The 18th European Conference on Artificial Intelligence

Proceedings

Workshop on Contexts and Ontologies

Monday July 21, 2008

Patras, Greece

P. Bouquet, J. Euzenat, C. Ghidini, D. L. McGuinness, V. de Paiva, G. Qi, L. Serafini, P. Shvaiko, H. Wache, A. Leger (Eds.)

Foreword

Contexts and ontologies play a crucial role in knowledge representation and reasoning. Computer systems which act intelligently need the ability to represent, utilize and reason about contexts and ontologies. Many projects devoted to the definition and usage of contexts as well as ontologies in intelligent KR systems. With the advent of the web and the ubiquitous connectivity, contexts and ontologies have become a relevant notion also in other, more recent, disciplines. Many application areas such as information integration, distributed knowledge management, semantic web, multi-agent systems, distributed reasoning, data grid and grid computing, pervasive computing and ambient intelligence as well as peer-to-peer information management systems, have acknowledged the need for methods to represent and reason about knowledge which is scattered in a large set of contexts and ontologies.

During the last decade, there has been a series of successful workshops and conferences on the development and application of contexts and ontologies. Three successful workshops have focused on combining the themes of ontologies and contexts, and have discussed them as complementary disciplines. The first two are “Contexts and Ontologies: Theory, Practice and Applications (C&O)” and “Context Representation and Reasoning (CRR)”. These two previous series of workshops were merged into the “Contexts and Ontologies: Representation and Reasoning (C&O:RR)” workshop. The C&O:RR workshop maintains the focus on the combination of contexts and ontologies while emphasizing representation and reasoning aspects of the research, the strong point of the CRR workshop. This workshop is the continuation of the C&O:RR 2007 under a broader title.

The organizers thank the members of our program committee for their careful work and our invited speakers, Dr. Andreas Herzig and Dr. Marco Schorlemmer for their inspiring contributions.

Organizing Committee

Program Committee

Acknowledgements

We appreciate the support from the EU under the IST project NeOn (IST-2006-027595)

Table of Contents

A data-intensive lightweight semantic wrapper approach to aid information integration.

Dave Braines[1], **Yannis Kalfoglou**, **Paul Smart**, **Nigel Shadbolt**[2] and **Jie Bao**[3]

Abstract. We argue for the flexible use of lightweight ontologies to aid information integration. Our proposed approach is grounded on the availability and exploitation of existing data sources in a networked environment such as the world wide web (instance data as it is commonly known in the description logic and ontology community). We have devised a mechanism using Semantic Web technologies that wraps each existing data source with semantic information, and we refer to this technique as SWEDER (Semantic Wrapping of Existing Data Sources with Embedded Rules). This technique provides representational homogeneity and a firm basis for information integration amongst these semantically enabled data sources. This technique also directly supports information integration though the use of context ontologies to align two or more semantically wrapped data sources and capture the rules that define these integrations. We have tested this proposed approach using a simple implementation in the domain of organisational and communication data and we speculate on the future directions for this lightweight approach to semantic enablement and contextual alignment of existing network-available data sources.

1 Introduction

A plethora of data is available in structured forms today, either in existing Semantic Web encodings such as OWL/RDF or, more likely, in more traditional formats such as XML, CSV, HTML or relational databases. This data is available to the consumer today, usually via URIs resolving to network or local addresses, but may also be sourced from directly referenced files and other non-URI referenced resources. The data itself can take any form, but we propose that it can be relatively easily semantically wrapped through a lightweight application of OWL/RDF to represent the data in the form of entities (classes), attributes (data properties) and relationships (object properties). The purpose of this lightweight semantic wrapping is to provide representational homogeneity for these existing data sources, thereby providing a firm semantic basis for any downstream consumption in potentially unknown contexts.

The details regarding how this semantic wrapping is achieved are peripheral to the scope of this paper which is to focus on how to leverage and capitalize on the end result: *semantically wrapped data sources*. Our focus in this paper is mainly on the subsequent creation of context ontologies to specifically capture the alignments between these semantically wrapped data sources, and we assume this representational homogeneity as a pre-requisite for our data sources. In practical terms this semantic wrapping is usually achieved through manual design and construction of a simple ontology, or reuse of an existing published ontology. Then the corresponding instance data is generated through the application of simple transformations of existing data-sources to the corresponding RDF/OWL representations. A suitable existing pattern for this work is that of RDFa[4], microformats[5] (and GRDDL[6]) which are popular techniques for semantically enriching existing web data sources today, albeit within existing web markup languages rather than as stand-alone ontology instance data as we are proposing.

We elaborate on making use of this semantically wrapped data in the next section where we introduce the notion of a lightweight form for representing the alignments or relationships between these existing data sources: *context ontologies* (section 2). These are used in a principled manner which we describe in section 2.1, and we apply in an example case in section 3. We go on to discuss proposed extensions to our work in section 3.1, related work in section 4, and conclude this paper in section 5.

2 Context ontologies

We adopt a dynamic notion of context which is not common to the formal notions presented in the AI literature (see, for example, the seminal work in [4] on formalizing contexts as first class objects). Our aim is to use context dynamically in order to capture and define each purpose for which data is used, specifically enabling it to be used by a consumer application. We do not take into account the initial context of existing data, as all data exists for a specific reason and with a specific format, but we treat this originating context as a precursor to our interest: how to enable seamless processing of many data sources by a variety of consumer applications in different contexts. In the simplest scenario, a consumer application will merely consume a single data source for further processing, and in this extremely simple case one could argue that the consumer application has added no additional context to the original data. This is an unlikely scenario for a consumer application of any real value, and it is more likely that a consumer application will consume multiple data sources and fuse them or otherwise make use of both data sources and the relationships between these sources. In this scenario we argue that the consumer application does add a context to these data sources, not least because a specific combination of multiple data sources has been selected to fulfil a particular need. This context is

[1] Emerging Technology Services, IBM United Kingdom Ltd., Hursley Park, Winchester, SO21 2JN, UK, email: dave_braines@uk.ibm.com

[2] School of Electronics and Computer Science, University of Southampton, UK, email: {y.kalfoglou,ps02v,nrs}@ecs.soton.ac.uk

[3] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY12180, USA, email: baojie@cs.rpi.edu

[4] http://www.w3.org/2006/07/SWD/RDFa/

[5] http://microformats.org/

[6] http://www.w3.org/2004/01/rdxh/spec

captured through the creation of a context ontology which specifically integrates the concepts from any semantically wrapped data sources that it references. This context ontology is then able to be easily used by the consumer application, thereby reading the various semantically wrapped data sources, processing the instance data and executing embedded rules to derive further information or alignments. The result of this could be published as instance data conforming to the context ontology and then made available for further consumption by unknown downstream consumer applications. For example, a context ontology may be created which aligns concepts from two semantically wrapped data sources containing geographic feature data and person location data. A consumer application can then use this new context ontology, execute the embedded rules to fuse these two data sources in specific ways, and infer the interesting intersection of these as defined within the context ontology. In this example case, the list of people attending specific geographic features of interest. This inferred additional data can then be used by the consumer application and can additionally be published as new instance data conforming to the context ontology.

Of course the fusion of two data sources by an application to achieve the results above could easily be achieved with existing technology, and does not require semantic representation. As a matter of fact, one could view the current trend of mashups[7] as a successful (usually non-semantic) form of such integrations. The specific benefits of semantically enabling the data sources and capturing the alignment representation and rules in a context ontology as defined in our approach lie in the representational homogeneity achieved through this approach, the self-defining and portable nature of the context ontologies and their embedded rules, and the ease with which consumer applications can use these context ontologies along with the appropriate semantically wrapped data sources. Further important capabilities are also enabled through the use of this approach, most notably the support for referencing common definitions via URIs to enable more rapid understanding and information integration.

A key aspect of our proposal is facilitating the creation, representation and consumption of information integration rules within these context ontologies, and this is something that existing OWL based solutions do not readily support. There are emerging standards in this area, notably SWRL[8] and potentially RIF[9], but for various reasons we have chosen a lightweight, pragmatic approach and use SPARQL[10] construct clauses to define these rules and store them as instances within our context ontology. This allows any SPARQL enabled endpoint to execute the rules and instantiate the inferred results directly from the construct clause held in the embedded rule without the need for any specific additional rule execution engine.

We store each actual SPARQL construct clause rule as instance data directly in the context ontology to which it applies, thus enabling these rules to be passed to the consumer application as part of the context ontology itself. In further iterations of this work we plan to introduce richer representation formats (such as SWRL, RIF) as these representations could be used to generate SPARQL construct clauses which would be executed as per our current solution. The use of these richer representation languages would expose the semantics of these information integration rules to consuming applications rather than the current solution which simply records the text of the SPARQL construct clause without providing any semantic representation of the rule which the SPARQL implements.

[7] http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
[8] http://www.w3.org/Submission/SWRL/
[9] http://www.w3.org/2005/rules/
[10] http://www.w3.org/TR/rdf-sparql-query/

2.1 A conceptual model

Conceptually and practically we use a two-tier model to represent each of our ontologies, both for the simple semantic wrappings of existing data sources, and for the subsequent context ontologies which are created to capture the alignments of ontologies. This two-tier approach allows for a clear separation between the representation of the model and the capture of any associated rules. An example of this two-tier doughnut shaped model is shown in figure 1. The model ontology is at the centre, and it is comprised of traditional ontology modeling concepts: entities, attributes and relationships, as described earlier. This ontology is imported into the outer ontology, which simply adds support for rules to be defined against the model. In our current implementation this takes the form of an import to a generic information integration rules ontology which enables the SPARQL construct based rules to be represented as instances of simple entities. The separation of these two aspects of our ontologies enables the rules to be captured separately to the model, thus offering us a flexible way in which to improve the rule representation solution in the future without affecting the model ontology, and it also enables us to easily use existing ontologies and wrap them with our rules. We label this technique SWEDER (Semantic Wrapping of Existing Data Sources with Embedded Rules).

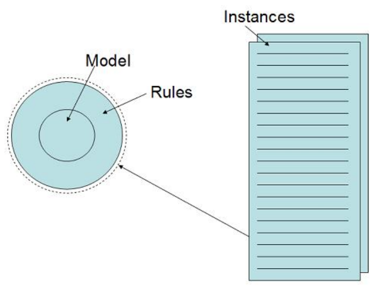


Figure 1. A doughnut shaped two-tier ontology model and associated instances.

The final aspect of our solution is the capture of context information for multiple ontologies, which we achieve via the creation of additional lightweight ontologies. These are the context ontologies we referred to previously and are built according to the same two-tier approach.

The context ontology defines any additional entities, attributes or relations which are relevant to the current context (in the inner model ontology), and also defines any instances of rules which are able to populate these additional items (in the outer rules ontology). The context ontology also imports any required source ontologies (which may of course be context ontologies themselves) and the new context ontology therefore captures the representation of this specific new context and embodies it in a semantic format consistent with the source ontologies. We visualize this approach in figure 2, and it should be noted that the imported ontologies can either be normal ontologies that exist already, or can be the semantically wrapped data source ontologies that we describe in this paper.

The final step is for a suitable consumer application to consume the context ontology and any associated instance data for the source ontologies. Since all of the integration rules are contained within the context ontology this consumer application simply invokes a standard process to extract all these rules (which are stored as SPARQL construct clause text), then executes them against the instance data

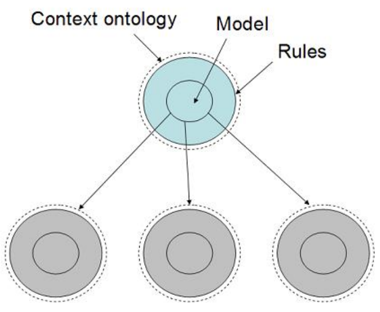


Figure 2. A typical context ontology.

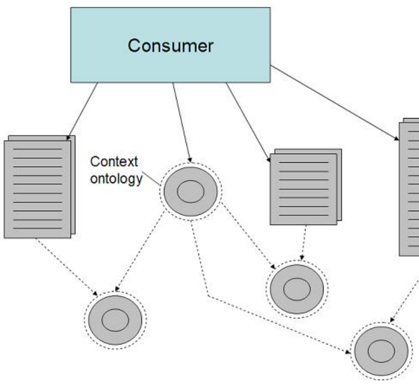


Figure 3. A consumer application interacts with context ontologies.

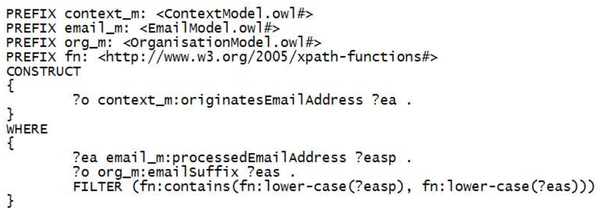
using an appropriate SPARQL endpoint. The results of these rule executions are that new instance data are created within the context ontology, and this can then be saved, published or further processed by the consumer application. The consumer application actually executes all the rules multiple times, until the set of all rule executions results in no further data being inferred. We depict diagrammatically the interaction with a consumer application in figure 3.

3 An example consumer application

In order to test our proposed technique we have applied SWEDER in the context of organisational and communication data. We used a variety of source data from existing applications, converting this to OWL/RDF based on simple ontologies defined in Protege[11]. The consumer application is built using the Jena framework from Hewlett Packard Labs[12] and the ARQ SPARQL processor for Jena. The source ontologies constructed in our example were:

- **Email** - this is a simple semantic representation of email data extracted from an email application. Includes `Email`, `EmailAddress` and `Tag` entities with multiple attributes and relations between them. (See figure 6);
- **Person** - this is a semantic representation of instant messaging system contacts and their groups. Includes `Person`, and `Relationship`. We could also extract this data from many sources such as FOAF[13], social network sites, etc.;
- **Organisation** - this is a semantic representation of basic organisation information such as *name*, *email suffix*, *homepage*, etc. The instance data for this was created specifically for the purposes of our experiment but could easily come from a CRM system or similar;
- **Project** - this a semantic representation of basic project information such as `project name`. This instance data for this was specifically created for the purposes of our experiment but could come from a DOAP[14] dataset or an application used to record project information.

It is noteworthy to point out that each of the above ontologies is completely stand-alone and requires no knowledge or understanding of data in the other. Conceptually these could have each been defined and created by different authors at different times, although for the purposes of our exercise we created each of these ourselves, but carefully ensured that each ontology was entirely separate from the others in terms of the constituent data and conceptual representation. Each of these ontologies makes use of appropriate RDF/OWL representations such as dependencies between properties, inverse relationships, etc. A single context ontology was then created to align the appropriate aspects of these source ontologies. Our approach supports multiple context ontologies, each for a specific alignment, but in our example case we used only one as we simply wished to demonstrate the value of these context ontologies and the recording of rules within them. The alignments we produced were captured using SPARQL construct rules that implement the following information integration tasks:



```
PREFIX context_m: <ContextModel.owl#>
PREFIX email_m: <EmailModel.owl#>
PREFIX org_m: <OrganisationModel.owl#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
CONSTRUCT
{
        ?o context_m:originatesEmailAddress ?ea .
}
WHERE
{
        ?ea email_m:processedEmailAddress ?easp .
        ?o org_m:emailSuffix ?eas .
        FILTER (fn:contains(fn:lower-case(?easp), fn:lower-case(?eas)))
}
```

Figure 4. A SPARQL example rule relating a person's email address to organisation.

- *EmailAddress to Person*: within the `Email` ontology (an excerpt of which is shown in figure 6), `EmailAddress` instances are created for each unique email address that is involved in sending or receiving an email. Each `EmailAddress` entity has a `rawEmailAddress` attribute containing the email address string which is that email address. Within the `Person` ontology a `Person` can have one or more values for the `emailAddress` attribute which contain their email address string(s). The rule we execute simply matches any `Person` with an `emailAddress` value which is identical to the `rawEmailAddress` of any `EmailAddress` entity. The construct clause populates a new relationship (object property) named `hasEmailAddress` on this `Person` and `hasPerson` on this `EmailAddress` to record the new inferred relationship between these two entities. We give an example of this SPARQL construct based rule in figure 5. From this we can subsequently infer a relationship between `Person` entities and `Email` entities and can now easily identify all emails that a `Person` has sent or received.

[11] Available from: http://protege.stanford.edu/
[12] Available from: http://jena.sourceforge.net/
[13] http://www.foaf-project.org/
[14] http://trac.usefulinc.com/doap

- *EmailAddress to Organisation*: within the Organisation ontology, Organisation instances are created, and each is populated with an emailSuffix string. Each EmailAddress entity has a rawEmailAddress attribute as described previously. The rule we execute matches any Organisation with an emailSuffix which is the same as the end of any EmailAddress entities rawEmailAddress attribute. The SPARQL construct clause populates a new relationship (object property) named originatesEmailAddress on this Organisation to record this inferred information.
- *Person to Organisation*: this builds on the previous rule, and identifies any EmailAddress which has a Person (hasPerson) and which has an Organisation (hasOrganisation). The rule then populates a new relationship (object property) named employsPerson on Organisation with a link to that Person. This rule relies on the previous two rules correctly instantiating EmailAddress to Person and EmailAddress to Organisation relationships. An example of such a rule is shown in figure 4. When we enable multiple rule executions this rule may infer additional information when it is run after the prerequisite rules.
- *Person to Project*: the Email ontology has multiple Email instances, many of which have already been tagged according to the name of the project that they relate to. This enables another rule to identify any Email with a hasTag text which is the same as any Project name or alternativeName attribute. This rule populates a new relationship (object property) named relatedEmail on Project and relatedProject on Email.
- *Project to Email* and *Project to Organisation*: these final two rules build on the same principles as before and identify each Project that has a relationship to a Person (or Organisation) and the Email which that Person (or Organisation) is involved with via the related EmailAddress entities. The results of these two rules are instantiated in the new hasEmail relationships (object properties) on Project and Organisation, and in the hasProject and hasOrganisation relationships on Email.

```
PREFIX context_m: <ContextModel.owl#>
PREFIX email_m: <EmailModel.owl#>
PREFIX person_m: <PersonModel.owl#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
CONSTRUCT
{
        ?p context_m:hasEmailAddress ?ea .
}
WHERE
{
        ?p person_m:emailAddress ?ease .
        ?ea email_m:processedEmailAddress ?easp .
        FILTER (fn:matches(?ease, ?easp, "i"))
}
```

Figure 5. A SPARQL example rule relating an email address to person.

The rules listed above are clearly very simple examples of the sorts of rules that may be desired by consumer applications and our proposed use of SPARQL construct clauses to represent these rules builds a flexible base against which richer and more complex rules can be written, limited only by the expressivity of SPARQL construct clauses.

In our simple demonstration we have built the consumer application to allow user navigation around this fused set of separate data sources, allowing the user to immediately see which emails relate to which projects, what organisations and people they are working with in the context of projects and so on. The instance data for this context ontology is also published out for potential further consumption simply through persisting it to an RDF/OWL file via the Jena API and making the URI of that file available to other consumers.

```
Email                                                    EmailAddress
  subject (String, functional)                             processedEmailAddress (String)
  body (String, functional)                                rawEmailAddress (String)
  emailType (String, functional)                           email (Email)
  notesUrl (String, functional)                            receivesEmail (Email)
  universalID (String, functional)                         receivesEmailDirect (Email)
  postedDate (dateTime, functional)                        receivesEmailOnCc (Email)
  hasResponse (Email)                                      receivesEmailOnBcc (Email)
  inResponseTo (Email, functional)                         sendsEmail (Email, functional)
  hasTag (Tag)
  emailAddress (EmailAddress)                            Tag
    emailAddressRecipient (EmailAddress)                   tagsEmail (Email, functional)
      emailAddressTo (EmailAddress)                        tagName (String, functional)
      emailAddressCc (EmailAddress)                        tagSource (String, functional)
      emailAddressBcc (EmailAddress)
    emailAddressFrom (EmailAddress, functional)
```

Figure 6. The hierarchy of Email ontology entities and their attributes and relationships.

3.1 Extensions

We recognise some shortcomings in our current solution and aim to carry out further investigation into a number of specific areas to address these, most notably:

- The use of reification to record whether each instance data triple is stated or inferred. At the moment any inferred instance data triples are simply instantiated as a result of the SPARQL construct execution, and can then not easily be differentiated from the instance data triples originating in the source ontologies (other than by looking at the namespace into which they are persisted). Using a reification technique we would be able to record relevant provenance data such as the rule(s) which instantiated the instance data triple, the time, the application, etc.
- As the W3C[15] standardization work on rules languages and interoperability continues to mature we plan to extend the notion of context ontologies to support the representation of rules written in SWRL, RIF or another appropriate richer representation. These richer representations of rules would allow semantic information about the composition of the rules themselves to be conveyed, and would be used to generate the required SPARQL construct clauses.
- We also plan to use a flexible approach for disseminating the results of the information integration rules we presented in the previous section. In [3] we propose a novel mechanism for sharing and distributing ontology alignment information, POAF (Portable Ontology Aligned Fragments). POAF is agnostic as to what the alignment format is or to the type of data source used. In that sense, we could deploy a variant of the POAF solution to share and distribute the rules described in the previous section and even the context ontologies they operate on.

We also observe some aspects which loosely relate to this work, and which we will review further in our ongoing work:

- Our approach enables a limited form of distributed reasoning as it allows each instance of a consuming application to consume different data and publish their results. In some cases the results of these distributed consumer applications can then be collected and further analysed as appropriate.

[15] http://www.w3.org/

- This approach can be used to efficiently publish summary information about potentially private data when appropriate. In the example case we see a scenario where employee email data is processed locally to identify interesting contextual information, and in some cases this contextual information may be able to be then published to a wider audience whereas the actual email data is not.
- Finally, our work so far has identified that the SPARQL construct technique for building rules can be used to implement some of the standard RDF-S/OWL entailments. We specifically demonstrate this for `rdfs:subPropertyOf`, `owl:SymmetricProperty` and `owl:inverseOf`. This is a pragmatic solution to these specific RDF-S/OWL entailments where we use the RDF-S/OWL semantics to define occurrences of these in our model ontologies in the normal way, but we generate SPARQL construct clauses from rule templates to specifically instantiate each actual rule occurrence. This has two main benefits from our pragmatic perspective: firstly, there is no need to use a reasoner in addition to the SPARQL end point processing, and secondly, that we can use the same technique to instantiate and persist the resulting data. We do not propose that this SPARQL construct clause based implementation of these standard entailments should be used in preference to the capabilities offered by existing reasoners, but we note it here as a further capability for this rule representation and execution technique that we have described here.

4 Related work

Different notions of contexts have been proposed and investigated in the past. For example in [5] the authors argue for different types of contexts that contribute information relevant to natural language understanding. Each context is used to serve a different purpose, similar to our work where we adopt a dynamic notion context that is closely related and dependent on the use of source data. Our work uses source data and a set of semantic wrappers to elicit context and represent it in lightweight ontologies. Similarly, context has been used in [2] to aid in ontology elicitation whereby certain features of context dictate the primitive ontological constructs that will form up an ontology.

Information integration, the driver behind our work with semantically wrapped data and context ontologies, is also the focus of [6] but the authors deploy different means to achieve that: they propose to use a special kind of context knowledge, namely assumption knowledge, which refers to a set of implicit rules about assumptions and biases that govern the source data. This is similar to our notion of rules that integrate information from semantically wrapped data (section 3) but we apply them at a later stage. A number of existing information integration solutions are being researched and implemented, and are often referred to as ontology alignment solutions. The INRIA alignment API and server[16] is a good example of such an ontology alignment API for expressing and sharing alignments. Our work on SWEDER is currently focused at a far simpler and pragmatic level than existing efforts such as these, but our use of SPARQL construct does enable rich expressivity when it comes to information integration rule construction.

Another interesting angle we investigate with the use of context ontologies is deploying rules to capture the dependencies between properties. This is similar to the work of [1] where the authors elaborate on a naming convention scheme which is based on a loose ontology that represents the notions of kind and superkind. Their aim is to ease data usability by providing a naming scheme that allows for classification of source data. In our work we use properties and super properties found in the context ontologies to aid information integration and grouping.

[16] Available from: http://alignapi.gforge.inria.fr/

5 Conclusion

We presented SWEDER: Semantic Wrapping of Existing Data Sources with Embedded Rules. A pragmatic approach to semantically enable existing sources of data and then utilise multiple semantically enabled sources of that data through the creation of context ontologies to capture the specific rules and any new entities, relationships or attributes arising from the new context. This technique allows us to store rules directly within the ontologies in such a way that they can be easily extracted and executed by common capability within any consuming application, specifically through the use of SPARQL construct clauses.

Details of a simple example application in the domain of organisation and collaboration information were given, with a description of some simple rules that have been written to integrate these separate semantically wrapped data sources into the new context, taking advantages of inherent relationships between the data in those sources. Finally, we described our planned future work in this area which involves, amongst other things, the use of reification techniques to capture the provenance of any data inferred as a result of rule execution, and the desire to user a richer representation format to capture the semantics of our rules in the future.

ACKNOWLEDGEMENTS

This research was sponsored by the US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] N. Cohen, P. Castro, and A. Misra, 'Descriptive naming of context data providers', in *Proceedings of the CONTEXT'05 Context Representation and Reasoning (CRR'05), Paris, France*, (July 2005).

[2] P. DeLeenheer and A. deMoor, 'Context-driven disambiguation in ontology elicitation', in *Proceedings of the AAAI'05 Workshop on Contexts and Ontologies (AAAI'05/W1), USA*, (July 2005).

[3] Y. Kalfoglou, P. Smart, D. Braines, and N. Shadbolt, 'POAF: Portable Ontology Aligned Fragments', in *Proceedings of the ESWC'08 International Workshop on Ontologies: Reasoning and Modularity (WORM'08), Tenerife, Spain*, (jun 2008).

[4] J. McCarthy, 'Notes on formalizing contexts', in *Proceedings of the 5th National Conference on Artificial Intelligence, Los Altos, CA, USA*, pp. 555–560, (1986).

[5] R. Porzel, H-P. Zorn, B. Loos, and R. Malaka, 'Towards a separation of pragmatic knowledge and contextual information', in *Proceedings of the 2nd International Workshop on Contexts and Ontologies (C&O 2006), Riva del Garda, Italy*, (August 2006).

[6] H. Zeng and R. Fikes, 'Extracting assumptions from incomplete data', in *Proceedings of the CONTEXT'05 Context Representation and Reasoning (CRR'05), Paris, France*, (July 2005).

Precompiling $\mathcal{ALC}$ TBoxes and Query Answering

Ulrich Furbach and **Claudia Obermaier** [1]

Abstract. Knowledge compilation is a common technique for propositional logic knowledge bases. The idea is to transform a given knowledge base into a special normal form ([11],[6]), for which queries can be answered efficiently. This precompilation step is very expensive but it only has to be performed once. We propose to apply this technique to knowledge bases defined in Description Logics. For this, we introduce a structure called linkless graph, for $\mathcal{ALC}$ concepts. Further we present an algorithm, based on path dissolution, which can be used for this precompilation step. We discuss an efficient satisfiability test as well as a subsumption test for precompiled concept descriptions. Finally we show how to extend this approach in order to precompile Tboxes and to use the precompiled Tboxes for efficient Tbox reasoning.

1 Introduction

Knowledge compilation is a technique for dealing with computational intractability of propositional reasoning. It has been used in various AI systems for compiling knowledge bases offline into systems, that can be queried more efficiently after this precompilation. An overview about techniques for propositional knowledge bases is given in [7]; more recently [6] discusses, how knowledge compilation techniques can be seen as DPLL-procedures. One of the most prominent successful applications of knowledge compilation is certainly in the context of belief networks ([5]). In this context the precompilation step, although it is very expensive, pays off because it only has to be performed once to the network, which is not changing too frequently. In the context of Description Logics, knowledge compilation has firstly been investigated in [1], where $\mathcal{FL}$ concept descriptions are approximated by $\mathcal{FL}^-$concept descriptions.

In this paper we propose to apply a similar technique to knowledge bases defined in Description Logics. There are several techniques for Description Logics which are related to our approach. An overview on precompilation techniques for description logics such as structural subsumption, normalization and absorption is given in [8]. To perform a subsumption check on two concepts, structural subsumption algorithms ([2]) transform both concepts into a normal form and compare the structure of these normal forms. However these algorithms typically have problems with more expressive Description Logics. Especially general negation, which is an important feature in the application of Description Logics, is a problem for those algorithms. The technique of structural subsumption algorithms is used in CLASSIC [12], GRAIL [13] and LOOM [9]. In contrast to structural subsumption algorithms our approach is able to handle general negation without problems.

Normalization ([3]) is another preprocessing technique for Description Logics, which eliminates redundant operators in order to determine contradictory as well as tautological parts of a concept. In many cases this technique is able to simplify subsumption and satisfiability problems.

Absorption ([14]) is a technique which tries to eliminate general inclusion axioms from a knowledge base. Both absorption and normalization have the aim of increasing the performance of tableau based reasoning procedures. In contrast to that, our approach extends the use of preprocessing. We suggest to transform the concept into a normal form called linkless graph which allows an efficient consistency test. For this consistency test a tableau procedure is not necessary anymore. Some subsumption queries can also be solved without a tableau algorithm. We will discuss that in Section 4.

In this paper we will consider the Description Logic $\mathcal{ALC}$ [2] and we adopt the concept of linkless formulae, as it was introduced in [10, 11]. The following section shortly introduces the idea of our precompilation. In Section 2 we describe linkless concept descriptions and give a transformation of $\mathcal{ALC}$ concept descriptions into linkless ones. This transformation is extended to precompile $\mathcal{ALC}$ Tboxes in Section 3. Further in Section 4 we discuss an efficient consistency test for precompiled concept descriptions.

2 Precompilation of $\mathcal{ALC}$ Concept Descriptions

The precompilation technique we use for an $\mathcal{ALC}$ concept C consists of two steps. In the first step C is transformed into a normal form by removing so called *links* occurring in C. The notion of a link has first been introduced for propositional logic ([10]). Intuitively links are contradictory parts of a concepts which therefore can be removed preserving equivalence.

In the second step of the precompilation process we consider role restrictions. Given for example $C = \exists R.B \sqcap \forall R.D$. According to the semantic of $\mathcal{ALC}$ it follows from $x \in C^I$ that there is an individual y with $(x, y) \in R^I$ and $y \in (B \sqcap D)^I$. The concept $B \sqcap D$ is precompiled in the second step of the precompilation. The second step is repeated recursively until all concept descriptions of reachable individuals are precompiled.

In the following we assume that concept descriptions in $\mathcal{ALC}$ are given in NNF, i.e., negation occurs only in front of concept names. Further the term *concept literal* denotes either a concept name or a negated concept name.

Definition 1 *For a given concept C, the set of its paths is defined as follows:*

$$\mathit{paths}(\bot) = \emptyset$$
$$\mathit{paths}(\top) = \{\emptyset\}$$
$$\mathit{paths}(C) = \{\{C\}\}, \text{ if C is a literal}$$
$$\mathit{paths}(C_1 \sqcap C_2) = \{X \cup Y | X \in \mathit{paths}(C_1) \; and \; Y \in \mathit{paths}(C_2)\}$$
$$\mathit{paths}(C_1 \sqcup C_2) = \mathit{paths}(C_1) \cup \mathit{paths}(C_2)$$

[1] University of Koblenz-Landau Germany, email: obermaie@uni-koblenz.de

The concept description $C = \neg A \sqcap (A \sqcup B) \sqcap \forall R.(E \sqcap F)$ has the two paths $p_1 = \{\neg A, A, \forall R.(E \sqcap F)\}$ and $p_2 = \{\neg A, B, \forall R.(E \sqcap F)\}$. We typically use p to refer to both the path and the conjunction of the elements of the path when the meaning is evident from the context.

In propositional logic a link means that the formula has a contradictory part. Furthermore if all paths of a formula contain a link, the formula is unsatisfiable. In Description Logics other concepts apart from complementary concept literals are able to form a contradiction. It is possible to construct an inconsistent concept description by using role restrictions. For example the concept $\exists R.C \sqcap \forall R.\neg C$ is inconsistent since it a) claims that there has to be an individual which is reachable via the role R and belongs to the concept C and b) claims that all individuals which are reachable via the role R have to belong to the concept $\neg C$. This clearly is not possible. We could say that the concept contains a link in the description of a reachable individual. Therefore in Description Logics it is not sufficient to consider links constructed by concept literals. We will take a closer look at role restrictions in Section 2.2.

Definition 2 *For a given concept C a* link *is a set of two complementary concept literals occurring in a path of C. The positive (negative) part of a link denotes its positive (negative) concept literal.*

Note that we regard $\bot$ and $\top$ as a complementary pair of concept literals. Obviously a path p is inconsistent, iff it contains a link or alternatively $\{\exists R.A, \forall R.B_1, \ldots, \forall R.B_n\} \subseteq p$ and all paths in $A \sqcap B_1 \sqcap \ldots \sqcap B_n$ are inconsistent. Note that a set of consistent paths uniquely determines a class of semantically equivalent concept descriptions. Further given a concept description C with a consistent path p, it is obvious that the interpretation of p is a model of C. And the other way around each model I of C is also a model of a consistent path of C.

Now we are able to define the term linkless.

Definition 3 *A concept C is called* linkless*, if C is in NNF and there is no path in C which contains a link.*

This special structure of linkless concepts allows us to consider each conjunct of a conjunction separately. Therefore satisfiability can be decided in linear time and it is possible to enumerate models very efficiently.

Note that a linkless concept description can still be inconsistent. Take $\forall R.B \sqcap \exists R.\neg B$ as an example. This example makes clear that it is not sufficient to remove links from a concept description. We also have to consider role restrictions. But first we learn how to remove links from a given concept description.

2.1 Removing Links

In this section a method to transform an $\mathcal{ALC}$ concept into an equivalent linkless $\mathcal{ALC}$ concept is introduced. In propositional logic one possibility to remove links from a formula is to use path dissolution ([10]). The idea of this algorithm is to eliminate paths containing a link. This technique will be used in our context as well.

Definition 4 *Let G be a concept description and A be a concept literal. The* path extension *of A in G, denoted by $CPE(A, G)$, is a concept G' containing exactly those paths in G which contain A. The* path complement *of A in G, denoted by $CPC(A, G)$, is the concept G' containing exactly those paths in G which do not contain A.*

Note that Definition 4 does not mention how to construct $CPE(A, G)$ and $CPC(A, G)$. The naive way would be to construct the disjunction of all respective paths in G. However there are more elaborate methods ([10]), producing a far more compact results.

Lemma 5 *For a concept G and a set of literals A, where all elements of A occur in G, the following holds:*
$G \equiv CPE(A, G) \sqcup CPC(A, G)$

We want to construct $CPE(D, G)$ and $CPC(D, G)$ for $G = (D \sqcup \forall R.E) \sqcap (C \sqcup \forall R.B)$. G has the paths: $c_1 = \{D, C\}$, $c_2 = \{D, \forall R.B\}$, $c_3 = \{\forall R.E, C\}$ and $c_4 = \{\forall R.E, \forall R.B\}$. This leads to $CPE(D, G) = D \sqcap (C \sqcup \forall R.B)$ and $CPC(D, G) = \forall R.E \sqcap (C \sqcup \forall R.B)$.

In the following $\overline{C}$ denotes the complement of a concept C, which is given in NNF and can be calculated simply by transforming $\neg C$ in NNF. Our next aim is to remove a link from a concept description. Therefore we define a dissolution step for a link $\{L, \overline{L}\}$ through a concept expression $G = G_1 \sqcap G_2$ (such that $\{L, \overline{L}\}$ is neither a link for G_1 nor G_2). Note that each path p through $G_1 \sqcap G_2$ can be split into the paths p_1 and p_2, where p_1 is a path through G_1 and p_2 is a path through G_2.

Definition 6 *Given a concept description $G = G_1 \sqcap G_2$ which contains the link $\{L, \overline{L}\}$. Further $\{L, \overline{L}\}$ is neither a link for G_1 nor G_2. W.l.o.g. L occurs in G_1 and $\overline{L}$ occurs in G_2. The dissolvent of G and $\{L, \overline{L}\}$ denoted by $Diss(\{L, \overline{L}\}, G)$, is*

$$\begin{aligned} Diss(\{L, \overline{L}\}, G) = &(CPE(L, G_1) \sqcap CPC(\overline{L}, G_2)) \sqcup \\ &(CPC(L, G_1) \sqcap CPC(\overline{L}, G_2)) \sqcup \\ &(CPC(L, G_1) \sqcap CPE(\overline{L}, G_2)) \end{aligned}$$

Note that $Diss(\{L, \overline{L}\}, G)$ removes exactly those paths from G which contain the link $\{L, \overline{L}\}$. Since these paths are inconsistent, $Diss(\{L, \overline{L}\}, G)$ is equivalent to G. This is stated in the next lemma where we use the standard set-theoretic semantics for $\mathcal{ALC}$. The interpretation of a concept C denoted by C^I is a subset of the domain and can be understood as the set of individuals belonging to the concept C in the interpretation I.

Lemma 7 *Let G be a concept description and $\{L, \overline{L}\}$ be a link in G such that $Diss(\{L, \overline{L}\}, G)$ is defined. Then for all x in the domain holds: $x \in G^{\mathcal{I}}$ iff $x \in Diss(\{L, \overline{L}\}, G)^{\mathcal{I}}$.*

By equivalence transformations and with the help of Lemma 5 the following lemma follows.

Proposition 8 *Let $\{L, \overline{L}\}$ and G be defined as in Definition 6. Then the following holds:*

$$\begin{aligned} Diss(\{L, \overline{L}\}, G) \equiv &(G_1 \sqcap CPC(\overline{L}, G_2)) \sqcup \\ &(CPC(L, G_1) \sqcap CPE(\overline{L}, G_2)) \\ Diss(\{L, \overline{L}\}, G) \equiv &(CPE(L, G_1) \sqcap CPC(\overline{L}, G_2)) \sqcup \\ &(CPC(L, G_1) \sqcap G_2) \end{aligned}$$

Now it is easy to see how to remove links: Suppose a concept description C in NNF is given and it contains a link $\{L, \overline{L}\}$. Then there must be conjunctively combined subconcepts G_1 and G_2 of C where the positive part L of the link occurs in G_1 and the negative part $\overline{L}$ occurs in G_2. In the first step we construct $CPE(L, G_1), CPC(L, G_1), CPE(\overline{L}, G_2)$ as well as

$CPC(\overline{L}, G_2)$. By replacing $G_1 \sqcap G_2$ in C by $Diss(\{L, \overline{L}\}, G_1 \sqcap G_2)$ we are able to remove the link.

Next we give an algorithm to remove all links in the way it is described above. In the following definition $G[G_1/G_2]$ denotes the concept one obtains by substituting all occurrences of G_1 in G by G_2.

Algorithm 9 *Let G be a concept description.*

$linkless(G) \overset{def}{=} G$, *if G is linkless.*

$linkless(G) \overset{def}{=} linkless(G[H/Diss(\{L, \overline{L}\}, H)])$,
where H is a subconcept of G and $\{L, \overline{L}\}$ is a link in H, such that $Diss(\{L, \overline{L}\}, H)$ is defined.

Theorem 10 *Let G be a concept description. Then $linkless(G)$ is equivalent to G and is linkless.*

Note that in the worst case this transformation leads to an exponential blowup of the concept description.

2.2 Handling Role Restrictions

In the previous section we learned how to remove all links form a given concept. Now we turn to the second step of the precompilation and consider role restrictions.

Definition 11 *Let C be a linkless concept, p be a path in C with $\exists R.A \in p$ and $A, B_1, \ldots, B_n$ be concepts. Further let $\{\forall R.B_1, \ldots, \forall R.B_n\} \subseteq p$ be the (possibly empty) set of all universal role restrictions w.r.t. R in p. Then the concept $C' \equiv A \sqcap B_1 \sqcap \ldots \sqcap B_n$ is called* R-reachable *from C. Further the concept $C'' \equiv B_1 \sqcap \ldots \sqcap B_n$ is called* potentially R-reachable *from C. p is called a path used to reach C' (potentially reach C”) from C.*

Note that it is possible that a concept description C' is (potentially) reachable from a concept description C via several paths. A concept description C' is called (potentially) reachable from a linkless concept description C, if it is (potentially) R-reachable from C for some role R. Further *universally (potentially) reachable* is the transitive reflexive closure of the relation *(potentially) reachable*. Given a concept C and a concept C' which is reachable from C. Since the concept C' is equivalent to the concept $linkless(C')$, we call both C' and $linkless(C')$ reachable from C.

For example the following linkless concept description:
$C = (\exists R.(D \sqcup E) \sqcup A) \sqcap \forall R.\neg D \sqcap \forall R.E \sqcap B$ which has the two different paths $p_1 = \{\exists R.(D \sqcup E), \forall R.\neg D, \forall R.E, B\}$ and $p_2 = \{A, \forall R.\neg D, \forall R.E, B\}$. The concept $C' \equiv (D \sqcup E) \sqcap \neg D \sqcap E$ is reachable from C via path p_1 using $\{\exists R.(D \sqcup E), \forall R.\neg D, \forall R.E\}$. However C' is not linkless.

In the second step of the precompilation we precompile, i.e remove all links from, all universally (potentially) reachable concepts. Further it is necessary to precompile all potentially universally reachable concepts as soon as we want to answer queries. For example the concept $\forall R.\neg D \sqcap \forall R.\neg E$ does not have any reachable concept descriptions since no existential role restriction w.r.t. the role R is present. However asking a query to this concept can introduce the missing existential role restriction and can make a concept description reachable. For example asking the subsumption query $\forall R.\neg D \sqcap \forall R.\neg E \sqsubseteq \forall R.(\neg D \sqcap \neg E)$ leads to checking the consistency of $\forall R.\neg D \sqcap \forall R.\neg E \sqcap \exists R.(D \sqcup E)$. So the transformation of the subsumption query to a consistency test introduced the missing existential role restriction and therefore makes a concept description reachable. Therefore those concepts have to be precompiled as well.

Since the concepts which are (potentially) reachable from another concept via a path p only depends on the role restrictions in occurring in p, we regard all paths containing the same role restrictions as equivalent. The result of the precompilation of a concept C can be represented by a rooted directed graph (N, E) i.e a directed graph with exactly one source. The graph consists of two different types of nodes: path nodes PN and concept nodes CN. So $N = CN \cup PN$. Whereas each path node in PN is a set of paths in C and each node in the CN is a linkless concept description. The set of edges is $E \subset (CN \times PN) \cup (PN \times CN)$. A concept node C_i has a successor node for each set of equivalent paths in C_i and further there is an edge from each path node to the concept nodes of (potentially) reachable concepts. These edges are labeled by a set of universally quantified role restrictions or by a set containing universally quantified role restrictions and one existential role restriction. This label indicates the role restrictions used to (potentially) reach a concept.

Definition 12 *The linkless graph of a concept C is defined as follows:*

- *If C does not contain any role restrictions, the precompilation of C is a rooted directed graph consisting of the one node $linkless(C)$ with one successor which is the set of paths of C.*
- *If C contains role restrictions, the precompilation of C is a rooted directed graph with root $linkless(C)$ and for each set P_i of equivalent paths in C there is a subsequent path node. There is an edge form a path node P_i to the linkless graph of concept node C', if C' is (potentially) reachable from C via one of the paths in P_i. This edge is labeled by the set of role restrictions used to reach C' from C.*

Since the depth of the linkless graph of a given concept C corresponds to the depth of nested role restrictions in C, the linkless graph is always finite. Further in case of the precompilation of a single concept description, the linkless graph is a rooted dag.

Consider for example the following concept with its four paths:

$$C \equiv (B \sqcap \neg E) \sqcup ((B \sqcup \neg A \sqcup (\exists R.A \sqcap A)) \sqcap \exists R.E \sqcap \forall R.F)$$

$p_1 = \{B, \neg E\}$ $\quad$ $p_2 = \{B, \exists R.E, \forall R.F\}$
$p_3 = \{\neg A, \exists R.E, \forall R.F\}$ $\quad$ $p_4 = \{\exists R.A, A, \exists R.E, \forall R.F\}$

There are three sets of equivalent paths: $\{p_1\}$, $\{p_2, p_3\}$ and $\{p_4\}$. The root of the linkless graph is C. For each set of equivalent paths, there is a successor path node. In the next step, reachable concepts are considered: for instance the concept $E \sqcap F$ is reachable via the paths in the second set of paths using the role restrictions $\{\exists R.E, \forall R.F\}$. Therefore there is an edge from the second path node to the concept node $E \sqcap F$ with $label(\langle\{p_2, p_3\}, E \sqcap F\rangle) = \{\exists R.E, \forall R.F\}$. In the same way, the precompilation of all (potentially) reachable concepts are combined with the path nodes. The result is the graph depicted in Fig. 1.

3 Precompilation of General Tboxes

When answering queries with respect to a general Tbox it is necessary to restrict reasoning such that only models of this Tbox are considered. As described in [2] we transform the given Tbox $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ into a meta constraint $\mathcal{M}$ with

$$\mathcal{M} = (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$$

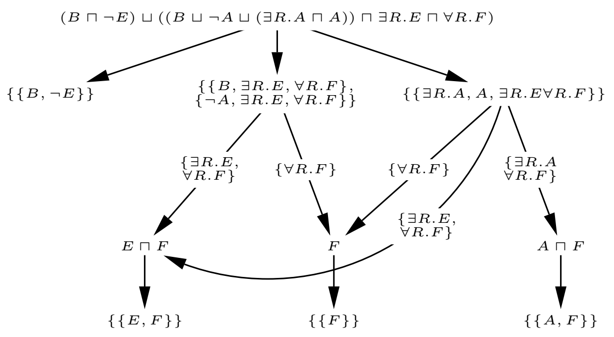

$(B \sqcap \neg E) \sqcup ((B \sqcup \neg A \sqcup (\exists R.A \sqcap A)) \sqcap \exists R.E \sqcap \forall R.F)$
$\{\{B, \neg E\}\}$
$\{\{B, \exists R.E, \forall R.F\}, \{\neg A, \exists R.E, \forall R.F\}\}$
$\{\{\exists R.A, A, \exists R.E \forall R.F\}\}$
$\{\exists R.E, \forall R.F\}$
$\{\forall R.F\}$
$\{\forall R.F\}$
$\{\exists R.A \forall R.F\}$
$\{\exists R.E, \forall R.F\}$
$E \sqcap F$
F
$A \sqcap F$
$\{\{E, F\}\}$
$\{\{F\}\}$
$\{\{A, F\}\}$

Figure 1. Example for a linkless graph

The idea of the linkless graph can be directly extended to represent precompiled Tboxes. We just construct the linkless graph for $\mathcal{M}$. Further, instead of just considering the concept nodes, each concept node C must also fulfill $\mathcal{M}$. So whenever there is a (potentially) reachable concept C', we precompile $C' \sqcap \mathcal{M}$ instead of just C'. In the case of precompiling a single concept description, the result is a linkless dag. In contrast to that the precompilation of a Tbox in general contains cycles and therefore leads to a linkless graph.

4 Properties of Precompiled Concept / Tboxes

Now we will consider some properties of a linkless graph in order to show that it is worthwhile to precompile a given concept description or a given Tbox into a linkless graph. We start by giving an efficient consistency check.

4.1 Consistency

Theorem 13 *Let C be a concept description and (N, E) its linkless graph with the root node H. Then holds: C is inconsistent iff $H = \bot$ or for each P with $\langle H, P\rangle \in E$ there is a concept C' which is reachable from C via one of the paths in P and the subgraph with root $linkless(C')$ is inconsistent.*

In the following we also use the term *inconsistent* for a linkless graph of an inconsistent concept. By adding a label sat to each concept node in the linkless graph, it can be ensured that no subgraph has to be checked more then once. At the beginning the sat label is set to the value $unknown$. Whenever during the consistency check a subgraph with root node C' is found to be (consistent) inconsistent, we set its sat label to ($true$) $false$. Only if it has the value $unknown$ it is necessary to perform a consistency check for this subgraph. Note that the use of the sat label does not only increase the efficiency of the consistency check. It furthermore prevents getting caught in cycles of the graph.

The consistency check described in Theorem 13 can be used to check the consistency of a precompiled Tbox as well. However it is important to use the sat label mentioned above, in order to ensure termination.

So to show that a precompiled concept is inconsistent, we have to compare all universally reachable concept description to $\bot$. Each of these checks can be done in constant time. Therefore the whole consistency check takes time linear to the number of universally reachable concepts.

As mentioned above, when precompiling a Tbox, the respective metaconstraint has to be added to every universally reachable concept. In the worst case there can be exponentially many universally reachable concepts. Given r different roles each with n existential role restrictions, m universal role restrictions which are all nested with depth d, in the worst case the number of universally reachable concepts is $r \cdot m \cdot 2^n \cdot d$. However in real world ontologies the number of universally reachable worlds is smaller. Furthermore precompiling a Tbox never increases the number of reachable concepts, contrariwise it usually decreases the number of reachable concepts. For example for the amino-acid [1] ontology $r = 5$, $d = 1$, $m = 3$ and $n = 5$. So in the worst case, there are 480 universally reachable worlds. But in reality, before the precompilation there are 170 and after the precompilation 154 reachable concepts.

4.2 Using the Linkless Graph to Answer Queries

Given the precompilation of a concept description, it is possible to answer certain subsumption queries very efficiently. In [4] an operator called conditioning is used as a technique to answer queries for a precompiled knowledge base. The idea of the conditioning operator is to consider $C \sqcap \alpha$ for a concept literal α and to simplify C according do α. Given for example $C = (B \sqcup E) \sqcap D$ and $\alpha = \neg B$, $C \sqcap \alpha$ can be simplified to $E \sqcap D$.

Definition 14 *Let C be a linkless concept description and $\alpha = C_1 \sqcap \ldots \sqcap C_n$ with C_i a concept literal. Then C conditioned by α, denoted by $C|\alpha$, is the concept description obtained by replacing each occurrence of C_i in C by $\top$ and each occurrence of $\overline{C_i}$ by $\bot$ and simplifying the conjunction according to the following simplifications:*

$\top \sqcap C = C$	$\top \sqcup C = \top$	$\bot \sqcap C = \bot$
$\bot \sqcup C = C$	$\exists R.\bot = \bot$	$\forall R.\top = \top$

It is clear that the conditioning operation is linear in the size of the concept description C. From the way $C|\alpha$ is constructed, it follows that $C|\alpha \sqcap \alpha$ is equivalent to $C \sqcap \alpha$ and obviously $C|\alpha \sqcap \alpha$ is linkless.

Definition 15 *Each concept literal is a* conditioning literal. *For each conditioning literal B, $\exists R.B$ and $\forall R.B$ are conditioning literals.*

Given a concept and a set of conditioning literals, in order to use conditioning for precompiled concepts we have to know how conditioning changes the set of paths in a concept description.

Definition 16 *Let P be a set of paths and α a set of concept literals. Then $\hat{P}$ denotes the set of paths obtained form P by*

1. *removing all elements of α from paths in P,*
2. *removing all paths from P, which contain an element whose complement is in α and*
3. *removing all paths p_1 from the remaining paths, if $p_2 \subset p_1$ for some $p_2 \in P$.*

Proposition 17 *Let C be a linkless concept description, P be the set of all paths in C and α a set of conditioning literals. Then the set of minimal paths of $C|\alpha$ is equal to $\hat{P}$.*

Next we want to use conditioning on linkless graphs. We give an algorithm for the conditioning operator for an arbitrary node of a linkless graph.

[1] http://www.co-ode.org/ontologies/amino-acid/2006/05/18/amino-acid.owl

Algorithm 18 *Let (N, E) be a linkless graph, $C \in N$ be a concept node and α a conditioning literal. Then C conditioned by α w.r.t. (N, E) denoted by $C|_{node}\alpha$ is the linkless graph obtained from (N, E) as follows:*

- *If α is a concept literal, then substitute $C|\alpha \sqcap \alpha$ for C. Further for each P with $\langle C, P\rangle \in E$ substitute $\hat{P}$ for P and add α to all paths in $\hat{P}$. If $\hat{P} = \emptyset$, remove its node and all its in- and outgoing edges.*
- *If $\alpha = QR.B$ with $Q \in \{\exists, \forall\}$ and B a conditioning literal, then substitute $C \sqcap QR.B$ for C and for each P with $\langle C, P\rangle \in E$ add $QR.B$ to all paths in P. Further*
  - *For all P whose paths do not contain a role restriction w.r.t. R: Create the linkless graph for B [1], add an edge from P to its root and label the edge with $\{QR.B\}$.*
  - *For all P whose paths contain role restrictions w.r.t. R:*
    - *For all P whose paths do not contain a universal role restriction w.r.t. R: Create the linkless graph for B, add an edge from P to its root and label the edge with $\{QR.B\}$.*
    - *If $Q = \forall$: add $QR.B$ to the labels of all edges $\langle P, C'\rangle \in E$ whose label contains a role restriction w.r.t. R and calculate $C'|_{node}B$.*
    - *If $Q = \exists$: for all edges $\langle P, C'\rangle \in E$ whose label contains only universal role restrictions w.r.t. R, copy the subgraph w.r.t. C' producing a new subgraph w.r.t. node C''. Create an edge from P to C'', label it with $label(\langle P, C'\rangle) \cup \{\exists R.B\}$ and calculate $C''|_{node}B$.*

During the calculation of $C|_{node}\alpha$ the depth of nested role restrictions in α decreases, hence the calculation always terminates. In fact, if the role restrictions are nested with maximal depth d, the conditioning only affects concept nodes which are reachable with d steps from the root node. Further the conditioning changes path nodes an labels of edges. So the complexity of the conditioning operator is linear to the number of concepts which are (potentially) universally reachable from the root node with d steps.

The conditioning operator for nodes can easily be extended to handle sets of conditioning literals.

Lemma 19 *Let C be concept description, H the root node of its linkless graph and α a set of conditioning literals. Then $C|\alpha$ is consistent iff $H|_{node}\alpha$ is consistent.*

Theorem 20 *Given a concept C, its linkless graph with root H and a subsumption query $C \sqsubseteq D$. If D is a concept literal, then $C \sqsubseteq D$ holds, iff $H|_{node}\neg D$ is inconsistent.*

Theorem 20 follows directly from Lemma 19, since $C \sqsubseteq D$ is equivalent to $C \sqcap \neg D$. We can use conditioning to combine C and $\neg D$. According to Lemma 19 $C|\neg D$ is consistent iff $H|_{node}\neg D$ is consistent. So we can construct $H|_{node}\neg D$ and test its consistency. If $H|_{node}\neg D$ is inconsistent, the subsumption query $C \sqsubseteq D$ holds. Theorem 20 can be easily extended for subsumption queries with a concept D, which is a in NNF and is constructed only using the connectives disjunction and negation.

Let's now consider the concept C whose linkless graph is presented in Fig. 1. If we want to answer the query $C \sqsubseteq B \sqcup \exists R.E \sqcup \exists R.A$ we have to condition the root of the linkless graph with $\{\neg B, \forall R.\neg E, \forall R.\neg A\}$. With the help of the consistency check mentioned above, we find out that the resulting graph is inconsistent. Therefore the subsumption query holds.

[1] Due to the structure of B, its linkless graph has a linear structure an can be constructed in time linear to the depth of nested role restrictions in B.

The linkless graph of a given Tbox $\mathcal{T}$ can be easily used to do Tbox reasoning. Let A and B be concepts both given in NNF and further A is constructed only using the connectives conjunction and negation and B is constructed only using the connectives disjunction and negation. If we want to check whether a subsumption $A \sqsubseteq_{\mathcal{T}} B$ holds, we have to check the consistency of $\mathcal{T} \sqcap A \sqcap \neg B$. Assuming that we have the linkless graph of $\mathcal{T}$, we only have to condition the root of the graph with the set of conjuncts in $A \sqcap \neg B$. We perform a consistency check for the resulting graph and if the graph is inconsistent, the subsumption holds.

Since in Tbox reasoning many queries are asked to the same Tbox, it is worthwhile to precompile the Tbox into a linkless graph. After that precompilation step, we can answer subsumption queries with the above mentioned structure very efficiently.

5 Future Work / Conclusion

In the next step, we want to investigate how to extend our approach to more expressive Description Logics for example $\mathcal{SHOIN}$, which is very important in the context of semantic web. Further it would be interesting to consider the satisfiability of concept descriptions which are almost linkless. In this context almost linkless means that the concept description is linkless outside of a certain scope.

Projection is a very helpful technique when different TBoxes have to be combined. Therefore we will investigate how to project linkless concept descriptions on a set of literals. Since linkless concept descriptions are closely related to a normal form which allows efficient projection, it is very likely that our normal form has this property too.

REFERENCES

[1] B. Selman and H. Kautz, 'Knowledge Compilation and Theory Approximation', *J. ACM*, **43**(2), 193–224, (1996).

[2] F. Baader et al., eds. *The Description Logic Handbook*. Cambridge University Press, 2003.

[3] P. Balsiger and A. Heuerding, 'Comparison of Theorem Provers for Modal Logics - Introduction and Summary.', in *TABLEAUX*, volume 1397 of *LNCS*, pp. 25–26. Springer, (1998).

[4] A. Darwiche, 'Decomposable Negation Normal Form', *Journal of the ACM*, **48**(4), (2001).

[5] A. Darwiche, 'A Logical Approach to Factoring Belief Networks', in *Proceedings of KR*, pp. 409–420, (2002).

[6] A. Darwiche and J. Huang, 'DPLL with a Trace: From SAT to Knowledge Compilation', in *Proceedings of IJCAI 05*, (2005).

[7] A. Darwiche and P. Marquis, 'A Knowlege Compilation Map', *Journal of Artificial Intelligence Research*, **17**, 229–264, (2002).

[8] I. Horrocks, 'Implementation and Optimization Techniques.', In Baader et al. [2], pp. 306–346.

[9] Robert M. MacGregor, 'Inside the LOOM Description Classifier.', *SIGART Bulletin*, **2**(3), 88–92, (1991).

[10] N. Murray and E. Rosenthal, 'Dissolution: Making Paths Vanish', *J. ACM*, **40**(3), 504–535, (1993).

[11] N. Murray and E. Rosenthal, 'Tableaux, Path Dissolution, and Decomposable Negation Normal Form for Knowledge Compilation', in *Proceedings of TABLEAUX 2003*, volume 1397 of *LNCS*. Springer, (2003).

[12] P. Patel-Schneider, D. McGuinness, and A. Borgida, 'The CLASSIC Knowledge Representation System: Guiding Principles and Implementation Rationale.', *SIGART Bulletin*, **2**(3), 108–113, (1991).

[13] A. Rector, S. Bechhofer, C. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon, 'The GRAIL concept modelling language for medical terminology.', *Artificial Intelligence in Medicine*, **9**(2), 139–171, (1997).

[14] D. Tsarkov and I. Horrocks, 'Description Logic Reasoner: System Description.', in *IJCAR*, eds., U. Furbach and N. Shankar, volume 4130 of *Lecture Notes in Computer Science*, pp. 292–297. Springer, (2006).

$\mathcal{K}$-$\mathcal{MORPH}$: A Semantic Web Based Knowledge Representation and Context-driven Morphing Framework

Sajjad Hussain[1] and **Syed Sibte Raza Abidi**[1]

Abstract. A knowledge-intensive problem is often not solved by an individual knowledge artifact; rather the solution needs to draw upon multiple, and even heterogeneous, knowledge artifacts. Each knowledge artifact may differ in terms of its modality, origin, and format; and may have different functional/operational roles in different problem-contexts. The synthesis of multiple knowledge artifacts to derive a 'comprehensive' knowledge artifact is a non-trivial problem. In this paper, we propose a semantic web based knowledge representation and morphing framework $\mathcal{K}$-$\mathcal{MORPH}$ that (a) semantically models the knowledge of various knowledge artifacts found in different modalities as ontologies; (b) semantically annotates the heterogeneous knowledge artifacts based on their respective ontologies; (c) represents the domain-specific constraints and specifications for the morphed knowledge, and treats them as a problem-context; (d) defines morphing constructs, to identify problem-specific knowledge components from the entire knowledge artifacts; (e) reconciles related knowledge components; and (f) generates a verified 'morphed' knowledge artifact that contains reconciled problem-specific knowledge from multiple artifacts. We discuss the architecture of a prototype medical knowledge morpher to show the need of knowledge morphing in medical domain.

1 INTRODUCTION

Knowledge originates in an assortment of knowledge artifacts–each artifact captures specific conceptual, contextual, functional and operational aspects of an underlying domain. For our purposes, we aim to apply knowledge for decision support and planning purposes. We argue that, central to knowledge-centric activities is the need to 'reason' over all available knowledge artifacts in order to (a) infer new knowledge, (b) test hypotheses, (c) suggest recommendations and actions, and (d) query rules to prove problem-specific assertions or theorems. The challenge, therefore, is to allow the reasoning process to simultaneously operate over multiple heterogeneous knowledge sources in order to derive a comprehensive reasoning outcome that builds on the different problem-specific perspectives dispersed across multiple knowledge artifacts that may differ in terms of modality and functional intensions. This challenge leads to the concept of 'knowledge morphing' that is defined as "the intelligent and autonomous fusion/integration of contextually, conceptually and functionally related knowledge objects that may exist in different representation modalities and formalisms, in order to establish a comprehensive, multi-faceted and networked view of all knowledge pertaining to a domain-specific problem"–Abidi 2005 [1].

From our perspective, which deals mainly with healthcare decision support [10], a knowledge artifact is basically a knowledge object, having a defined representation formalism, that encapsulates a specific kind of knowledge. The key knowledge modalities that we deal with are (a) explicit knowledge that is represented in terms of the following knowledge artifacts–clinical practice guidelines, clinical pathways and medical literature [2, 9]; (b) experiential knowledge represented as past cases and medical records; (c) observational knowledge that is derived from operational data and represented as data models and induced rules. Our knowledge morphing solution aims to synthesize these different knowledge artifacts, as per the problem description–i.e. the problem's *context*.

In this paper, we present our approach to pursue knowledge morphing. We propose a Semantic Web based Knowledge Morphing framework $\mathcal{K}$-$\mathcal{MORPH}$ that focuses on two aspects:

1. *Knowledge Representation*: Domain-specific knowledge representation is achieved through the use of *ontologies*. For each type of knowledge artifact we have developed a specific ontology that firstly models the generic structure (i.e. the form) of the knowledge artifact and then encodes the knowledge inherent within the artifact (i.e. its function) as an instance of the ontology [2, 9]. Representation of the problem that is mitigating knowledge morphing is pursued through the definition of a *problem-context* that encapsulates the problem specification–i.e. input and output elements, intension of the solution based on the morphed knowledge and domain-specific constraints.
2. *Context-driven Knowledge Morphing*: The knowledge morphing process comprises three main tasks: (i) specification of the *morphing construct* that explicitly defines the morphing intension, potential problem-specific knowledge constructs within the candidate knowledge artifacts, problem-context and morphing functions; (ii) knowledge morphing through *ontology reconciliation* based on a *proof-level ontology alignment* mechanism that synthesizes multiple artifact-specific ontology sub-constructs to yield a comprehensive multi-facted morphed knowledge object; and (iii) *validation and verification* of the morphed knowledge.

2 Knowledge Morphing

In principle, knowledge morphing aims to generate a comprehensive knowledge artifact with respect to a specific problem context. In practice, knowledge morphing aims to reconcile multiple knowledge resources–in our case knowledge artifacts represented as dis-

[1] NICHE Research Group, Faculty of Computer Science, Dahousie University, Canada, email: {hussain, sraza}@cs.dal.ca

tinct ontologies–to generate a morphed knowledge artifact. We argue that typically knowledge artifacts entail knowledge that is broader then a specific problem's scope. For instance, in healthcare a clinical guideline may contain knowledge about the diagnosis, treatment, prognosis and follow-up care for a particular disease. Therefore, we posit that the integration of entire knowledge artifacts unnecessarily exacerbates the complexity of establishing interoperability between multiple artifacts for no meaningful purpose. Rather, our approach for knowledge morphing follows three steps: (i) identify the knowledge components (or sub-artifacts) within a knowledge artifact that are pertinent towards the given problem description; (ii) extract the identified sub-artifacts as candidate constructs for knowledge morphing. Given that the original knowledge artifacts are represented as ontologies, the sub-artifacts will be represented as sub-ontologies that are validated for conceptual consistency and completeness; and (iii) reconcile or align the sub-ontologies to generate a new sub-ontology that represents the 'morphed' knowledge artifact as shown in Figure 1. In this way, our knowledge morphing approach pursues highly-specific ontology alignment guided by the problem's context–i.e. a single knowledge morphing context (akin to a query) forms the basis of the process. This also means that as the problem context changes a new morphed knowledge artifact will be developed. The re-usability of morphed knowledge is another interesting problem that we will be subsequently investigating.

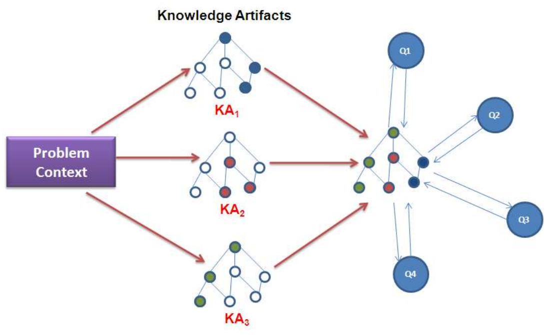


Figure 1. Knowledge Morphing

It may be noted that the literature suggests other approaches to knowledge morphing problem from different perspectives. ECOIN is one notable framework that performs semantic reconciliation of independent data sources, under a defined context [7]. Semantic reconciliation is performed at the context level by defining *conversion functions* between contexts as a network. ECOIN approach believes on the *single ontology, multiple views* notion [7], and introduces the notion of *modifiers* to explicitly describe the multiple specializations/views of the concepts used in different data sources. It exploits the modifiers and conversion functions, to enable context mediation between data sources, and reconcile and integrate source schemas with respect to their conceptual specializations. Another recent initiative towards knowledge morphing is the OpenKnowledge project [5]. The OpenKnowledge framework supports the knowledge sharing among different knowledge artifacts, not by sharing their asserted statements, instead by sharing their *interaction models*. An interaction model provides a context in which knowledge can be transmitted between two (or more) knowledge sources (peers). This approach has a closer relevance with semantic service composition [8], where each interaction model (stands for a knowledge source) can be seen as a service that interacts with other services based on their service descriptions and business logics.

3 $\mathcal{K}$-$\mathcal{MORPH}$ ARCHITECTURE

We adopt a Semantic Web (SW) architecture [3] to address the problem of knowledge morphing. Given that at the core of knowledge morphing is the need to semantically model the different knowledge artifacts, we believe that the SW offers a logic-based framework to (a) semantically model the knowledge of various knowledge artifacts found in different modalities as ontologies; (b) semantically annotate the heterogeneous knowledge artifacts based on their respective ontologies; (c) capture and represent the underlying domain concepts, and the semantic relationships that are inherent within a problem-context, in terms of a domain ontology; (d) ensure interoperability between multiple ontologically defined knowledge artifacts; and (e) maintaining changes, evolution and management of ontologies.

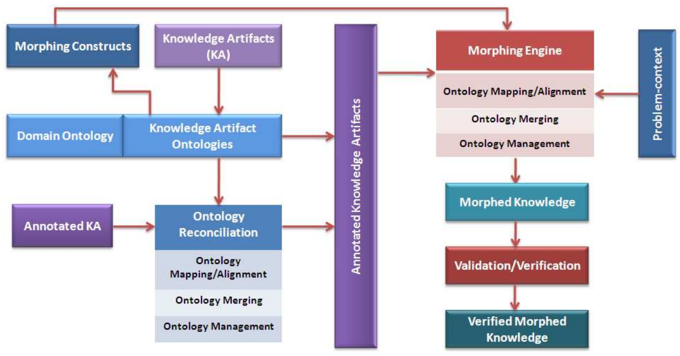


Figure 2. High-level schematic of $\mathcal{K}$-$\mathcal{MORPH}$

$\mathcal{K}$-$\mathcal{MORPH}$ comprises the following elements (see Figure 2).

1. *Domain Ontology* is used to capture and represent explicit domain knowledge in terms of generic and standardized concepts.
2. *Knowledge Artifact Ontologies* represent the structure and content of different knowledge artifacts–each knowledge artifact type is represented by its unique knowledge artifact ontology. These ontologies are both guided by and reflect the domain ontology.
3. *Knowledge Artifact Annotation* is the process to annotate the content of a knowledge artifact with respect to its corresponding knowledge artifact ontology. An annotated knowledge artifact is called an *Ontology-encoded Knowledge Artifact* (OKA).
4. *Morphing Constructs* specify the problem of knowledge morphing for a given context in terms of declarative knowledge dictating how to operate with the available knowledge artifacts to derive the problem-specific knowledge components (or sub-ontologies).
5. *Ontology Reconciliation* process involves the alignment of two (or more) candidate OKAs to yield morphed knowledge.
6. *Reconciliation of Other Annotated Ontologies* deals with knowledge that is annotated using other ontologies and attempts to find correspondences between ontologies, and then mapping the annotated ontologies into knowledge artifact ontologies [6].
7. *Morphing Engine* is the main component that handles the knowledge morphing process through proof-level ontology alignment. It takes as input a problem-context, OKAs and morphing constructs and then performs the following:

(a) Identifies the knowledge components in a knowledge artifact ontology that have relevance with the problem-context.

(b) Maps/aligns all identified knowledge components.

(c) Finds inconsistencies in aligned knowledge components.

(d) Merges knowledge components via merging rules.

8. *Validation and Verification*: The morphed knowledge can be validated by employing proof engines, and verified against the expert knowledge.

The above-mentioned $\mathcal{K}$-$\mathcal{MORPH}$ elements are described below.

3.1 Knowledge Representation and Annotation via Ontologies

In $\mathcal{K}$-$\mathcal{MORPH}$, a necessary step for knowledge morphing is to pursue knowledge formalization in order to support domain-specific inferencing based on declarative and procedural knowledge. Declarative knowledge describes the domain concepts, potential problems and probable solutions. Such declarative knowledge can be causal, qualitative, descriptive or quantitative. Procedural knowledge describes how to apply the knowledge to actually solve domain-specific problems, whilst taking into account, and satisfying the unique operational constraints of a domain-specific institution.

We use ontologies to model a knowledge artifact as it allows (i) formalization of domain-specific knowledge; (ii) conceptualization of the knowledge along declarative and procedural dimensions; (iii) annotation of the knowledge based on an ontological model; (iv) reuse and evolution of the knowledge; (v) use of standard terms and concepts; and (vi) identification of similar knowledge components that can potentially be aligned to achieve knowledge morphing. For our purposes, an Ontology is formally defined as follows:

Definition 1 (Ontology) *Let $\mathcal{V}$ be the set of structured vocabulary, and $\mathcal{A}_x$ be the set of axioms about $\mathcal{V}$, which are formulated in formal language $\mathcal{L}$. An ontology O is defined by the following tuple:*

$$O := \langle \mathcal{L}, \mathcal{V}, C, H_C, R, H_R, I, \mathcal{A}_x \rangle$$

where, concepts $C \subseteq \mathcal{V}$ of the schema are arranged in a subsumption hierarchy H_C. Binary relations $R \subseteq \mathcal{V}$ exist between pairs of concepts. Relations can also be arranged in a subsumption hierarchy H_R. (Meta-)Data is constituted by instances $I \subseteq \mathcal{V}$ of specific concepts. Additionally, one can define axioms $\mathcal{A}_x = \mathcal{L}(\mathcal{V})$ which can be used to infer knowledge from already asserted knowledge.

An Ontology $O' := \langle \mathcal{L}, \mathcal{V}, C', H'_C, R', H'_R, I', \mathcal{A}'_x \rangle$ is a sub-ontology of O, where $C' \subseteq C, H'_C \subseteq H_C, R' \subseteq R, H'_R \subseteq H'_R, I' \subseteq I, \mathcal{A}'_x \subseteq \mathcal{A}_x$; and written as $O' \prec O$.

3.1.1 Domain Ontology and Knowledge Artifact Ontology

In $\mathcal{K}$-$\mathcal{MORPH}$, knowledge artifacts are represented using two different (but inter-related) ontologies, namely: (i) *Domain Ontology*; and (ii) *Knowledge Artifact Ontology*. A domain ontology serves as a high-level ontology that describes the fundamental concepts of the domain–i.e. declarative knowledge. It serves two purposes: (i) Standardization of the domain-specific concepts and relations defined in the knowledge artifact ontologies; and (ii) Specification of abstract knowledge links between contextually and functionally congruent knowledge components in different knowledge artifact ontologies. The execution of these knowledge links, through proof engines, eventually leads to knowledge morphing.

A knowledge artifact ontology serves as a lower-level ontology that captures both the structure and content of a particular knowledge artifact–such as a practice guidelines [2], past cases and so on. Each knowledge artifact is represented by an individual knowledge artifact ontology that models the semantic relations inherent in the knowledge artifact, and characterizes the procedural knowledge as a sequence of control structures. Each control structure may deal with the identification, rationalization, ordering, execution and quantification of a domain-specific action and its effects.

3.1.2 Contextualizing Ontologies

Ontologies and contexts are used to model a domain with different views. Ontologies define a shared model that provides a global perspective, whereas contexts are used to realize a local aspect of a domain. Contextualizing an ontology deals with an adaptation of its ontology model to support a local view [11, 12]. In $\mathcal{K}$-$\mathcal{MORPH}$, each knowledge artifact ontology models the procedural knowledge of a knowledge artifact. However, the intended semantics and implementation details of each procedure may vary in different contexts. Contextualizing a knowledge artifact ontology can provide its local view that models (i) a specific interpretation of its ontology concepts, and (ii) an implementation of its procedural knowledge that can be applied in a particular context.

3.2 Morphing Constructs

In order to capture the behaviour of context, under which two or more knowledge artifacts can morphed to solve a specific problem, we defined a *Morphing Construct*. The morphing construct supervises the knowledge morphing process (see section 3.4), and provides a context for determining when, where and how two or more knowledge artifacts need to be reconciled. A Morphing Construct is a tuple that contains context-specific knowledge components and is formally defined as follows:

Definition 2 (Context Declaration) *A Context Declaration $\mathcal{C}_x = \langle l, \mathcal{A}'_x \rangle$ is a tuple comprised of a context label l, and a set of axioms $\mathcal{A}'_x \subseteq \mathcal{A}_x$ that specifies the problem-context and domain-specific constraints, under which ontology-encoded knowledge artifacts are allowed to morph.*

Definition 3 (Morphing Construct) *Let $O_\mathcal{K}$ be a knowledge artifact ontology. Morphing construct $\mathcal{M}_c = \langle O'_\mathcal{K}, \mathcal{C}_x, c_\mathcal{D} \rangle$ is a tuple of a contextualized knowledge artifact sub-ontology $O'_\mathcal{K} \prec O_\mathcal{K}$, a context declaration $\mathcal{C}_x$, and a domain concept $c_\mathcal{D}$ from a domain ontology $O_\mathcal{D}$.*

Example # 1:

1. Let $O'_\mathcal{K} = CPG'$ is a contextualized sub-ontology of CPG (see section 4) [2].
```
CPG' = [hasRecommendation(X,R),
hasDecisionCriteria(R,C), hasFollowup(R,F),
intendedPatient(X,P), hasTimeInterval (R, T), ...]
```
2. Let $\mathcal{C}_x = Cx_1 =$
```
<cdss,[(forLocation(cdss, halifax) ∨ forLocation(cdss,
toronto)), hasResources (cdss, patient_care),
applicableTo(cdss, resident_patient),
hasPractioners(cdss, family_physician),
hasInclusionCriteria (cdss, evidence_based_Recom),
hasExclusionCriteria (cdss, follow-up_Recom), ...]>
```
3. Let $c_\mathcal{D} = cpgBasedRecom$ a concept from the Domain Ontology

An example morphing construct can be written as

$$Mc_1 =< CPG', Cx_1, cpgBasedRecom >$$

A contextualized sub-ontology $O'_{\mathcal{K}}$ represents how certain knowledge components of a knowledge artifact ontology $O_{\mathcal{K}}$ can be utilized under a problem-context. The above example shows an example morphing construct Mc_1 for a knowledge artifact ontology for Clinical Practice Guidelines (CPG) (see section 4) [2]. CPG' is defined as a contextualized sub-ontology that can be utilized under the problem-context Cx_1, augmenting the domain concept $c_{\mathcal{D}} = cpgBasedRecom$. By the declarative knowledge of morphing constructs, sub-ontologies are served as contextualized ontologies of given knowledge artifact ontologies that provide all the contextually-relevant knowledge components that need to be reconciled, to produce a morphed knowledge artifact.

3.3 Ontology Reconciliation

Our knowledge morphing approach is based on the reconciliation of sub-ontologies to yield a unified 'morphed sub-ontology'. Ontology reconciliation among ontologies is normally performed by (i) identifying conceptual similarities among two source ontologies; (ii) aligning and mapping sources ontologies based on identified similarities; (iii) merging, integrating, mediating source ontologies based on found mappings/alignments; and (iv) finding and resolving semantic inconsistencies in reconciled ontologies [6].

Mapping and alignment between ontologies have been carried out based on their lexical, conceptual, and structural similarities [6]. We believe such mappings can became more 'trustworthy' by finding similarities among entities that are driven from the underlying ontology axioms; and so their proofs. Alignments established between entities that takes account in the underling ontology axioms and their proofs, are called *proof-based alignments*. The underling ontology axioms and proofs are served as a declarative semantic model for describing a domain that ontology relates to. By identifying *proof-based alignment* candidates, mappings and alignments will then be consistent with the semantic model, and befitted with the declarative knowledge provided by the ontology axioms.

A *proof-level ontology* is an ontology where each of its triples ($\langle subject, predicate, object\rangle$) are entailed by triples that are not necessarily from the same ontology. An ontology (that represents a relational schema) can be seen as a proof-level ontology, where each of its triples are asserted facts, and are entailed by null (denoted as $\perp \models T$). An ontology at the proof-level can provide the justifications behind inferred instances based on ontology-based and user-defined axiomatic systems (that are modeled in $L(V) = Ax$).

We argue that proof-level ontologies can serve as better candidates for ontology alignment process. If *proof-based alignment* is established among two (inferred) entities in triples T_1 and T_2 from two proof-level ontologies along their proofs ($\mathbb{T}' \models T_1$ and $\mathbb{T}'' \models T_2$, where $\mathbb{T}', \mathbb{T}'' \subseteq \mathbb{T}$), then entities appear in their justifications (modeled as set of ontology triples $\mathbb{T}', \mathbb{T}'' \subseteq \mathbb{T}$) can be treated as the next alignment candidates. *Proof-based alignment* approach ensures that such alignment candidates are aligned in a target ontology.

Proof-based alignment not only finds a similarity between entities, but also maintains the relationship between aligned entities with their original proof structures. After an entity e in one ontology is *proof-based aligned* with an entity f from another ontology, additional proofs can be generated for the new aligned entity f. Such proofs will be analogous to the proof of e. Analogous proofs represent similar reasoning strategies used in a particular domain but expressed in different terminologies.

3.4 Morphing Engine

Our *Morphing Engine* inputs the problem-context, ontology-encoded knowledge artifacts (OKAs), domain ontology, and morphing constructs. It employs the ontology reconciliation process, supervised by the morphing constructs and domain axioms; and generates a morphed knowledge artifact. Morphing constructs lead to identify the contextualized OKAs to be reconciled; whereas a domain ontology provides domain axioms that specify domain-specific constrains to be fulfilled during the morphing process. An abstract process of morphing engine is defined as follows:

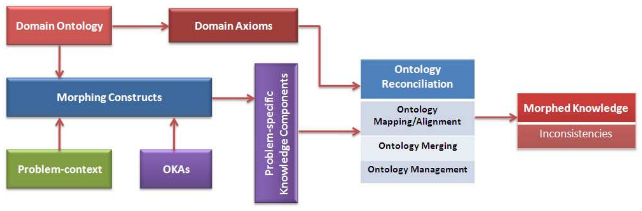


Figure 3. $\mathcal{K}$-$\mathcal{MORPH}$: Morphing Engine

Definition 4 (Knowledge Morphing Process) *Let $\mathbb{O}_{\mathcal{K}}$ be a set of ontology-encoded knowledge artifacts, $\mathbb{O}_{\mathcal{D}}$ be the set of domain ontologies, $\mathbb{C}_x$ be the set of problem-contexts, Π be the set of morphing constructs, and $\mathcal{I} \subseteq Ax$ be the set of logical inconsistencies in the morphed OKA. Knowledge Morphing Process is then the function*

$$\mathcal{MORPH} : \mathbb{O}_{\mathcal{K}} \times \mathbb{O}_{\mathcal{D}} \times \mathbb{C}_x \times 2^{\Pi} \longrightarrow \mathbb{O}_{\mathcal{K}} \times 2^{\mathcal{I}}$$

An abstract architecture of our morphing engine is shown in Figure 3. It first employs the problem-context to determine the problem-specific knowledge components from different knowledge artifact ontologies using morphing constructs. Morphing constructs also delivers the correspondence between identified knowledge components and domain concepts (see section 3.2). Domain ontology provides *Domain Axioms* that describe the semantic relationships among domain concepts. Once the correspondence between the knowledge components and domain concepts is achieved, the morphing engine employs the ontology reconciliation process that (i) computes the semantic correspondence between knowledge components based on the semantic relationships between their corresponding domain concepts; (ii) aligns and then merges knowledge components based on their correspondence; (iii) identifies and resolves semantic inconsistencies, if present; and (iv) generates a morphed (ontology-encoded) knowledge artifact, and unresolved inconsistencies in it.

3.5 Evaluation: Validation and Verification

Once the morphed knowledge artifact is generated, $\mathcal{K}$-$\mathcal{MORPH}$ employs an evaluation process to validate the morphed knowledge. Some of the approaches [4] we plan to involve for evaluating the morphed OKA, are as follows: (i) evaluating, whether results generated from the morphed OKA in a particular application under a specific context are 'satisfactory'; (ii) evaluating logical consistencies, by checking whether the morphed OKA model is consistent with pre-defined domain-specific theories provided by domain experts; (iii) evaluating the morphed OKA against a "golden standard", if available; and (iv) evaluating, whether a pre-defined structure and design principles are maintained in the morphed OKA.

4 USING $\mathcal{K}$-$\mathcal{MORPH}$ FOR CLINICAL DECISION-MAKING

Clinical decision making involves an active interplay between various medical knowledge artifacts to derive pragmatic solutions for a clinical problem [10]. We are currently developing a prototype *Medical Knowledge Morpher* (as shown in Figure 4) that deals with the three different medical knowledge artifacts, namely, (i) Electronic Patient Records (EPR), (ii) Clinical Practice Guidelines (CPG), and (iii) Clinical Pathways (CP). Each knowledge artifact, despite targeting the same domain knowledge, has a different purpose. For instance, CPGs are systematically developed disease-specific recommendations to assist clinical decision-making in accordance with the best evidence [2]. CP serve as institution-specific workflows that guide the care process in line with the evidence-based medical knowledge found in CPG [9]. EPR are containers of patient's longitudinal medical information.

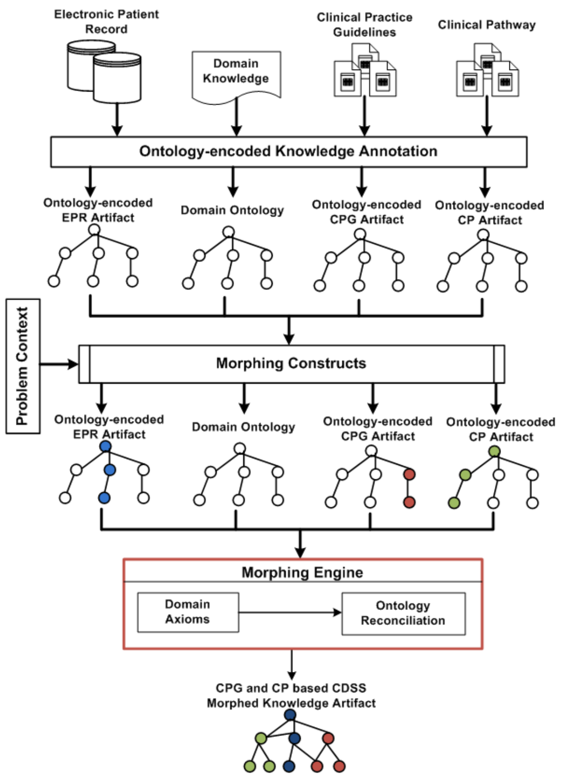

Electronic Patient Record
Domain Knowledge
Clinical Practice Guidelines
Clinical Pathway
Ontology-encoded Knowledge Annotation
Ontology-encoded EPR Artifact
Domain Ontology
Ontology-encoded CPG Artifact
Ontology-encoded CP Artifact
Problem Context
Morphing Constructs
Ontology-encoded EPR Artifact
Domain Ontology
Ontology-encoded CPG Artifact
Ontology-encoded CP Artifact
Morphing Engine
Domain Axioms
Ontology Reconciliation
CPG and CP based CDSS Morphed Knowledge Artifact

Figure 4. Medical Knowledge Morphing

For clinical decision making we need an active interplay between these three distinct artifacts as follows: The EPR determines the clinical context that in turn determines which CPG need to be referred to make the 'right' clinical decisions. Based on the context, the morphing construct will determine the clinical intention, the knowledge needs for the given intention and the knowledge resources to be utilized. The two knowledge sources in this case–i.e. the CPG and CP –both now need to be integrated to optimally apply the knowledge for clinical decision making. The CPG will provide the declarative knowledge and it needs to be aligned with the procedural knowledge contained by CP. Knowledge morphing is therefore needed at two levels: (a) morphing the different knowledge components from multiple knowledge artifacts of the same type–i.e. recommendations from multiple CPGs; and (b) morphing different knowledge artifact types–i.e. synthesizing CPG and CP. The morphed knowledge artifact will consist of operational relations between EPR, CPG, and CP knowledge artifacts and serve as a holistic knowledge artifact to support clinical decision making in terms of (a) evidence-based recommendations based CPG-based knowledge, based on the patient scenario recorded in EPR, and also (b) institution-specific workflow knowledge to pragmatically execute the recommendations.

5 CONCLUDING REMARKS

Optimal and complete decision support needs a comprehensive knowledge-base. Developing such a self-contained knowledge-base as an independent entity is a challenging undertaking. One possible approach is to systematically leverage multiple knowledge sources to develop a comprehensive knowledge-base–such a composite knowledge-base not only manifests the specializations of its constituent sources but also broadens the knowledge coverage whilst maintaining the uniqueness and independence of the original knowledge sources. In this paper, we presented our knowledge morphing approach, and the $\mathcal{K}$-$\mathcal{MORPH}$ framework, to pursue the development of a comprehensive, multi-facted knowledge-base. We are currently developing a prototype medical knowledge morpher to support clinical decision-making process.

ACKNOWLEDGEMENTS

This research is funded by a grant from Agfa Healthcare (Canada).

REFERENCES

[1] SSR Abidi, 'Medical knowledge morphing: Towards the integration of medical knowledge resources', *18th IEEE International Symposium on Computer-Based Medical Systems, Dublin*, (June 23-24 2005).

[2] SSR Abidi and Shapoor Shayegani, 'Modeling the form and function of clinical practice guidelines: An ontological model to computerize clinical practice guidelines', in *Knowledge Management for Healthcare Processes Workshop at ECAI 2008*, Patras, Greece, (July 21-25 2008).

[3] T Berners-Lee, J Hendler, and O Lassila, 'The semantic web', *Scientific American*, **284**(5), 34–43, (2001).

[4] Janez Brank, Marko Grobelnik, and Dunja Mladenic, 'A survey of ontology evaluation techniques', in *conference on Data Mining and Data Warehouses (SiKDD)*, (2005).

[5] Dave Robertson et. al., 'Open knowledge: Semantic webs through peer-to peer interaction', Technical Report DIT-06-034, University of Trento, Povo, Italy, (May 2006).

[6] Jérôme Euzenat and Pavel Shvaiko, *Ontology matching*, Springer-Verlag, Heidelberg (DE), 2007.

[7] Aykut Firat, Stuart Madnick, and Benjamin Grosof, 'Contextual alignment of ontologies in the ecoin semantic interoperability framework', *Inf. Technol. and Management*, **8**(1), 47–63, (2007).

[8] Fausto Giunchiglia, Fiona McNeill, and Fiona McNeill, 'Web service composition via semantic matching of interaction specifications', in *WWW2007*, Banff, Canada, (May 8-12). ACM Press.

[9] K. Hurley and SSR Abidi, 'Ontology engineering to model clinical pathways: Towards the computerization and execution of clinical pathways', in *20th IEEE Symposium on Computer-Based Medical Systems*, Maribor, Slovenia, (June 20-22 2008). IEEE Press.

[10] VL Patel, JF Arocha, and J Zhang, *Cambridge Handbook of Thinking and Reasoning*, chapter Thinking and reasoning in medicine, Cambridge University Press, 2004.

[11] Aviv Segev and Avigdor Gal, 'Putting things in context: A topological approach to mapping contexts to ontologies', *Journal on Data Semantics IX, LNCS 4601*, 113–140, (2007).

[12] Tony Veale and Yanfen Hao, 'Corpus-driven contextualized categorization', in *Contexts and Ontologies workshop at ECAI 2006*, Trento, Italy, (August 2006).

Cognitive context and syllogisms from ontologies for handling discrepancies in learning resources

Christiana Panayiotou and **Brandon Bennett**[1]

Abstract. The deployment of learning resources on the web by different experts has resulted in the accessibility of multiple viewpoints about the same topics. In this work we assume that learning resources are underpinned by ontologies. Different formalizations of domains may result from different contexts, different interpretation of terminology, different vocabularies to define concepts, incomplete knowledge and conflicting knowledge of the experts deriving the ontologies. We define the notion of *cognitive learning context* that refers to multiple and possibly inconsistent ontologies about a single topic. We then discuss how this notion relates to the cognitive states of *ambiguity* and *inconsistency*. Discrepancies in viewpoints can be identified via the inference of conflicting arguments from consistent subsets of statements. Two types of arguments are discussed, namely arguments inferred directly from taxonomic relations between concepts and arguments about the necessary and jointly sufficient features that define concepts.

1 Introduction

Learning resources are becoming increasingly available to the learners on the web. As a result a learner may have access to multiple learning resources about the same topic. We assume that each learning resource is underpinned by an ontology. Ontologies of the same domain may be represented at various degrees of abstraction and granularity. Reasons can be traced to different points of view and experience of the experts deriving the ontologies. It can also be due to different degrees of completeness of ontologies. The learner may not be able to determine whether discrepancies in ontologies arise due to incompleteness of knowledge, due to disagreement between ontologies, or due to differences in the perspectives giving rise to different viewpoints.

This paper's purpose is twofold. Firstly, to formalize the cognitive state of ambiguity and inconsistency arising when a learner encounters incomplete ontologies of learning resources about a topic. In order to address the problem of cognitive ambiguity and confusion of learners we allow resources with conflicting or different information to be part of the same cognitive context. We assume that the context is related to the goal of the learning activity (referred to as the *focus* of the learning activity) rather than on the compatibility of the resources referred to by the context. As a consequence, the context may involve multiple domains, if multiple domain points of view are relevant to the learning topic. For example, the topic may involve the points of view of multiple domains like psychology, social science and anthropology in order to form a particular position.

Secondly, we propose a proof-theoretic approach to the automatic derivation of arguments from ontologies. Out of the resources available to the learner, she only needs to consider those subsets of statements that are relevant to her reasoning. We use the notion of *sub-ontology* to describe consistent subsets of ontologies that can be used to form the cognitive context of the learner. We also suggest that differences in ontologies can be identified via the use of argumentation and we formalize two different types of arguments that are useful in learning. These are syllogistic arguments following from hierarchical relations in ontologies and arguments about necessary and jointly sufficient features of concepts. The rest of this paper is outlined as follows. Section 2 reviews related work on the definition of context based on the locality assumption and paraconsistent logics modeling inferences from inconsistent theories. In section 3 we discuss the notions of cognitive learning context, cognitive ambiguity and inconsistency arising from the resources taking part in a learning activity. Section 4 discusses our approach to defining syllogistic arguments and arguments from necessary and jointly sufficient features for the definition of concepts. Section 5 summarizes the main issues discussed and briefly outlines current research.

2 Related Work

The Local Model Semantics [6] provide a foundation for reasoning with contexts which is based on two main principles: the principle of locality and the principle of compatibility. The first states that reasoning requires only a part of what is potentially available [6]. The principle of compatibility states that there is compatibility among the kinds of reasoning performed in different contexts [6], thus assumes a relatedness between different contexts by some meaningful relation of subsets of local models. In this paper we focus on the cognitive context of a learner in a learning situation. The principle of locality, discrepancies in the ontologies of learning resources and the assumption that the available information may be incomplete affect the way learners interpret the information and can be used to model the cognitive state of the learner.

The assumption of possible inconsistency in theories or knowledge bases has been addressed via a number of logics. For example paraconsistent, many-valued logics and modal logics are among the ones used widely to model inconsistency. Notable uses of paraconsistent and possible world semantics to model mental models and epistemic states can be traced to the works of [5] and [9]. Fagin and Halpern [5] consider each agent as a society of minds rather than a single mind. Lokhorst [8], inspired by the local reasoning models of Fagin and Halpern [5], developed a two-valued split-patients local reasoning model as a structure: $M = \langle W, w_0, \Psi, S, R, V \rangle$, where W is a set of possible worlds, w_0 is the actual world in W, Ψ is a set

[1] School of Computing, University of Leeds, UK, email: {cpaa,brandon}@comp.leeds.ac.uk

of "minds", S is a function from W into the set of non-empty subsets of Ψ (i.e. S maps a world to the set of minds in which this world is possible) and R is a function from Ψ into $W \times W$. If we consider each mind in Lockhorst's model as a different ontology obtained independently of each other we might argue that the above model is suitable to be allied as a local reasoning model of a learner. However, this model would imply that the learner is unable to associate information from different resources.

The paraconsistent logic LEI is based on the idea of multiple observers having diverging views about a certain state of affairs. It extends classical logic with the formula $p?$ where $p?$ is satisfied whenever p holds in all plausible worlds. Unlike the traditional modal logics approach to modeling necessity and possibility, the LEI employs two satisfaction relations: the credulous and the skeptical approach. Martins et al. [9] provided a multiple world semantics to the above idea where each plausible world corresponds to a particular view of the world. The above approach is useful in comparing beliefs derived by the credulous vs. skeptical entailment relation which is different from the focus of this paper.

Unlike the above model, we assume that the learner is able to compare information obtained from different ontologies for its relevance, its validity, and for drawing inferences. Where the relevance between concepts used in different ontologies cannot be established the learner is bound to feel confused. We therefore need to combine two levels of reasoning: a local reasoning level which considers each ontology locally and the meta-epistemic level, at which the agent compares inferences drawn locally in each ontology and determines compatibility with other ontologies.

Our work is influenced by significant relevant work in the area of representing and combining information from different ontologies using context formalisms, e.g. [2, 7, 3]. In this paragraph we briefly discuss how our work relates to the use of ontologies to represent context. Bouquet et al. [2] introduced the notion of *contextual ontology* where the contents of an ontology are kept local and explicit mappings are used to associate the contents (e.g. concepts, roles, etc) of one ontology to the content of another. He addressed the fact that each ontology may represent its own local domain rather than a unique shared domain and provided the semantics of *local domains*. In line with this approach, we also assume that each ontology (representing a learning resource) used in the learning activity has its own local domain and interpretation function. Although correspondences between ontologies may be represented via bridge rules [6] or simple default rules where applicable, these may not always be known to the learner. The above assumptions are important in the construction of derivations whenever assumptions from one ontology can be combined to make inferences in another ontology.

The notion of *context space* addressed in [2] is similar to the idea upon which the *cognitive learning context* of a learner is based in this paper, namely that a context consists of a set of resources. However, the focus of our work is on the representation of the notions of cognitive ambiguity and inconsistency rather than on modeling the mappings of concepts between different ontologies. Of particular importance to us then are the relevance of the ontologies being used in the learning task and the plausible epistemic alternatives in case of information incompleteness.

3 Cognitive Learning Context, Ambiguity and Inconsistency

In this paper we represent each epistemology via its underlying ontology. In this project we use $OWL-DL$ as an ontology representation language because it is a decidable fragment of description logic and expressive enough to satisfy our need for the representation of concepts, roles and hierarchies that give rise to the type of arguments formalized in this work.

3.1 Ontology

An Ontology in this paper is described as a structure $\langle T, A\rangle$ where T denotes a DL TBox (i.e. a set of terminological) axioms and A denotes a DL ABox (i.e. a set of grounded assertions). Each ontology has its own signature consisting of a disjoint set of relation names, concept names and constant names of individuals. We denote the signature of an OWL ontology O_i by $Sig(O_i) \equiv R \cup C \cup N$, where R denotes the relation names, C the concept names and N the set of individual names. The interpretation I_i of $Sig(O_i)$ is the structure $\langle D_i, \cdot^{I_i}\rangle$ where D_i is the domain of the ontology and $\cdot^{I_i}$ is the interpretation function such that: $C^{I_i} \subseteq D_i$, $R^{I_i} \subseteq {D_i}^n$ (in OWL is $D_i \times D_i$). Assume a set of resources $R_1, \ldots, R_n$ underpinned by a set of ontologies $E_1, \ldots, E_n$, respectively. Then we define the local learning context of a learner as follows:

3.2 Cognitive Learning Context

The local reasoning learning context of a learner L is defined as a structure

$$\Upsilon \equiv \langle E, W, \delta, \eta, sit, \sigma, I^*\rangle$$

where
$E = \{O_1, \ldots O_n\}$ is the set of possibly inconsistent ontologies referred to by the learner, where each O_i has its own vocabulary. W is a non-empty set of epistemic alternatives (possible worlds). A subset of terminological axioms and assertions selected from an ontology O_i is referred to as *sub-ontology* and is denoted by $Sub(O_i)$. The function δ associates each sub-ontology (set of statements selected from an ontology), $Sub(O_i)$, to a set of *compatible epistemic alternatives* in W. The phrase *compatible epistemic alternatives* refers to the possible epistemic states such that if the associated resource contains p or infers p, each of the epistemic alternatives also infers or entails p. We assume that the *focus, η,* of the learning activity is either is a proposition. Also sit denotes the actual situation the learner is in during the learning activity. The relevance function σ accesses subsets of ontologies that are relevant to the focus of the epistemic activity. Therefore σ maps an ontology and a proposition denoting the focus of the learning activity to a set of propositions of the ontology (sub-ontology) relevant to the focus. Assume that Φ denotes the set of all possible propositions and SO_i is the set of sub-ontologies that can be created out of each ontology. Then $\sigma : O_i \times \Phi \rightarrow SO_i$. The axiomatization of relevance is currently under development and not provided in this paper. I^* is an interpretation function on the joint vocabulary

$$V = \bigcup_{i=1}^{n} Sig(O_i)$$

such that:

1. For each axiom $\alpha_i \in T_i$, $\alpha^{I^*} = \alpha^{I_i}$
2. For each axiom $\alpha_j \not\in T_i$ and $\alpha_j \in T_j$ there exist $Sub(O_i)$-compatible subsets of epistemological alternatives $W_1 \subseteq W$ and $W_2 \subseteq W$ such that $W_1 \models \alpha_j$ and $W_2 \not\models \alpha_j$. Note that $W \models \phi$ if and only if for each world $w \in W$ we have $w \models \phi$ for any formula ϕ.

Using the above definition of the cognitive state of a learner, we are now able to discuss the cognitive states of ambiguity, ignorance and inconsistency.

3.3 Cognitive Ambiguity

Intuitively, a learner reaches a cognitive state of ambiguity whenever she has access to more than one plausible epistemic alternatives and the learner is unable to choose one. The Oxford English Dictionary defines ambiguity as: *wavering of opinion, hesitation, doubt, uncertainty, as to one's course, or, capable of being understood in two or more ways, or, doubtful, questionable, indistinct, obscure, not clearly defined and lastly, admitting more than one interpretation or explanation; of double meaning or several possible meanings* (in [4]). The notion of ambiguity in our case refers to the interpretation of incompleteness of information contained in learning resources by the learner. We assume that a learner becomes aware of the incompleteness of a learning resource when she compares it with her background knowledge or with another resource. The set of resources relevant to the subject of the learning activity may change in each situation according to the focus of the learning activity. Assume a unified signature Σ which consists of the union of all the signatures $Sig(O_i^{'})$ (defined as above). To simplify matters, we assume that any two identical non-logical symbols of two resources R_1 and R_2 are considered the same unless there is evidence to the contrary. The following defaults enable us to draw inferences based on default correspondences between identical symbols across ontologies.

$$\frac{[R_1 : C(x)] : [R_2 : C(x)] \leftrightarrow [R_1 : C(x)]}{[R_2 : C(x)]} \tag{1}$$

Default rule 1 states that if there is no inference inconsistent to $[R_2 : C(x)] \leftrightarrow [R_1 : C(x)]$ in R_2 then $R_2 : C(x)$ can be asserted in R_2. A similar default inference rule is used for relations between concepts and names of individuals.

$$\frac{[R_1 : R(x, y)] : [R_2 : R(x, y)] \leftrightarrow [R_1 : R(x, y)]}{[R_2 : R(x, y)]} \tag{2}$$

The biconditional used in the inference rules aims to maintain consistency with mappings of terms between different vocabularies. For example, when two people (P_1 and P_2 say) are viewing a scene from opposite sites then $P_1 : right \leftrightarrow P_2 : left$. Further assume that $P_i : right \rightarrow \neg P_i : left$ holds for each person. Then obviously, it is inconsistent to assume that $P_1 : right \leftrightarrow P_2 : right$. Note that the intended meaning of the notions of $P_i : right$ and $P_i : left$ for each $i \in \{1, 2\}$ is independent of the situation of P_i. However the actual assignment of terms is dependent on their situation.

Let us consider $O_1^{'} = Sub(O_1)$ and $O_2^{'} = Sub(O_2)$ of two different ontologies O_1 and O_2 as above. We use the notation $[O_i^{'} \setminus O_j^{'}]_T$ to denote all the terminological axioms of $O_i^{'}$ that are not included in $O_j^{'}$. For example, assume $O_1^{'} = \{ENC \sqsubseteq OOL, VB \sqsubseteq ENC\}$ and $O_2^{'} = \{INH \sqsubseteq OOL, VB \sqsubseteq INH\}$. Then, $[O_1^{'} \setminus O_2^{'}]_T = \{ENC \sqsubseteq OOL\}$.

Now assume that O_2 does not include any axiom associating the concepts of ENC and INH. If there was an association (e.g. $Disjoint(ENC, INH)$) then $\{ENC \sqsubseteq OOL\}$ might not be a possibility at all. However, since there is no information associating the two concepts in O_2, then $O_2^{'}$ is compatible with two sets of epistemic alternatives: the first set is the one in which the axiom holds and the second in which it doesn't. Using this approach we define cognitive ambiguity as the situation in which the learner can see possible epistemic alternatives of a resource which are compatible with the resource but inconsistent with each other.

3.4 Cognitive Inconsistency (Confusion)

Intuitively, we assume that Cognitive inconsistency arises when in the actual world of the learner, information about a topic is conflicting. This is evidenced by conflicting information from different resources. It is different from cognitive ambiguity in that cognitive ambiguity appears as a consequence of possible epistemic alternatives due to lack of knowledge. The cognitive state of inconsistency can be explained via the existence of conflicting arguments from different learning resources. The cognitive state of ambiguity arises from the *possibility* of inconsistency between incomplete resources due to absence of information to the contrary.

In the next section we argue that the method of argumentation can be used to determine inconsistencies between conflicts or ambiguities between ontologies. Inconsistencies are determined via the derivation of refuting arguments from different resources related to the *focus* of the learning activity.

4 Syllogistic Arguments and Ontological Taxonomic Relations.

The process of argumentation is important during interaction with a learner in order to determine discrepancies in conceptualizations of the learner and the tutor or the learner and the learning resources related to the focus of the learning activity. It is also important for the recognition of differences or inconsistencies in ontologies automatically. In the next section we discuss the formalization of two types of arguments that can be inferred from ontologies, namely syllogisms and arguments about necessary and jointly sufficient features associated to the definition of concepts.

An Ontology may include one or more hierarchies of concepts that can be used to infer categorical statements.

4.1 Concept hierarchy

A *concept hierarchy* is a structure $\mathcal{H} = \langle C_{\mathcal{H}}, R_{\mathcal{H}} \rangle$ where $C_{\mathcal{H}}$ is a set of concepts, st. $C_{\mathcal{H}} \subseteq C$ of the ontology O, and $R_{\mathcal{H}} = \{Disjoint, SubclassOf, Intersects, ComplementOf\}$ and every concept in $C_{\mathcal{H}}$ is associated with another concept via a relation in $R_{\mathcal{H}}$. OWL-DL provides all of relations in $R_{\mathcal{H}}$ and therefore a hierarchy can be represented in it. We are interested in those interpretations of a hierarchy that satisfy all the taxonomic relations within the hierarchy. A model, $\mathcal{M}_{\mathcal{H}}$ of $\mathcal{H}$ is an interpretation I of $\mathcal{H}$ where all the taxonomic relations in $R_{\mathcal{H}}$ are satisfied. The semantics of ontological primitives used in a taxonomic hierarchy are as follows: If $C_1, C_2 \in C_{\mathcal{H}}$ then $subclassOf(C_1, C_2)$ if and only if $C_1^I \subseteq C_2^I$, $Disjoint(C_1, C_2)$ if and only if $C_1^I \cap C_2^I = \emptyset$, $Intersects(C_1, C_2)$ if and only if $C_1^I \cap C_2^I \neq \emptyset$, and $ComplementOf(C_1) = \mathcal{U} \setminus C_1^I$. Obviously, $\mathcal{M}_{\mathcal{H}}$ is a sub-model of $\mathcal{M}$ and therefore any entailment of $\mathcal{M}_{\mathcal{H}}$ is an entailment of $\mathcal{M}$.

The above set-theoretic semantics of taxonomic primitives are used to represent syllogisms and arguments from necessary and jointly sufficient properties for the representation of concepts. Bennett[1] showed (see 4.4) that set-equations can be translated to equivalent propositional formulae subject to certain constraints. Consequently syllogisms can be tested for their validity against a propositional theorem prover.

4.2 Categorical statements

Generalized statements of the form: *Every X is a Y* or *Every X has the property of Y* can be inferred from taxonomic hierarchies and can be combined to form *syllogistic arguments*. These statements are referred to as *categorical statements*. A syllogism [11] is a particular type of argument that has two premises and a single conclusion and all statements in it are categorical propositions.

4.2.1 Individuals

In ontologies, a distinction is made between individuals and classes. In the consequent we argue that the set equations that can be used to represent ontological primitives can be translated to propositional logic formulae that can be used to test validity of arguments. To simplify computation and to prove whether an individual belongs to a class (or a refutation that an individual belongs to a class) we represent individuals as singular sets consisting of that individual only. In this way we treat individuals as classes during inference. An ontology may include one or more hierarchies of concepts that can be used to infer syllogisms.

4.2.2 Syllogisms

Syllogisms form a particular type of arguments that are constructed from generalized statements (categorical statements). There are four basic categorical statements which can be combined to produce 64 patterns of Syllogistic Arguments. These are shown below together with the corresponding ontological primitives:

Categorical Statement	**Ontological Primitive**
Every S is a P	SubclassOf(S, P)
No S is a P	SubclassOf(S, ComplementOf(P))
Some S is a P	Intersects(S, P)
Some S is not P	Intersects(S, ComplementOf(P))

However, only 27 of them are valid syllogisms. This suggests the need to check the validity of syllogisms constructed from ontologies and exchanged during interaction with the learner.

4.3 Necessary and Sufficiency Conditions Arguments.

The classical view of the representation of concepts states that the features representing a concept are *singly necessary* and *jointly sufficient* to define a concept. In line with the above view we propose the following definitions for the *necessary* and *jointly sufficient* features representing a concept.

4.3.1 Necessary Features for the Representation of a Concept

Intuitively, a feature ϕ is *singly necessary* for the definition of C if and only if existence of C implies existence of ϕ. Assume a feature ϕ. We define a set Φ consisting of all individuals of the domain which have property ϕ (e.g. via the onProperty restriction in OWL-DL). Then, ϕ is a necessary property for the representation of concept C if and only if $C^I \subseteq \Phi$. An example of a refutal to the assumption that ϕ is a necessary feature for C is the derivation of an individual that belongs to C and to a class disjoint with Φ.

4.3.2 Jointly Sufficient Features for the Representation of a Concept

Let $\{\Phi_1, \ldots, \Phi_n\}$ represent the set of concepts corresponding to features $\phi_1, ..., \phi_n$ respectively. Then $\phi_1, ..., \phi_n$ are jointly sufficient for the representation of concept C if and only if $\{\Phi_1 \cap, \ldots, \cap\Phi_n\} \subseteq C^I$. An example of a refutal (i.e. an attacking argument) to the above assumption would be the existence of an individual that has these properties but does not belong to C. Conflicting arguments about these notions can be used to differentiate concept definitions between different ontologies.

4.4 Bennett's theory

Bennett [1] proved that set equations can be translated to equivalent universal equations which can in turn be converted to propositional logic formulae and can be tested for their validity with a Gentzen theorem prover. The theorem expressing the correspondence between set equations to universal equations is called *classical entailment correspondence theorem.* Although his theory was intended primarily for reasoning with mereological relations it is applicable in our case for reasoning with the type of arguments described above. This is because the mereological relations being represented using this theory closely resemble the set-theoretic semantics attributed to the ontological primitives describing associations between concepts in ontologies. Based on Bennett's *classical entailment correspondence theorem* we were able via a small adaptation to derive a *taxonomic entailment correspondence theorem* which is very similar to the theorem described above but concerns hierarchical relations. This is stated as follows:

$$M_H \models \phi \, if \; and \; only \; if \; M_{C+} \models \tau = \mathcal{U} \tag{3}$$

where $\mathcal{U}$ is the universe of discourse. The *Taxonomic entailment correspondence theorem* shows the correspondence between taxonomic relations and universal set equations. As in [1], we can avoid unintended taxonomic relations captured during the translation from universal set equations to propositional formulae, by the use of *entailment constraints* [1]. In order to avoid excessive technical details which are beyond the scope of this paper, we focus on the use of the above theory to our work. In particular, it can be used to convert each categorical statement in a syllogistic argument to its corresponding propositional form which can be tested efficiently against a propositional theorem prover.

4.5 Conflicts between arguments

Intuitively, a set of arguments consists of a minimal set of premises (here categorical statements) used in the derivation of a claim. In this paper we focus on strict arguments that are inferred via the classical entailment relation. Two arguments conflict with each other (attack) if either (i) the claim of one argument is inconsistent with the claim of the other argument (i.e. *rebutal* [10]) or (ii) the claim of one argument is inconsistent with one of the other premises of the other argument (i.e. *undercutting* [10]) or (iii) one argument's premises are inconsistent with the other argument's premises. Since a syllogism is defined entirely in terms of categorical expressions then two syllogistic arguments conflict each other if any expression in one argument is inconsistent with an expression in the other argument.

5 Conclusion and Future Work

In this paper we introduced the notion of cognitive learning context that refers to multiple and possibly inconsistent ontologies. Differences in ontologies can be identified via arguments that can be inferred from consistent subsets of ontologies. We show that syllogistic arguments can be inferred from ontological primitives and we represent the necessary and sufficient properties of concepts used to argue in learning situations. Current work focuses on the axiomatization of relevance of modules, the argumentation theory and its use in the representation of the cognitive state of the learner.

REFERENCES

[1] Brandon Bennett, *Logical Representations for Automated Reasoning about Spatial Relationships*, Ph.D. dissertation, School of Computer Studies, The University of Leeds, October 1997.

[2] Paolo Bouquet, Fausto Giunchiglia, and Frank van Harmelen, 'Contextualizing ontologies', Dit-04-013, Department of Information and Communication Technology, University of Trento, (February 2004).

[3] Paolo Bouquet, Luciano Serafini, and Heiko Stoermer, 'Introducing Context into RDF Knowledge Bases', in *Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop*. CEUR Workshop Proceedings, (December 2005).

[4] Michelle Dalmau, 'Ambiguity as a conceptual framework for design', Technical report.

[5] R. Fagin and J. Halpern, 'Awareness and limited reasoning', *Artificial Intelligence*, **34**, 39–76, (1988).

[6] Chiara Ghidini and Fausto Giunchiglia, 'Local models semantics, or contextual reasoning = locality + compatibility', *Artificial Intelligence*, **127**, 221–259, (2001).

[7] R. Guha, R. McCool, and R. Fikes, 'Contexts for the semantic web', in *Proceedings of the International Semantic web Conference*, eds., Sheila A. McIlraith, Dimitris Plexoudakis, and Frank van Harmelen. Lecture Notes in Computer Science, (November 2004).

[8] Gert-Jan C. Lokhorst, 'Counting the minds of split-brain patients', *Logique and Analyse*, 315–324, (1996).

[9] T. Martins, A, M. Pequeno, and T. Pequeno, 'A multiple worlds semantics for a paraconsistent nonmonotonic logic', in *lecture notes in pure and applied mathematics, paraconsistency: the logical way to the inconsistent*, eds., Walter Carnielli and Marcelo Coniglio, volume 228, pp. 187–209, (2002).

[10] Henry Prakken, 'On dialogue systems with speech acts, arguments and counterarguments'.

[11] D. Walton, *Foundamentals of Critical Argumentation*, Cambridge University Press, 2006.

A Sequence-based Ontology Matching Approach

Alsayed Algergawy, Eike Schallehn and **Gunter Saake**[1]

Abstract. The recent growing of the Semantic Web requires the need to cope with highly semantic heterogeneities among available ontologies. Ontology matching techniques aim to tackle this problem by establishing correspondences between ontologies' elements. An intricate obstacle faces the ontology matching problem is its scalability against large number and large-scale ontologies. To tackle these challenges, in this paper, we propose a new matching framework based on Prüfer sequences. The proposed approach is applicable for matching a database of XML trees . Our approach is based on the representation of XML ontologies as sequences of labels and numbers by the Prüfer's method that constructs a one-to-one correspondence between schema ontologies and sequences. We capture ontology tree semantic information in Label Prüfer Sequences (LPS) and ontology tree structural information in Number Prüfer Sequences (NPS). Then, we develop a new structural matching algorithm exploiting both LPS and NPS. Our Experimental results demonstrate the performance benefits of the proposed approach.

1 Introduction

The Semantic Web (SW) is evolving towards an open, dynamic, distributed, and heterogenous environments. The core of the SW is ontology, which is used to represent our conceptualizations. The Semantic Web puts the onus of ontology creation on the user by providing common ontology languages such as XML, RDF(S) and OWL. However, ontologies defined by different applications usually describe their domains in different terminologies, even they cover the same domain. In order to support ontology-based information integration, tools and mechanisms are needed to resolve the semantic heterogeneity problem and align terms in different ontologies. Ontology matching plays the central role in these approaches. *Ontology matching* is the task of identifying correspondences among elements of two ontologies [5, 20, 15].

Due to the complexity of ontology/schema matching, it was mostly performed manually by a human expert. However, manual reconciliation tends to be a slow and inefficient process especially in large-scale and dynamic environments such as the Semantic Web. Therefore, the need for automatic semantic schema matching has become essential. Consequently, many ontology/schema matching systems have been developed for automating the matching process, such as Cupid [17], COMA [6], Similarity Flooding [18], LSD [7], BTreeMatch [12], Spicy [2], GLUE [8, 9], OntoBuilder [21], QOM [13], and S-Match [14]. Moreover, most of these approaches have been developed and tested using small-scale schemas. The primary focus was on matching effectiveness. Unfortunately, the effectiveness of automatic match techniques typically decreases for larger schemas. In particular, matching the complete input schemas may lead not only to long execution times, but also poor quality due to the large search space. Therefore, the need for efficient and effective algorithms has been arisen.

Recently, matching algorithms are introduced to focus on matching large-scale and large number schemas and ontologies, i.e. considering the efficiency aspect of matching algorithms, such as COMA++ [1, 11], QOM [13], Bellflower [23], and PORSCHE [22]. Most of these systems rely heavily on either rule-based approaches or learner-based approaches. In the rule-based systems, schemas to be matched are represented as schema trees or schema graphs which in turn requires traversing these trees (or graphs) many times. On the other hand, learning-based systems need much pre-effort to train its learners. As a consequence, especially in large-scale schemas and dynamic environments, matching performance declines radically.

Motivated by the above challenges and by the fact that the most prominent feature for an XML schema is its hierarchical structure, in this paper, we propose a novel approach for matching XML schemas. In particular, we develop and implement the *XPrüM* system, which consists mainly of two parts —schema preparation and schema matching. Schemas to be matched are first parsed and represented internally using rooted ordered labeled trees, called schema trees. Then, we construct a Prüfer sequence for each schema tree. Prüfer sequences construct a one-to-one correspondence between schema trees and sequences. We capture schema tree semantic information in Label Prüfer Sequences (LPS) and schema tree structural information in Number Prüfer Sequences (NPS). LPS is exploited by a linguistic matcher to compute terminological similarities between schemas elements.

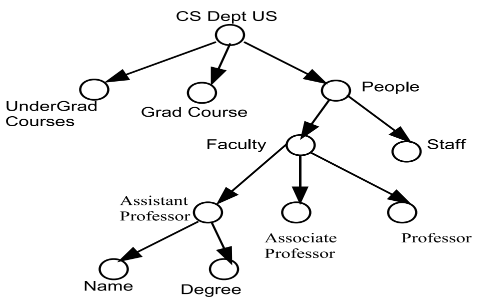
Linguistic matching techniques may provide false positive matches. Structural matching is used to correct such matches based on structural contexts of schema elements. Structural matching relies on the notion of node[2] context. In this paper, we distinguish between three types of node contexts depending on its location in the schema tree. These types are *child context, leaf context* and *ancestor context*. We exploit the number sequence representation of the schema tree to extract node contexts for each tree node in an efficient way. Then, for each node context, we apply its associated algorithm. For example, the leaf context similarity between two nodes is measured by extracting leaf context for each node as a gap vector. Then, we apply the cosine measure between two gap node vectors. Other context similarity measures are determined similarly.

By representing schema trees as Prüfer Sequences we need to traverse these trees only once to construct these sequences. Then, we develop a novel structural matching algorithm which captures semantic information existing in label sequences and structural information embedded in number sequences.

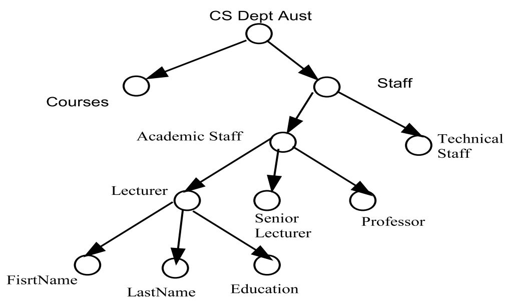
The paper is organized as follows: Section 2 introduces basic concepts and definitions. Section 3 describes our proposed approach. Section 4 presents experimental results. Section 5 gives concluding

[1] Magdeburg University, Germany, email: {alshahat, eike, saake}@ovgu.de

[2] In this paper the terms node and element are interchangeable



(a) Ontology Tree OT1



(b) Ontology Tree OT2

Figure 1: Computer science department ontologies

remarks and our proposed future work.

2 Preliminaries

The semantics conveyed by ontologies can be as simple as a database schema or as complex as the background knowledge in a knowledge base. Our proposed approach is concerning only with the first case, which gives the assumption that ontologies in this paper are represented as XML schemas.

XML schemas can be modeled as rooted, ordered, labeled trees $T = (N, E, Lab)$, called ontology trees. N is the set of tree nodes, representing different XML schema components. E is the set of edges, representing the relationships between schema components. Lab is the set of node labels, representing properties of each node. We categorize nodes into *atomic nodes*, which have no outgoing edges and represent leaf nodes and *complex nodes*, which are the internal nodes in the ontology tree.

2.1 Motivations

To the best of our knowledge no work for matching XML schemas exists which is based on *Prüer Sequences*. Most of existing matching systems rely heavily on matching schema trees/graphs. In addition, these systems perform poorly when dealing with large-scale ontologies. These shortcomings have motivated us to develop an XML schema matching system based on Prüfer Sequences, called *XPrüM*.

Our proposed system is also a schema-based approach. However, each schema tree is represented using two sequences which capture both tree semantic information and tree structure. This leads to a space efficient representation and an efficient time response when compared to the state-of-the-art systems.

2.2 A Matching Example

To describe the operation of our approach, we use the example found in [9]. It describes two XML ontologies, shown in Figure 1 (a and b), that represent organization in universities from different countries and have been widely used in the literature. The task is to discover semantic correspondences between two schemas' elements.

3 The Proposed Approach

In this section, we shall describe the core parts of the XPrüM system. As shown in Fig.2, it has two main parts: *ontology preparation* and *ontology matching*. First, ontologies are parsed using a SAX parser[3]

[3] http://www.saxproject.org

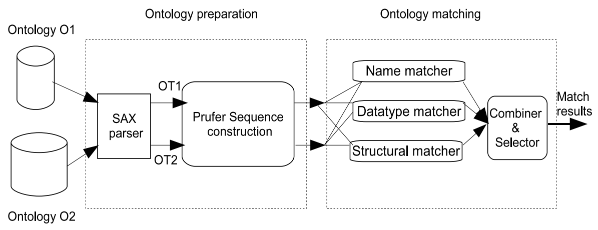


Figure 2: Matching Process Phases

and represented internally as ontology trees. Then, using the Prüfer sequence method, we extract both label sequences and number sequences. The ontology matching part discovers the set of matches between two ontologies employing both sequences.

3.1 Prüfer Sequences Construction

We now describe the tree sequence representation method, which provides a bijection between ordered, labeled trees and sequences. This representation is inspired from classical *Prüfer sequences* [19] and particularly from what is called *Consolidated Prüfer Sequence CPS* proposed in [24].

CPS of an ontology tree OT consists of two sequences, Number Prüfer Sequence *NPS* and Label Prüfer Sequence *LPS*. They are constructed by doing a *post-order traversal* that tags each node in the ontology tree with a unique traversal number. NPS is then constructed iteratively by removing the node with the smallest traversal number and appending its parent node number to the already structured partial sequence. LPS is constructed similarly but by taken the node labels of deleted nodes instead of their parent node numbers. Both NPS and LPS convey completely different but complementary information—NPS that is constructed from unique post-order traversal numbers gives wealthy tree structure information and LPS gives the labels for tree nodes. CPS representation thus provides a bijection between ordered, labeled trees and sequences. Therefore, CPS = (NPS, LPS) uniquely represents a rooted, ordered, labeled tree, where each entry in the CPS corresponds to an edge in the schema tree. For more details see [24].

Example 1. Consider ontology trees *OT1* and *OT2* shown in Figure 3, each node is associated with its *OID* and its post order number. Table 1 illustrates CPS for $OT1$ and $OT2$. For example, CPS of $OT1$ can be written as the $NPS(OT1)$= 11 11 5 5 8 8 8 10 10 11 -, and the $LPS(OT1).name$= *UnderGrad Courses, Grad Courses,*

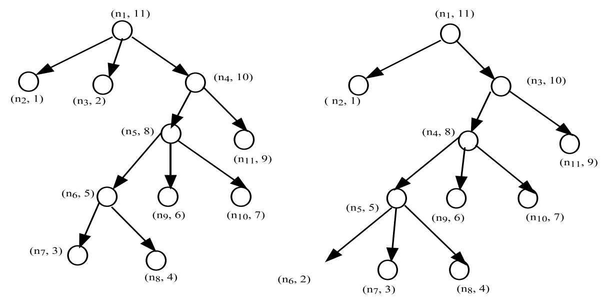


Figure 3: Nodes OIDs and corresponding post-order numbers

, Name, Degree, Assistant Professor, Associate Professor, Professor, Faculty, Staff, People, CS Dept US.

Table 1: CPS of Ontology Trees

Schema Tree ST1				Schema Tree ST2			
NPS	*LPS*			*NPS*	*LPS*		
	OID	*name*	*type/data type*		*OID*	*name*	*type/data type*
11	n_2	UnderGrad Courses	element/string	11	n_2	Courses	element/string
11	n_3	Grad Courses	element/string	5	n_6	FirstName	element/string
5	n_7	Name	element/string	5	n_7	LastName	element/string
5	n_8	Degree	element/string	5	n_8	Education	element/string
8	n_6	Assistant Professor	element/-	8	n_5	Lecturer	element/-
8	n_9	Associate Professor	element/string	8	n_9	SeniorLecturer	element/string
8	n_{10}	Professor	element/string	8	n_{10}	Professor	element/string
10	n_5	faculty	element/-	10	n_4	AcademicStaff	element/-
10	n_{11}	Staff	element/string	10	n_{11}	TecnicalStaff	element/string
11	n_4	People	element/-	11	n_3	Staff	element/-
-	n_1	CS Dept US	element/-	-	n_1	CS Dept Aust	element/-

3.1.1 CPS Properties

In the following, we list the structural properties behind CPS representation of ontology trees. If we construct a CPS=(NPS, LPS) from an ontology tree OT, we could classify these properties into:

- *Unary Properties*: for every node n_i has a postorder number *k*,
  1. *atomic node*: n_i is an atomic node iff $k \notin NPS$
  2. *complex node*: n_i is a complex node iff $k \in NPS$
  3. *root node*: n_i is the root node (n_{root}) iff $k = max(NPS)$, where *max* is a function which returns the maximum number in NPS.
- *Binary Properties*
  1. *edge relationship*: each entry $CPS_i = (k_i, LPS_i)$ represents an edge from the node whose post-order number is k_i to a node n_i = $LPS_i.OID$. This property shows both child and parent relationships. This means that the node $n_i = LPS_i.OID$ is an immediate child for the node whose post-order number k_i.
  2. *sibling relationship*: $\forall$ two entries $CPS_i = (k_i, LPS_i)$ and $CPS_j = (k_j, LPS_j)$, the two nodes $n_i = LPS_i.OID$ and $n_j = LPS_j.OID$ are two sibling nodes iff $k_i = K_j$.

3.2 Matching Algorithms

Ontology matching algorithms operate on the sequential representation of two ontology trees *ST1* and *ST2* and discover semantic correspondences between them. Generally speaking, the process of ontology matching is performed, as shown in Figure 2, in two phases—*element matchers* and *combiner & selector*.

First, a degree of similarity is computed automatically for all element pairs using the element matcher phase. Recent empirical analysis shows that there is no single dominant element matcher that performs best, regardless of the data model and application domain [10]. As a result, we should exploit different kinds of element matchers. In our approach, we make use of *name matcher*; to exploit elements'names, *datatype matcher*; to exploit elements' types/datatypes, and *structural matcher*; to exploit elements' structural contexts. After a degree of similarity is computed, how to combine different similarities from different element matchers and select top-K mappings are addressed in the second phase

3.2.1 Name Matcher

The aim of this phase is to obtain an initial matching between elements of two ontology trees based on the similarity of their names. To compute linguistic similarity between two elements'names s_1 and s_2, we use the following three similarity measures. The first one is $sim_{edit}(s_1, s_2) = \frac{max(|s_1|,|s_2|) - editDistance(s_1,s_2)}{max(|s_1|,|s_2|)}$ where *editDistance(*s_1*,* s_2*)* is the minimum number of character insertion and deletion operations needed to transform one string to the other. The second similarity measure is based on the number of different trigrams in the two strings: $sim_{tri}(s_1, s_2) = \frac{2 \times |tri(s_1) \bigcap tri(s_2)|}{|tri(s_1)| + |tri(s_2)|}$ where *tri(*s_1*)* is the set of trigrams in s_1. The third similarity measure is based on Jaro-Winkler distance, which is given by $sim_{jaro}(s_1, s_2) = \frac{1}{3} \times (\frac{m}{|s_1|} + \frac{m}{|s_2|} - \frac{m-t}{m})$ where *m* is the number of matching characters and *t* is the number of transpositions. The linguistic matching between two ontology tree nodes is computed as the combination (weighted sum) of the above three similarity values.

3.2.2 Datatype Compatibility

We propose the use of datatype compatibility to improve initial linguistic similarity. To this end, we make use of built-in XML datatypes hierarchy [4] in order to compute datatype compatibility coefficients. Based on XML schema datatype hierarchy, we build a datatype compatibility table as the one used in [17]. After computing datatype compatibility coefficients we can adjust linguistic similarity values. The result of the above process is an adjusted linguistic similarity matrix.

3.2.3 Structural Matcher

Our structural matching algorithm is motivated by the fact that the most prominent feature in an XML schema is its hierarchical structure. This matcher is based on the node context, which is reflected by its ancestors and its descendants. The descendants of an element include both its immediate children and the leaves of the subtrees rooted at the element. The immediate children reflect its basic structure, while the leaves reflect the element's content. In this paper, as in [16, 3], we consider three kinds of node contexts depending on its position in the ontology tree:

- *The child context* of a node n_i is defined as the set of its immediate children nodes including attributes and subelements. The child context of an atomic node is an empty set. Using the edge relationship property, we could identify immediate children of a complex node and their count. The number of immediate children of a non-leaf node from the *NPS* sequence is obtained by counting its post-order traversal number in the sequence, and then we could identify these children. For example, in *Example 1, consider OT1*,

[4] `http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/`

the post-order number of node n_4 is 10. This number repeats two times. This means that it has two immediate children $\{n_5, n_{11}\}$.

- *The leaf context* of a node n_i is defined as the set of leaf nodes of subtrees rooted at node n_i. We notice that nodes whose post-order numbers do not appear in the *NPS* sequence are atomic nodes; *CPS' atomic property*. From this notice and from the child context we could recursively obtain the leaf context for a certain node. For example, in *Example 1, consider OT1*, nodes $\{n_2, n_3, n_7, n_8, n_9, n_{10}, n_{11}\}$ are leaf nodes set. Node n_6 has two children n_7, n_8, which are leaves. Then they are the leaf context set of node n_6.
- *The ancestor context* of a node n_i is defined as the path extending from the root node to the node n_i. The ancestor of the root node is an empty path. For a non-atomic node, we obtain the ancestor context by scanning the *NPS* sequence from left to right and identifying the numbers which are greater than post-order number of the node until the first occurrence of the root node. While scanning from left to right, we ignore nodes whose post-order numbers are less than post-order numbers of already scanned nodes. For a leaf node, the ancestor context is the ancestor context of its parent union the parent itself. For example, in *Example 1, consider OT1*, the ancestor context of node n_5 (non-atomic node) is the path n_1/ n_4/ n_5. While the ancestor context of node n_9 (atomic node) is the path n_1/n$_4$/ n$_5$/ n$_9$.

Structural Context Similarity Algorithm Structural node context defined above relies on the notion of path and set. In order to compare two ancestor contexts, we essentially compare their corresponding paths. On the other hand, in order to compare two child contexts and/or leaf contexts, we need to compare their corresponding sets. In the following we describe how to compute the three structural context measures:

1. *Child Context Algorithm:* To obtain the child context similarity between two nodes, we compare the two child context sets for the two nodes. To this end, we first extract the child context set for each node. Second, we get the linguistic similarity between each pair of children in the two sets. Third, we select the matching pairs with maximum similarity values. And finally, we take the average of best similarity values.
2. *Leaf Context Algorithm:* Before we delve into details of computing leaf context similarity, we shall first introduce the notion of *gap* between two tree nodes.

Definition The gap between two nodes n_i and n_j in an ontology tree *OT* is defined as the difference between their post-order numbers.
To compute the leaf context similarity between two nodes, we compare their leaf context sets. To this end, first, we extract the leaf context set for each node. Second, we determine the gap between each node and its leaf context set. We call this vector the *gap vector*. Third, we apply the cosine measure between two vectors.

Example 2. For the two ontology trees shown in *Example 1*. The leaf context set of *OT1.*n_1 is *leaf_set* (n_1)=$\{n_2, n_3, n_7, n_8, n_9, n_{10}, n_{11}\}$ and the leaf context set of *OT2.*n_1 is *leaf_set* (n_1)=$\{n_2, n_6, n_7, n_8, n_9, n_{10}, n_{11}\}$. The gap vector of *ST1.*n_1 is v_1 =*gapvec(OT1.*n_1*)*=$\{$10,9,8,7,5,4,2$\}$ and the gap vector of *OT2.*n_1 is v_2 =*gapvec(OT2.*n_1*)*=$\{$10,9,8,7,5,4,2$\}$. The cosine measure CM of the two vectors gives *CM(*v_1,v_2*) =1.0*. Then the leaf context similarity between nodes *OT1.*n_1 and *OT2.*n_1 is 1.0.

3. *Ancestor Context Similarity:* The ancestor context similarity captures the similarity between two nodes based on their ancestor contexts. To compute the ancestor similarity between two nodes n_i and n_j, first we extract each ancestor context from the *CPS* sequence, say path P_i for n_i and path P_j for n_j. Second, we compare two paths. To compare two paths, we use three of four scores established in [4] and reused in [3]. These scores are combined to compute the similarity between two paths P_i and P_j *psim* as follows:

$$psim(P_i, P_j) = LCS_n(P_i, P_j) - \gamma GAPS(P_i, P_j) - \delta LD(P_i, P_j) \quad (1)$$

where γ and δ are positive parameters ranging from 0 to 1 that represent the comparative importance of each factor. The three scores are: (1)$LCS_n(P_i, P_j)$ used to measure longest common subsequences between two paths normalized by the length of the first path, (2) $GAPS(P_i, P_j)$ used to ensure that the occurrences of two paths'nodes are close to each other, and (3)$LD(P_i, P_j)$ used to give higher values to source paths whose lengths is similar to target paths.

Putting it all together: Our complete structural matching algorithm is as follows: The algorithm accepts *CPS(NPS, LPS)* for each schema tree and the linguistic similarity matrix as inputs and produces a structural similarity matrix. For each node pairs, context similarity is computed using child context, ancestor context, and leaf context. The three context values are then combined.

4 Experimental Evaluation

To implement the solution we have installed above algorithms using Java platform. We ran all our experiments on 2.4GHz Intel core2 processor with 2GB RAM running Windows XP. We shall describe the data sets used through evaluation and our experimental results.

4.1 Data Sets

We experimented with the data sets shown in Table 2. These data sets were obtained from[5] [6] [7]. We choose these data sets since they capture different characteristics in the numbers of nodes (schema size), their depth (the number of nodes nesting) and represent different application domains, see Table 2.

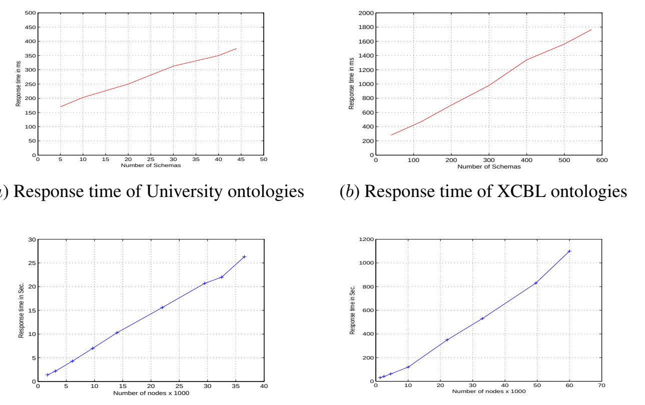
Table 2: Data set details

Domain	*No. of ontologies/nodes*	*Ontology size*
University	44/550	< 1KB
XCBL	570/3500	< 10 KB
OAGIS	4000/36000	<100 KB
OAGIS	100/65000	>100 KB

4.2 Measures for Match Performance

The *XPrüM* system considers both performance aspects —matching effectiveness and matching efficiency. However due to space limit and according the paper outline, we consider matching efficiency in more details. To this end, we use the response time as a function of the number of schemas and the number of nodes to measure matching efficiency.

[5] http://www.cs.toronto.edu/db/clio/testSchemas.html
[6] http://www.xcbl.com
[7] http://www.oagi.org



(a) Response time of University ontologies (b) Response time of XCBL ontologies

(c) Res. time of OAGIS ontology'nodes 1 (d) Res. time of OAGIS ontology'nodes 2

Figure 4: Performance analysis of XPrüM system with real world ontologies

4.3 Experimental Results

XPrüM Quality: To show matching effectiveness of our system, ontologies shown in Figure 1 have been used. XPrüM could discover 9 true positive matches among 11 and 2 false positive matches with F-measure of 81%. Compared to COMA++ [1] which could discover only 5 true positive matches and miss 6 false negative matches with F-measure of 62%, our system demonstrates better quality.

XPrüM Efficiency: Figure 4 shows that *XPrüM* achieves high scalability across all three domains. The system could identify and discover correspondences across 44 schemas including 550 nodes from the university domain in a time of 0.4 second, while the approach needs 1.8 seconds to match 570 schemas including approximately 3500 nodes from the XCBL domain. This demonstrates that *XPrüM* is scalable with large number of schemas. To demonstrates the system scalability with large-scale schemas, we performed two set of experiments. The first one is tested with the OAGIS domain whose schemas'sizes ranging between $10KB$ and $100KB$. Figure 4(c) shows that the system needs 26 seconds to match 4000 schemas containing 36000 nodes. The second set is performed also using the OAGIS domain containing 100 schemas whose sizes are greater than $100KB$. *XPrüM* needs more than 1000 seconds to match 65000 nodes as shown in Figure 4(d).

5 CONCLUSION

With the proliferation of the Semantic Web ontologies, the development of automatic techniques for ontology matching will be crucial to their success. In this paper, we have addressed an intricate problem associated to the ontology matching problem—matching scalability. To tackle this, we have proposed and implemented the *XPrüM* system, a hybrid matching algorithm to automatically discover semantic correspondences between XML schemas. The system starts with transforming schemas into ontology trees and then constructs a consolidated Prüfer sequence *CPS* for each schema tree which construct a one-to-one correspondence between schema trees and sequences. We capture schema tree semantic information in Label Prüfer Sequences and schema tree structural information in Number Prüfer Sequences.

Experimental results have shown that *XPrüM* has a high scalability with respect to large number and large-scale schemas. Moreover, it could preserve matching quality beside its scalability. *XPrüM* has other features including: it is almost automatic; it does not make use of any external dictionary; moreover, it is independent on data mode and application domain of matched schemas.

In our ongoing work, we should consider the second aspect of semantics conveyed by ontologies, i.e. modeling ontologies using RDF(s) or OWL, to deal with background knowledge. This helps us to apply the sequence-based approach on other applications and domains such as image matching and the Web service discovery.

REFERENCES

[1] D. Aumueller, H.H. Do, S. Massmann, and E. Rahm, 'Schema and ontology matching with COMA++', in *SIGMOD Conference*, (2005).

[2] A. Bonifati, G. Mecca, A. Pappalardo, and S. Raunich, 'The spicy project: A new approach to data matching', in *SEBD*. Turkey, (2006).

[3] A. Boukottaya and C. Vanoirbeek, 'Schema matching for transforming structured documents', in *DocEng'05*, pp. 101–110, (2005).

[4] D. Carmel, N. Efraty, G. Landau, Yo. Maarek, and Y. Mass, 'An extension of the vector space model for querying xml documents via xml fragments', *SIGIR Forum*, **36**(2), (2002).

[5] L. Ding, P.Kolari, Z. Ding, S. Avancha, T. Finin, and A. Joshi, 'Using ontologies in the semantic web: A survey', in *TR-CS-05-07*, (2005).

[6] H. H. Do and E. Rahm, 'COMA- A system for flexible combination of schema matching approaches', in *VLDB 2002*, pp. 610–621, (2002).

[7] A. Doan, 'Learning to map between structured representations of datag', in *Ph.D Thesis*. Washington University, (2002).

[8] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A.Halevy, 'Learning to match ontologies on the semantic web', *Knowledge and Information Systems*, **12**(4), 303–319, (2003).

[9] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, *Ontology matching: A machine learning approach*, Handbook on Ontologies, International Handbooks on Information Systems, 2004.

[10] C. Domshlak, A. Gal, and H. Roitman, 'Rank aggregation for automatic schema matching', *IEEE on KDE*, **19**(4), 538–553, (April 2007).

[11] C. Drumm, M. Schmitt, H.-H. Do, and E. Rahm, 'Quickmig - automatic schema matching for data migration projects', in *Proc. ACM CIKM07*. Portugal, (2007).

[12] F. Duchateau, Z. Bellahsene, and M. Roche, 'An indexing structure for automatic schema matching', in *SMDB Workshop*. Turkey, (2007).

[13] M. Ehrig and S. Staab, 'QOM - quick ontology mapping', in *International Semantic Web Conference 2004:*, pp. 683–697, (2004).

[14] F. Giunchiglia, M. Yatskevich, and P. Shvaiko, 'Semantic matching: Algorithms and implementation', *Journal on Data Semantics*, **9**, 1–38, (2007).

[15] Y. Kalfoglou and M. Schorlemme, 'Ontology mapping: the state of the art', *The Knowledge Engineering Review*, **18**(1), 1–31, (2003).

[16] M. L. Lee, L. Yang, W. Hsu, and X. Yang, 'Xclust: Clustering XML schemas for effective integration', in *CIKM'02*, pp. 63–74, (2002).

[17] J. Madhavan, P. Bernstein, and E. Rahm, 'Generic schema matching with cupid', in *VLDB 2001*, pp. 49–58. Roma, Italy, (2001).

[18] S. Melnik, H. Garcia-Molina, and E. Rahm, 'Similarity flooding: A versatile graph matching algorithm and its application to schema matching', in *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, (2002).

[19] H. Prufer, 'Neuer beweis eines satzes uber permutationen', *Archiv für Mathematik und Physik*, **27**, 142–144, (1918).

[20] E. Rahm and P. Bernstein, 'A survey of approaches to automatic schema matching', *VLDB Journal*, **10**(4), 334–350, (2001).

[21] H. Roitman and A. Gal, 'Ontobuilder: Fully automatic extraction and consolidation of ontologies from Web sources using sequence semantics', in *EDBT 2006 Workshops*, (2006).

[22] K. Saleem, Z. Bellahsene, and E. Hunt, 'PORSCHE: Performance oriented schema mediation', *Accepted for publication in Information Systems Journal*, (2008).

[23] M. Smiljanic, *XML Schema Matching Balancing Efficiency and Effectiveness by means of Clustering*, Ph.D. dissertation, Twente University, 2006.

[24] S. Tatikonda, S. Parthasarathy, and M. Goyder, 'LCS-trim: Dynamic programming meets XML indexing and querying', in *VLDB'07*, pp. 63–74, (2007).

Semantic Alignment of Context-Goal Ontologies

Mellal-Labidi Nacima [1] and **Dapoigny Richard**[2] and **Foulloy Laurent**[3]

Abstract. Several distributed systems need to inter-operate and exchange information. Ontologies are gained the popularity in AI community as a means for enriching the description of information and make their context more explicit. Thus, to enable Interoperability between systems, it is necessary to align ontologies describing them in a sound manner. Our main interest is focused on ontologies describing systems functionalities. We treat these lasts as goals to achieve. In general, a goal is related to the realization of an action in a particular context. Therefore, we call ontologies describing goals and their context Context-Goal [4] Ontologies. Most of the methodologies proposed to reach interoperability are semi automatic, they are based on probabilities or statistics and not on mathematical models. The purpose of this paper is to investigate an approach where the alignment of C-G Ontologies achieves an automatic and semantic interoperability between distributed systems based on a mathematical model "Information Flow".

1 Background and Outlines

The exponential growth of information and resources exchanged between different systems increases the rate of heterogeneous information and makes their understanding and analysis very difficult. A crucial problem arising from this heterogeneity concerns the preservation of the meaning (sense) of the exchanged information. This is what we call the *Semantic Interoperability*. A definition is commonly accepted for semantic interoperability: *it gives meaning to the exchanged information and ensures that this sense is common in all systems between which exchanges must be done* [9] [18]. Therefore, distributed systems may combine the received information with the local one and treat the whole in a consistent way.

To ensure semantic interoperability, the exchanged information between systems must be described in a formal structure preserving its semantics. The great challenge is omnipresent in the knowledge Engineering field, where the proposed methodologies and techniques collect, identify, analyze, organize and share knowledge between different organizations. Among these techniques, *ontologies* are gained the popularity in AI community as a means for enriching the description of information and make their context more explicit. They represent an efficient and promising way to implement this is through the use of Ontologies, an explicit specification of conceptualization [8].

The semantic interoperability needs the use of methodologies which establish semantically links between the services provided by the communicating entities of the distributed system. In literature, discovering these links is called *ontologies alignment* , it aims to find connections between concepts belonging to different ontologies within a single application.

After a careful look at the different theories related to these topics, such as the MAFRA framework developed for the mapping of distributed ontologies [14]. Using the Bridge notion, MAFRA allows to create semantic relations between two (source and target) ontologies and apply such relations in translating source ontology instances into target ontology instances.

GLUE is another framework. It is a system that employs learning techniques to semi-automatically create semantic mappings between ontologies[7]. PROMPT is a tool for merging ontologies.

In [11], the approach mainly builds on the IF-Map method to map ontologies in the domain of computer science departments from five UK universities. Their method is also complemented by harvesting mechanisms for acquiring ontologies, translators for processing different ontology representation formalisms and APIs for web enabled access of the generated mappings. In [16], first-order model theory are investigated to formalize and automatize the issues arising with semantic interoperability for which they focused on particular understandings of semantics. But it would be desirable to provide a theoretical framework that accommodates various different understandings of semantics depending on the semantic interoperability. Authors made the first step towards semantic integration by proposing a mathematically sound application of channel theory to enable semantic interoperability of separate ontologies representing similar domains. In general, these works present some insufficiencies. They are semi-automatic (MAFRA, IF-MAP), or centered on probabilities (GLUE) or based on syntactic similarities (PROMPT). For that, it is useful to develop applications in order to automat the ontology alignment.

In the context of distributed systems, the communicating subsystems differ, generally, by a set of services that each one provides to the other, in order to realize global goals. These goals are directly related to the provided services : accomplish a service is done with an objective and achieve a goal is also performing a particular service. In this spirit, several researches are interested in goal notion and in its representation [5], [19], [10], [17] and [2]. These works have proved the effectiveness of system modeling using goals. For this reason, we are interested in our research to the goal oriented modeling. The majority of these works are centered on the description of the functionalities of systems and their components. They have denoted a functionality of a component as a " verb+noun " style for representing the component's activities (or actions) and its operands which needs an ontological schema for functional knowledge which specifies not only the data structure but also the conceptual viewpoint for capturing the target world [12]. Following this approach, we associate with each goal some possible actions (at least one) in order to fulfill the intended goal. At this level, there is a lack of formal terminology [13]. To remedy this problem and represent goal semantics

[1] University of LARBI-BEN-MHIDI (Algeria) and University of Savoie (FRANCE), email: nacima.mellal@univ-savoie.fr
[2] University of Savoie, France
[3] University of Savoie, France
[4] we abbreviate Context-Goal as C-G

formally, our approach uses the linguistical aspect based on the principle of linking names to verbs via lexical relations. For example, the structure associated with the name *Temperature* can contain verbs: to measure, to reduce, to increase .. These verbs express the intention or the function related to this name. More precisely, if we analyze two goals "to measure the temperature of a room " or "to measure the temperature of a human being ", we note that the action is always the same but its context is different. This brings us to introduce the concept *Context of a goal*. Our work is focused around two issues that are, on the one hand, the representation and reasoning on goals and, on the other hand, the resolution of semantic interoperability.

The present paper is divided into five sections. In the first one, we present a short background on the existing researches in our domain. The second section introduces a case study to illustrate our approach. In the third section, the C-G ontology construction is defined. The fourth section presents the appropriate part of IF model. The fifth section is centered on the principle of the C-G ontologies alignment.

2 Case Study

To illustrate our different definitions, we propose a case study which concerns an open channel hydraulic system composed of three subsystems (S1, S2, S3) connected with a CAN network. The first system is situated upriver. It performs two pressure measurements from a Pitot sensor: dynamic and static pressure. The measurement of velocity depends on these two pressures. S1 provides, also, the value of the water level in its area. S2 is similar to S1. It has, in addition, an actuator able to regulate the water level. The actuator is a sort of gate activated by the difference of two values of velocity (local velocity and received velocity from distant system-S1 or S3-) and modifies its position with the help of a brushless motor. The actuator is activated only if the velocity comes from a remote upstream system. S3 is the downstream system, it is similar to S1

S2 plays a central role. It must satisfy the global goal which is the regulation of the level by activating the gate according to the velocity information received from the upstream system. The problem appears very simple, but if we consider that a system identical to S2 is a downstream system from S3, then things become less trivial. Indeed, S2 may receive a velocity information from S1, but also from S3. It is then necessary to select the good information. On an other hand, S1 may send two identical information, but from two sequences of different actions. Again, it is necessary to discern the sequence of actions in order to choose the right connection to S2. To solve this problem, we base, on one hand, on a formal representation of goals and their contexts, and on the other hand, on the semantic coordination mechanism of systems using the Information Flow (IF).

3 Context-Goal Ontologies

3.1 Formalization of Context-Goal pairs

We will see in this section how to formalize the knowledge of a given system in terms of C-G Ontologies. We have introduced the context notion which is related to a goal, where this last is the result of an action on a given context (see figure 1). The pairs C-G will be related to form plans in order to satisfy goals.

The context is employed in different disciplines : computer science (mainly the Artificial Intelligence), cognitive science, linguistics, philosophy, psychology or yet in the application areas such as medicine or legislation. In the applications of "Context-Aware" field, the context is considered as an information which characterizes a situation: “ *By context, we refer to any information that characterizes a situation related to the interaction between humans, applications, and the surrounding environment. Context-aware applications*

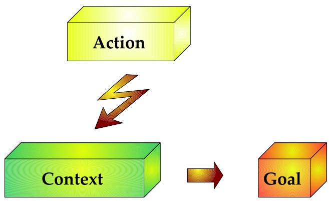


Figure 1. Context-Goal pair

promise richer and easier interaction” [6]. According to Brézillon, the context is always relative to some thing : state, time, object, situation .. " *Le contexte sert de guide au focus, c'est-à-dire au sous-ensemble d'éléments qui sont pertinents pour la tâche en cours* " [3].

In [4], a context is expressed in record of dependent types based on an intuitionist logic. This record is a sequence fields in which labels l_i correspond to certain types T_i, ie, every field may depend to values of precedent fields. We use a simplified version of this approach where a First Order Logic is employed. Contexts are modeled by a structure of knowledge including entities, constraints and proposals. These lasts are extracted from a domain ontology. Based on this idea, we formalize contexts distinguishing two categories:

- **Context Type** : a context type C is constructed from a set of types of objects $\{T_1, T_2, ..T_m\}$ describing entities, properties and/or constraints. We formalize context by the following tuple:
- **Context Token** : a context token c introduce the objects (instantiated types) called tokens which represent an instance sequence of types $\{t_1, t_2, ..t_n\}$. A context token is defined by the tuple:

$$C = \begin{bmatrix} l_1 & : T_1 \\ l_2 & : T_2 \\ & \dots \\ example & \\ Z & : Zone \\ \kappa_1 & : Static(Z) \end{bmatrix} \qquad c = \begin{bmatrix} l_1 & : t_1 \\ l_2 & : t_2 \\ & \dots \\ example & \\ z_1 & : Zone1 \\ \kappa_1 & : Static(z_1) \end{bmatrix}$$

l_i are labels.
We say that a context (token or type) is valid if and only if the conjunction of the tuple elements are valid.

A goal is defined by the result of an action associated to a particular context. When the context is a type context, we speak about a " *Goal type*" and in the other case we say " *Goal token*". Each pair "C-G" is associated with a real action corresponding generally to the execution of a function within the computational meaning.

The functional aspect of the association C-G is preserved adopting the following rules:

Rule 1 : At a given context, a single action is associated.
Rule 2 : several contexts may correspond to one action because the action can be carried out in several contexts of execution.
In a given system, we denote the C-G pair by $(C_i, \gamma_j)^{(k)}$, such as : C_i is a given context, γ_j is goal and k is a system.

According to our case study, we give the following example extracted from the thesis [15]. The context $C_5^{(1)}$ corresponds to an action " *receive* " where the result is a velocity value received from another system. The basic constraint is the good functioning of the network ($\kappa_1 : Online(R)$). The result is the goal type $\gamma_4^{(2)}$.

$$C_5^{(2)} = \begin{bmatrix} R & : Network \\ \kappa_1 & : Online(R) \\ V & : Velocity \\ Pos & : Distance \\ \kappa_2 & : Sent(V, Pos) \end{bmatrix} \rightarrow \gamma_4^{(2)} = \begin{bmatrix} g & : Received(V, Pos) \end{bmatrix}$$

3.2 Relations between C-G pairs

In general, an ontology should satisfy an explicit definition of concepts and relations among them. In our approach, the pairs Context-Goal are the ontology concepts and the relations between them are the causal order of goals achievement. We define two relations between the pairs : Causal Dependency and Subsumption Dependency.

3.3 Causal Dependency

We define the inherent causality in systems by dependance relations which base on the inclusion of types.

If a goal associated to a given context is realized, then it is possible to find another context in which this but appears. In other words, if this goal is included in the structure of another but it becomes possible to connect the two pairs. This result may be generalized to several contexts. We can formalize this in the following way:

Definition 1 *Contextual Inclusion*
Let (C, γ) and (C', γ') be two pairs of context types and goals (resp. tokens) such as γ be a goal type representing the result of a given action on C. If C' contain γ, then we say that γ is included in C' and we write:

$$\gamma \subseteq C'$$

Example: $\gamma_2^{(1)} \subseteq C_5^{(1)}$, where $\gamma_2^{(1)}$ is the goal type corresponding to the action of measuring the velocity value.

We define the causal dependency by:

Definition 2 *Causality*
A pair C-G $(C_l, \gamma_m)^{i(k)}$ of abstraction level i in a system k is in causal relation with the pair $(C_{l+1}, \gamma_{m+1})^{i(k)}$ in the same level and system if $\gamma_m^{i(k)} \subseteq C_{l+1}$ and we denote:

$$(C_l, \gamma_m)^{i(k)} \preceq (C_{l+1}, \gamma_{m+1})^{i(k)}$$

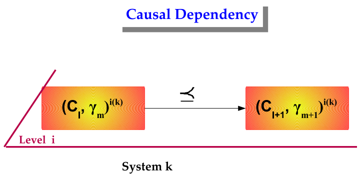


Figure 2. Causal Dependency between C-G pairs

The validity of a pair Context-Goal (C', γ') depends on the completion of the goal γ. This means that the pair (C, γ) "*causes*" the occurrence of (C', γ').

When the pairs are not of the same level, we propose for that the relation *Subsumption Dependency*:

3.4 Subsumption Dependency

Definition 3 *Subsumption of pairs Context-Goal*
A pair Context-Goal $(C_q, \gamma_r)^{i+1(k)}$ of level $i+1$ on a system k subsumes a plan $(C_l, \gamma_m)^{i(k)}, ..., (C_{l+p}, \gamma_{m+p})^{i(k)}$ of level i on the same system if the realization of $\gamma_r^{i+1(k)}$ depends to the realization of all the goals in the types sequence $(\gamma_m, ..., \gamma_{m+p})^{i(k)}$ which we note:

$$(C_l, \gamma_m)^{i(k)}, \ldots, (C_{l+p}, \gamma_{m+p})^{i(k)} \sqsubseteq (C_q, \gamma_r)^{i+1(k)}$$

Example: $(C_1, \gamma_1)^{1(1)}, (C_2, \gamma_1)^{1(1)}, (C_3, \gamma_2)^{1(1)}, (C_5, \gamma_4)^{1(1)} \sqsubseteq (C_1, \gamma_1)^{2(1)}$ where $\gamma_1^{2(1)}$ goal type of level 2 which corresponds to the measurement of the velocity in system S1.

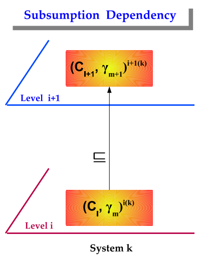


Figure 3. Subsumption Dependency between Context-Goal pairs

3.5 C-G Ontologies Formalization

The C-G ontology $\mathcal{O}$ obeys to the following formal structure:
Definition 4 *C-G ontology*

$$\mathcal{O} = (\mathcal{G}, \preceq, \sqsubseteq)$$

Where $\mathcal{G}$ is the set of Context-Goal types pairs, $\preceq$, the relation of causal dependency between Context-Goal types pairs and $\sqsubseteq$, the subsumption dependency relation.

We have established a knowledge representation (goals) by defining an intentional ontology. This ontology includes pairs of Context-Goal types, allows firstly the representation knowledge and, secondly, provides a conceptual basis to use the IF model in order to link the subsystems of a distributed system. We recall the posed problem in the case study where the system S2 must regulate the level of water in the channel. To solve the problem, we use a mathematical model (IF) which connects Context-Goal ontologies automatically, so connects distributed systems.This model is summarized in the following section.

4 Summary of IF-Model

The IF model is proposed by J.Barwise and J.Seligman [1] to define a logic for distributed systems. In our approach, we define by the model a formal framework in order to implement the ontology alignment process. Let A be a distributed system, each component of A may be described by entities of some types. The types are related by constraints and represent the local behavior of these entities. This is called in IF model by *Local Logic*. To connect entities of different components in a distributed system, the *Information Channel* is used, a tool characterized by the *distributed logic* which expresses formally the links between entities.

When we specify the problem of the alignment in terms of IF model, ontologies are described by local logics and the distributed logic models the alignment.

4.1 Local Logic

Definition 5 *"Classification"*
A classification A is a triple $< tok(A), typ(A), \models_A >$, which consists of:

1. *a set $tok(A)$ of objects to be classified known as the instances or particulars of A that carry information,*
2. *a set $typ(A)$ of objects used to classify the instances, the types of A,*
3. *a binary classification relation $\models_A$ between $tok(A)$ and $typ(A)$ that tells one which tokens are classified as being of which types.*

The notation $a \models_A \alpha$ must be understood as "instance a is of type α in A". IF classifications are related through infomorphisms.

Definition 6 *"Infomorphism"*
Let A and B be classifications. An infomorphism denoted $f =< f^\wedge, f^\vee >$: $A \Leftrightarrow B$ is a contravariant pair of functions $f^\wedge$: $typ(A) \to typ(B)$ and $f^\vee$: $tok(B) \to tok(A)$ which satisfies the fundamental property:

$$f^\vee(b) \models_A \alpha \text{ iff } b \models_B f^\wedge(\alpha) \tag{1}$$

for each $\alpha \in typ(A)$ and $b \in tok(B)$

Definition 7 *"Theory"*
We call $Th(A) = \langle typ(A), \vdash_A \rangle$ a theory generated by from the classification A where the constraints on the set of sequents are satisfied by every token in A.

Let $\alpha \in typ(A) : \Gamma \subseteq typ(A), \Gamma' \subseteq typ(A), \Delta \subseteq typ(A), \Delta' \subseteq typ(A), \Sigma' \subseteq typ(A)$,
$\Sigma_0 \subseteq typ(A), \Sigma_1 \subseteq typ(A)$, the theory $Th(A)$ generated from a classification A is called regular if it satisfies the following conditions:
Property 1 • *The Identity: $\alpha \vdash_T \alpha$*
- *The Weakening: if $\Gamma \vdash_T \Delta$ then $\Gamma, \Gamma' \vdash_T \Delta, \Delta'$*
- *The Global cut: if $\Gamma, \Sigma_0 \vdash_T \Delta, \Sigma_1$ for any partition $\langle \Sigma_0, \Sigma_1 \rangle$ of subsets Σ', then $\Gamma \vdash_T \Delta$*

Definition 8 *Local logic*
Let $A =< tok(A), typ(A), \models_A>$ a classification, $Th(A) =< typ(A), \vdash_A>$ a theory generated from A. The local logic $\mathcal{L}_A$ is defined as:

$$\mathcal{L}_A =< A, Th(A), N_A >$$

such as: : $\forall a \in N_A \subseteq tok(A)$, a satisfies the constraints of $Th(A)$. N_A is called the set of normal tokens.
The local logic posses the following characteristics:

Property 2 *Let $\mathcal{L}_A$ be a local logic generated from a classification A. $\mathcal{L}_A$ is valid if N_A = $tok(A)$;*
- *$\mathcal{L}_A$ is complete if every sequent satisfied by every normal token is a constraint in $\mathcal{L}_A$.*

Property 3 *Inverse image of a local logic*
Let $\mathcal{L}_A$ and $\mathcal{L}_B$ be two local logics, such as :
$\mathcal{L}_A =< A, Th(A), N_A >$
$\mathcal{L}_B =< B, Th(B), N_B >$
f is an infomorphism connecting A and B, such, ($f : A \to B$).
The inverse image of $\mathcal{L}_B$ by f, denoted $f^{-1}[\mathcal{L}_B]$, is the local logic generated from a classification A, the theory $f^{-1}[Th(A)]$ and the set of normal tokens :

$$\{a \in tok(A) | a = f(b) \textit{ for some } b \in \mathrm{N}_A\}$$

Property 4 *Let A and B be two classifications, f an infomorphism connecting A and B ($f : A \to B$) and $\mathcal{L}_B$ a local logic generated from the classification B,*
- *If $\mathcal{L}_B$ is complete, then $f^{-1}[\mathcal{L}_B]$ is complete.*
- *If f is surjective on tokens ($f^\vee$) and $\mathcal{L}_B$ is valid, then $f^{-1}[\mathcal{L}_B]$ is valid.*

4.2 Information Channel

A distributed system is modeled in IF model by a set of classifications connected by infomorphisms.

An information channel $\mathcal{C}$ consists in the connection of different classifications $A_{i \in I}$ with a core classification C through infomorphisms h_i. The infomorphisms are defined in the domain of A_i and the codomain in C.

Definition 9 *" Information Channel "*
Let $\{A_{i\in \mathrm{I}}\}$ be an indexed family of classifications and let C be a classification. Having a set of infomorphisms $\{h_i\}$, an information channel which formalize the connections between $\{A_{i\in \mathrm{I}}\}$ and C is defined by:

$$\mathcal{C}= \{h_i : A_{i\in \mathrm{I}} \Leftrightarrow C\}$$

Definition 10 *Distributed Logic*
The distributed logic of an information channel $\mathcal{C}$ is the local logic on the sum $\sum_{i\in \mathrm{I}} A_i$ mentioned by $\mathcal{L}_C$. The distributed logic is denoted $DLog_\mathcal{C}(\mathcal{L}_C)$.

The distributed logic is justified by the local logic of the core classification. We presented in this section the main algebraic tools of the IF model.

5 Aligning C-G Ontologies

We will show in this section, how to succeed an automatic alignment of C-G ontologies using the IF model which aims to connect entities of different systems in terms of information channel. In these systems, types serve to classify objects and obey specific relations. The set defines a distributed logic.

As mentioned in the previous section, the IF model introduces a consequence relationship $\vdash$ on a set of types. This relation can find from a given type t_1, the corresponding type t_2 through the relation $\vdash$ (where t_1 and t_2 belong to different sets of types).

According to the IF model, these entities are related via the information channel which preserves the information during its transmission between systems. The IF model is a good mean to achieve the alignment of ontologies since it can provide a theory and a logic which links entities belonging to different systems.

The process steps are summarized as follows:

1. **Identification of possible classifications in each system according to their associated ontologies:** For every goal of level 2, we identify a classification. The types of the classification are the pairs C-G. The tokens are the goal types included in these pairs. The binary relation expresses the inclusion of a goal in a pair C-G. Let us take the example of classification according to our example :

Table 1. Classification B_1 associated to goal of level 2 $\gamma_1^{2(2)}$

$\models_{B_1}$	B_1				
	$(C_1, \gamma_1)^{1(2)}$	$(C_2, \gamma_1)^{1(2)}$	$(C_3, \gamma_2)^{1(2)}$	$(C_4, \gamma_3)^{1(2)}$	$(C_1, \gamma_1)^{2(2)}$
γ_1^2	1	1	1	1	1
γ_2^2	0	0	1	1	1
γ_3^2	0	0	0	1	1

2. **Generation of their possible theories:** from every classification, we identify the corresponding theory (see definition 7).
3. **Construction of the Information Channel:** the important step in the process. To identify the distributed logic, we introduce a case where a pair C-G, in a given ontology, is not valid. According to our case study, the pair $(C_5, \gamma_4)^{1(2)}$ is not valid because the action *receive* does not happened in the S2 system, so the constraint κ_2 : $Sent(V, Pos)$ is not satisfied. Then it is important to connect this pair to the corresponding pair $(C_i, \gamma_j)^{k(l)}$ in distant systems. It is clear that is not possible to connect it with all pairs, because a combinatorial explosion may be produced when the number of pair is very high. For that we propose two filtering(s):

(a) First Filtering: we assume a partial alignment of pairs C-G of initial classifications via a key classification K. The goal $\gamma_4^{1(2)}$ is the type of K and the tokens are a and b. We observe that the goal is appeared in K as a type but in the initial classifications it appears as a token. Then, it is useful to introduce the flip of classifications by transposing rows and columns. These classifications are connected via infomorphisms. In our case, the condition of infomorphisms aims to identify the candidate classifications by searching goals which are identical semantically to $\gamma_4^{1(2)}$ or included in its context.

(b) Second Filtering: To choose the corresponding classification (the corresponding pair C-G) from the candidates and connect them through the information channel, the core classification C must be generated. The types of C are the disjoint union of goals. The tokens are the cartesian product of pairs C-G. Its binary relation expresses the fact that goals are part of one set of types or not. We generate from C the relevant theory $Th(C)$. According to our case study, $Th(C)$ introduces three sequent(s) :
$\{\gamma_1^1, \gamma_2^1, \gamma_4^1\} \vdash_C \{\gamma_1^2, \gamma_2^2, \gamma_4^2\}$
$\{\gamma_2^1, \gamma_4^1\} \vdash_C \{\gamma_1^2 \gamma_2^2, \gamma_4^2\}$
$\{\gamma_1^3, \gamma_2^3, \gamma_4^3\} \vdash_C \{\gamma_1^2, \gamma_2^2, \gamma_4^2\}$

To choose the relevant sequent, we propose an elimination rule: "*We eliminate a sequence of goals if at least one of these goals do not verify the contexts in the C-G ontology*"

Applying this rule, only the first sequent is satisfied, because all constraints are satisfied. For example in the pair $(C_6, \gamma_5)^{1(2)}$ the sequence $\{\gamma_1^1, \gamma_2^1, \gamma_4^1\}$ coming from the S1 system verifies all the constraints. The central constraints are κ_1 and κ_2, where κ_1 means that the received velocity must be sent from an upstream system, so S1. But as we have reported, S1 may send two values of velocity (real and estimated velocity), to choose the relevant one, we have introduced in C_6 the κ_2 which is satisfied by the first sequent.

$$\overrightarrow{\gamma}_5^{(2)} = \left[g \ : Calculated(V_1 - V_2, Pos_1) \right.$$

$$\left[\begin{array}{ll} V_1 & : Velocity \\ Pos_1 & : Distance \\ V_2 & : Velocity \\ Pos_2 & : Distance \\ p_1 & : Calculated(V_1, Pos_1) \\ p_2 & : Received(V_2, Pos_2) \\ \kappa_1 & : More - Than(Pos_1, Pos_2) \\ \kappa_2 & : Not - Estimated(V_2) \end{array} \right.$$

From C classification and $Th(C)$, we generate the local logic $LL(C)$ on C. This allows to generate the distributed logic $DL(C)$. It is the inverse image of $LL(C)$ (see definition 10) which expresses the links between different classifications. We succeed an automatic and semantic alignment of C-G ontologies.

6 Conclusion

We have analyzed the dual problem of knowledge representation and the alignment of ontologies. For that we have proposed a methodology addressing two fundamental axes. The first runs around the representation of knowledge where the goals of each system must be formally represented preserving their semantic aspect. These goals are related to their contexts. We have used ontologies which are an effective means for this aspect. The second axis is the achievement of semantic inter-operability of systems, our methodology allows a semantic and automatic alignment of C-G Ontologies. For that, we base on the mathematical IF model to success this alignment formalize connections between C-G ontologies in terms of information channel.

Concerning ongoing work, we investigate the application of Information Channel theory in industrial environments where goal structures generalize the role concept to the industrial or business framework and where the context is replaced by a business context.

REFERENCES

[1] Jon Barwise and Jerry Seligman, *Information Flow: The Logic of Distributed Systems*, number 44, Cambridge University Press, 1997.

[2] Bresciani, Giorgini, Giunchiglia, Mylopoulos, and Perini, 'TROPOS: An agent-oriented software development methodology', *Autonomous Agents and Multi-Agent Systems*, **8**(3), 203–236, (May 2004).

[3] Patrick Brézillon and Charles Tijus, 'Une représentation basée sur le contexte des utilisateurs à travers leurs pratiques', in *Ecole de gestion des Connaissances Workshop "Modélisation des Utilisateurs dans les IHM*, pp. 23–31, Nice, (2005).

[4] Richard Dapoigny and Patrick Barlatier, 'Deriving behavior from goal structure for the intelligent control of physical systems', in *Informatics in Control, Automation and Robotics*, pp. 51–58, (2007).

[5] Robert Darimont, Emmanuelle Delor, Philippe Massonet, and Axel van Lamsweerde, 'Grail/kaos: An environment for goal-driven requirements analysis, integration and layout.', in *RE*, (1997).

[6] Anind Dey, Gregory Abowd, and Daniel Salber, 'A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications', *Human-Computer Interaction*, **16**(2, 3 and 4), 97–166, (2001).

[7] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy, *Ontology matching: A machine learning approach*, 2003.

[8] Thomas Gruber, 'A translation approach to portable ontology specifications', *Knowledge Acquisition*, **5**(2), 199–220, (1993).

[9] Fabrice Jouanot, 'Un modèle sémantique pour l'interopérabilité de systèmes d'information.', in *INFORSID*, pp. 347–364, (2000).

[10] Igor Jurisica, John Mylopoulos, and Eric Yu. Using ontologies for knowledge management: An information systems perspective, 1999.

[11] Yannis Kalfoglou and Marco Schorlemmer, 'If-map: an ontology mapping method based on information flow theory', *Journal on Data Semantics*, **1**(1), 98–127, (october 2003).

[12] Kitamura and Mizoguchi, 'Ontology-based description of functional design knowledge and its use in a functional way server'. 2003.

[13] Lind M, 'Modeling goals and functions of complex industrial plant', *Journal of Applied Artificial Intelligence*, (1994).

[14] Alexander Maedche, Boris Motik, Nuno Silva, and Raphael Volz, 'MAFRA - A Mapping Framework for Distributed Ontologies', in *Proc. of 13th European Conference on Knowledge Engineering and Knowledge Management (EKAW)*, Siquenca, Spain, (2002).

[15] Nacima MELLAL, *Réalisation de l'interopérabilité sémantique des systèmes, basée sur les ontologies et les flux d'information*, Ph.D. dissertation, LISTIC-Université de Savoie, 2007.

[16] Marco Schorlemmer and Yannis Kalfoglou, 'Using information flow theory to enable semantic interoperability', in *Catalan Conference on Artificial Intelligence (CCIA '03)*, (2003).

[17] Axel van Lamsweerde, 'Goal-oriented requirements engineering: a guided tour', in *5th International Conference on Requirements Engineering*, pp. 249–262, (2001).

[18] François Vernadat, 'Interoperable enterprise systems: Architectures, methods and metrics', Technical report, LGIPM, Université de Metz, France, (2007).

[19] Eric Yu, 'Towards modelling and reasoning support for early-phase requirements engineering', in *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, pp. 226–235, (1997).

The production of documents from ontologies

Dana Dannélls[1]

Abstract. The production of documents from an ontology is a challenging task which requires a significant effort from a natural language generator. Addressing this problem involves a careful examination of how the knowledge formalized in an ontology can be realized. We have started to exploit the abilities of generating natural language texts from a Web Ontology Language (OWL) and to examine how the content of the ontology can be rendered in different textual degrees that support reader and listener preferences. In this paper we present our line of research and exemplify some of the advantages and challenges we encountered while attempting to generate different contexts from a domain specific OWL ontology.

1 Introduction

A major challenge for a language generator developer who wishes to make use of semantic web ontologies is how to alter the input knowledge-base, so as to verbally express contents of some knowledge for different purposes. This task becomes even harder when the user preferences such as the preferred language, text length and syntax must be computed.

Our research project aims to adapt the presentation of a text content for a specific readership from Web ontologies. At this stage we do not deal with the production of grammatical sentences with the appropriate content, but rather focus on the subtle content realization of parts of the ontology to enhance readability. We recognize a number of elements which have an effect on the process of text production from an ontology and which we have started to examine:

1. the selection of axioms;
2. the representation order of the selected axioms;
3. the verbalization and realization of the ontology terms used to express classes and relations.

We utilized a domain specific OWL ontology and started to exploit the role of linguistic contexts from this expressive language with emphasis on verbalization and realization of the relations between concepts. It appears that many relations have a specific interpretation in the ontology depending on the context in which they appear, hereby given the user's preferences and the ontology knowledge, linguistic realizations exhibit great variations. We illustrate some of these variations and discuss their implications for text productions.

In this paper we give an overview of previous work on generation from ontologies and discuss a number of the advantages and challenges that Web ontology languages pose to language generators (section 2). We provide a description of the domain ontology and the domain ontology language in section 3. We exemplify the difficulties in verbalizing the ontology-relations which we came across while attempting to produce coherent and cohesive texts in section 4.

[1] Natural Language Processing Research Unit, Department of Swedish Language, University of Gothenburg, email: dana.dannells@svenska.gu.se

2 Background

There are many definitions for the term *ontology* [11]. In this context, an ontology is defined as a structured framework for modeling the concepts and relationships of some domain expertise, which provides the structural and semantic ground for computer based processing of domain knowledge. To allow better use of ontologies in applications, traditional ontology language standards such as DAML+OIL and OWL[2] have been specified by the World Wide Web Consortium (W3C). One of the purposes of these established standards is to enable better communication between humans and machines in which information is given a well defined meaning.

2.1 Generating from ontologies

Generation techniques deal with the process of converting semantic representation into surface form in a particular language. The features of the text produced are normally chosen with respect to a particular target reader group. There have been successful attempts to develop natural language generation tools that generate texts from Web ontology languages [1, 2, 12, 13].

Wilcock [12] presents an approach in which the concepts defined in the ontology are used to generate the lexicon. Bontcheva and Wilks [2] concentrate on the semantic representations encoded in a Web ontology language and discuss how these can be exploited to generate text summaries. They point out the content of the ontology itself as a major factor for the quality of the output. Gawronska and Erlendsson [4] show how biological ontologies as Kyoto Encyclopedia of Genes and Genomes, may be utilized for generating graphs representing the essential contents of biomedical scientific articles.

Mellish and Sun [8] describe the large extent of linguistic material in existing Web ontologies and its complexity. They exemplify how an extended text with multiple sentences can be generated from class axioms, i.e. subsumption relationships between classes and properties.

Similarly to [13, 2], this work is concerned with generating various textual variation in descriptions of individuals on the basis of a domain-specific ontology. As opposed to [8], this approach deals with individuals and requires manual input of the lexicon. In contrast to [13] where templates are used, we intend to utilize a grammar-based surface realiser to enhance linguistic variations in the generated texts.

2.2 Opportunities and challenges

As pointed out by many authors, there are several advantages which make Web ontology languages such as OWL particularly suitable to generate from. For example axioms can be seen as forming a graph in

[2] http://www.w3.org/TR/

which routes between axioms correspond to different possible transitions in a coherent text [7]; axioms can be used to accommodate the system to different contextual degrees and user needs; the use of multiple-inheritance converts the class hierarchy into a directed graph and not a tree structure.

Web ontologies provide implicit information about a domain. This advantage has been exploited by a number of Natural Language Generation (NLG) systems [9] who use the ontology knowledge and description logics (DL) to complete generation related tasks. In many domain ontologies the ontology terms used to express classes and relations are similar to their lexical entry, which in many aspects facilitate the generation tasks. However, natural languages are ambiguous, so even ontologies which do not make a distinction between the ontology terms and natural language words used to describe them, contain a lot of ambiguities that need to be resolved.

To reveal implicit information about a concept, inferences have to be drawn. These inferences, that are mostly based on DL [10], can be rendered differently depending on the target user and language. Furthermore, it is necessary to fully understand what the knowledge formalized in an ontology actually states before natural language words can be expressed. The content and knowledge formalized in an ontology leads to ambiguous content interpretations, it also create problems when trying to verbalize the data represented on this basis. This has brought with it an awareness of the need to encode linguistic knowledge about concepts directly into ontologies [5].

3 The domain ontology model

The work described in this paper is based on the CIDOC Conceptual Reference Model (CRM) ontology,[3] which is an initiative to construct an ontology within the Cultural Heritage (CH) domain. The CIDOC ontology consists of 81 relations and 244 concepts and is available in various formats, among them the OWL-lite ontology. It contains facts about concepts (sets of objects) and roles (binary relations) and provides a conceptual model that subscribes an object-centred view of the CH domain.

3.1 Population and maintenance

The CIDOC-CRM ontology does not contains facts about individuals (single objects) and the first step to start with is to populate the ontology. Populating an ontology mainly involve adding new individuals. In our work this includes lexical entries, as well as new concepts and relations. On the task of ontology population, most of the work that has been carried out relates to information extraction from unstructured natural language text or semi-structured HTML pages [6].

The process of ontology population described here was conducted manually, population is based on a small corpus of CH texts that we have collected from internal museum repositories. Following the guidelines given by the reference document [3] for filling in concept-values along with a thorough analysis of the information content, we have so far enriched the ontology with a total of 150 new objects including instances and classes. Each object was assigned with its lexical lemma that links to a lexical string-name.

3.2 The ontology terminology

An OWL ontology (lite or DL) has a description logic based semantics which consists of a set of axioms. Axioms assert facts about concepts (Tbox) and facts about individuals (Abox). Roles are usually asserted in the form of inclusion axioms.

As with any representation of an OWL ontology, the CIDOC CRM ontology contains classes (concepts) that define a group of individuals that belong together because they share some properties (roles). A subclass is a class that is a specialization of another class (its superclass). According to the CRM documentation, *specialization* means: (1) all instances of the subclass are also instances of its superclass; (2) the intension of the subclass extends the intension of its superclass; (3) the subclass inherits the definition of all of the properties declared for its superclass in addition to having one or more properties of its own.

Properties server to define relationships of a specific kind between two classes. A property can have a subproperty which is a specialization of another property (its superproperty). A property must be defined with reference to both its domain and range. The term *specialization* in the context of properties has similar meaning as for classes with additional restrictions, i.e: (4) the domain of the subproperty is the same as the domain of its superproperty or a superclass of that domain; (5) the range of the subproperty is the same as the range of its superproperty or the subclass of that range.

4 Linguistic realizations from the ontology

In the semantics of OWL, a given axiom may match multiple rules and therefore may have multiple possible realizations. In this section we exemplify some of the challenges (see section 2.2) which are related to surface realization aspects in a given context, following the representation of the CIDOC-CRM ontology model.

4.1 An OWL representation

The following example is taken from our ontology and it describes the class *EdelfeltProduction*. This class comprises a set of productions that has been carried out by Albert Edelfelt. According to the CRM reference document: “a production can present activities, that are designed to, and succeed in, creating one or more new items”.

This particular example presents the production of a portrait that took place in France and was made by Albert Edelfelt between 1880 and 1890.

```
<museum:EdelfeltProduction rdf:about=”#EdelfeltPortraitProduction”>
 <crm:P14F.carried_out_by>
  <crm:E21.Person rdf:about=”#AlbertEdelfelt”/>
  </crm:P14F.carried_out_by>
  <crm:P12F.occurred_in_the_presence_of>
  <crm:E21.Person rdf:about=”#AlbertEdelfelt”/>
  </crm:P12F.occurred_in_the_presence_of>
 <crm:P7F.took_place_at>
  <crm:E48.Place_Name rdf:about=”#France”/>
  </crm:P7F.took_place_at>
 <crm:P4F.has_time_span>
<crm:E49.Time_Appellation rdf:about=”#1880-1890”/>
  </crm:P4F.has_time_span>
</crm:E12.Production>
```

The class *EdelfeltProduction* is a subclass of *E12.Production*. *E12.Production* has multiple inheritance classes, i.e. *E11.Modification* and *E63.Beginning_of_Existence*, this is shown

[3] http://cidoc.ics.forth.gr/

below.[4]

```
<owl:Class rdf:about="&crm;E12.Production">
 <rdfs:subClassOf rdf:resource="&crm;E11.Modification />
 <rdfs:subClassOf rdf:resource="&crm;E63.Beginning_of_Existence
/>
</owl:Class>
```

E11.Modification is a subclass of *E7.Activity* and *E63.Beginning_of_Existence* is a subclass of *E5.Event*, hence the inferred relation *P12F.occurred_in_the_presence_of*.

4.2 Surface realizations

Given an ontology, populated by individuals, given some user preferences, the task is to verbalize and realize the knowledge contained in the ontology. A direct realization of the above ontology fragment results in the following text:

> This portrait production was carried out by Albert Edelfelt. The portrait production occurred in the presence of Albert Edelfelt. The portrait production took place in France. The portrait production has time span 1880-1890.

Inferred knowledge When dealing with inferred knowledge, domain-dependent aspects need to be taken into consideration. As relations carry different information when they appear in different conceptual contexts. For example, following the above ontology fragment, we can note that the inferred relation *P12F.occurred_in_the_presence_of*, carries redundant information. We can draw the conclusion that in the context of *EdelfeltPortraitProduction* this relation does not contribute with new information and could be therefore eliminated, or "selected" and verbalized instead of the relation *carried_out_by*.

Nevertheless, when a production describes an activity which has resulted in, for example, a movie production such as the movie "The lord of the ring", an inferred relation such as *occurred in the presence of* will not provide redundant information and will probability be necessary to include in the output text. Thus, it is important to make a distinction between conceptual contexts where inferred relations provide new information and cannot be eliminated.

Verbalization The choice of a lexical entry encoding a relation is both domain and user dependent, for example, the relation *carried_out_by* could be verbalized as either "painted by" or "created by" depending on the concepts it describes. Furthermore, the choice between synonyms for the relation *created_by* are various: "produce by", "bring out by", "develop by", "acquire by", etc. Some differences in categorizations or internal makeup must be present if the difference in information content is to be consequential.

When verbalizing the description about the instance *EdelfeltPortraitProduction* we want to establish a text which is similar to the following:

> This portrait production was carried out by Albert Edelfelt. The production took place in France. It covers the period 1880-1890.

Humans can recognize that semantic representations are intimately linked, this realization process could also be automated rather easily. However, the problem of how words and other linguistic phenomena might be integrated with the internal representations that support reasoning is yet to be explored.

[4] The notation &crm; is used as a shortcut for the complete URL to the CIDOC-CRM ontology.

5 Conclusions and future work

We have outlined an ongoing research that addresses the problems encountered while attempting to generate coherent and cohesive texts from a Web ontology language. This work is currently based on a domain specific lightweight CIDOC-CRM ontology. Text planning follows the ontology axioms structure; the assertional part of the ontology is developed manually; both the terminological part and the assertional part are applied to present parts of the ontology.

Our examples illustrate the challenges posed for language generators. We have shown that relations might have a particular, quite specific interpretation in an ontology depending on the context in which they appear. The choice of a lexical entry encoding a relation is both domain and user dependent. Even though OWL provides powerful reasoning opportunities for natural language generators, there still exists a need to find automatic domain-dependent solutions. We highlighted the problem of inferable relations that are necessary or unnecessary in a particular context, a task which is not trivial for machines and yet to be approached.

This project is only in its early stages. Exploiting OWL for realization purposes and finding general, domain-independent solutions requires a considerable amount of work. In the near future we are planning to address issues related to content selection and lexical determination of relations between concepts, a task which depends on the chosen semantic content, the concept it describes, the class hierarchy that is utilized to represent the concept, and the target language.

REFERENCES

[1] K. Bontcheva, 'Generating tailored textual summaries from ontologies.', in *ESWC*, pp. 531–545, (2005).

[2] K. Bontcheva and Y. Wilks, 'Automatic report generation from ontologies: the miakt approach', in *Nineth International Conference on Applications of Natural Language to Information Systems*, (2004).

[3] N. Crofts, M. Doerr, T. Gill, S. Stead, and M. Stiff, *Definition of the CIDOC Conceptual Reference Model*, the version 4.2.4 of the reference document edn., March 2008. http://cidoc.ics.forth.gr/docs/cidoc_crm_version_4.2.4_March2008_.pdf.

[4] B. Gawronska and B. Erlendsson, 'Syntactic, semantic and referential patterns in biomedical texts: towards in-depth text comprehension for the purpose of bioinformatics.', in *Proceedings of the 2nd International Workshop on Natural Language Understanding and Cognitive Science NLUCS*, pp. 68–77, Miami, USA, (May 2005).

[5] J. Judgem, M. Sogrin, and A. Troussov, 'Galaxy:ibm ontological network miner', *In Sören Auer, Christian Bizer, Claudia Müller, and Anna V. Zhdanova,*, **113**, 157–160, (2007).

[6] V. Karkaletsis, A Valarakos, and C.D. Spyropoulos, 'Populating ontologies in biomedicine and presenting their content using multilingual generation', in *AIME*, pp. 256–265, (2005).

[7] C. Mellish and J. Z. Pan, 'Natural language directed inference from ontologies', *Artificial Intelligence*, **172**(10), 1285–1315, (2008).

[8] C. Mellish and X. Sun, 'The semantic web as a linguistic resource: Opportunities for natural language generation', *Knowl.-Based Syst.*, **19**(5), 298–303, (2006).

[9] S. Daniel Paiva, 'A survey of applied natural language generation systems', Technical Report ITRI-98-03, Information Technology Research Institute(ITRI), University of Brighton, UK, (1998).

[10] E. Reiter and C. Mellish, 'Using classication to generate text', in *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL92)*, (1992).

[11] S. Staab and R. Studer, *Handbook on Ontologies*, International Handbooks on Information Systems, Springer, 2004.

[12] G. Wilcock, 'Talking owls: Towards an ontology verbalizer', in *Human Language Technology for the Semantic Web and Web Services*, pp. 109–112, (2003). Sanibel Island, Florida.

[13] G. Wilcock and K. Jokinen, 'Generating responses and explanations from rdf/xml and daml+oil.', in *Knowledge and Reasoning in Practical Dialogue Systems IJCAI-2003*, pp. 58–63, (2003). Acapulco.