mean that embedding an ontology in a DS does not provide interesting knowledge. Indeed, the global semantics level is very much influenced, and this is crucial in ontology integration. Next section shows how a DS can be used to define a new DL ontology that integrate the knowledge from all the system.

## 4.2 Localizing Distributed Knowledge

Here, we show how to interpret a DS as a standard DL ontology, by building a standard DL interpretation out of a distributed one. The multiple signatures of the DS ontologies can be gathered into one vocabulary in the following way.

**Definition 7.** The integrated vocabulary  $V_S$  of a distributed system  $S = \langle \mathbf{O}, \mathbf{A} \rangle$  is defined as follows:

- for all primitive concept (resp. primitive role, resp. individual)  $X_i$  in ontology  $O_i$ , there exists a primitive concept (resp. primitive role, resp. individual)  $X_i^{\rightarrow}$  in  $V_S$ ;

- for all constructed concept (resp. constructed role)  $X_i$  of ontology  $O_i$  appearing in the axioms or alignments of  $S$ , there exists a **primitive** concept (resp. **primitive** role)  $X_i^{\rightarrow}$  in  $V_S$ .

The integrated vocabulary can be interpreted by a standard DL interpretation. An integrated interpretation is a specific DL interpretation defined out of a given distributed interpretation. Informally, it can be described as the mathematical composition of the local interpretation functions and the equalizing function.

**Definition 8.** Given a distributed interpretation  $\mathcal{I} = \langle \mathbf{I}, \epsilon \rangle$  of a system  $S$ , the integrated interpretation  $\mathcal{I}^{\rightarrow}$  built out of  $\mathcal{I}$  is a DL interpretation of  $V_S$  defined as follows:

- for all primitive concept  $C_i^{\rightarrow}$  of  $V_S$ ,  $(C_i^{\rightarrow})^{\mathcal{I}^{\rightarrow}} = \{\epsilon_i(x); x \in C_i^{I_i}\}$ ;
- for all primitive role  $R_i^{\rightarrow}$  of  $V_S$ ,  $(R_i^{\rightarrow})^{\mathcal{I}^{\rightarrow}} = \{\langle \epsilon_i(x), \epsilon_i(y) \rangle; \langle x, y \rangle \in R_i^{I_i}\}$ ;
- for all individual  $a_i^{\rightarrow}$  of  $V_S$ ,  $(a_i^{\rightarrow})^{\mathcal{I}^{\rightarrow}} = \epsilon_i(a_i^{I_i})$ .

Of course, this interpretation can be extended to interpret constructed concepts like  $\exists R^{\rightarrow}.C^{\rightarrow}$ . Be careful not to confuse complex concept  $\exists R^{\rightarrow}.C^{\rightarrow}$  and the *primitive* concept  $(\exists R.C)^{\rightarrow}$ . See Sect. 4 for details.

**Deduction in the integrated vocabulary:** Since integrated interpretations are standard DL interpretations, they may satisfy DL axioms over the integrated vocabulary. Theo. 1 shows how distributed satisfaction influence integrated interpretation satisfaction.

**Theorem 1.** Let  $\mathcal{I} = \langle \mathbf{I}, \epsilon \rangle$  be a distributed interpretation of a DS which contains concepts  $C_i, D_i$ , roles  $R_i, S_i$ , individuals  $a_i, b_i, o_1, \dots, o_n$  in ontology  $O_i$  and concept  $C_j$ , role  $R_j$ , individual  $a_j$  in ontology  $O_j$ .

$$\begin{array}{ll}
 I_i \models i : C_i(a_i) \implies \mathcal{I}^{\rightarrow} \models C_i^{\rightarrow}(a_i^{\rightarrow}) & \mathcal{I} \models i : C_i \xleftrightarrow{\subseteq} j : C_j \implies \mathcal{I}^{\rightarrow} \models C_i^{\rightarrow} \sqsubseteq C_j^{\rightarrow} \\
 I_i \models i : R_i(a_i, b_i) \implies \mathcal{I}^{\rightarrow} \models R_i^{\rightarrow}(a_i^{\rightarrow}, b_i^{\rightarrow}) & \mathcal{I} \models i : R_i \xleftrightarrow{\subseteq} j : R_j \implies \mathcal{I}^{\rightarrow} \models R_i^{\rightarrow} \sqsubseteq R_j^{\rightarrow} \\
 I_i \models i : C_i \sqsubseteq D_i \implies \mathcal{I}^{\rightarrow} \models C_i^{\rightarrow} \sqsubseteq D_i^{\rightarrow} & \mathcal{I} \models i : C_i \xleftrightarrow{\perp} j : C_j \implies \mathcal{I}^{\rightarrow} \models C_i^{\rightarrow} \sqsubseteq \neg(C_j^{\rightarrow}) \\
 I_i \models i : a_i = b_i \implies \mathcal{I}^{\rightarrow} \models a_i^{\rightarrow} = b_i^{\rightarrow} & \mathcal{I} \models i : R_i \xleftrightarrow{\perp} j : R_j \implies \mathcal{I}^{\rightarrow} \models R_i^{\rightarrow} \sqsubseteq \neg(R_j^{\rightarrow}) \\
 I_i \models i : a_i \xleftrightarrow{\subseteq} j : b_j \implies \mathcal{I}^{\rightarrow} \models a_i^{\rightarrow} = a_j^{\rightarrow} & \mathcal{I} \models i : a_i \xleftrightarrow{\subseteq} j : C_j \implies \mathcal{I}^{\rightarrow} \models C_j^{\rightarrow}(a_i^{\rightarrow})
 \end{array}$$

Moreover, the following assertions hold:

$$\begin{array}{ll}
 \mathcal{I}^{\rightarrow} \models C_i^{\rightarrow} \sqcup D_i^{\rightarrow} \sqsubseteq (C_i \sqcup D_i)^{\rightarrow} & \mathcal{I}^{\rightarrow} \models R_i^{\rightarrow} \sqcup S_i^{\rightarrow} \sqsubseteq (R_i \sqcup S_i)^{\rightarrow} \\
 \mathcal{I}^{\rightarrow} \models (C_i \sqcup D_i)^{\rightarrow} \sqsubseteq C_i^{\rightarrow} \sqcup D_i^{\rightarrow} & \mathcal{I}^{\rightarrow} \models (R_i \sqcup S_i)^{\rightarrow} \sqsubseteq R_i^{\rightarrow} \sqcup S_i^{\rightarrow} \\
 \mathcal{I}^{\rightarrow} \models C_i^{\rightarrow} \cap D_i^{\rightarrow} \sqsubseteq (C_i \cap D_i)^{\rightarrow} & \mathcal{I}^{\rightarrow} \models R_i^{\rightarrow} \cap S_i^{\rightarrow} \sqsubseteq (R_i \cap S_i)^{\rightarrow} \\
 \mathcal{I}^{\rightarrow} \models (\exists R_i.C_i)^{\rightarrow} \sqsubseteq \exists (R_i^{\rightarrow}). (C_i^{\rightarrow}) & \mathcal{I}^{\rightarrow} \models (R_i^{\rightarrow})^{\neg} \sqsubseteq (R_i^{\neg})^{\rightarrow} \\
 \mathcal{I}^{\rightarrow} \models \exists R_i^{\rightarrow}. \top \sqsubseteq (\exists R_i. \top)^{\rightarrow} & \mathcal{I}^{\rightarrow} \models (R_i^{\rightarrow})^{\neg} \sqsubseteq (R_i^{\neg})^{\rightarrow} \\
 \mathcal{I}^{\rightarrow} \models (\{o_1, \dots, o_n\})^{\rightarrow} \sqsubseteq \{o_1^{\rightarrow}, \dots, o_n^{\rightarrow}\} & \mathcal{I}^{\rightarrow} \models (R_i^{\rightarrow})^+ \sqsubseteq (R_i^+)^{\rightarrow} \\
 \mathcal{I}^{\rightarrow} \models \{o_1^{\rightarrow}, \dots, o_n^{\rightarrow}\} \sqsubseteq (\{o_1, \dots, o_n\})^{\rightarrow} & \mathcal{I}^{\rightarrow} \models (R_i \circ S_i)^{\rightarrow} \sqsubseteq R_i^{\rightarrow} \circ S_i^{\rightarrow}
 \end{array}$$

Each of the previous assertions is quite easy to prove, but fastidious. Therefore, we do not reproduce them here but interested readers can find them online.<sup>3</sup> For all other

<sup>3</sup> <http://www.inrialpes.fr/exmo/people/zimmer/DL2007Proof.pdf>

constructors and subsumptions, counter examples can be found where the interpretation does not satisfy them. This can also be found online.

This theorem allows compiling axioms that are satisfied by all the models of a DS. Therefore, it is possible to build a new ontology that integrate knowledge from distributed ontologies and alignments. Such an ontology correctly represents knowledge of the system, but might not be complete. Nonetheless, Theo. 1 can be used as the basis of a compilation algorithm which integrate aligned ontologies in a modular way. The compiled ontology itself could be embedded in a distributed system.

## 5 Conclusion and further work

We have proposed a new formalism for distributed systems composed of ontologies and ontology alignments. Its semantics has the advantage of being able to compose correspondences (*i.e.*, to deduce a new alignment from a chain of alignments exists from the first to the second ontology). Given some restrictions, it also offers strong robustness, since the absence of A-Box and nominals, together with only cross-ontology subsumption correspondences, guarantee that local deduction is invariant to the change of the outer system (*i.e.*, alignments and other ontologies). Finally, it seems to be a good candidate semantics for ontology integration and modularization, because of its two-level semantics.

However, it still needs theoretical investigation. The most important work in the continuity of what is proposed here is the design of a deduction procedure. This paper already provides correct deductive rules, but completeness is not guaranteed in the general case. Developing a tableau-like algorithm is hard because the two levels of semantics interact with each others, although they are not processable together by usual methods. Such a procedure would open the way to implementation and tests.

## References

1. J. Bao, D. Caragea, and V. Honavar. On the semantics of linking and importing in modular ontologies. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference*, volume 4273 of *LNCS*, pages 72–86. Springer, November 2006.
2. A. Borgida and L. Serafini. Distributed Description Logics: Directed domain correspondences in federated information sources. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2002: Confederated International Conferences CoopIS, DOA, and ODBASE 2002*, volume 2519 of *LNCS*. Springer, 2002.
3. C. Ghidini and L. Serafini. Distributed First Order Logics. In Dov M. Gabbay and Maarten de Rijke, editors, *Frontiers of Combining Systems 2*, volume 7 of *Studies in Logic and Computation*, pages 121–139, Baldock, Hertfordshire, UK, 2000. Research Studies Press.
4. P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, World Wide Web Consortium (W3C), 2004.
5. A. Zimmermann and J. Euzenat. Three semantics for distributed systems and their relations with alignment composition. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference*, volume 4273 of *LNCS*, pages 16–29. Springer, November 2006.

## Modal Logic Applied to Query Answering and the Case for Variable Modalities

Evgeny Zolin

School of Computer Science  
The University of Manchester  
zolin@manchester.ac.uk

**Abstract.** We present a query answering technique based on notions and results from modal Correspondence Theory. It allows us to answer a wide family of conjunctive queries by polynomial reduction to knowledge base satisfiability problem. An advantage of this technique lies in its uniformity: it does not depend on a Description Logic (DL), so that extending a DL does not invalidate the algorithm. Thus, the problem of answering queries in this family is decidable in any decidable DL. The construction also leads to an idea of extending the modal language with so called variable modalities, whose syntax and semantics is introduced in the paper. On the one hand, this yields a broader family of queries that can be answered with the same technique. On the other hand, modal logic with variable modalities is interesting per se, and we formulate some natural (open) questions about this logic.

### 1 Introduction

Developing languages and algorithms for reasoning with ontologies is a crucial aspect of the Semantic Web activity. Among reasoning tasks, querying is a fundamental mechanism for extracting information from a KB. Two most important reasoning services involving queries are query answering and query containment (also called subsumption); they are mutually reducible (and we focus on query answering here). While the complexity of DLs is now well understood [1, 11], the decidability and complexity issues for query answering in expressive DLs have only recently got partial or complete solutions (see [5, 8] and references therein). For the expressive DLs that underpin the state-of-the-art web ontology languages OWL DL and OWL 1.1, even the decidability of query answering is not established yet.

Usually, query answering techniques are developed for a specific DL, and the more expressive is a DL, the more complex becomes the query answering algorithm. In this paper, we address the problem from a different perspective: we develop a *uniform* technique for answering a certain *family* of queries, which means that the algorithm is independent of a DL in which a KB is formulated. Therefore, extending the expressivity of a DL does not hurt the algorithm (in contrast to other approaches where, e.g., introducing transitive roles can invalidate the algorithm, cf. [5, 8]). The basic and most prominent uniform algorithm

for query answering is so called *rolling-up* technique (see, e.g., [5]) applicable to the family of *tree-like* queries whose root is the only distinguished variable. Given such a query  $q(x)$ , the algorithm transforms it into an *equivalent* concept  $C$ , i.e., in any model, the extensions of  $C$  and  $q(x)$  coincide. Hence, to answer the query  $q(x)$  is the same as to retrieve all instances of the concept  $C$  (and the latter task is reducible to KB unsatisfiability). The starting point for us is an observation that, in general, the equivalence of a concept to a query is sufficient, but not necessary for them to have the same answers. It turns out that the proper relation between  $C$  and  $q(x)$  that guarantees them to have the same answers is closely related (or, as we conjecture, even equivalent) to the relation of *local correspondence* known from modal Correspondence Theory [2, 6, 9].

To illustrate how this works, consider a cyclic query  $q(x) \leftarrow xRx$ ; it is not equivalent to any  $\mathcal{ALC}$  concept, as follows from the tree model property, so rolling-up is not applicable. Now recall that reflexivity  $xRx$  is expressible by (i.e., *locally corresponds* on Kripke frames to) a modal formula  $p \rightarrow \Diamond p$ , where  $p$  is a propositional *variable* (its interpretation on a Kripke frame is universally quantified). Then we introduce a *fresh* concept name  $P$  (i.e., whose interpretation is not constrained by a KB) and translate this modal formula into a DL concept  $\neg P \sqcup \exists R.P$ , which, as we prove, always has the same answers as our query  $q(x)$ .

In general, given a query  $q(x)$  from a family specified below, we invoke an algorithm from Correspondence Theory [6] to build a modal formula  $\varphi$  that locally corresponds to  $q(x)$ ; then we translate  $\varphi$  into a concept (usually, in the same DL as the query) by introducing a fresh concept name for each variable in  $\varphi$ ; the resulting concept, as we show, has exactly the same answers as the original query  $q(x)$  over *any* KB in any DL. The details of this technique are described in Sect. 2. A natural question arises: what if we additionally allow to use fresh role names? In terms of modal logic, this means that a formula  $\varphi$  will contain what we call *variable modalities*. We introduce syntax and semantics for such a modal language in Sect. 3 and generalise the query answering technique to this setting, which results in a wider family of queries that can be answered within this approach. Finally, in Sect. 4 we conclude with formulating some open problems concerning our query answering technique, as well as definability and first-order correspondence for the modal logic with variable modalities.

## 2 Queries answered by concepts

Since our results are applicable to *any* DL (extending  $\mathcal{ALC}$ ), we do not need to describe expressive DLs, and we only briefly recall the definition of  $\mathcal{ALC}$  and  $\mathcal{ALCI}$  and fix some notation. The vocabulary consists of finite sets of *concept names* CN, *role names* RN, and *individual names* (or *constants*) IN. Concepts of  $\mathcal{ALC}$  are defined by the following syntax:

$$C ::= \perp \mid A \mid \neg C \mid C \sqcap D \mid \forall R.C, \quad \text{where } A \in \text{CN}, R \in \text{RN}.$$

Other connectives are taken as customary abbreviations, e.g.,  $C \rightarrow D$  stands for  $\neg(C \sqcap \neg D)$ . In  $\mathcal{ALCI}$ , inverse roles  $R^-$  can additionally be used in place of  $R$ .

A *terminology* (or a TBox)  $\mathcal{T}$  is a finite set of *axioms* of the form  $C \sqsubseteq D$ , where  $C, D$  are arbitrary concepts. An ABox  $\mathcal{A}$  is a finite set of *assertions* of the form  $a:C$  and  $aRb$ , where  $a, b \in \text{IN}$ ,  $C$  is a concept and  $R$  a role. Finally, a *knowledge base*  $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A} \rangle$  consists of a TBox  $\mathcal{T}$  and an ABox  $\mathcal{A}$ .

**Definition 1. (Semantics)** An *interpretation*  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  consists of non-empty *domain*  $\Delta^{\mathcal{I}}$  and an interpretation function  $\cdot^{\mathcal{I}}$  that maps:

- each constant  $a \in \text{IN}$  to an element  $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ ,
- each concept name  $C \in \text{CN}$  to a subset  $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ ,
- each role name  $R \in \text{RN}$  to a binary relation  $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ ;

and is extended to concepts, roles, axioms and assertions as follows:

$$\begin{array}{l|l|l} \perp^{\mathcal{I}} = \emptyset & (R^-)^{\mathcal{I}} = \{ \langle e, d \rangle \mid \langle d, e \rangle \in R^{\mathcal{I}} \} & \mathcal{I} \models C \sqsubseteq D \text{ iff } C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\ (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (\forall R.C)^{\mathcal{I}} = \{ e \in \Delta^{\mathcal{I}} \mid d \in C^{\mathcal{I}} \} & \mathcal{I} \models a:C \text{ iff } a^{\mathcal{I}} \in C^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} & \text{for all } d \text{ such that } \langle e, d \rangle \in R^{\mathcal{I}} & \mathcal{I} \models aRb \text{ iff } \langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}} \end{array}$$

Here  $\mathcal{I} \models \Phi$  stands for ‘ $\mathcal{I}$  satisfies  $\Phi$ ’. An interpretation is called a *model* of a KB if it satisfies all its TBox axioms and ABox assertions. A knowledge base  $\mathcal{KB}$  *entails*  $\Phi$  (notation:  $\mathcal{KB} \models \Phi$ ) if  $\mathcal{I} \models \Phi$ , for all models  $\mathcal{I}$  of  $\mathcal{KB}$ .

**Definition 2. (Queries)** A *conjunctive query* is an expression of the form

$$q(\vec{x}) \leftarrow \exists \vec{y} (t_1(\vec{x}, \vec{y}) \wedge \dots \wedge t_n(\vec{x}, \vec{y})),$$

where  $\vec{x}, \vec{y}$  are tuples of (*distinguished*, resp., *non-distinguished*) variables, and each *atom*  $t_i(\vec{x}, \vec{y})$  is of the form  $u:C$  (*concept atom*) or  $uRv$  (*role atom*), where  $C$  is a concept,  $R$  a role, and  $u, v$  are either variables from  $\vec{x}, \vec{y}$  or constants.<sup>1</sup> A query without concept atoms is called *relational*. Queries with 0 and 1 distinguished variables are called *boolean* and *unary* resp. Given an interpretation  $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ , a query  $q$  of arity  $m$  is interpreted as follows:

$$q^{\mathcal{I}} := \{ \vec{e} \in \Delta^m \mid \mathcal{I} \models \exists \vec{y} (t_1(\vec{e}, \vec{y}) \wedge \dots \wedge t_n(\vec{e}, \vec{y})) \}.$$

The *answer set* of a query  $q(\vec{x})$  w.r.t. a knowledge base  $\mathcal{KB}$  is defined as the set of tuples of constants  $\vec{a}$  that satisfy the query  $q$  in all models of  $\mathcal{KB}$ :

$$\text{ans}_{\mathcal{KB}}(q) := \{ \vec{a} \in \text{IN} \mid \mathcal{KB} \models q(\vec{a}) \}.$$

The following is the main notion of our paper (in fact, it can be formulated for an arbitrary first-order formula  $q$  in the appropriate language, cf. Example 2).

**Definition 3.** A unary query  $q(x)$  is *answered by* a concept  $C$  (written as  $q(x) \approx C$ ) if, for any  $\mathcal{KB}$  and any  $a \in \text{IN}$ , we have:  $\mathcal{KB} \models q(a) \Leftrightarrow \mathcal{KB} \models a:C$ ; in other words, if the queries  $q(x)$  and  $x:C$  always have the same answer set.<sup>2</sup>

A boolean query  $q$  is *answered by* a concept  $C$  (notation:  $q \approx C$ ) if, for any  $\mathcal{KB}$ , the equivalence holds:  $\mathcal{KB} \models q \Leftrightarrow \mathcal{KB} \models a:C$ , where  $a$  is a fresh constant.

<sup>1</sup> In what follows, w.l.o.g. we consider queries without constants, since constants can be eliminated at the price of introducing nominals:  $xRa$  is equivalent to  $xRz \wedge z:\{a\}$ .

<sup>2</sup> Strictly speaking, we should define:  $q(x) \approx C$  over a DL  $\mathcal{L}$  iff  $q(x)$  and  $x:C$  have the same answers for any  $\mathcal{KB}$  formulated in  $\mathcal{L}$ . However, we shall not complicate the matters, since in all our results whenever  $q \approx C$  holds, it hold in fact for any DL  $\mathcal{L}$ .

Our task is to determine what kind of queries can be answered by concepts and when these concepts can be found efficiently (and preferably in the same language as the query). For this aim, we will use results from a branch of modal logic called the Correspondence Theory. Briefly, this theory is devoted to questions whether a modally definable class of frames is also first-order definable, and if so, whether the corresponding first-order formula can be found efficiently (and similarly in the other direction). The background information on that theory can be found in [2, Chap. 3]; here we will recall its basic notions.

Formulas of the (multi-)modal language with  $n$  modalities  $\Box_i$  and a countable set of propositional variables  $p_i$  are defined by the following syntax:

$$\varphi ::= \perp \mid p_i \mid \varphi \rightarrow \psi \mid \Box_i \varphi.$$

As K. Schild observed in [10], this language is a notational variant of  $\mathcal{ALC}$  with  $n$  role names  $R_i$ .<sup>3</sup> Exploiting this fact, whenever  $\varphi$  is a modal formula, we denote by  $C_\varphi$  the concept obtained from  $\varphi$  by replacing  $\Box_i$  with  $\forall R_i$  and  $p_i$  with *fresh* concept names  $P_i$  (it is convenient to reserve a countable set of fresh concept names, i.e., that will not occur in any KB or query).

**Definition 4 (Kripke semantics).** A *frame*  $F = \langle \Delta, r_1, \dots, r_n \rangle$  consists of a non-empty set  $\Delta$  and  $n$  binary relations  $r_i$  on  $\Delta$ . A model  $M = \langle F, \nu \rangle$  (based on  $F$ ) consists of a frame  $F$  and a *valuation* of variables  $p_i \subseteq \Delta$ . The notion “a formula  $\varphi$  is *true* at a point  $e \in \Delta$  in a model  $M$ ” (notation:  $M, e \models \varphi$ ) is defined inductively:  $M, e \not\models \perp$ ;  $M, e \models p_i$  iff  $e \in p_i$ ;  $M, e \models \varphi \rightarrow \psi$  iff  $M, e \not\models \varphi$  or  $M, e \models \psi$ ;  $M, e \models \Box_i \varphi$  iff  $M, d \models \varphi$  for all  $d \in \Delta$  such that  $\langle e, d \rangle \in r_i$ . A formula  $\varphi$  is *valid* (at a point  $e$ ) in a frame  $F$  (notation:  $F, e \Vdash \varphi$  or  $F \Vdash \varphi$  resp.) if it is true (at this point) in all models based on  $F$ .

**Definition 5 (Correspondence).** Let  $\varphi$  be a modal formula,  $\alpha(x)$  and  $\beta$  first-order formulae in the vocabulary  $\{R_1, \dots, R_n, =\}$  with one and no free variables, resp. We say that  $\alpha(x)$  *locally corresponds* to  $\varphi$  (notation:  $\alpha(x) \rightsquigarrow \varphi$ ) if  $F \models \alpha(e) \Leftrightarrow F, e \Vdash \varphi$ , for any frame  $F$  and any its point  $e$ . Similarly,  $\beta$  *globally corresponds* to  $\varphi$  (notation:  $\beta \rightsquigarrow \varphi$ ) if  $F \models \beta \Leftrightarrow F \Vdash \varphi$ , for any frame  $F$ .

For example, a formula  $p \rightarrow \Diamond p$  corresponds (both locally and globally) to reflexivity, whereas  $\Box(\Box p \rightarrow p) \rightarrow \Box p$  is valid in a frame  $F$  iff  $F$  is transitive and has no infinite ascending chains, which is a (monadic) second-order property.

We are ready to establish a relationship between the correspondence relation ( $q \rightsquigarrow \varphi$ ) and query answering ( $q \approx C_\varphi$ ). We conjecture that they are equivalent; however, we have succeeded to prove only one implication and partially the converse one. For proofs, see a paper [13] and a recent technical report [12].

**Theorem 6 (Reduction).** *Let  $q(x)$  be a unary relational<sup>4</sup> query,  $\varphi$  a modal formula. If  $q(x)$  locally corresponds to  $\varphi$ , then the query  $q(x)$  is answered by the  $\mathcal{ALC}$ -concept  $C_\varphi$ . In symbols:  $q(x) \rightsquigarrow \varphi \implies q(x) \approx C_\varphi$ .*

*Similarly for boolean queries and global correspondence: if  $q \rightsquigarrow \varphi$  then  $q \approx C_\varphi$ .*

<sup>3</sup> Note that the words ‘correspondence theory’ in the title of his paper refer to this observation only and have nothing to do with modal Correspondence Theory.

<sup>4</sup> Queries that additionally involve concept atoms  $y:C$  are covered by Theorem 10.

**Lemma 7 (Partial converse).** (1) *Suppose that a unary relational query  $q(x)$  is answered by a concept  $C_\varphi$ , i.e.,  $q(x) \approx C_\varphi$ , for some modal formula  $\varphi$ . Then:*  
 a)  $F \models q(e) \Rightarrow F, e \Vdash \varphi$ , for any frame  $F$  and any its point  $e$ ;  
 b)  $F \models q(e) \Leftarrow F, e \Vdash \varphi$ , for any finitely branching<sup>5</sup> frame  $F$  and any point  $e$ .  
 (2) *The same holds for boolean queries and global validity ( $F \Vdash \varphi$ ), but only for ‘finite’ instead of ‘finitely branching’ frames in (b).*

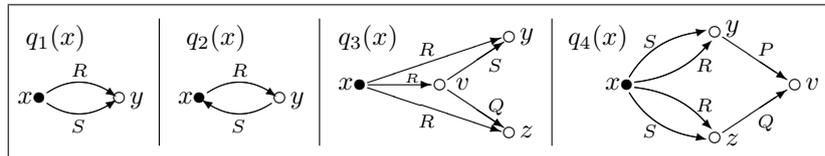
It is undecidable to determine whether a given FO formula corresponds to some modal formula [3]. There were few attempts to find FO fragments for which the problem is decidable.<sup>6</sup> We apply (and extend) those results and identify several families of queries for which the answering concept can be built efficiently. The family  $\mathcal{K}$  below stems from so called *Kracht’s fragment* [6]; the family  $\mathcal{Z}$  contains queries beyond that fragment. Queries from both  $\mathcal{K}$  and  $\mathcal{Z}$  are answered by  $\mathcal{ALC}$ -concepts. By “forgetting” the direction of edges<sup>7</sup> in queries, we obtain a family  $\mathcal{E}$  of queries that are answered by  $\mathcal{ALCI}$ -concepts. The formal description of these families of relational queries (and their non-relational analogues), the corresponding algorithm and the proofs can be found in [13].

**Corollary 8.** *There exist a (polynomial) algorithm that takes a unary relational conjunctive query  $q(x)$  from the following families  $\mathcal{K}$ ,  $\mathcal{Z}$ , and  $\mathcal{E}$  and returns a concept in  $\mathcal{ALC}$  (for  $q \in \mathcal{K} \cup \mathcal{Z}$ ) or  $\mathcal{ALCI}$  (for  $q \in \mathcal{E}$ ) that answers this query. Furthermore:  $\mathcal{K} \not\subseteq \mathcal{Z}$ ,  $\mathcal{Z} \not\subseteq \mathcal{K}$ , and  $(\mathcal{K} \cup \mathcal{Z}) \subset \mathcal{E}$ . So, for any DL  $\mathcal{L}$ , the problem of answering queries within these families has the same complexity as  $\mathcal{L}$  itself.*

- Family  $\mathcal{K}$ :** take an oriented tree with the root  $x$ , and add any number of oriented chains linking  $x$  with other nodes (in any direction).
- Family  $\mathcal{Z}$ :** take an “anti-tree” (i.e., an oriented tree with edges directed from leaves to the root); merge all its leaves and denote the resulting node by  $x$ .
- Family  $\mathcal{E}$ :** the graph of a query is connected and has no cycles that consist of non-distinguished variables only (i.e., any cycle contains the node  $x$ ).

**Example 1.** The queries  $q_1(x) \leftarrow xRy \wedge xSy$  and  $q_2(x) \leftarrow xRy \wedge ySx$  are answered by the concepts  $\forall R.Y \rightarrow \exists S.Y$  and  $X \rightarrow \exists R.\exists S.X$  resp. The following two queries witness that the families  $\mathcal{K}$  and  $\mathcal{Z}$  are incomparable w.r.t. inclusion:

Query in $\mathcal{K} \setminus \mathcal{Z}$ :	$q_3(x) \leftarrow xRv \wedge vSy \wedge vQz \wedge xRy \wedge xRz$
Concept:	$\forall R.(Y \sqcap Z) \rightarrow \exists R.(\exists S.Y \sqcap \exists Q.Z)$
Query in $\mathcal{Z} \setminus \mathcal{K}$ :	$q_4(x) \leftarrow xRy \wedge xSy \wedge yPv \wedge xRz \wedge xSz \wedge zQv$
Concept:	$\forall R.(Y \sqcap Z) \rightarrow \exists S.((Y \sqcap \exists P.V) \sqcup (Z \sqcap \exists Q.\neg V))$



<sup>5</sup> I.e., any point of the frame has a finite number of successors.

<sup>6</sup> And much more results in the “modal to first-order” direction; see [4] for an overview.

<sup>7</sup> It is convenient to represent a relational query as an oriented graph, whose nodes are distinguished ( $\bullet$ ) and non-distinguished ( $\circ$ ) variables and edges are role atoms.

**Example 2.** Boolean (not necessarily conjunctive) queries can be answered similarly; namely, we can check for *modally definable* properties of roles [2, Ch. 3]. For instance, whether a role  $R$  is reflexive, transitive, dense, euclidean, confluent etc. in a KB can be checked, by Theorem 6, using the DL-translations of the modal formulae  $p \rightarrow \diamond p$ ,  $\diamond p \rightarrow \diamond \diamond p$ ,  $\diamond \diamond p \rightarrow \diamond p$ ,  $\diamond p \rightarrow \Box \diamond p$ ,  $\diamond \Box p \rightarrow \Box \diamond p$  resp.

### 3 Modal logic with variable modalities

The results obtained above inspire to introduce a natural extension of modal logic, which is interesting per se and in addition yields to a wider family of queries that can be answered with the same technique. Recall that in order to answer, e.g., a query  $q(x) \leftarrow xRx$ , we introduced a fresh concept name  $P$  and then proved that  $q(x)$  has the same answers as the concept  $\neg P \sqcup \exists R.P$ . It is not hard to see that, without fresh concept names, only tree-like queries can be answered (and this will coincide with the rolling-up technique). So it were fresh concept names that enabled us to answer cyclic queries. Now, can we gain even more if we allow to use fresh role names? As we show below, the answer is Yes (but additional queries are usually not *conjunctive*, but rather first-order formulae of other kinds).

Recall that in the definition of the *validity* of a modal formula in a frame, we quantify over interpretations of propositional variables (i.e., unary predicates), but interpretation of modalities (i.e., binary predicates) is fixed. In other words, the standard modal logic is a logic of *constant* modalities and propositional *variables*. Therefore it is natural to consider a notion of validity, in which the rôle of unary and binary predicates is symmetric. So, we extend modal logic with *variable* modalities and propositional *constants*.

The vocabulary of the *mixed modal logic* consists of  $m$  propositional constants  $A_1, \dots, A_m$ ,  $n$  constant modalities  $\Box_1, \dots, \Box_n$ , and countable sets of propositional variables  $p_i$  and variable modalities  $\Box_i$ . The syntax for formulae is:

$$\varphi ::= \perp \mid p_i \mid A_i \mid \varphi \rightarrow \psi \mid \Box_i \varphi \mid \Box_i \varphi$$

**Definition 9 (Semantics).** A *frame*  $F = \langle \Delta, \vec{\alpha}, \vec{r} \rangle$  consists of a non-empty set  $\Delta$ , a list  $\vec{\alpha}$  of  $m$  unary predicates  $\alpha_i \subseteq \Delta$ , and a list  $\vec{r}$  of  $n$  binary relations  $r_i \subseteq \Delta \times \Delta$ . A *model*  $M = \langle F, \vec{\pi}, \vec{s} \rangle$  consists of a frame  $F$  and countable sequences of unary predicates  $\pi_i \subseteq \Delta$  and binary relations  $s_i \subseteq \Delta \times \Delta$ .

The notion “a formula  $\varphi$  is *true* at a point  $e \in \Delta$  in a model  $M$ ” is defined inductively: boolean cases are standard;  $M, e \models p_i$  iff  $e \in \pi_i$ ;  $M, e \models A_i$  iff  $e \in \alpha_i$ ;  $M, e \models \Box_i \varphi$  iff  $M, d \models \varphi$  for all  $d \in \Delta$  such that  $\langle e, d \rangle \in r_i$ ; and similarly for  $\Box_i$  and  $s_i$ . The notion of *validity* of a formula (at a point) in a frame is standard, but note that here saying “for all models  $M$ ” involves quantification over unary *binary* predicates  $s_i$  (hence it is no longer a *monadic* second-order notion).

The notion of correspondence is defined as in Def. 5, but for first-order formulae in the vocabulary  $\{A_1, \dots, A_m, R_1, \dots, R_n, =\}$ . The mixed modal language is much more expressive. For instance,  $F \models \Box p \rightarrow \Box p$  iff  $r = \Delta \times \Delta$ ; and

$F \models p \rightarrow \Box p$  iff  $|\Delta| = 1$ ; these properties are not expressible in the standard modal language (see. [12] for more examples). It turns out that almost all the results from the previous section can be generalised to the mixed modal logic, with even more elegant formulations, as we do not need to rule out concept atoms  $x:C$  from queries now. In what follows, whenever  $\varphi$  is a mixed modal formula, by  $C_\varphi$  we denote a concept obtained from  $\varphi$  by replacing  $\Box_i$  with  $\forall R_i$ ,  $\Box_i$  with  $\forall S_i$ , and  $p_i$  with  $P_i$ , where concept names  $P_i$  and role names  $S_i$  are fresh (symbols  $A_i$  are left unchanged). The following results are proved in [12].

**Theorem 10 (Reduction).** *Let  $q(x)$  be a unary query,  $\varphi$  a mixed modal formula. If  $q(x)$  locally corresponds to  $\varphi$ , then the query  $q(x)$  is answered by the  $\mathcal{ALC}$ -concept  $C_\varphi$ . In symbols:  $q(x) \leftrightarrow \varphi \implies q(x) \approx C_\varphi$ .*

*Similarly for boolean queries and global correspondence: if  $q \leftrightarrow \varphi$  then  $q \approx C_\varphi$ .*

**Lemma 11 (Partial converse).** **(1)** *Suppose that a unary query  $q(x)$  is answered by a concept  $C_\varphi$ , i.e.,  $q(x) \approx C_\varphi$ , for some mixed modal formula  $\varphi$ . Then  $F \models q(e)$  implies  $F, e \Vdash \varphi$ , for any frame  $F$  and any its point  $e$ .*

**(2)** *The same holds for boolean queries and the global validity ( $F \Vdash \varphi$ ).*

**Example 3 (Mary likes all cats).** Suppose that a  $\mathcal{KB}$  contains an individual Mary, a concept Cat and a role Likes, and we want to express a boolean query whether “Mary likes all cats”. A straightforward way to do this is to write a concept subsumption:  $\text{Cat} \sqsubseteq \exists \text{Likes}^- . \{\text{Mary}\}$ , but it contains an inverse role and a nominal, even if the language of  $\mathcal{KB}$  does not, thus increasing the complexity of reasoning [11]. This query can also be formulated using role negation:  $\text{Mary} : \forall \neg \text{Likes}. \neg \text{Cat}$ , again with an increase of the complexity [7]. The solution we propose enables one to express this query in  $\mathcal{ALC}$ . To this end, note that a mixed modal formula  $\Box p \rightarrow \Box(A \rightarrow p)$  locally corresponds to a first-order formula  $q(x) := \forall y (A(y) \rightarrow xRy)$  (see [12] for a proof). Now let  $A$  stand for Cat and  $R$  for Likes, then our query “Mary likes all cats” can be represented as  $q(\text{Mary})$ . Finally, we apply Theorem 10 and conclude that  $q(\text{Mary})$  holds w.r.t.  $\mathcal{KB}$  iff Mary is an instance (w.r.t.  $\mathcal{KB}$ ) of the following concept (where the concept name SomeConc and the role name SomeRel are fresh):

$$\forall \text{Likes}. \text{SomeConc} \rightarrow \forall \text{SomeRel}. (\text{Cat} \rightarrow \text{SomeConc}).$$

## 4 Conclusions and open questions

One of the achievements of this paper is the established relationship between query answering in DL and Correspondence Theory (Theorems 6 and 10). It allowed us to build a uniform query answering algorithm for some families of conjunctive queries. Furthermore, a modal logic with variable modalities was introduced; although it is quite a natural extension of the standard modal logic, it has not been considered in the literature, to the best of our knowledge.

There is a number of natural problems left open, answers to which would complete the whole picture. Here we mention some of them.

- Q1** Do the converses of Theorems 6 and 10 hold?
- Q2** Which conjunctive queries locally/globally correspond to modal formulae (with or without  $\Box$ 's)? At least, are these families of queries decidable?
- Q3** Which conjunctive queries can be answered by  $\mathcal{ALC}$ -concepts (with or without fresh role names)? By Theorems 6 and 10, queries from **Q2** form a subset of queries from **Q3**, and by Corollary 8, they contain the families  $\mathcal{K}$  and  $\mathcal{Z}$ .
- Q4** The same questions **Q2** and **Q3** for the logics  $\mathcal{ALCT}$ ,  $\mathcal{ALCQ}$ , and  $\mathcal{ALCQT}$ .
- Q5** The expressive power of the mixed modal logic: What classes of frames are definable (i.e., an analogue of Goldblatt-Thomason theorem [2, Th. 3.19])? Which of them are first-order definable?

### Acknowledgements

The research was supported by EPSRC (GR/S63168/01). The author would like to thank Ulrike Sattler for the help during the research, Birte Glimm for fruitful discussions, and the referees for useful comments and suggestions.

### References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.
3. A. Chagrov, L. Chagrova. The truth about algorithmic problems in Correspondence Theory. In *Advances in Modal Logic*, vol. 6, pp. 121–138, College Publications, 2006.
4. W. Conradie, V. Goranko, D. Vakarelov. Elementary canonical formulae: A survey on syntactic, algorithmic, and model-theoretic aspects. *Advances in Modal Logic*, vol. 5, pp. 17–51, King's College Publications, 2004.
5. B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for Description Logics with transitive roles. In *Proc. of DL'06*, pp. 3–14, vol. 189 in CEUR, 2006.
6. M. Kracht. How completeness and Correspondence Theory got married. In M. de Rijke, editor, *Diamonds and Defaults*, pp. 175–214. Kluwer, 1993.
7. C. Lutz and U. Sattler. Mary likes all cats. In *Proc. of DL'2000*, pp. 213–226, 2000.
8. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in *SHIQ*. In *Proc. of DL'06*, pp. 62–73, vol. 189 in CEUR, 2006.
9. H. Sahlqvist. Correspondence and completeness in the first- and second-order semantics for modal logic. In S. Kanger, editor, *Proc. of the 3rd Scand. Logic Symp.*, Uppsala, 1973. North-Holland Publishing Company, Amsterdam, 1975.
10. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI'91*, pp. 466–471, 1991.
11. E. Zolin. *A Navigator for the Complexity of Description Logics*. Web tool. Available at <http://www.cs.man.ac.uk/~ezolin/logic/>
12. E. Zolin. Modal logic with variable modalities with applications to querying knowledge bases. Technical Report, The University of Manchester, 2007. Available at [http://www.cs.man.ac.uk/~ezolin/logic/pdf/var\\_mod\\_2007\\_TR.pdf](http://www.cs.man.ac.uk/~ezolin/logic/pdf/var_mod_2007_TR.pdf)
13. E. Zolin. Query answering based on modal correspondence theory. In *Proc. of the 4th "Methods for Modalities" Workshop (M4M-4)*, pp. 21–37, 2005.

# The Minimal Finite Model Visualization as an Ontology Debugging Tool

Guntis Barzdins and Martins Barinskis

IMCS, University of Latvia  
 martins.barinskis@gmail.com, guntis.barzdins@mii.lu.lv

**Abstract.** We present an new *Protégé* plugin for constructing a minimal satisfiability model of an OWL ontology and visualizing it in the original music score notation.

## 1 Introduction

The ontology satisfiability is a property that indicates whether all classes defined in the ontology can possibly have any instances. Satisfiability implies consistency. The described *Protégé* plugin [1] complements the traditional ontology satisfiability debugging tools [2]: if an automatically constructed minimal model of the ontology contradicts the author’s intentions, then the ontology itself is either wrong or incomplete.

Currently, we consider only ontologies with a finite satisfiability model. Here we focus on the model visualization enhancements, while the overall approach is described in [3].

## 2 Acquiring and Visualizing a Satisfiability Model

Let us consider a simple OWL DL pizza ontology:

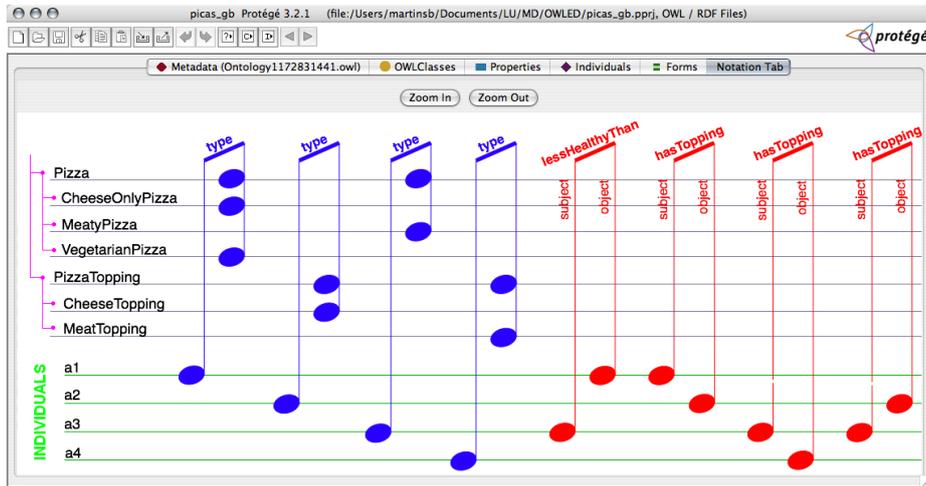
```
Ontology(
  Class(Pizza partial restriction(hasTopping
    someValuesFrom(PizzaTopping)) owl:Thing)
  Class(MeatyPizza complete restriction(hasTopping
    someValuesFrom(MeatTopping)) Pizza
    restriction(lessHealthyThan someValuesFrom(VegetarianPizza)))
  Class(CheeseOnlyPizza complete restriction(hasTopping
    someValuesFrom(CheeseTopping)) Pizza
    restriction(hasTopping allValuesFrom(CheeseTopping)))
  Class(VegetarianPizza complete complementOf(restriction(hasTopping
    someValuesFrom(MeatTopping)) Pizza)
  Class(PizzaTopping partial owl:Thing)
  Class(CheeseTopping partial PizzaTopping)
  Class(MeatTopping partial PizzaTopping)
  DisjointClasses(CheeseTopping MeatTopping)
  DisjointClasses(PizzaTopping Pizza)
  ObjectProperty(hasTopping domain(Pizza) range(PizzaTopping))
)
```

Pellet and Fact++, the popular OWL DL/OWL 1.1 reasoners, do not output an ontology satisfiability model. Therefore, to build a model, we map the OWL ontology to the FOL formula and use *Mace4*, a FOL model builder:

```

relation(CheeseOnlyPizza(_), [ 1, 0, 0, 0 ]),
relation(CheeseTopping(_), [ 0, 1, 0, 0 ]),
relation(MeatTopping(_), [ 0, 0, 0, 1 ]),
relation(MeatyPizza(_), [ 0, 0, 1, 0 ]),
relation(Pizza(_), [ 1, 0, 1, 0 ]),
relation(PizzaTopping(_), [ 0, 1, 0, 1 ]),
relation(VegetarianPizza(_), [ 1, 0, 0, 0 ]),
relation(hasTopping(_,_), [ 0, 1, 0, 0,
                          0, 0, 0, 0,
                          0, 1, 0, 1,
                          0, 0, 0, 0 ]),
relation(lessHealthyThan(_,_), [ 0, 0, 0, 0,
                                0, 0, 0, 0,
                                1, 0, 0, 0,
                                0, 0, 0, 0 ])
    
```

The created satisfiability model is further visualized using an original “music score notation”:



The traditional class hierarchy and individuals are visualized as lines that are interconnected by the notes (relations). Filtering can be used to navigate larger models.

## References

1. Barinskis, M., Barzdins, G.: OWL DL Satisfiability Model Visualization Plugin for Protege: <http://apps.lumii.lv/satmodviz/>
2. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL Ontologies. In: WWW'05, ACM Press (2005) 633–640
3. Barinskis, M., Barzdins, G.: Satisfiability Model Visualization Plugin for Deep Consistency Checking of OWL Ontologies. In: OWLED 2007 (to be published). (2007)

## Structural Theory of Science as a Systematic Framework for the Design of DL's and CD's for E-Science

Hansje Braam

Utrecht University, Faculty of Humanities  
hansje.braam@let.uu.nl

### 1 Context

The construction of ontologies (DL-instances, Concrete Domains) for E-Science (in my individual case E-Humanities) is quite hard:

- i) in many cases we have to cope with ontological divergencies, up to completely disjunct ontologies, as many competing theories (and scientific practices) rely on non-common features, entity-classes and relations;
- ii) many scientifically relevant predicates have arity  $> 2$ , a potential problem for DL(CD)'s;
- iii) in science (natural science, but humanities as well) formalisms are used with high complexity (mathematics, formal grammars, predicate logics, restricted natural language etc.);
- iv) higher order predicates occur -especially in the humanities- (f. e. it is possible to reason about certain modes of interpretation). It is a.o. this semantical complexity which distinguishes science from folk cognition.

I propose to use Structural Theory of Science (the classical reference [1], see also [2]) as a common framework for the analysis of scientific theories (and practices) in which the specifications of adequate ontologies can be related and subsequently used in the systematic construction of the DL's and CD's.

### 2 Structural Theory of Science

Structural theory of science (STS) is basically a semantical approach to scientific theories. It focusses on a model-theoretical exposition of the meaning of theories. Thus, it is not restricted to formal deductive theories. STS has been applied to wide range of scientific theories, including physics, linguistics, psychology, economy and theory of literature. It also facilitates the exact study of theory dynamics and intertheoretical relations. Theories are not approached as deductive systems of statements (axioms, propositions, theorems etc.) but as possibly complex expressions about the relations between certain sets of models. Models are tuples of domains and predicates of the theory. Simply put, four levels are relevant:

- i) the set of the intended applications  $I_A(T)$  (in most cases, that what the theory

is about: the empirical field);

ii) the set of all possible models for the basic ontology of the theory  $M_{pp}(T)$ , excluding its theoretical notions (e.g. including only what the theory takes for granted);

iii) the set of all possible models for the ontology of the theory  $M_p(T)$ , which includes the domains and ranges of its theoretical predicates;

iv) the set of models of the theory itself  $M(T)$ , the theoretical predicates are specified in full.

An approximating function  $F$  relates the set of intended applications to  $M$ : the empirical claim of a theory  $T$  is that  $F(I_A(T)) \subset M(T)$ . Generally we have  $F(I_A(T)) \subseteq M(T) \subseteq M_p(T)$  and  $M_{pp}(T)$  is a projection of  $M_p(T)$ . In DL-terms one could associate  $I_A$  with possible models for the ABox of  $T$  and  $M$  with all models for its TBox. Example: for a simple theory  $ILT$  of interpretation of literary English works we could have  $M_{pp}(ILT)$  as a set of tuples  $\langle Text, Interpreter \rangle$ ,  $M_p(ILT)$  as tuples from  $M_{pp}(ILT)$  extended with the full *Interprete*-relation (i.e.  $Interpreter \times Text \times Interpretation$ ), where *Interpretation* is a  $ILT$ -qualified subset of *Text*. The specification of  $M(ILT)$  (by specification of *Interprete*) may well be non-computational.  $I_A(ILT)$  covers English literary works with interpretations by certain interpreters. The intertheoretical relations which can be defined on these 4 levels are quite relevant for epistemological and methodological reasons. These relations are therefore also important in the academic educational realm.

STS is expressed using set-theoretical concepts [4] and cannot be used as an implementation language, it is much too strong for that. However, it can be used as a common context for the study of possible DL and CD versions for fragments of scientific theories. In this way we can have a theoretical environment for systematic analysis and development of DL/CD for E-Science. Important issues are how the STS concepts of theoreticity, theory-nets, reduction and emergency [3] can be connected to relations between ontologies in the DL/OWL world. An urgent need is the extension of STS (and DL's) towards the incorporation of empirical experimentation. This will need extensive analysis of experimentation and data gathering itself [5] as well as action logics.

## References

1. Balzer, W., Ulises Moulines, C., Sneed, J. D.: *An Architectonic for Science. The Structuralist Program*, Reidel, Dordrecht etc. 1987.
2. Balzer, W., Sneed, J. D., Ulises Moulines, C. (eds.): *Structuralist Knowledge Representation. Paradigmatic Examples*, Rodopi, Amsterdam etc. 2000.
3. Bickle, J.: Concepts Structured through Reduction: a Structuralist Resource Illuminates the Consolidation-Long-Term Potentiation (LTP) Link, *Synthese*, **130**, 2002, 123-133.
4. Suppes, P.: *Representation and Invariance of Scientific Structures*, CSLI, Stanford, 2002.
5. Gooding, D.: *Experiment and the Making of Meaning. Human Agency in Scientific Observation and Experiment*, Kluwer, Dordrecht etc. 1990.

## Integrating Semantic Annotations in Bayesian Causal Models

Hector G. Ceballos and Francisco J. Cantu

Tecnologico de Monterrey, Mexico  
 {ceballos, fcantu}@itesm.mx

### 1 Introduction

Probabilistic reasoning has been powered by the formalization of causality theory through Bayesian causal models[1]. Even when its semantic is flexible enough to model complex problems, it has to deal with the problem of interoperability between models. In the research community the necessity of contexts for these models has been pointed out. We need means to represent the context on which the causal model is developed and the meaning of causal model events in the real world.

### 2 Semantic Bayesian Causal Models

We introduce *Semantic Bayesian Causal Models*(SBCM) which integrate a causal model with a semantic layer into an intelligent agent. An SBCM works as an inference engine in an intelligent agent in stochastic environments and is basically constituted by a Bayesian causal model to represent and reason about causal relationships among events, and semantic annotations in an ontology recognized by other agents that describe these events. A SBCM is represented by:

$$M = \langle V, U, G_{VU}, P(v_i|pa_i, u_i), P(u), C, Z, F, A, O, B \rangle \quad (1)$$

where  $V$  is the set of endogenous variables,  $U$  is the set of exogenous variables,  $G_{VU}$  is a causal graph consisting of variables in  $V \times U$ ,  $P(v)$  is the Bayesian probabilistic distribution,  $P(u)$  is a probabilistic distribution used to explain bias in the system or interference produced by external factors,  $C \subset V$  represents endogenous variables that can be manipulated by agent (control variables),  $Z$  is the subset of endogenous variables that cannot be manipulated by agent (co-variates),  $F \in Z$  represents agent objectives which we interpret as final cause in Causality theory,  $A$  is the set of semantic annotations over  $V$  expressed through Description Logics (DL) statements in terms of OWL ontology  $O$ , and  $B$  is the set of current beliefs expressed as interventions ( $V_i = v_i$ )<sup>1</sup>.

Agent inference process is performed at two levels: semantic and causal. Former enables common understanding between agents meanwhile the latter summarizes agent experience and guides its behavior. In the first phase, agent perceives the environment through its sensors and transforms its perceptions into

<sup>1</sup> Capital letters represent variables ( $V_i$ ) meanwhile small letter represents values( $v_i$ )

DL statements expressed in a given ontology. Then perceptions are compared against annotations over variables in  $Z$  to determine if any covariate can be instantiated (node instantiation phase). The result of this process is a set of interventions over  $Z$  that are integrated with current beliefs ( $B$ ). Annotations associated to every variable ( $A_i$ ) will be expressed as queries in triplet format (SPARQL). The result of running a variable query over current perceptions will determine if the variable is activated (intervened). A special variable in the query will be bind to the variable value in the intervention. If  $A_i$  doesn't contain this special variable,  $Z$  variable is made true when perceptions match annotations. Otherwise, is made false using a kind of negation as failure.

In the second phase beliefs are revised with current perceptions and resulting interventions are applied to the causal model producing an instantiated causal model used to perform the inference. Plans aligned to reach  $F$  are identified and through a heuristic the most feasible plan and action are selected. Selected action is represented by an intervention over a control variable ( $C_w = c_w$ ). Annotations over  $C_w$  are instantiated with  $c_w$  and triplets resulting are used to encode action.

### 3 Conclusions

Having annotations over causal model variables enables matching variables among different causal models and calculating a distributed causal effect[2] through nodes sharing the same semantic content. Agents will be in position to exchange information about causal relationships influencing other agents behavior to enforce cooperation.

Besides, semantic information associated to variables presenting an irregular behavior (noise) would lead to causal relationships discovery. Semantic information dismissed in the node instantiation phase can be used for this purpose. This way, we are in a position of not just learning probabilistic distributions but the causal structure too[3].

The final purpose of this model is to develop agents that reason over a network of causal relationships guided by Causality theory introduced by Aristotle and mathematical models developed by J. Pearl. We call this architecture *Causal Agent*.

### References

1. Pearl, J.: Causality. Models, Reasoning, and Inference. Cambridge University Press (2000)
2. Maes, S., Meganck, S., Manderick, B.: Identification of causal effects in multi-agent causal models. In: IASTED International Conference on Artificial Intelligence and Applications. (2005) 178–182
3. Flores-Quintanilla, J., Morales-Menendez, R., Ramirez-Mendoza, R., Garza-Castanon, L., Cantu-Ortiz, F.: Towards a new fault diagnosis system for electric machines based on dynamic probabilistic models. In: American Control Conference, 2005. Proceedings of the 2005. Volume 4., IEEE (2005) 2775–2780

## Ontological Modeling for Neurovascular Disease Studies: Issues in the Adoption of DL

Gianluca Colombo, Marco Antoniotti, Daniele Merico, Flavio De Paoli, and Giancarlo Mauri

Dipartimento di Informatica, Sistemistica e Comunicazione  
Università degli Studi di Milano–Bicocca  
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy  
{gianluca.colombo, marco.antoniotti,}@disco.unimib.it

### 1 Introduction

We describe the ontological modeling issues encountered in the EU-funded NEUROWEB project. The aim of the project is to support association studies in the field of neurovascular medicine, integrating the different local repositories maintained by the clinical partners. Specifically association studies are carried out by searching statistical correlations between a feature and an *aggregate state* (i.e., *phenotype*), such as the occurrence of a complex/multi-factorial pathology. In medicine, and thus also in the specific neurovascular domain, the occurrence of a phenotype is implicitly asserted through the diagnostic process, an activity that is deeply rooted in the expert knowledge of the clinical community. Therefore, the major ontological commitment is to define phenotypes having a shared semantic, and connected to the data stored in local repositories. Since association studies are extremely sensitive to noise, a knowledge-intensive treatment is required.

### 2 A Critical Assessment of Existing Bio-Ontologies

Independent analyses of common-use, generalist biomedical ontologies (such as SNOMED-CT) have often revealed significant deficiencies. These are not primarily due to DL expressiveness limits, but rather to the adoption of too generic semantics for relations, which lead to inconsistencies in specific cases. For instance, Ceusters et al. [1] provide examples of erroneous subsumption relations in SNOMED-CT, as a consequence of using relationships that are too generic in their scope. This sort of representation defects is usually grounded on an insufficient conceptual modeling due to the lack of expert core knowledge and, in turn, it also leads to neglecting semantic issues being discussed in the formal logic community. Clinically-grounded phenotypes, the core entities to be modeled, are aggregate concepts, which can be deconstructed into fundamental building blocks. In analogy to the work done in the genomic field [2], we identify four conceptual components: (A) the *anatomical part* interested, (B) the *value*

observed, (C) the *device or method* used for the diagnostic exam, (D) the *etiology*, i.e. the complex of causes and risk factors concurring to the pathogenesis of the disorder. The modeling issues to be faced are two-fold, conceptual and formal: (1) clinical phenotypes are not explicitly described (e.g., in manuals); on the contrary, they are grounded on the core expert knowledge guiding the diagnostic process. (2) the aggregate nature of phenotypes requires a mereological treatment; therefore it is necessary to explicitly take into account the different formal semantics of part-of relations, as described both by foundational works in knowledge representation [3, 4] and in biomedicine ontological modeling [5, 6] fields. We argue that the availability of core expert knowledge is the enabling factor of an adequate use of formal relation semantics; as a matter of fact, in order to develop the conceptual model, we oriented Knowledge Acquisition to explicitly formulate the otherwise implicit mereological semantics assumed by the domain experts. Without core expert knowledge sources, the semantic specification of such relationships would be ungrounded. This strategy has led to the development of a two-layered, inter-connected ontological framework [7]: the top level is a taxonomy of pathology types and subtypes; the bottom level enables to represent the building blocks underlying the pathological phenotypes, exploiting the different mereological relations already identified (anatomical part, value, diagnostic device, etiology). The formal language adopted is *SHIQ* providing the required expressive power for mereological relations, as discussed in [3]. The specific experience acquired through the NEUROWEB project suggests that semantic refinements, spawned by formal contributions, provide valuable guidance to improve the Knowledge Engineering task. In turn, an advancement in the field of Knowledge Representation requires an adequate Knowledge Engineering methodology, capable of supporting the adoption of a DL enriched of expressiveness for ontological modeling in specific domains.

## References

1. Ceusters, W., Smith, B., Flanagan, J.: Ontology and medical terminology: Why description logics are not enough. In: Proceedings of TEPR 2003 - Towards an Electronic Patient Record, San Antonio, Texas (2003)
2. Bard, J.B.L., Rhee, S.Y.: Ontologies in biology: Design, applications and future challenges. *Nature Reviews Genetics* **6**(5) (2004/03) 213–222
3. Sattler, U.: Description logics for the representation of aggregated objects. In W.Horn, ed.: Proceedings of the 14th European Conference on Artificial Intelligence, IOS Press, Amsterdam (2000)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The description logic handbook: theory, implementation, and applications. Cambridge University Press, New York, NY, USA (2003)
5. Hahn, U., Schulz, S.: Towards a broad-coverage biomedical ontology based on description logics. In: Pacific Symposium on Biocomputing. (2003) 577–588
6. Bodenreider, O., Stevens, R.: Bio-ontologies: current trends and future directions. *Brief Bioinform* **7**(3) (2006) 256–274
7. Colombo, G., Merico, D., Antoniotti, M., Frisoni, G., De Paoli, F., Mauri, G.: An ontological modeling approach to neurovascular disease study: the neuroweb case. *BMC Bioinformatics* (2007) to appear.

## An ExpTime Tableau-based Decision Procedure for *ALCQI*

Yu Ding and Volker Haarslev  
Concordia University, Montreal, Canada

### 1 Motivation and Notions

The *algebraic method*, when used to deal with *qualified number restrictions*, is realized by an *atomic decomposition* step that generates an exponential number of sub-problems[HTM01]. It is not very clear this approach could lead to worst-case optimal tableau-based decision procedures for the fundamental *concept satisfiability* problem. Interestingly, it is known from practice that the *algebraic method* has better *run-time performance*. So far the practical success however is confined in DLs without *inverse roles*, and it is unclear how to use the *algebraic method* for DLs having both *Q* and *I*. All these are baffling and beg for answers.

Firstly, we confirm that the *algebraic method* in general, according to the well-known result on *integer linear programming*[Pap81], leads to a worst-case ExpTime tableau-based decision procedure for the *concept satisfiability* problem. Secondly, we extend the *algebraic method* to DLs with *inverse roles*. To realize two goals simultaneously, several ingredients must be prepared to form a recipe. Naturally, the well-known *global (sub)tableaux caching technique* and the *atomic decomposition* principle must be selected. But neither of the two is directly applicable to DLs with inverse roles, extra ingredients are given below.

**Tableau Structure:** The *tableau structure* (TS) is a *labeled graph*. Each node is labeled with a set of *concepts*, each edge is labeled with a *role*. Additionally, (1) each node is labeled with (0 to many) algebraic objects<sup>1</sup>; (2) each edge is associated with two variables, one for *indicator* and one for *cardinality number*.

**Propositional Branch:** This notion abstracts the execution of the  $\sqcap$ -rule and the  $\sqcup$ -rule (common of *tableau expansion rules* for DLs in the *ALC*-family).

**Cut Formulae:** Given a concept *E* subject to *concept satisfiability test* w.r.t. a GCI  $\top \sqsubseteq G$  in *ALCQI*, for each modal subformula (of *E* and *G*) of the form  $\exists^{\geq n} R.C$  (where *R* is any role),  $\top \sqsubseteq C \sqcup \tilde{C} \sqcup \exists^{\leq 0} R^-. \top$  is a *cut formula*<sup>2</sup>.

**Constraints Fine-Tuning:**<sup>3</sup> In the *tableau structure*, let *x* and *y* be neighbors, *x* is completed and *y* is not completed, *x* has a *R*<sup>-</sup>-edge to *y*, we have:  
 – if  $\exists^{\leq n} R.C \in \mathcal{B}(y)$  and  $C \in \mathcal{C}(y)$ , then  $\exists^{\leq n-1} R.C \in \mathcal{B}'(y)$ ; else  $\exists^{\leq n} R.C \in \mathcal{B}'(y)$ ;  
 – if  $\exists^{\geq n} R.C \in \mathcal{B}(y)$  and  $C \in \mathcal{C}(y)$ , then  $\exists^{\geq n-1} R.C \in \mathcal{B}'(y)$ ; else  $\exists^{\geq n} R.C \in \mathcal{B}'(y)$ ;  
 where  $\mathcal{B}(y)$  denotes the *working propositional branch*, and  $\mathcal{B}'(y)$  its *fine-tuning*. The adjustment of restrictions of a *propositional branch* at a node based on the *cut-sets* ( $\mathcal{C}(y)$ , concepts from cut formulae) at its neighbors is called *fine-tuning*.

**Atomic Decomposition and Integer Linear Programs:** What is typical of the *algebraic approach* is the building of one *integer linear program* from the decomposition of *role fillers* for a group of (*qualified*) *number restrictions*.

<sup>1</sup> Each object represents an integer program  $[A|B]x = b$ , where *A* is  $m \times (2^m - 1)$  coefficient matrix, *B* is  $m \times m$  matrix, *b* is a *m*-vector, s.t.  $x \geq 0$ , integer.

<sup>2</sup> It can be read in *ALCQI* as  $\exists^{\geq 1} R^-. \top \sqsubseteq C \sqcup \tilde{C}$  or in *ALCI* as  $\exists R^-. \top \sqsubseteq C \sqcup \tilde{C}$ .

<sup>3</sup> Due to space limit, the fine-tuning shown here is correct only for tree structures.

The *atomic decomposition* forms all combinations<sup>4</sup> about *role fillers* and their negations. Each such combination is considered as a conjunction, and will be formed as a successor node. The unsat. of one combination (i.e. one successor) is reflected by setting the indicator variable to 0, which in turn might affect the feasibility of the *integer program* at the *fine-tuned working propositional branch*.

## 2 A Short Description of the Decision Procedure

**Decision Procedure:** There are two major data structures: **Nogood** (for unsat *caching* across different tableau structures) and **Witness** (for static *blocking* of tableau nodes within one tableau structure). The procedure starts building a tableau structure (TS) from a root node labeled with the concept subject to satisfiability test. It switches to explore a different TS if the current one can not be saturated clash-free. When switching to explore another TS, at least one new **Nogood** will be found. Thus, at most  $2^{O(n)}$  number of TSs will be explored<sup>5</sup>.

**Complexity Analysis:** Three factors are considered : (1) the maximum cost per *tableau node*; (2) the maximum size of a *tableau structure*; (3) the maximum number of tableau structures necessarily to form. We allow  $2^{O(n^c)}$  cost per node for some constant  $c$ , but require the size of each tableau structure be of  $2^{O(n)}$  and only  $2^{O(n)}$  such tableau structures be formed. So, the total cost is  $2^{O(n^{\max(1,c)})}$ .

**Conclusion:** For DLs with both *inverse roles* and (*qualified*) *number restrictions*, we have introduced restricted *cut-formulae* to guarantee the soundness of the *global (sub)-tableaux caching*[DM99], and introduced the *fine-tuning* of qualified number restrictions to adapt the *algebraic method* to DLs with inverse roles. The former can be treated like GCIs; the latter can be integrated in the *tableau expansion rules* as shown in [HTM01] for *SHQ*. Solving *integer linear program* is known to be NP-complete, and so is the *propositional satisfiability* problem. Both problems possibly appear simultaneously in each tableau node. The decision procedure is designed to respect these facts. Further, the *inconsistency propagation rules*[DM99] are extended to work on *tableau structures* labeled with *algebraic objects*. The decision procedure is a worst-case ExpTime framework for *SHOQ* and *SHIQ*. Details and proofs are omitted. Comments and references to important literature are regrettably impossible to give here.

## References

- [DM99] Francesco M. Donini and Fabio Massacci. Exptime tableaux for alc. *Artificial Intelligence*, 124:87–138, 1999.
- [HTM01] Volker Haarslev, Martina Timmann, and Ralf Möller. Combining tableaux and algebraic decision procedures for dealing with qualified number restrictions in description logics. *IJCAR-2001*, pages 39–48, 2001.
- [Pap81] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.

<sup>4</sup> Regardless of the differences, the *atomic decomposition* and the *integer linear programs* have intricate connections to the *choose-rule* and  $\leq$ -rule.

<sup>5</sup> To explore one TS, either DFS or BFS suffices. There is no assumption about the order in which TSs are explored. The switching could be implemented either as restarting or backjumping, or any sound method that shares previous computations.

## Tools for the QuOnto System Conversion between OWL and DL-Lite<sub>F</sub> with Protégé - OWL Plug-in

Manuel Dioturni<sup>1</sup>, Maurizio Iacovella<sup>2</sup>

<sup>1</sup> Student at University of Rome "La Sapienza", <sup>2</sup> Engineer at Epistemica S.r.l.  
manuel\_dioturni@hotmail.com, m.iacovella@epistemica.com

### 1 Introduction

The QuOnto System [1] was developed to implement efficient reasoning algorithms over Knowledge Bases with large amount of instances. It is based on a particular Description Logic, called DL-Lite<sub>F</sub> [2], briefly described as:

$$\begin{array}{l} Cl \longrightarrow A \mid \exists R \mid Cl_1 \sqcap Cl_2 \\ Cr \longrightarrow A \mid \exists R \mid \neg A \mid \neg \exists R \\ R \longrightarrow P \mid P^- \\ \text{TBox assertions : } Cl \sqsubseteq Cr, (\text{funct } R) \end{array}$$

It represents a good trade-off between expressive power and computational complexity of sound and complete reasoning, in particular, it is tailored to capture basic ontology languages and allows to answer to complex queries expressed on ontologies in LOG-SPACE compared to the data complexity (the data size) [3]. This system doesn't feature yet a user interface nor managing tools and it uses a particular XML syntax to write the ontology, this makes it not easy-to-use. Protégé is an open-source ontology editor based on Java that has a plug-in architecture which makes it flexible and easy to be extended. In addition it can read and write ontologies in OWL format, that is the standard language proposed by W3C. This features have driven the idea to use it as a framework to manage some element of the QuOnto System.

### 2 Description

The first work for this objective is to translate ontologies written in OWL language to an XML document following the proprietary XML Schema used as TBox input for QuOnto.

The translation from OWL to DL-Lite<sub>F</sub> is not always possible because DL-Lite<sub>F</sub> is a strict subset of OWL, in particular of OWL-Lite, the least expressive fragment of OWL, which presents some constructs (e.g., some kinds of role restrictions) that cannot be expressed in DL-Lite<sub>F</sub>, and that make reasoning in OWL-Lite not-tractable if the formalism is not restricted [1]; so it makes the ontology check necessary to know if it is DL-Lite<sub>F</sub> expressible, and in this case, we would submit it to the QuOnto engine.

When the ontology has been defined in Protégé [4] it is possible to access its OWL code through the Protégé API [5]. This allows the complete list of the ontology components to be obtained, and subsequently checked and translated.

The ontology check consists of the analysis of the constructs involved in the definition of the particular OWL ontology in order to know whether a corresponding one in DL-Lite<sub>F</sub> exists for each one: we use Protégé API to scan the whole ontology structure inspecting each OWL element so to recognize, depending on the nesting of the elements, if the conversion can take place; only in this case we can obtain the corresponding ontology in DL-Lite<sub>F</sub> (QuOnto compliant) that now we are able to submit, through the QuOnto API, to the QuOnto reasoner. At this point, it is possible to invoke QuOnto for a consistency check.

This issue has represented the critical task: to design the whole mapping existing between OWL constructs and DL-Lite<sub>F</sub> constructs (if the mapping is possible), so to pass by the original ontology in OWL to an equivalent ontology in DL-Lite<sub>F</sub>, where “equivalent” means that “they have the same logical models”.

### 3 Conclusions and future work

We have shown how it is possible to extend the features of a largely used ontology editor to translate an ontology from OWL to DL-Lite<sub>F</sub>, in order to carry out reasoning tasks on it using the QuOnto System. The use of the Protégé API has been a valid way to proceed, having it offered useful features to access to the OWL source code. Thanks to this Protégé plug-in, testing the QuOnto capabilities and comparing it with other reasoners will be easier for the user.

In future, we would investigate how to link Protégé directly to the QuOnto Core API to allow to the user to manage this system in a more simple and practical way.

### References

1. D.Calvanese, G.De Giacomo, D.Lembo, M.Lenzerini and R.Rosati. Tailoring OWL for Data Intensive Ontologies. In *Proc. of the Workshop on OWL: Experiences and Directions (OWLED 2005)*, 2005
2. A.Acciarri, D.Calvanese, G.De Giacomo, D.Lembo, M.Lenzerini, M.Palmieri and R.Rosati. QuOnto: Querying ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, 2005.
3. D.Calvanese, G.De Giacomo, D.Lembo, M.Lenzerini and R.Rosati. DL-Lite: Tractable Description Logics for Ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, 2005
4. M.Horridge, H.Knublauch, A.Rector, R.Stevens and C.Wroe. A practical guide to building OWL ontologies using the Protégé-OWL plug-in and CO-ODE tools. (*August 27, 2004*).
5. H.Knublauch. Protégé OWL-API Programmer’s Guide.

## Using Off-the-Shelf Reasoners for Reasoning over Distributed ABoxes

Florian Fuchs and Michael Berger

Siemens AG, Corporate Technology, Intelligent Autonomous Systems  
 Otto-Hahn-Ring 6, 81739 Munich, Germany  
 {florian.fuchs.ext|m.berger}@siemens.com

### 1 Introduction

In the recent past, there has been a growing interest in reasoning over large ABoxes. As an example, consider a large-scale infrastructure monitoring system, where types of monitored events are modelled as concepts in the TBox and actual observations are represented as individuals with corresponding assertions in the ABox. ABox reasoning (such as instance retrieval and instance check) is used for analyzing the acquired monitoring data.

In many scenarios, the ABox data is produced distributedly, e.g. by the sensor system which acquired the observation. Today’s established reasoners (RacerPro, Pellet, Fact++, KAON2) require the complete ABox to be available locally for query answering. However, centralizing the ABox is frequently not an option: Data is owned by different organizations, who do not want it to be replicated outside of their control. Data volumes may be too large to be handled by a single reasoner and induce high network traffic. High data rates prevent effective caching.

Existing approaches to reasoning over distributed ABoxes [1] cannot directly be adopted as they assume several semantically-mapped TBoxes. As they propose to extend the underlying reasoning algorithm, they require to modify an existing reasoning implementation, which is only possible for open source implementations and requires in-depth knowledge about implementation details.

This work therefore investigates an alternative approach, where reasoner instances are considered black boxes and process a query using subqueries to other relevant reasoner instances. We assume a set of reasoner instances  $\{R_i\}$  with a common TBox  $\mathcal{T}$  and a distributed ABox  $\mathcal{A} = \bigcup_i \mathcal{A}_i$ , where  $\mathcal{A}_i$  is the ABox subset available at  $R_i$ . The goal is to answer (grounded) conjunctive queries  $Q = q_1 \wedge \dots \wedge q_n$  (with  $q_i$  (named) concept or role query terms), i.e. to determine all (named) variable bindings for which  $\langle \mathcal{T}, \mathcal{A} \rangle \models Q$ .

Each reasoner instance consists of a *query processing component* and a *reasoning component* with a local knowledge base. The query processing component processes the incoming query, generates subqueries to other relevant reasoner instances and adds their answers to the local knowledge base. Then the local reasoning component simply answers the original query w.r.t. the updated local knowledge base and returns the result. Off-the-shelf reasoners can be reused as a

reasoning component. The query processing component is separate and supports different query answering strategies. Apart from conjunctive queries, the query language allows retrieval of explicit ABox assertions. This setup provides a flexible framework for query answering over distributed ABoxes. We design query answering strategies for different combinations of DL expressivity and types of ABox distribution and investigate their properties such as answer completeness.

A naive strategy downloads all ABoxes  $\mathcal{A}_i$  to the local knowledge base using subqueries that retrieve all assertions. The reasoning component checks the consistency of  $\bigcup_i \mathcal{A}_i$  and answers  $Q$ . This strategy delivers complete answers for arbitrary ABox distributions and the DL supported by the reasoning component.

A second strategy can be used for partitioned ABoxes, i.e.  $H_{\mathcal{A}_i} \cap H_{\mathcal{A}_j} = \emptyset$  for all  $i, j, i \neq j$  where  $H_{\mathcal{A}_i}$  is the set of individuals that occur in assertions in  $\mathcal{A}_i$ . In this case,  $\mathcal{A}$  is consistent iff  $\mathcal{A}_i$  is consistent for every  $i$ . A variable binding satisfies a conjunct  $q_k$  w.r.t.  $\langle \mathcal{T}, \mathcal{A} \rangle$  iff there are  $i, k$  so that the binding satisfies  $q_k$  w.r.t.  $\langle \mathcal{T}, \mathcal{A}_i \rangle$  (see also [2]). So the query  $Q$  is answered by retrieving satisfying variable bindings for each conjunct  $q_k$  from each  $R_i$  (this can be optimized by using appropriate index structures). The resulting bindings are then consolidated w.r.t. the original conjunctive query. This strategy delivers complete answers also for expressive DLs such as *SHIQ*.

A third strategy allows shared individuals among ABox subsets ( $N_{shared} = \bigcup_{i,j,i \neq j} H_{\mathcal{A}_i} \cap H_{\mathcal{A}_j}$ ), but imposes the following restrictions: For all  $a \in N_{shared}$  only concept assertions  $C_{shared}(a)$  and role assertions  $R(a, b), R(b, a)$  with  $R \sqsubseteq R_{shared}$ ,  $b$  arbitrary, are allowed.  $C_{shared}$  is disjoint from all other named concepts,  $R \sqsubseteq R_{shared}$  does not occur in any concept definition and role axiom except role hierarchy. These restrictions still allow to express the underlying graph structure of a monitored infrastructure, for example. Moreover, consistency checks and query conjuncts can be answered locally, which allows to process conjunctive queries as described for the previous strategy. Answers are complete for expressive DLs such as *SHIQ*.

We have implemented this framework and use it as a testbed for experiments with different query answering strategies: Reasoner instances use an extended SPARQL query engine as query processing component and Pellet as reasoning component. They communicate using SPARQL and the SPARQL protocol. Future work includes developing new strategies with less restrictions on ABox distribution. We investigate the spectrum between the two extremes of (i) downloading all ABox assertions (naive strategy) for gaining complete answers and (ii) downloading only ABox assertions about selected individuals for gaining a certain guaranteed degree of completeness.

## References

- [1] Serafini, L., Tamin, A.: Distributed Instance Retrieval in Heterogeneous Ontologies. In: *Proc. of Workshop Semantic Web Applications and Perspectives*, 2005.
- [2] Pothipruk, P., Governatori, G.: A Formal Ontology Reasoning with Individual Optimization: A Realization of the Semantic Web. In: *Proc. of Conf. on Web Information Systems Engineering*, 2005.

# Explaining Subsumption and Patching Non-Subsumption with Tableaux Methods

Thorsten Liebig and Stephan Scheele and Julian Lambertz  
Ulm University, D-89069 Ulm, Germany

{thorsten.liebig|stephan.scheele|julian.lambertz}@uni-ulm.de

May 15, 2007

## Explaining Subsumption and Unsatisfiability

Our approach follows the idea of adapting a tableaux algorithm for explaining subsumption and unsatisfiability as suggested in [1, 4]. The explanation follows the proof and uses the structure of a tableaux tree for the production of natural-language explanations. Our method covers the DL *SHIN* including GCIs. We rely on a data structure for annotated terms and nodes to keep book of dependencies between expressions and tableaux nodes and to identify those that contribute to a clash (are relevant). We saturate the tableaux tree to search for alternative explanation candidates. An evaluation function then chooses the best option according to the shortest length. Future work aims at evaluation functions based on empirical knowledge to differentiate explanation options w.r.t. explanation quality. Irrelevant expressions are hidden in the explanation. Furthermore we have developed a technique to optimize an explanation by aggregating a sequence of similar explanation steps into one assertion that a user can drill down if desired. Concerning role hierarchies we have implemented a general methodology to explain the inheritance of properties between roles. Our prototype TABEX is implemented in Lisp (CLOS) and capable of explaining within *SHF* including GCIs. TABEX improves the explanation quality by aggregating explanation steps and utilizing the idea of drill-down and roll-up. Optionally it can utilize RacerPro for optimization purposes.

## Explaining and Patching Non-Subsumption

When applying the techniques from above, a non-subsumption results in at least one model for the corresponding negated subsumption query. Each of this models is caused by an unclosed tableaux path and can be interpreted as counter examples wrt. the subsumption query. The idea of **explaining non-subsumption** is quite similar to the approach for subsumption. All explanations of unclosed paths build up the explanation that ends when there are no more successors generated and constrained by the subsumer **and** subsumee.

The problem with **patching a non-subsumption** is the infinite search space caused by the infinite many ways of closing at least one of the potentially many unclosed tableaux paths. Therefore, our idea is to constrain the search space by concentrating on a set of common errors of unexperienced users studied in [5]. The errors from [5] considered in this approach are: allquantification instead of existential quantification, wrong use of negation in combination with a quantor ( $\neg\forall r.(..)$  vs.  $\forall r.(..)$  and  $\neg\exists r.(..)$  vs.  $\exists r.(..)$ ) and use of a partial instead of a complete definition. Our strategy looks for symptoms of the described errors within an unclosed path. If we find such symptoms in a node, we investigate all involved concept definitions in order to identify the error.

Although concentrating on the mentioned errors reduces the search space significantly, the approach often leads to many suggestions. A first idea to rate suggestions is to leave suggestions leading to a trivial (partial) subsumption unconsidered as mentioned in [3]. A second idea is that the user wants to patch the non-subsumption but does not want the class hierarchy to change elsewhere (compare with [2]). As a first simple metric we compute the differences a suggestion would cause within the direct sub- and superconcepts for all concepts of the KB with help of an external reasoner. A suggestion with little impact on the hierarchy is rated better than one with a high impact. Furthermore, a suggestion patching the non-subsumption is preferred than one which does not.

## Conclusion and Outlook

We argue that tableaux-based methods are valuable for explaining as well as providing clues for repairing an unwanted non-subsumption. In comparison to axiom pinpointing our approach currently is restricted in language expressivity, but more robust with respect to large amounts of axioms and provides more detailed explanations. An extension to ABox explanations seems possible.

## References

- [1] A. Borgida, E. Franconi, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. Explaining  $\mathcal{ALC}$  subsumption. In *Proc. of DL99*, pages 37–40, 1999.
- [2] C. Elsenbroich, O. Kutz, and U. Sattler. A Case for Abductive Reasoning over Ontologies. In *Proc. of OWL: Experiences and Directions WS*, 2006.
- [3] A. Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, University of Maryland College Park, 2006.
- [4] T. Liebig and M. Halfmann. Explaining Subsumption in  $\mathcal{AL}\mathcal{E}\mathcal{H}\mathcal{F}_{R^+}$  TBoxes. In *Proc. of DL05*, pages 144–151, 2005.
- [5] A. L. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In *EKAW*, pages 63–81, 2004.

## Model Checking of Restricted CTL\* formulas using $\mathcal{ALCK}_{R^+}$

Taufiq Rochaeli and Claudia Eckert

Department of Computer Science  
Technische Universitt Darmstadt  
{rochaeli,eckert}@sec.informatik.tu-darmstadt.de

**Introduction** The purpose of model checking technique is to verify whether the implemented computer program (the model) satisfies the specified requirements (the formulas). Now let  $K$  be the Kripke model representing the behavior of the system and  $R$  be the set of formulas. The model checking process verifies whether every formula in  $R$  is satisfied by the model, which has  $I$  as the set of initial states. Formally, it checks whether  $\forall f \in R, \forall s \in I : K, s \models f$  is true or not. We propose an approach to perform model checking of the restricted CTL\* formula based on the top of description logics reasoning services. This approach allows us to build the ontology of the atomic propositions used both in the model and in the formula.

We realize our approach by performing the following steps: (i) representing the model in the assertion box (ABox) of the description-logic based knowledge base, (ii) defining the ontology of atomic propositions in the terminology box (TBox) of the knowledge base, (iii) defining a part of the formal semantics of CTL\* logic on top of the DL semantics, (iv) defining the translation rules for the formula.

**Representation of the Kripke Model and the Ontology of Atomic Propositions** The Kripke model is represented as a set of axioms  $\mathcal{A}_M$ . There are three types of axiom in  $\mathcal{A}_M$ . The first type of the axiom,  $\text{State}(x)$ , defines the state  $s$  of Kripke model  $K$ , where  $x$  is a unique individual name representing state  $s$ . The second type of the axiom,  $\text{next}(x_i, x_j)$ , defines the state transition between  $s_i$  and  $s_j$ . The last kind of axiom,  $V_i(x)$ , is created for each  $p_i \in L(s)$ . The concept  $V_i$  represents the atomic proposition  $p_i$ . For each state  $s$  of the Kripke model  $K$ , an assertion is created in  $\mathcal{A}_M$ , where  $x$  is a unique individual name representing the state  $s$ . The state transitions  $(s_i, s_j)$  are defined by the assertions in form of  $\text{next}(x_i, x_j)$ . The set of atomic propositions that label a state are represented as an assertion about instance membership. That is, for each  $p_i \in L(s)$  an assertion  $V_i(x)$  is created in  $\mathcal{A}_M$ . The concept  $V_i$  represents the atomic proposition  $p_i$ . The ontology of atomic propositions is defined in  $\mathcal{T}_{AP}$ , which has the purpose as the common vocabulary.

**Formal Semantics of CTL\* Logic in Description Logics** The basic idea in *emulating* the formal semantics of CTL\* temporal logic on top of the description logic semantics is to construct the Kripke model in the ABox. Furthermore, we require the epistemic operator  $\mathbf{K}$ , which is presented in [1], to fix the interpretation of the Kripke model represented in the ABox. Therefore, the epistemic operator  $\mathbf{K}$  immediately appears in the concepts representing atomic propositions and the roles representing state transitions. (see equations 1, 7 - 10.)

$$\begin{aligned}
 s \models p_i &\Leftrightarrow \mathcal{KB} \models \mathbf{KV}_i(x) & (1) \\
 s \models \mathbf{E}(g_1) &\Leftrightarrow \mathcal{KB} \models D_1(x) & (2) \\
 \pi \models g_1 &\Leftrightarrow \mathcal{KB} \models D_1(\sigma_1) & (3) \\
 \pi \models \neg g_1 &\Leftrightarrow \mathcal{KB} \models \neg D_1(\sigma_1) & (4) \\
 \pi \models g_1 \vee g_2 &\Leftrightarrow \mathcal{KB} \models (D_1 \sqcup D_2)(\sigma_1) & (5) \\
 \pi \models g_1 \wedge g_2 &\Leftrightarrow \mathcal{KB} \models (D_1 \sqcap D_2)(\sigma_1) & (6) \\
 \pi \models \mathbf{X} g_1 &\Leftrightarrow \mathcal{KB} \models (\exists \mathbf{Knext}. D_1)(\sigma_1) & (7) \\
 \pi \models \mathbf{G} g_1 &\Leftrightarrow \langle \mathcal{T}_M \cup \{D_{aux} \equiv D_1 \sqcap \exists \mathbf{Knext}. D_{aux}\}, \mathcal{A}_M \rangle \models D_{aux}(\sigma_1) & (8) \\
 \pi \models \mathbf{F} g_1 &\Leftrightarrow \mathcal{KB} \models (D_1 \sqcup \exists \mathbf{Knext}^+. D_1)(\sigma_1) & (9) \\
 \pi \models g_1 \mathcal{U} g_2 &\Leftrightarrow \langle \mathcal{T}_M \cup \{D_{aux} \equiv D_2 \sqcup (D_1 \sqcap \exists \mathbf{Knext}. D_{aux})\}, \mathcal{A}_M \rangle & (10) \\
 &\quad \models D_{aux}(\sigma_1)
 \end{aligned}$$

**Fig. 1.** Formal semantics of a subset of CTL\* logic in  $\mathcal{ALCK}$ .

**Translating CTL\* Formulas into TBox Concepts** The restricted CTL\* formulas has the form of  $\mathbf{E}[\phi]$ , where  $\phi$  is the LTL formula enclosed in a  $\mathbf{E}$  path quantifier. The first step is to decompose  $\phi$  into a parse tree. Now we can perform the translation of the LTL formula  $\phi$ , by visiting the nodes in the parse tree starting from the leaf nodes and toward the root node. Suppose that we have a counter  $idx$ , which has the initial value 0. For each visit in the node, we create a new axiom in the TBox  $\mathcal{T}_f$  according to the rules in Fig. 2 and increase the counter  $idx$ . The concepts  $D_i$  and  $D_j$  refer to the concepts created after translating the previous sub formulas  $g_i$  and  $g_j$ , respectively.

LTL formula	concepts added to $\mathcal{T}_f$
$p_i$	$D_{idx} \equiv \mathbf{KV}_i$
$\neg g_i$	$D_{idx} \equiv \neg D_i$
$g_i \vee g_j$	$D_{idx} \equiv D_i \sqcup D_j$
$g_i \wedge g_j$	$D_{idx} \equiv D_i \sqcap D_j$
$\mathbf{X} g_i$	$D_{idx} \equiv \exists \mathbf{Knext}. D_i$
$\mathbf{G} g_i$	$D_{idx} \equiv D_i \sqcap \exists \mathbf{Knext}. D_{idx}$
$\mathbf{F} g_i$	$D_{idx} \equiv D_i \sqcup \exists \mathbf{Knext}^+. D_i$
$g_i \mathcal{U} g_j$	$D_{idx} \equiv D_j \sqcup (D_i \sqcap \exists \mathbf{Knext}. D_{idx})$

**Fig. 2.** Translation table of a subset of CTL\* sub formulas into  $\mathcal{ALCK}$  concepts.

**Performing the Model Checking** Let  $K$  and  $s$  be the Kripke model representing the dynamic behavior of a system and the initial state of the model. The formula  $f$  represents the desired property. The model checking  $K, s \models f$  can be performed as an instance checking  $\mathcal{KB}_M \models C(x)$ , whereas  $\mathcal{KB}_M : \langle \mathcal{T}_{AP} \cup \mathcal{T}_f, \mathcal{A}_M \rangle$ .

## References

1. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. *ACM Trans. Comput. Logic* **3** (2002) 177–225

## Description Logics in the Calculus of Structures

Jean-David Roubach<sup>1,2</sup>, Pascal Yim<sup>2</sup>, and Joaquín Rodríguez<sup>1</sup>

<sup>1</sup> INRETS – ESTAS  
 Villeneuve d’Ascq, France  
 {roubach,rodriguez}@inrets.fr  
<sup>2</sup> École Centrale de Lille – LAGIS  
 Villeneuve d’Ascq, France  
 pascal.yim@ec-lille.fr

**Abstract.** We introduce a new proof system for the description logic  $\mathcal{ALC}$  in the framework of the calculus of structures, a structural proof theory that employs *deep inference*. This new formal presentation introduces positive proofs for description logics. Moreover, this result makes possible the study of sub-structural refinements of description logics, for which a semantics can now be defined.

### 1 A calculus of structures for description logics

Proof systems in the calculus of structures are defined by a set of deep inference rules operating on *structures*[1]. The rules are said to be *deep* because unlike the sequent calculus for which rules must be applied at the root of sequents, the rules of the calculus of structures can be applied at any depth inside a structure.

As noted by Schild[2],  $\mathcal{ALC}$  is a syntactic variant of propositional multimodal logic  $K_{(m)}$ . Therefore, since this logic involves no interaction between its modalities, its proof system in the calculus of structures can be straightforwardly extended from a proof system of unimodal  $K$  in the calculus of structures, such as the cut-free proof system  $\text{SKSg}_K$  described in [3].

Let  $\mathcal{A}$  be a countable set equipped with a bijective function  $\bar{\cdot} : \mathcal{A} \rightarrow \mathcal{A}$ , such that  $\bar{\bar{A}} = A$ , and  $\bar{A} \neq A$  for every  $A \in \mathcal{A}$ . The elements of  $\mathcal{A}$  are called primitive concepts, and two of them are denoted by  $\top$  and  $\perp$  such that  $\bar{\top} := \perp$  and  $\bar{\perp} := \top$ .

The set  $\mathcal{R}$  of *prestructures* of  $\mathcal{ALC}$  concepts is defined by the following grammar, where  $A$  is a primitive concept and  $R$  is a role name:

$$C, D ::= \top \mid \perp \mid A \mid \bar{C} \mid (C, D) \mid [C, D] \mid \exists R.C \mid \forall R.C .$$

On the set  $\mathcal{R}$ , the relation  $=$  is defined to be the smallest congruence relation induced by the following equations.

Associativity	Commutativity
$(C, (D, E)) = ((C, D), E)$	$[C, D] = [D, C]$
$[C, [D, E]] = [[C, D], E]$	$(C, D) = (D, C)$

Units	Negation	Roles
$(C, \top) = C$	$\overline{(C, D)} = [\bar{C}, \bar{D}]$	$\overline{\forall R.C} = \exists R.\bar{C}$
$[C, \perp] = C$	$\overline{[C, D]} = (\bar{C}, \bar{D})$	$\overline{\exists R.C} = \forall R.\bar{C}$
$[\top, \top] = \top$	$\bar{\bar{C}} = C$	$\forall R.\top = \top$
$(\perp, \perp) = \perp$		$\exists R.\perp = \perp$

A *structure* is an element of  $\mathcal{R}/\equiv$ , i.e. an equivalence class of prestructures. For a given structure  $C$ , the structure  $\bar{C}$  is called its *negation*. *Contexts* are defined by the following syntax, where  $C$  stands for any structure:  $S ::= \{\circ\} \mid [C, S] \mid (C, S)$ .

An *inference rule* is a scheme of the kind  $\rho \frac{S\{C\}}{S\{D\}}$ . This rule specifies a step of rewriting inside a generic context  $S\{\circ\}$ . A *proof* in a given system, is a finite chain of instances of inference rules in the system, whose uppermost structure is the unit  $\top$ .

## 2 System $\text{SKSg}_{\mathcal{ALC}}$

The following set of rules defines the sound and complete cut-free proof system  $\text{SKSg}_{\mathcal{ALC}}$  for  $\mathcal{ALC}$  in the calculus of structures :

$$\begin{array}{ccc}
 \text{i}\downarrow \frac{S\{\top\}}{S[C, \bar{C}]} & & \text{i}\uparrow \frac{S(C, \bar{C})}{S\{\perp\}} \\
 \text{w}\downarrow \frac{S\{\perp\}}{S\{C\}} & \text{s} \frac{S([C, D], E)}{S[C, (D, E)]} & \text{w}\uparrow \frac{S\{C\}}{S\{\top\}} \\
 \text{c}\downarrow \frac{S[C, C]}{S\{C\}} & & \text{c}\uparrow \frac{S\{C\}}{S(C, C)} \\
 \text{k}\downarrow \frac{S\{\forall R.[\bar{C}, D]\}}{S[\forall R.\bar{C}, \forall R.D]} & & \text{k}\uparrow \frac{S(\exists R.\bar{C}, \exists R.D)}{S\{\exists R.(\bar{C}, D)\}}
 \end{array}$$

## References

1. Brünnler, K.: Deep Inference and Symmetry in Classical Proofs. Logos Verlag, Berlin (2004)
2. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proceedings of the International Joint Conference on Artificial Intelligence. (1991)
3. Stewart, C., Stouppa, P.: A systematic proof theory for several modal logics. In: Advances in Modal Logic. Volume 5., King's College (2005)

## Approximating OWL-DL Ontologies

Edward Thomas and Jeff Z. Pan

Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, UK

**Abstract.** In this poster, we propose to recast the idea of knowledge compilation into approximating OWL DL ontologies with DL-Lite ontologies, against which query answering has only polynomial data complexity. We identify a useful category of queries for which our approach also guarantees completeness. Furthermore, we paper report on the implementation of our approach in the ONTOSEARCH2 system.

### 1 Introduction

Ontologies play a key role in the Semantic Web [1], for which W3C has standardised the Web Ontology Language OWL [2]; in this poster, we are interested in the OWL DL sub-language (‘DL’ for Description Logic) due to the existence of reasoning support. A growing library of ontologies is available online, covering a wide range of human knowledge. For this large body of knowledge to be usable by Semantic Web applications, a framework that allows efficient query answering over ontologies is required.

Query answering over OWL DL is a hard problem. It has been shown that the complexity of ontology entailment in  $\mathcal{SHOIN}(\mathbf{D}^+)$ , i.e., OWL DL, is NEXPTIME. This indicates query answering over OWL DL ontologies is at least NEXPTIME. Approximation has been identified as a potential way to reduce the complexity of reasoning over OWL DL ontologies. Existing approaches are mainly based on syntactic approximation of ontological axioms and queries. All these approaches could introduce unsound answers. To the best of our knowledge, we have not seen any published framework on sound (and possibly incomplete) approximations for ontology query answering, not to mention efficient ones.

We propose to recast the idea of knowledge compilation [3] into semantic approximation of OWL DL ontologies. The idea of knowledge compilation is simple: users can write statements in an expressive representation language and these such statements can be compiled into a restricted language that allows efficient inference. In this way, users do not have to use a less expressive language which might be too limited for practical applications. In [3], Selman and Kautz showed how propositional logical theories can be compiled into Horn theories that approximate the original information; they also applied this idea on subsumption reasoning for the Description Logic  $\mathcal{FL}$ . In this poster, we investigate applying knowledge compilation on query answering over OWL DL ontologies. Namely, we propose approximating OWL DL ontologies [4] (or simply source ontologies) with corresponding DL-Lite [5, 6] ontologies (or simply target ontologies), against which query answering has only polynomial data complexity, and provide algorithms to compute target DL-Lite ontologies.

## 2 Implementation

We have implemented the approximation algorithm in the ontology search and query tool, ONTOSEARCH2 [7]. Unlike existing syntactic approximation approach, our approach always guarantees sound answers for conjunctive and disjunctive queries over ontologies. For queries without non-distinguished variables (which modern OWL DL reasoners like Racer and KAON2 disallow anyway), our approach guarantees both soundness and completeness. Our preliminary evaluations in the ONTOSEARCH2 system shows that our approach is very scalable: ONTOSEARCH2 outperforms existing OWL DL reasoners significantly. In general, our results indicate that the user can still have efficient querying answering support when they use expressive ontology languages, such as OWL DL and OWL 1.1.

For our future work, one of the main challenges is to extend the query language to support datatype properties and some datatype predicates/built-ins. Secondly, we will investigate benchmarks for querying answering over more complex ontologies. Furthermore, it is interesting to see how to make it more complete for queries with non-distinguished variables. Last but not least, we shall investigate further optimisations on calculating target ontologies and how to apply our semantic approximations to other light-weight ontology languages.

## References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* **284**(5) (2001) 34–43
2. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL Web Ontology Language Semantics and Abstract Syntax. Technical report, W3C (2004) W3C Recommendation.
3. Selman, B., Kautz, H.: Knowledge Compilation and Theory Approximation. *Journal of the ACM (JACM)* **43** (1996) 193–224
4. Pan, J.Z., Thomas, E.: Approximating OWL-DL Ontologies. In: *Proc. of AAAI 2007 (In Press)*. (2007)
5. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R., Vetere, G.: DL-Lite: Practical Reasoning for Rich DLs. In: *Proc. of the DL2004 Workshop*. (2004)
6. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., , Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: *Proc. of AAAI 2005*. (2005)
7. Pan, J.Z., Thomas, E., Sleeman, D.: ONTOSEARCH2: Searching and Querying Web Ontologies. In: *Proc. of WWW/Internet 2006*. (2006) 211–218

## Practical Conforming Datatype Groups

Dave Turner and Jeremy J. Carroll {DavidT,Jeremy.Carroll}@hp.com

HP Laboratories, Bristol, UK

**Abstract.** The proposed OWL 1.1 language is based on the description logics *SRIQ* and *SHOQ*( $D_n$ ), whose features include the use of  $n$ -ary datatype predicates. The means of specifying such predicates, which must be usefully expressive without breaking decidability properties, is omitted; rectifying this omission is nontrivial.

### 1 Discussion

A finite predicate conjunction over a datatype group  $\mathcal{G} = (\Delta_{\mathbf{D}}, \mathbf{D}_{\mathcal{G}}, \Phi_{\mathcal{G}}^1, \Phi_{\mathcal{G}})$  is a statement of the form  $\bigwedge_{j=1}^k p_j(v_1^{(j)}, \dots, v_{n_j}^{(j)})$  where  $p_j$  is an  $n_j$ -ary predicate in  $\mathbf{D}_{\mathcal{G}} \cup \Phi_{\mathcal{G}}^1 \cup \Phi_{\mathcal{G}}$ . The datatype group  $\mathcal{G}$  is said to be *conforming* if

1.  $\mathbf{D}_{\mathcal{G}}$ ,  $\Phi_{\mathcal{G}}^1$  and  $\Phi_{\mathcal{G}}$  are closed under negation,
2. a binary inequality predicate  $\neq_d \in \Phi_{\mathcal{G}}$  is defined for each datatype  $d \in \mathbf{D}_{\mathcal{G}}$ , and importantly
3. the satisfiability of finite predicate conjunctions over  $\mathcal{G}$  is decidable.

Pan and Horrocks [1] present a datatype group containing the predicate `kmtrsPerMile = D(k, m, "k=1.6*m")` for converting between distances measured in kilometers and miles. Elsewhere they suggest a predicate of objects that are small enough to have no additional postage cost: `smallObj = D(1, w, "1+w<10")`.

**Theory of Arithmetic** A naïve generalisation of this syntax is too expressive and quickly leads to non-conforming datatype groups. For example, consider a datatype group containing

`integerAddition = D(i, j, k, "i=j+k")`  
and `integerMultiplication = D(i, j, k, "i=j*k")`

viewed as ternary predicates over `integerD`. The satisfiability problem over this datatype group amounts to the solution of Diophantine equations, which is Hilbert’s tenth problem and is known to be undecidable.

Note that a datatype group containing either one of these predicates could be conforming; it is the presence of both which forces undecidability. The instability of conformingness under merging of intersecting datatype groups is not an artificial problem: in the context of OWL there are a small, finite number of base datatypes and one would expect to want to merge intersecting datatype groups reasonably often. The open-world style of the Semantic Web encourages such merges, so it would be inappropriate to forbid them completely. Furthermore, without both multiplication and addition, end-users would not even be able to convert between °F and °C: `fahrenheitToCelsius = D(f, c, "f=1.8*c+32.0")`.

**Implementation of Arithmetic** An example from [1] declares that the Yangtze river is 3937.5 miles long and uses the `kmtrsPerMile` predicate to deduce that it is also 6300.0km long, using the XML Schema datatype `float`. If instead we start from a length of 3937.501 miles, then

$$\begin{aligned} \langle 6300.0015, 3937.501 \rangle &\in \llbracket \text{kmtrsPerMile} \rrbracket \\ \text{and } \langle 6300.0015, 3937.5007 \rangle &\in \llbracket \text{kmtrsPerMile} \rrbracket, \end{aligned}$$

so that the Yangtze river may be deduced also to be 3937.5007 miles long.

We implemented a system to do conversions between `floats` representing lengths in km, m, cm, mm,  $\mu\text{m}$ , inches, feet, yards, fathoms, poles, chains, furlongs, statute and nautical miles and leagues, and deduced the length of the Yangtze to be both 6335.3584km and 6361.8555km, and nearly 800,000 other values, starting from a declaration that its length in miles is 3937.5. These rounding errors were highly dependent on the structure of the definitions of the units, as multiplication in `float` is not associative.

Additionally, suppose the Volga river were declared to be 3668.8003km long, then it would have no value for its `lengthInMile` property at all, since

$$\begin{aligned} \langle 3668.8000, 2293.0000 \rangle &\in \llbracket \text{kmtrsPerMile} \rrbracket \\ \langle 3668.8005, 2293.0002 \rangle &\in \llbracket \text{kmtrsPerMile} \rrbracket \\ \text{and } \nexists x \in \text{float}. &2293.0 < x < 2293.0002 \end{aligned}$$

Notice that this cannot be remedied by using the arbitrary-precision `decimal` instead of the fixed-precision `float`: for example the temperature 75.0°F has no corresponding `decimal` representation in °C.

In practice, many applications do not require the declarative style of arithmetic that datatypes like `kmtrsPerMile` would allow. Instead, a procedural approach is adequate. For example, a user may be happy that the Volga can be deduced to be 2293.0km long, and may be equally happy with 2293.0002km, as long as one, and only one, of the options is chosen.

## 2 Conclusions and Future Work

Datatype groups are motivated by the requirements of DL users to be able to express complex constraints simultaneously on multiple data values. However, there has been little discussion regarding datatype groups that satisfy these user requirements whilst also being conforming and computationally feasible. We have shown that this question is far from trivial; see [2] for more details.

## References

1. Pan, J.Z., Horrocks, I.: Web Ontology Reasoning with Datatype Groups. In Fensel, D., Sycara, K., Mylopoulos, J., eds.: Proc. of the 2nd International Semantic Web Conference (ISWC2003). (2003)
2. Turner, D., Carroll, J.J.: Practical conforming datatype groups. HP Technical Report HPL-2007-37 (2007)

## Author Index

Ait-Kaci, Hassan	147	Fadhil, Amineh	267
Alferes, Jose Julio	347	Fanizzi, Nicola	275
Antoniotti, Marco	529	Farouk, Toumani	363
Anureev, Igor	459	Ferrari, Mauro	219
Araújo, Rudi	155	Fiorentini, Camillo	219
Artale, Alessandro	163	Fiorino, Guido	219
Baader, Franz	17, 171	Fuchs, Florian	535
Barinskis, Martins	523	Garanina, Natalia	459
Barzdins, Guntis	523	Ghidini, Chiara	283
Ben-David, Shoham	179	Glimm, Birte	65
Benatallah, Boualem	187	Goczyla, Krzysztof	291
Berger, Michael	535	Gore, Rajeev	299
Bernardi, Raffaella	195	Haarslev, Volker	53, 267, 307, 531
Bienvenu, Meghyn	203	Hacid, Mohand-Said	187
Bong, Yusri	483	Hitzler, Pascal	347, 395, 403
Borgida, Alex	1, 211	Hladik, Jan	17
Bozzato, Loris	219	Homola, Martin	315
Braam, Hansje	525	Horrocks, Ian	41, 65, 419
Calvanese, Diego	29, 163, 195, 227, 235	Hu, Zhengguo	499
Cantú Ortíz, Francisco	527	Iacovella, Maurizio	533
Carroll, Jeremy	545	Ianni, Giovambattista	259
Ceballos, Hector	527	Ji, Qiu	435
Chang, Liang	243	Kaya, Atila	323
Chen, Xiaoping	491	Kazakov, Yevgeny	41
Colombo, Gianluca	529	Keet, C. Maria	331
Cuenca Grau, Bernardo	41	Kharlamov, Evgeny	235
d’Amato, Claudia	275	Khizder, Vitaliy	339
De Giacomo, Giuseppe	29, 227	Knorr, Matthias	347
De Paoli, Flavio	529	Kontchakov, Roman	76, 163
Di Noia, Tommaso	443	Krötzsch, Markus	355
Di Sciascio, Eugenio	443	Krennwallner, Thomas	259
Ding, Yu	53, 531	Krisnadhi, Adila	88
Dioturni, Manuel	533	Lambertz, Julian	537
Donini, Francesco	443	Leger, Alain	187, 363
Du, Jianfeng	251	Lembo, Domenico	227, 371
Eckert, Claudia	539	Lenzerini, Maurizio	29, 227
Eiter, Thomas	259	Levesque, Hector	12
Espinosa, Sofia	323	Liebig, Thorsten	537
Esposito, Floriana	275	Lin, Fen	243

Lin, Zuoquan	395, 403	Stanchev, Lubomir	123
Lisi, Francesca Alessandra	379	Stoilos, Giorgos	427
Lubyte, Lina	387	Straccia, Umberto	467
Lutz, Carsten	88, 100	Stuckenschmidt, Heiner	475
Möller, Ralf	323	Suntisrivaraporn, Boontawee	171
Ma, Yue	395, 403	Tessarì, Sergio	283, 387
Mauri, Giancarlo	529	Thomas, Edward	427, 543
Melzer, Sylvia	323	Thorne, Camilo	195
Merico, Daniele	529	Tishkovsky, Dmitry	135
Milicic, Maja	112	Toman, David	123, 339
Miller, Renée J.	13	Toumani, Farouk	187
Mosca, Alessandro	411	Trefler, Richard	179
Motik, Boris	419	Turhan, Anni-Yasmin	483
Nguyen, Linh Anh	299	Turner, Dave	545
Nutt, Werner	235	Visco, Giulio	467
Pai, Hsueh-Ieng	307	Waloszek, Aleksander	291
Palmonari, Matteo	411	Waloszek, Wojciech	291
Pan, Jeff Z.	427, 435, 543	Weddell, Grant	123, 179, 339
Penaloza, Rafael	17, 171	Wessel, Michael	323
Pinto, HSofia	155	Wolter, Frank	76
Poggi, Antonella	227	Wu, Jian	499
Pound, Jeffrey	123	Wu, Jiewen	53
Qi, Guilin	403, 435	Yang, Fangkai	491
Qiu, Lirong	243	Yim, Pascal	541
Ragone, Azzurra	443	Zakharyashev, Michael	76, 163
Rey, Christophe	187, 363	Zhang, Changli	499
Rochaeli, Taufiq	539	Zimmermann, Antoine	507
Rodriguez, Joaquin	541	Zolin, Evgeny	515
Rosati, Riccardo	29, 227, 451		
Roubach, Jean-David	541		
Rudolph, Sebastian	355		
Ruzzi, Marco	371		
Ryzhikov, Vladislav	163		
Sattler, Ulrike	41, 65		
Scheele, Stephan	537		
Schindlauer, Roman	259		
Schmidt, Renate	135		
Serafini, Luciano	283		
Shearer, Rob	419		
Shen, Yi-Dong	251		
Shi, Zhongzhi	243		
Shilov, Nikolay	459		
Shiri, Nematollaah	307		
Stamou, Giorgos	427		