Preface

During the recent decade, handling uncertainty has started to play an important role in ontology languages, especially in application areas like the Semantic Web, Bio-medicine, and Artificial Intelligence. For this reason, there is currently a strong research interest in Description Logics (DLs) that allow for dealing with uncertainty. The subject of the *First International Workshop on Uncertainty in Description Logics (UniDL '10)* was how to deal with uncertainty and imprecision in DLs. This encompasses approaches that enable probabilistic or fuzzy reasoning in DLs, but the workshop was also open for approaches based on other uncertainty formalisms. The workshop focused on the investigation of reasoning problems and approaches for solving them, including especially tractable ones. For classical DL reasoning problems such as subsumption and satisfiability, algorithms that can handle uncertainty exist, but they are still less well-investigated than in the case of standard DLs without uncertainty. For novel reasoning services, such as query answering, computation of generalizations, modules, or explanations, it is not yet clear how to realize them in DLs that can express uncertainty.

Topics of interest included but were not limited to: (1) modeling of uncertain knowledge in DLs; (2) different formalizations of uncertainty for DLs; (3) formal semantics for uncertain information in DLs; (4) extensions of DL reasoning problems to uncertainty; (5) reasoning algorithms for DLs with uncertainty; in particular, (6) tableau algorithms for probabilistic DLs or fuzzy DLs; (7) tractable DLs with uncertainty; (8) complexity of uncertain reasoning; (9) system descriptions for implemented reasoning algorithms in uncertain DLs; and (10) novel applications of DLs with uncertainty.

These proceedings contain the papers presented at the *First International Workshop on Uncertainty in Description Logics (UniDL '10)*, which was held in Edinburgh, UK, July 20, 2010. It contains 5 technical papers and 2 system descriptions, which were selected in a careful reviewing process, where each paper was reviewed by at least three program committee members. These proceedings also contain an extended abstract of the invited talk.

We wish to thank all authors who submitted papers and all workshop participants for fruitful discussions. We are grateful to Ralf Möller for his invited talk at the workshop. We would like to thank the program committee members and external referees for their timely expertise in carefully reviewing the submissions. Many thanks also to the developers of the EasyChair Conference System, which we used for the reviewing process and the preparation of these proceedings.

July 2010

Thomas Lukasiewicz
Rafael Peñaloza
Anni-Yasmin Turhan

Workshop Organization

Program Chairs

Thomas Lukasiewicz (Oxford University, UK)
Rafael Peñaloza (TU Dresden, Germany)
Anni-Yasmin Turhan (TU Dresden, Germany)

Program Committee

Eyal Amir (University of Illinois, Urbana-Champaign, USA)
Fernando Bobillo (University of Zaragoza, Spain)
Simona Colucci (Technical University of Bari, Italy)
Fabio G. Cozman (University of Sao Paulo, Brazil)
Manfred Jaeger (Aalborg University, Denmark)
Pavel Klinov (University of Manchester, UK)
Ralf Möller (Hamburg University of Technology, Germany)
Mathias Niepert (University of Mannheim, Germany)
Guilin Qi (Southeast University, China)
Stefan Schlobach (Vrije Universiteit Amsterdam, The Netherlands)
Luciano Serafini (IRS Trento, Italy)
Giorgos Stoilos (Oxford University, UK)
Umberto Straccia (ISTI-CNR, Italy)

External Referees

Abner Guzman-Rivera (University of Illinois, Urbana-Champaign, USA)

Sponsors

This workshop was generously supported by Elsevier's Artificial Intelligence Journal.

A Probabilistic Abduction Engine for Media Interpretation based on Ontologies

Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, Michael Wessel

Hamburg University of Technology, Germany

Abstract. For multimedia interpretation, and in particular for the combined interpretation of information coming from different modalities, a semantically well-founded formalization is required in the context of an agent-based scenario. Low-level percepts, which are represented symbolically, define the observations of an agent, and interpretations of content are defined as explanations for the observations.
We propose an abduction-based formalism that uses description logics for the ontology and Horn rules for defining the space of hypotheses for explanations (i.e., the space of possible interpretations of media content), and we use Markov logic to define the motivation for the agent to generate explanations on the one hand, and for ranking different explanations on the other.
This work has been funded by the European Community with the project CASAM (Contract FP7-217061 CASAM) and by the German Science Foundation with the project PRESINT (DFG MO 801/1-1).

1 Introduction

For multimedia interpretation in the context of an agent-based scenario, and for the combined interpretation of information coming from different modalities in particular, a semantically well-founded formalization is required. Low-level percepts, which are represented symbolically, define the observations of an agent w.r.t. some content, and interpretations of the content are defined as explanations for the observations.

We propose an abduction-based formalism that uses description logics for the ontology and Horn rules for defining the space of hypotheses for explanations (i.e., the space of possible interpretations of media content). Additionally, we propose the use of Markov logic to define the motivation for the agent to generate explanations on the one hand, and for ranking different explanations on the other.

Based on a presentation of the most important preliminaries in Section 2, the abduction and interpretation procedures are discussed in detail in Section 3. Optimization techniques for the probabilistic abduction engine are pointed out. In Section 4, a complete example is given, showing the main approach using intermediate steps. Section 7 summarizes this paper.

2 Preliminaries

In this chapter, the most important preliminaries are specified in order to make this document self-contained.

2.1 Preliminaries on Description Logics

For specifying the ontology used to describe low-level analysis results as well as high-level interpretation results, a less expressive description logic is applied to facilitate fast computations. We decided to represent the domain knowledge with the DL $\mathcal{ALH}_f{}^-(\mathcal{D})$ (restricted attributive concept language with role hierarchies, functional roles and concrete domains). We shortly describe our nomenclature in order to make this paper self-contained. For details see [Baader et al., 2003].

In logic-based approaches, atomic representation units have to be specified. The atomic representation units are fixed using a so-called signature. A DL *signature* is a tuple $\mathcal{S} = (\mathbf{CN}, \mathbf{RN}, \mathbf{IN})$, where $\mathbf{CN} = \{A_1, ..., A_n\}$ is the set of concept names (denoting sets of domain objects) and $\mathbf{RN} = \{R_1, ..., R_m\}$ is the set of role names (denoting relations between domain objects). The signature also contains a component $\mathbf{IN}$ indicating a set of individuals (names for domain objects).

In order to relate concept names and role names to each other (terminological knowledge) and to talk about specific individuals (assertional knowledge), a knowledge base has to be specified. An $\mathcal{ALH}_f{}^-$ *knowledge base* $\Sigma_{\mathcal{S}} = (\mathcal{T}, \mathcal{A})$, defined with respect to a signature $\mathcal{S}$, is comprised of a terminological component $\mathcal{T}$ (called *Tbox*) and an assertional component $\mathcal{A}$ (called *Abox*). In the following we just write Σ if the signature is clear from context. A Tbox is a set of so-called *axioms*, which are restricted to the following form in $\mathcal{ALH}_f{}^-$:

(I) Subsumption	$A_1 \sqsubseteq A_2$, $R_1 \sqsubseteq R_2$
(II) Disjointness	$A_1 \sqsubseteq \neg A_2$
(III)Domain and range restrictions for roles	$\exists R.\top \sqsubseteq A$, $\top \sqsubseteq \forall R.A$
(IV)Functional restriction on roles	$\top \sqsubseteq (\leq 1\, R)$
(V) Local range restrictions for roles	$A_1 \sqsubseteq \forall R.A_2$
(VI)Definitions with value restrictions	$A \equiv A_0 \sqcap \forall R_1.A_1 \sqcap ... \sqcap \forall R_n.A_n$

With axioms of form (I), concept (role) names can be declared to be subconcepts (subroles) of each other. Axioms of form (II) denote disjointness between concepts. Axioms of type (III) introduce domain and range restrictions for roles. Axioms of the form (IV) introduce so-called *functional* restrictions on roles, and axioms of type (V) specify local range restrictions (using value restrictions, see below). With axioms of kind (VI) so-called definitions (with necessary and sufficient conditions) can be specified for concept names found on the left-hand side of the $\equiv$ sign. In the axioms, so-called *concepts* are used. Concepts are concept names or expressions of the form $\top$ (anything), $\bot$ (nothing), $\neg A$ (atomic negation), $(\leq 1\, R)$ (role functionality), $\exists R.\top$ (limited existential restriction), $\forall R.A$ (value restriction) and $(C_1 \sqcap ... \sqcap C_n)$ (concept conjunction).

Knowledge about individuals is represented in the Abox part of Σ. An Abox $\mathcal{A}$ is a set of expressions of the form $A(a)$ or $R(a, b)$ (concept assertions and role assertions, respectively) where A stands for a concept name, R stands for a role name, and a, b stand for individuals. Aboxes can also contain equality ($a = b$) and inequality assertions ($a \neq b$). We say that the unique name assumption (UNA) is applied, if $a \neq b$ is added for all pairs of individuals a and b.

In order to understand the notion of logical entailment , we introduce the semantics of $\mathcal{ALH}_f{}^-$. In DLs such as $\mathcal{ALH}_f{}^-$, the semantics is defined with interpretations $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\triangle^{\mathcal{I}}$ is a non-empty set of domain objects (called the domain of $\mathcal{I}$) and $\cdot^{\mathcal{I}}$ is an interpretation function which maps individuals to objects of the domain ($a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$), atomic concepts to subsets of the domain ($A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$) and roles to subsets of the cartesian product of the domain ($R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$). The interpretation of arbitrary $\mathcal{ALH}_f{}^-$ concepts is then defined by extending $\cdot^{\mathcal{I}}$ to all $\mathcal{ALH}_f{}^-$ concept constructors as follows:

$$\begin{array}{ll}
\top^{\mathcal{I}} & = \triangle^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = \emptyset \\
(\neg A)^{\mathcal{I}} & = \triangle^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(\leq 1\, R)^{\mathcal{I}} & = \{u \in \triangle^{\mathcal{I}} \mid (\forall v_1, v_2)\, [((u, v_1) \in R^{\mathcal{I}} \wedge (u, v_2) \in R^{\mathcal{I}}) \rightarrow v_1 = v_2] \\
(\exists R.\top)^{\mathcal{I}} & = \{u \in \triangle^{\mathcal{I}} \mid (\exists v)\, [(u, v) \in R^{\mathcal{I}}]\} \\
(\forall R.C)^{\mathcal{I}} & = \{u \in \triangle^{\mathcal{I}} \mid (\forall v)\, [(u, v) \in R^{\mathcal{I}} \rightarrow v \in C^{\mathcal{I}}]\} \\
(C_1 \sqcap ... \sqcap C_n)^{\mathcal{I}} & = C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}
\end{array}$$

In the following, the *satisfiability* condition for axioms and assertions of an $\mathcal{ALH}_f{}^-$-knowledge base Σ in an interpretation $\mathcal{I}$ are defined. A concept inclusion $C \sqsubseteq D$ (concept definition $C \equiv D$) is satisfied in $\mathcal{I}$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp. $C^{\mathcal{I}} = D^{\mathcal{I}}$) and a role inclusion $R \sqsubseteq S$ (role definition $R \equiv S$), if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ (resp. $R^{\mathcal{I}} = S^{\mathcal{I}}$). Similarly, assertions $C(a)$ and $R(a, b)$ are satisfied in $\mathcal{I}$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ resp.

$(a, b)^\mathcal{I} \in R^\mathcal{I}$. If an interpretation $\mathcal{I}$ satisfies all axioms of $\mathcal{T}$ resp. $\mathcal{A}$ it is called a *model* of $\mathcal{T}$ resp. $\mathcal{A}$. If it satisfies both $\mathcal{T}$ and $\mathcal{A}$ it is called a model of Σ. Finally, if there is a model of Σ (i.e., a model for $\mathcal{T}$ and $\mathcal{A}$), then Σ is called satisfiable.

We are now able to define the entailment relation $\models$. A DL knowledge base Σ *logically entails* an assertion α (symbolically $\Sigma \models \alpha$) if α is satisfied in all models of Σ. For an Abox $\mathcal{A}$, we say $\Sigma \models \mathcal{A}$ if $\Sigma \models \alpha$ for all $\alpha \in \mathcal{A}$.

2.2 Substitutions, Queries, and Rules

Sequences, Variable Substitutions and Transformations A *variable* is a name of the form *String* where *String* is a string of characters from {A...Z}. In the following definitions, we denote places where variables can appear with uppercase letters.

Let V be a set of variables, and let $\underline{X}, \underline{Y_1}, \ldots, \underline{Y_n}$ be sequences $\langle \ldots \rangle$ of variables from V. $\underline{z}$ denotes a sequence of individuals. We consider sequences of length 1 or 2 only, if not indicated otherwise, and assume that $(\langle X \rangle)$ is to be read as (X) and $(\langle X, Y \rangle)$ is to be read as (X, Y) etc. Furthermore, we assume that sequences are automatically flattened. A function *as_set* turns a sequence into a set in the obvious way.

A *variable substitution* $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ is a mapping from variables to individuals. The application of a variable substitution σ to a sequence of variables $\langle X \rangle$ or $\langle X, Y \rangle$ is defined as $\langle \sigma(X) \rangle$ or $\langle \sigma(X), \sigma(Y) \rangle$, respectively, with $\sigma(X) = i$ and $\sigma(Y) = j$. In this case, a sequence of individuals is defined. If a substitution is applied to a variable X for which there exists no mapping $X \leftarrow k$ in σ then the result is undefined. A variable for which all required mappings are defined is called *admissible* (w.r.t. the context).

Grounded Conjunctive Queries Let $\underline{X}, \underline{Y_1}, \ldots, \underline{Y_n}$ be sequences of variables, and let $Q_1, \ldots, Q_n$ denote concept or role names.

A query is defined by the following syntax.

$$\{(\underline{X}) \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

The sequence $\underline{X}$ may be of arbitrary length but all variables mentioned in $\underline{X}$ must also appear in at least one of the $\underline{Y_1}, \cdots, \underline{Y_n}$: $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$.

Informally speaking, $Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ defines a conjunction of so-called *query atoms* $Q_i(\underline{Y_i})$. The list of variables to the left of the sign | is called the *head* and the atoms to the right of are called the query *body*. The variables in the head are called distinguished variables. They define the query result. The variables that appear only in the body are called non-distinguished variables and are existentially quantified.

Answering a query with respect to a knowledge base Σ means finding admissible variable substitutions σ such that $\Sigma \models \{\sigma(Q_1(\underline{Y_1})), \ldots, \sigma(Q_n(\underline{Y_n}))\}$. We say that a variable substitution $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ introduces *bindings* $i, j, \ldots$ for variables $X, Y, \ldots$. Given all possible variable substitutions σ, the *result* of a query is defined as $\{(\sigma(\underline{X}))\}$. Note that the variable substitution σ is applied before checking whether $\Sigma \models \{Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}$, i.e., the query is *grounded* first.

For a query $\{(?y) \mid Person(?x), hasParticipant(?y, ?x)\}$ and the Abox $\Gamma_1 = \{HighJump(ind_1), Person(ind_2), hasParticipant(ind_1, ind_2)\}$, the substitution $[?x \leftarrow ind_2, ?y \leftarrow ind_1]$ allows for answering the query, and defines bindings for $?y$ and $?x$.

A *boolean* query is a query with $\underline{X}$ being of length zero. If for a boolean query there exists a variable substitution σ such that $\Sigma \models \{\sigma(Q_1(\underline{Y_1})), \ldots, \sigma(Q_n(\underline{Y_n}))\}$ holds, we say that the query is answered with *true*, otherwise the answer is *false*.

Later on, we will have to convert query atoms into Abox assertions. This is done with the function *transform*. The function *transform* applied to a set of query atoms $\{\gamma_1, \ldots \gamma_n\}$ is defined as $\{transform(\gamma_1, \sigma), \ldots, transform(\gamma_n, \sigma)\}$ where
$transform(P(\underline{X}), \sigma) := P(\sigma(\underline{X}))$.

Rules A rule r has the following form $P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ where P, $Q_1, \ldots, Q_n$ denote concept or role names with the additional restriction (safety condition) that $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$.
Rules are used to derive new Abox assertions, and we say that a rule r is *applied* to an Abox $\mathcal{A}$. The function call $apply(\Sigma, P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}), \mathcal{A})$ returns a set of Abox assertions $\{\sigma(P(\underline{X}))\}$ if there exists an admissible variable substitution σ such that the answer to the query

$$\{() \mid Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}$$

is *true* with respect to $\Sigma \cup \mathcal{A}$.[1] If no such σ can be found, the result of the call to $apply(\Sigma, r, \mathcal{A})$ is the empty set. The application of a set of rules $\mathcal{R} = \{r_1, \ldots r_n\}$ to an Abox is defined as follows.

$$apply(\Sigma, \mathcal{R}, \mathcal{A}) = \bigcup_{r \in \mathcal{R}} apply(\Sigma, r, \mathcal{A})$$

The result of $forward\_chain(\Sigma, \mathcal{R}, \mathcal{A})$ is defined to be $\emptyset$ if $apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup \mathcal{A} = \mathcal{A}$ holds. Otherwise the result of *forward_chain* is determined by the recursive call
$apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup forward\_chain(\Sigma, \mathcal{R}, \mathcal{A} \cup apply(\Sigma, \mathcal{R}, \mathcal{A}))$.

For some set of rules $\mathcal{R}$ we extend the entailment relation by specifying that $(\mathcal{T}, \mathcal{A}) \models_{\mathcal{R}} \mathcal{A}_0$ iff $(\mathcal{T}, \mathcal{A} \cup forward\_chain((\mathcal{T}, \emptyset), \mathcal{R}, \mathcal{A})) \models \mathcal{A}_0$.

2.3 Probabilistic Knowledge Representation

The basic notion of probabilistic knowledge representation formalisms is the so-called *random experiment.* A *random variable* X is a function assigning a value to the result of a random experiment. The random experiment itself is not represented, so random variables are functions without arguments, which return different values at different points of time. Possible values of a random variable comprise the so-called *domain* of the random variable. In the sequel, we will use *boolean* random variables, whose values can be either 1 or 0 (*true* or $false$, respectively).

Let $\vec{X} = \{X_1, ..., X_n\}$ be the ordered set of all random variables of a random experiment. An *event* (denoted $\vec{X} = \vec{x}$) is an assignment $X_1 = x_1, ..., X_n = x_n$ to all random variables. In case $n = 1$ we call the event simple, otherwise the event is called complex. A certain vector of values $\vec{x}$ is referred to as a *possible world.* A possible world can be associated with a *probability value* or *probability* for short. Hence, the notion of a possible world can be used as a synonym for an event, and depending on the context we use the former or the latter name. In case of an event with a boolean random variable X, we write x as an abbreviation for $X = true$ and $\neg x$ as an abbreviation for $X = false$.

Mappings of events to probabilities (or assignment of probabilities to events) are specified with so-called *probability assertions* of the following syntax: $P(\vec{X} = \vec{x}) = p$, where $\vec{X}$ is a vector of random variables, and p is a real value between 0 and 1 (it is assumed that the reader is familiar with Kolmogorov's axioms of probability). In the special case of a simple event (single random variable, $n = 1$) we write $P(X = x) = p$. The probability value p of an event is denoted as $P(\vec{X} = \vec{x})$ (or $P(X = x)$ in the simple case). In its raw form a set of probabilistic assertions is called a *probabilistic knowledge base* (with signature $\vec{X}$).

A mapping from the domain of a random variable X to probability values $[0, 1]$ is called a *distribution.* For distributions we use the notation $\mathbf{P}(X)$. Distributions can be defined for (ordered) sets of random variables as well. In this case we use $\mathbf{P}(X_1, \ldots, X_n)$ as a denotation for a mapping to the n-dimensional cross product of $[0, 1]$. For specifying a distribution, probability assertions for *all* domain values must be specified, and the values p must sum up to 1. In case all random variables of a random experiment are involved, we speak of a *(full) joint probability distribution* (JPD), otherwise the expression is said to denote a *marginal distribution* (projection of the n-dimensional space of probability values to a lower-dimensional space with m dimensions). The expression $\mathbf{P}(X_1, \ldots, X_m, X_{m+1} = x_{m+1}, \ldots, X_l = x_l)$ denotes an m-dimensional distribution with known

[1] We slightly misuse notation in assuming $(\mathcal{T}, \mathcal{A}) \cup \Delta = (\mathcal{T}, \mathcal{A} \cup \Delta)$. If $\Sigma \cup \mathcal{A}$ is inconsistent the result is well-defined but useless. It will not be used afterwards.

values $x_{m+1}, \ldots, x_l$. In slight misuse of notation, we sometimes write $\vec{e}$ for these known values (e stands for evidence). The fragment $\vec{e}$ need not necessarily be written at the end in the parameter list of $\mathbf{P}$.

A *conditional probability* for a set of random variables $X_1, ..., X_m$ is denoted with $P(X_1 = x_1, ..., X_m = x_m \mid \vec{e})$ or, in distribution form, we write $\mathbf{P}(X_1, ..., X_m \mid \vec{e})$ (conditional probability distribution). This distribution can be also written as $\frac{\mathbf{P}(\vec{X}, \vec{e})}{\mathbf{P}(\vec{e})}$.

For a probabilistic knowledge base, formal inference problems are defined. We restrict our attention to the two most convenient probabilistic inference problems: A *conditional probability query* is the computation of the joint distribution of a set of m random variables conditioned on $\vec{e}$ and is denoted with

$$P_X(x_1 \wedge ... \wedge x_m \mid \vec{e}) = ?.$$

where $vars(x_1, \ldots, x_m) \cap vars(\vec{e}) = \emptyset$ and $vars(x_1, \ldots, x_m) \cup vars(\vec{e}) \subseteq X$ with $vars$ specified in the obvious way. Note that x_i indicates $X_i = x_i$. In the following we have the distribution form of the above query:

$$\mathbf{P}_X(X_1, ..., X_m \mid \vec{e}) = ?.$$

If the set of random variables X is known from the context, the subscript $_X$ is often omitted.

The Maximum A Posteriori (MAP) inference returns the most-likely state of query atoms given the evidence. Based on the MAP inference, the "most probable world" given the evidence is determined as a set of events. The MAP inference problem given a distribution for a set of random variables X is formalized as follows:

$$MAP_X(\vec{e}) := \vec{e} \cup argmax_{\vec{x}} P(\vec{x} | \vec{e}) \tag{1}$$

where $vars(\vec{x}) \cap vars(\vec{e}) = \emptyset$ and $vars(\vec{x}) \cup vars(\vec{e}) = X$.

For both inference problems, conditional probability queries as well as the MAP problem, different kinds of algorithms exist, which possibly exploit additional assertions (such as, e.g., conditional independence assumptions in so-called Bayesian networks, or factored probability distribution specifications as in so-called Markov networks). In the next subsection, we focus on the latter formalism.

2.4 Markov Logic

The formalism of Markov logic [Domingos and Richardson, 2007] provides a means to combine the expressivity of first-order logic augmented with the formalism of Markov networks [Pearl, 1988]. The Markov logic formalism uses first-order logic to define "templates" for constructing Markov networks. The basic notion for this is a called a Markov logic network.

A Markov logic network $MLN = (\mathcal{F}_{MLN}, \mathcal{W}_{MLN})$ consists of an ordered multiset of first-order formulas $\mathcal{F}_{\mathcal{MLN}} = \{F_1, ..., F_m\}$ and an ordered multiset of real number weights $\mathcal{W} = \{w_1, ..., w_m\}$. The association of a formula to its weight is by position in the ordered sets. For a formula $F \in \mathcal{F}_{MLN}$ with associated weight w we also write wF (weighted formula). Thus, a Markov logic network can also be defined as a set of weighted formulas. Both views can be used interchangeably. As a notational convenience, for ordered sets we nevertheless sometimes write $\vec{X}, \vec{Y}$ instead of $\vec{X} \cup \vec{Y}$.

In contrast to standard first-order logics such as predicate logic, relational structures not satisfying a formula F_i are not ruled out as models. If a relational structure does not satisfy a formula associated with a large weight it is just considered to be quite unlikely the "right" one.

Let $C = \{c_1, ..., c_m\}$ be the set of all constants mentioned in $\mathcal{F}_{MLN}$. A *grounding* of a formula $F_i \in \mathcal{F}_{MLN}$ is a substitution of all variables in the matrix of F_i with constants from C. From all groundings, the (finite) set of grounded atomic formulas (also referred to as *ground atoms*) can be obtained. Grounding corresponds to a domain closure assumption. The motivation is to get rid of the quantifiers and reduce inference problems to the propositional case.

Since a ground atom can either be true or false in an interpretation (or world), it can be considered as a boolean random variable X. Consequently, for each MLN with associated random variables $\vec{X}$, there is a set of possible worlds $\vec{x}$. In this view, sets of ground atoms are sometimes used to denote worlds. In this context, negated ground atoms correspond to *false* and non-negated ones to *true*. We denote worlds using a sequence of (possibly negated) atoms.

When a world $\vec{x}$ violates a weighted formula (does not satisfy the formula) the idea is to ensure that this world is less probable rather than impossible as in predicate logic. Note that weights do not directly correspond to probabilities (see [Domingos and Richardson, 2007] for details).

For each possible world of a Markov logic network $MLN = (\mathcal{F}_{MLN}, \mathcal{W}_{MLN})$ there is a probability for its occurrence. Probabilistic knowledge is required to obtain this value. As usual, probabilistic knowledge is specified using a probability distribution. In the formalism of Markov networks the full joint probability distribution of a Markov logic network MLN is specified in symbolic form as $\mathbf{P}_{MLN}(\vec{X}) = (P(\vec{X} = \vec{x}_1), \ldots, P(\vec{X} = \vec{x}_n))$, for every possible $\vec{x}_i \in \{true, false\}^n$, $n = |\vec{X}|$ and $P(\vec{X} = \vec{x}) := log\_lin_{MLN}(\vec{x})$ (for a motivation of the log-linear form, see, e.g., [Domingos and Richardson, 2007]), with log_lin being defined as

$$log\_lin_{MLN}(\vec{x}) = \frac{1}{Z} exp\,(\sum_{i=1}^{|\mathcal{F}_{MLN}|} w_i n_i(\vec{x}))$$

According to this definition, the probability of a possible world $\vec{x}$ is determined by the exponential of the sum of the number of true groundings (n_i) formulas $F_i \in \mathcal{F}_{MLN}$ in $\vec{x}$, multiplied with their corresponding weights $w_i \in \mathcal{W}_{MLN}$, and finally normalized with

$$Z = \sum_{\vec{x} \in \vec{X}} \exp\,(\sum_{i=1}^{|\mathcal{F}_{MLN}|} w_i n_i(\vec{x})), \tag{2}$$

the sum of the probabilities of all possible worlds. Thus, rather than specifying the full joint distribution directly in symbolic form as we have discussed before, in the Markov logic formalism, the probabilistic knowledge is specified implicitly by the weights associated with formulas. Determining these formulas and their weights in a practical context is all but obvious, such that machine learning techniques are usually employed for knowledge acquisition.

A *conditional probability query for a Markov logic network MLN* is the computation of the joint distribution of a set of m events involving random variables conditioned on $\vec{e}$ and is denoted with

$$P_{MLN}(x_1 \wedge \ldots \wedge x_m \mid \vec{e})$$

The semantics of this query is given as:

$$P_{rand\_vars(MLN)}(x_1 \wedge \ldots \wedge x_m \mid \vec{e}) \; w.r.t. \; \mathbf{P}_{MLN}(rand\_vars(MLN))$$

where $vars(x_1, \ldots, x_m) \cap vars(\vec{e}) = \emptyset$ and $vars(x_1, \ldots, x_m) \subseteq rand\_vars(MLN)$. and the function $rand\_vars$ function is defined as follows: $rand\_vars((\mathcal{F}, \mathcal{W})) := \{P(\underline{C}) \mid P(\underline{C})$ is mentioned in some grounded formula $F \in \mathcal{F}\}$. Grounding is accomplished w.r.t. all constants that appear in $\mathcal{F}$. An algorithm for answering queries of the above form is investigated in [Gries and Möller, 2010].

In the case of Markov logic, the definition of the MAP problem given in (1) can be rewritten as follows. The conditional probability term $P(\vec{x}|\vec{e})$ is replaced with with the Markovian formula:

$$MAP_{MLN}(\vec{e}) := \vec{e} \cup argmax_{\vec{x}} \frac{1}{Z_e} \exp\left(\sum_i w_i n_i\,(\vec{x}, \vec{e})\right) \tag{3}$$

Thus, for describing the most-probable world, MAP returns a set of events, one for each random variable used in the Markov network derived from MLN. In the above equation, $\vec{x}$ denotes the hidden variables, and Z_e denotes the normalization constant which indicates that the normalization process is performed over possible worlds consistent with the evidence $\vec{e}$. In the next equation, Z_e is removed since it is constant and it does not affect the *argmax* operation. Similarly, in order to optimize the

MAP computation the exp function is left out since it is a monotonic function and only its argument has to be maximized:

$$MAP_{MLN}(\vec{e}) := \vec{e} \cup argmax_{\vec{x}} \sum_i w_i n_i \left(\vec{x}, \vec{e}\right) \tag{4}$$

The above equation shows that the MAP problem in Markov logic formalism is reduced to a new problem which maximizes the sum of weights of satisfied clauses.

Since the MAP determination in Markov networks is an **NP**-hard problem [Domingos and Richardson, 2007], it is performed by exact and approximate solvers. The most commonly used approximate solver is MaxWalkSAT algorithm, a weighted variant of the WalkSAT local-search satisfiability solver. The MaxWalkSAT algorithm attempts to satisfy clauses with positive weights and keep clauses with negative weights unsatisfied.

It has to be mentioned that there might be several worlds with the same maximal probability. But at this step, only one of them is chosen non-deterministically.

2.5 Combining Markov Logic and Description Logics

Since $\mathcal{ALH}_f{}^-$ is a fragment of first-order logic, its extension to the Markovian style of formalisms is specified in a similar way as for predicate logic in the section before. The formulas in Markov logic correspond to Tbox axioms and Abox assertions. Weights in Markov description logics are associated with axioms and assertions.

Groundings of Tbox axioms are defined analogously to the previous case.[2] Abox assertions do not contain variables and are already grounded. Note that since an $\mathcal{ALH}_f{}^-$ Abox represents a relational structure of domain objects, it can be directly seen as a possible world itself if assertions not contained in the Abox are assumed to be false.

For appropriately representing domain knowledge in CASAM, weights are possibly used only for a subset of the axioms of the domain ontology. The remaining axioms can be assumed to be *strict*, i.e., assumed to be true in any case. A consequence of specifying strict axioms is that lots of possible worlds $\vec{x}$ can be ruled out (i.e., will have probability 0 by definition).

A *Markov DL knowledge base* Σ_M is a tuple $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is comprised of a set $\mathcal{T}_s$ of strict axioms and a set $\mathcal{T}_w$ of weighted axioms and $\mathcal{A}$ is comprised of a set $\mathcal{A}_s$ of strict assertions and a set $\mathcal{A}_w$ of weighted assertions. Referring to axioms, a proposal for CASAM is to consider strictness for the domain ontology patterns (I)–(IV):

(I)	subsumption	$A_1 \sqsubseteq A_2,\ R_1 \sqsubseteq R_2$
(II)	disjointness	$A_1 \sqsubseteq \neg A_2$
(III)	domain and range restrictions	$\exists R.\top \sqsubseteq A,\ \top \sqsubseteq \forall R.A$
(IV)	functional roles	$\top \sqsubseteq (\leq 1\, R)$

The main justification treating axioms as strict is that the subsumption axioms, disjointness axioms, domain and range restrictions as well as functional role axioms (in combination with UNA) are intended to be true in any case such that there is no need to assign large weights to them.

In [Gries and Möller, 2010] we show that Gibbs sampling with deterministic dependencies specified in an appropriate fragment remains correct, i.e., probability estimates approximate the correct probabilities. We have investigated a Gibbs sampling method incorporating deterministic dependencies and conclude that this incorporation can speed up Gibbs sampling significantly. For details see [Gries and Möller, 2010]. The advantage of this probabilistic approach is that initial ontology engineering is done as usual with standard reasoning support and with the possibility to add weighted axioms and weighted assertions on top of the strict fundament. Since lots of possible worlds do not have to be considered because their probability is known to be 0, probabilistic reasoning will be significantly faster.

[2] For this purpose, the variable-free syntax of axioms can be first translated to predicate logic.

3 Probabilistic Interpretation Engine

In this chapter, the abduction procedure is defined by the abduction algorithm CAE. Additionally, a media interpretation agent is described by defining a probabilistic interpretation algorithm *Interpret*.

3.1 Computing Explanations

In general, abduction is formalized as $\Sigma \cup \Delta \models_{\mathcal{R}} \Gamma$ where background knowledge (Σ), rules ($\mathcal{R}$), and observations (Γ) are given, and explanations (Δ) are to be computed. In terms of DLs, Δ and Γ are Aboxes and Σ is a pair of Tbox and Abox.

Abox abduction is implemented as a non-standard retrieval inference service in DLs. In contrast to standard retrieval inference services where answers are found by exploiting the ontology, Abox abduction has the task of acquiring what should be added to the knowledge base in order to answer a query. Therefore, the result of Abox abduction is a set of hypothesized Abox assertions. To achieve this, the space of abducibles has to be previously defined and we do this in terms of rules.

We assume that a set of rules $\mathcal{R}$ as defined above (see Section 2.2) are specified, and define a non-deterministic function *compute_explanation* as follows.

- $\mathit{compute\_explanation}(\Sigma, \mathcal{R}, \mathcal{A}, P(\underline{z})) = \mathit{transform}(\Phi, \sigma)$ if there exists a rule

  $$r = P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}) \in \mathcal{R}$$

  that is applied to an Abox $\mathcal{A}$ such that a minimal set of query atoms Φ and an admissible variable substitution σ with $\sigma(\underline{X}) = \underline{z}$ can be found, and the query $Q := \{() \mid \mathit{expand}(P(\underline{z}), r, \mathcal{R}, \sigma) \setminus \Phi\}$ is answered with *true*.
- If no such rule r exists in $\mathcal{R}$ it holds that $\mathit{compute\_explanation}(\Sigma, \mathcal{R}, \mathcal{A}, P(\underline{z})) = \emptyset$.

The goal of the function *compute_explanation* is to determine what must be added (Φ) such that an entailment $\Sigma \cup \mathcal{A} \cup \Phi \models_{\mathcal{R}} P(\underline{Z})$ holds. Hence, for *compute_explanation*, abductive reasoning is used. The set of query atoms Φ defines what must be hypothesized in order to answer the query Q with *true* such that $\Phi \subseteq \mathit{expand}(P(\underline{X}), r, \mathcal{R}, \sigma)$ holds. The definition of *compute_explanation* is non-deterministic due to several possible choices for Φ.

The function application $\mathit{expand}(P(\underline{Z}), P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}), \mathcal{R})$ is also defined in a non-deterministic way as

$$\mathit{expand}'(Q_1(\sigma'(\underline{Y_1})), \mathcal{R}, \sigma) \cup \cdots \cup \mathit{expand}'(Q_n(\sigma'(\underline{Y_n})), \mathcal{R}, \sigma)$$

with $\mathit{expand}'(P(\underline{Z}), \mathcal{R}, \sigma)$ being $\mathit{expand}(P(\sigma'(\underline{z})), r, \mathcal{R}, \sigma')$ if there exist a rule $r = P(\underline{X}) \leftarrow \ldots \in \mathcal{R}$ and $\langle P(\underline{X}) \rangle$ otherwise. The variable substitution σ' is an extension of σ such that:

$$\sigma' = [X_1 \leftarrow z_1, X_2 \leftarrow z_2, \ldots] \tag{5}$$

The above equation shows the mapping of the free variables if it is not already defined. This means the free variables in the body of each rule are mapped to individuals with unique IDs.

We say the set of rules is backward-chained, and since there might be multiple rules in $\mathcal{R}$, backward-chaining is non-deterministic as well. Thus, multiple explanations are generated.[3]

[3] In the expansion process, variables have to be renamed. We neglect these issues here.

3.2 The Abduction Procedure

In the following, we devise an abstract computational engine for "explaining" Abox assertions in terms of a given set of rules. Explanation of Abox assertions w.r.t. a set of rules is meant in the sense that using the rules some high-level explanations are constructed such that the Abox assertions are entailed. The explanation of an Abox is again an Abox. For instance, the output Abox represents results of the content interpretation process. The presentation in slightly extended compared to the one in [Castano et al., 2008]. Let the agenda $\mathfrak{A}$ be a set of Aboxes Γ and let Γ be an Abox of observations whose assertions are to be explained. The goal of the explanation process is to use a set of rules $\mathcal{R}$ to derive "explanations" for elements in Γ. The explanation algorithm implemented in the CASAM abduction engine works on a set of Aboxes $\mathfrak{I}$.
The complete explanation process is implemented by the CAE function:

```
**Function** CAE($\Omega$, $\Xi$, $\Sigma$, $\mathcal{R}$, $S$, $\mathfrak{A}$):
**Input:** a strategy function $\Omega$, a termination function $\Xi$, a background knowledge $\Sigma$, a set of rules $\mathcal{R}$, a scoring function $S$, and an agenda $\mathfrak{A}$
**Output:** a set of interpretation Aboxes $\mathfrak{I}'$
$\mathfrak{I}' := \{assign\_level(l, \mathfrak{A})\}$;
**repeat**
    $\mathfrak{I} := \mathfrak{I}'$;
    $(\mathcal{A}, \alpha) := \Omega(\mathfrak{I})$  // $\mathcal{A} \in \mathfrak{I}$, $\alpha \in \mathcal{A}$ s.th. $requires\_fiat(\alpha^l)$ holds;
    $l = l + 1$;
    $\mathfrak{I}' := (\mathfrak{A} \setminus \{\mathcal{A}\}) \cup assign\_level(l, explanation\_step(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha))$;
**until** $\Xi(\mathfrak{I})$ *or no* $\mathcal{A}$ *and* $\alpha$ *can be selected such that* $\mathfrak{I}' \neq \mathfrak{I}$ ;
**return** $\mathfrak{I}'$
```

where $assign\_level(l, \mathfrak{A})$ is defined by a lambda calculus term as follows:

$$assign\_level(l, \mathfrak{A}) = map(\lambda(\mathcal{A}) \bullet assign\_level(l, \mathcal{A}), \mathfrak{A}) \tag{6}$$

$assign\_level(l, \mathfrak{A})$ takes as input a superscript l and an agenda $\mathfrak{A}$.

In the following, $assign\_level(l, \mathcal{A})$ is defined which superscripts each assertion α of the Abox $\mathcal{A}$ with l if the assertion α does not already have a superscript:

$$assign\_level(l, \mathcal{A}) = \left\{\alpha^l \mid \alpha \in \mathcal{A}, \alpha \neq \beta^i, i \in \mathbb{N}\right\} \tag{7}$$

Note that l is a global variable, its starting value is zero and it is incremented in the CAE function. The *map*[4] function is defined as follows:

$$map(f, X) = \bigcup_{x \in X} \{f(x)\} \tag{8}$$

It takes as parameters a function f and a set X and returns a set consisting of the values of f applied to every element x of X.
CAE function applies the strategy function Ω in order to decide which assertion to explain, uses a termination function Ξ in order to check whether to terminate due to resource constraints and a scoring function S to evaluate an explanation.

The function Ω for the explanation strategy and Ξ for the termination condition are used as an oracle and must be defined in an application-specific way. In our multimedia interpretation scenario we assume that the function $requires\_fiat(\alpha^l)$ is defined as follows:

[4] Please note that in this report, the expression *map* is used in two different contexts. The first one MAP denotes the Maximum A Posteriori approach which is a sampling method whereas the second one *map* is a function used in the $assign\_level(l, \mathfrak{A})$ function.

$$requires\_fiat(\alpha^l) = \begin{cases} true & \text{if } l = 0 \\ false & \text{if } l \neq 0 \end{cases}$$

The function *explanation_step* is defined as follows.

explanation_step$(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)$:

$$\bigcup_{\Delta \in compute\_all\_explanations(\Sigma,\mathcal{R},S,\mathcal{A},\alpha)} consistent\_completed\_explanations(\Sigma, \mathcal{R}, \mathcal{A}, \Delta).$$

We need two additional auxiliary functions.

consistent_completed_explanations$(\Sigma, \mathcal{R}, \mathcal{A}, \Delta)$:

$$\{\Delta' \mid \Delta' = \Delta \cup \mathcal{A} \cup forward\_chain(\Sigma, \mathcal{R}, \Delta \cup \mathcal{A}), consistent_{\Sigma}(\Delta')\}$$

compute_all_explanations$(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)$:

$$maximize(\Sigma, \mathcal{R}, \mathcal{A}, \{\Delta \mid \Delta = compute\_explanation(\Sigma, \mathcal{R}, \alpha), consistent_{\Sigma \cup \mathcal{A}}(\Delta)\}, S).$$

The function $maximize(\Sigma, \mathcal{R}, \mathcal{A}, \Delta s, S)$ selects those explanations $\Delta \in \Delta$s for which the score $S(\Sigma, \mathcal{R}, \mathcal{A}, \Delta)$ is maximal, i.e., there exists no other $\Delta' \in \Delta s$ such that $S(\Sigma, \mathcal{R}, \mathcal{A}, \Delta') > S(\Sigma, \mathcal{R}, \mathcal{A}, \Delta)$. The function $consistent_{(\mathcal{T},\mathcal{A})}(\mathcal{A}')$ determines if the Abox $\mathcal{A} \cup \mathcal{A}'$ has a model which is also a model of the Tbox $\mathcal{T}$.

Note the call to the nondeterministic function *compute_explanation*. It may return different values, all of which are collected.

In the next Section we explain how probabilistic knowledge is used to (i) formalize the effect of the "explanation", and (ii) formalize the scoring function S used in the CAE algorithm explained above. In addition, it is shown how the termination condition (represented with the parameter Ξ in the above procedure) can be defined based on the probabilistic conditions.

3.3 The Interpretation Procedure

The interpretation procedure is completely discussed in this section by explaining the interpretation problem and presenting a solution to this problem. The solution is presented by a probabilistic interpretation algorithm which calls the CAE function described in the previous section. In the given algorithm, a termination function, and a scoring function are defined. The termination function determines if the interpretation process can be stopped since at some point during the interpretation process it makes no sense to continue the process. The reason for stopping the interpretation process is that no significant changes can be seen in the results. The defined scoring function in this section assigns scores to interpretation Aboxes.

Problem The objective of the interpretation component is the generation of interpretations for the observations. An interpretation is an Abox which contains high level concept assertions. Since in the artificial intelligence, the agents are used for solving the problems, in the following the same problem is formalized in the perspective of an agent:
Consider an intelligent agent and some percepts in an environment where the percepts are the analysis results of KDMA and HCI. The objective of this agent is finding explanations for the existence of percepts. The question is how the interpretation Aboxes are determined and how long the interpretation process must be performed by the agent. The functionality of this Media Interpretation Agent is presented in the *MI_Agent* algorithm in Section 3.4.

Solution In the following, an application for a probabilistic interpretation algorithm is presented which gives a solution to the mentioned problem. This solution illustrates a new perspective to the interpretation process and the reason why it is performed. Assume that the media interpretation component receives a weighted Abox $\mathcal{A}$ from KDMA and HCI which contains observations. In the following, the applied operation $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ in the algorithm is explained:

The $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ function determines the probability of the Abox $\mathcal{A}$ with respect to the Abox $\mathcal{A}'$, a set of rules $\mathcal{R}$, a set of weighted rules $\mathcal{WR}$, and the Tbox $\mathcal{T}$ where $\mathcal{A} \subseteq \mathcal{A}'$. Note that $\mathcal{R}$ is a set of forward and backward chaining rules. The probability determination is performed based on the Markov logic formalism as follows:

$$P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T}) = P_{MLN(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})}(\vec{Q}(\mathcal{A}) \mid \vec{e}(\mathcal{A}')) \tag{9}$$

$\vec{Q}(\mathcal{A})$ denotes an event composed of all assertions which appear in the Abox $\mathcal{A}$. Assume that Abox $\mathcal{A}$ contains n assertions $\alpha_1, \ldots, \alpha_n$. Consequently, the event for the Abox $\mathcal{A}$ is defined as follows:

$$\vec{Q}(\mathcal{A}) = \langle \alpha_1 = true \wedge \ldots \wedge \alpha_n = true \rangle \tag{10}$$

where $a_1, \ldots, a_n$ denote the random variables of the MLN. Assume that an Abox $\mathcal{A}$ contains m assertions $\alpha_1, \ldots, \alpha_m$. Then, the evidence vector $\vec{e}(\mathcal{A})$ defined as follows.

$$\vec{e}(\mathcal{A}) = \langle a_1 = true, \ldots, a_m = true \rangle \tag{11}$$

In order to answer the query $P_{MLN(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})}(\vec{Q}(\mathcal{A}) \mid \vec{e}(\mathcal{A}'))$ the function $MLN(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ is called. This function builds the Markov logic network MLN based on the Aboxes $\mathcal{A}$ and $\mathcal{A}'$, the rules $\mathcal{R}$, the weighted rules $\mathcal{WR}$ and the Tbox $\mathcal{T}$ which is a time consuming process. Note that in theory the above function is called not only once but several times. In a practical system, this might be handled more efficiently. This function returns a Markov logic network $(\mathcal{F}_{MLN}, \mathcal{W}_{MLN})$, which we define here as follows using a tuple notation:

$$MLN(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T}) = \bigcup \begin{cases} \{(\alpha, w)\} & \text{if } \langle w, \alpha \rangle \in \mathcal{A} \\ \{(\alpha, \infty)\} & \text{if } \alpha \in \mathcal{A} \\ \{(\alpha, w)\} & \text{if } \langle w, \alpha \rangle \in \mathcal{A}' \\ \{(\alpha, \infty)\} & \text{if } \alpha \in \mathcal{A}' \\ \{(\alpha, \infty)\} & \text{if } \alpha \in \mathcal{R} \\ \{(\alpha, w)\} & \text{if } \langle w, \alpha \rangle \in \mathcal{WR} \\ \{(\alpha, \infty)\} & \text{if } \alpha \in \mathcal{T} \end{cases}$$

In the following, the interpretation algorithm *Interpret* is presented:

```
**Function** Interpret($\mathfrak{A}$, $CurrentI$, $\Gamma$, $\mathcal{T}$, $\mathcal{FR}$, $\mathcal{BR}$, $\mathcal{WR}$, $\epsilon$)
**Input:** an agenda $\mathfrak{A}$, a current interpretation Abox $CurrentI$, an Abox of observations $\Gamma$, a Tbox $\mathcal{T}$, a set of forward chaining rules $\mathcal{FR}$, a set of backward chaining rules $\mathcal{BR}$, a set of weighted rules $\mathcal{WR}$, and the desired precision of the results $\epsilon$
**Output:** an agenda $\mathfrak{A}'$, a new interpretation Abox $NewI$, and Abox differences for additions $\Delta_1$ and omissions $\Delta_2$
$i := 0$ ;
$p_0 := P(\Gamma, \Gamma, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ ;
$\Xi := \lambda(\mathfrak{A}) \bullet \{i := i + 1; p_i := \max_{\mathcal{A} \in \mathfrak{A}} P(\Gamma, \mathcal{A} \cup \mathcal{A}_0, \mathcal{R}, \mathcal{WR}, \mathcal{T});$ **return** $\mid p_i - p_{i-1} \mid < \frac{\epsilon}{i}\}$;
$\Sigma := (\mathcal{T}, \emptyset)$;
$\mathcal{R} := \mathcal{FR} \cup \mathcal{BR}$;
$S := \lambda((\mathcal{T}, \mathcal{A}_0)), \mathcal{R}, \mathcal{A}, \Delta) \bullet P(\Gamma, \mathcal{A} \cup \mathcal{A}_0 \cup \Delta, \mathcal{R}, \mathcal{WR}, \mathcal{T})$;
$\mathfrak{A}' := CAE(\Omega, \Xi, \Sigma, \mathcal{R}, S, \mathfrak{A})$;
$NewI = argmax_{\mathcal{A} \in \mathfrak{A}'}(P(\Gamma, \mathcal{A}, \mathcal{R}, \mathcal{WR}, \mathcal{T}))$;
$\Delta_1 = AboxDiff(NewI, CurrentI)$; // additions
$\Delta_2 = AboxDiff(CurrentI, NewI)$; // omissions
**return** $(\mathfrak{A}', NewI, \Delta_1, \Delta_2)$;
```

In the above algorithm, the termination function Ξ and the scoring function S are defined by lambda calculus terms. The termination condition Ξ of the algorithm is that no significant changes can be seen in the successive probabilities p_i and p_{i-1} (scores) of the two successive generated interpretation Aboxes in two successive levels $i-1$ and i. In this case, the current interpretation Abox *CurrentI* is preferred to the new interpretation Abox *NewI*. In the next step, the CAE function is called which returns agenda $\mathfrak{A}'$. Afterwards, the interpretation Abox *NewI* with the maximum score among the Aboxes $\mathcal{A}$ of $\mathfrak{A}'$ is selected. Additionally, the Abox differences Δ_1 and Δ_2 respectively for additions and omissions among the interpretation Aboxes *CurrentI* and *NewI* are calculated. In the following, the strategy condition Ω is defined which is one of the parameters of CAE function:

```
Function $\Omega(\mathfrak{I})$
Input: a set of interpretation Aboxes $\mathfrak{I}$
Output: an Abox $\mathcal{A}$ and a fiat assertion $\alpha$
$\mathfrak{A} := \left\{\mathcal{A} \in \mathfrak{I} \mid \neg\exists \mathcal{A}' \in \mathfrak{I}, \mathcal{A}' \neq \mathcal{A} : \exists {\alpha'}^{l'} \in \mathcal{A}' : \forall \alpha^l \in \mathcal{A} : l' < l\right\}$;
$\mathcal{A} := random\_select(\mathfrak{A})$;
$min\_\alpha_s = \left\{\alpha^l \in \mathcal{A} \mid \neg\exists {\alpha'}^{l'} \in \mathcal{A}', {\alpha'}^{l'} \neq \alpha^l, l' < l\right\}$;
return $(\mathcal{A}, random\_select(\{min\_\alpha_s\}))$;
```

In the above strategy function Ω, the agenda $\mathfrak{A}$ is a set of Aboxes $\mathcal{A}$ such that the assigned superscripts to their assertions are minimum. In the next step, an Abox $\mathcal{A}$ from $\mathfrak{A}$ is randomly selected. Afterwards, the min_α_s set is determined which contains the assertions α from $\mathcal{A}$ whose superscripts are minimum. These are the assertions which require explanations. The strategy function returns as output an Abox $\mathcal{A}$ and an assertion α which requires explanation.

3.4 The Media Interpretation Agent

In the following, the *MI_Agent* function is presented which calls the *Interpret* function:

```
Function MI_Agent($Q$, $Partners$, $Die$, $(\mathcal{T}, \mathcal{A}_0), \mathcal{FR}, \mathcal{BR}, \mathcal{WR}, \epsilon$)
Input: a queue of percept results $Q$, a set of partners $Partners$, a function $Die$ for the
termination process, a background knowledge set $(\mathcal{T}, \mathcal{A}_0)$, a set of forward chaining rules $\mathcal{FR}$, a
set of backward chaining rules $\mathcal{BR}$, a set of weighted rules $\mathcal{WR}$, and the desired precision of the
results $\epsilon$
Output: –
$CurrentI = \emptyset$;
$\mathfrak{A}'' = \{\emptyset\}$;
repeat
    $\Gamma := extractObservations(Q)$;
    $W := MAP(\Gamma, \mathcal{WR}, \mathcal{T})$ ;
    $\Gamma' := Select(W, \Gamma)$;
    $\mathfrak{A}' := filter(\lambda(\mathcal{A}) \bullet consistent_\Sigma(\mathcal{A}), map(\lambda(\mathcal{A}) \bullet \Gamma' \cup \mathcal{A} \cup \mathcal{A}_0 \cup forward\_chain(\Sigma, \mathcal{FR}, \Gamma' \cup \mathcal{A} \cup \mathcal{A}_0),$
        $\{select(MAP(\Gamma' \cup \mathcal{A} \cup \mathcal{A}_0, \mathcal{WR}, \mathcal{T}), \Gamma' \cup \mathcal{A} \cup \mathcal{A}_0) \mid \mathcal{A} \in \mathfrak{A}''\}))$;
    $(\mathfrak{A}'', NewI, \Delta_1, \Delta_2) := Interpret(\mathfrak{A}', CurrentI, \Gamma', \mathcal{T}, \mathcal{FR}, \mathcal{BR}, \mathcal{WR} \cup \Gamma, \epsilon)$;
    $CurrentI := NewI$;
    $Communicate(\Delta_1, \Delta_2, Partners)$;
    $\mathfrak{A}'' := manage\_agenda(\mathfrak{A}'')$;
until $Die()$ ;
```

where the *filter* function is defined as follows:

$$filter(f, X) = \bigcup_{x \in X} \begin{cases} \{x\} & \text{if } f(x) = true \\ \emptyset & \text{else} \end{cases} \tag{12}$$

The *filter* function takes as parameters a function f and a set X and returns a set consisting of the values of f applied to every element x of X.

In the MI_Agent function, the current interpretation $CurrentI$ and the agenda $\mathfrak{A}''$ are initialized to empty set. Since the agent performance is an incremental process, it is defined by a $repeat - until$ loop. The percept results Γ are sent by KDMA and HCI to the queue Q. In order to take the observations Γ from the queue Q, the MI_Agent calls the *extractObservations* function.
The $MAP(\Gamma \cup \mathcal{A}, \mathcal{WR}, \mathcal{T})$ function determines the most probable world of observations Γ with respect to a set of weighted rules $\mathcal{WR}$ and the Tbox $\mathcal{T}$. This function performs actually the mentioned MAP process in Chapter 2. It returns a vector W which consists of a set of zeros and ones assigned to the ground atoms of the considered world. The assertions with assigned zeros and ones are called respectively, negative and positive assertions.
The *Select*(W, Γ) function selects the positive assertions from the bit vector W in the input Abox Γ. The selected positive assertions which require explanations are also known as fiat assertions. This operation returns as output an Abox Γ' which has the following characteristic: $\Gamma' \subseteq \Gamma$.
In the next step, a set of forward chaining rules $\mathcal{FR}$ is applied to all the Aboxes of $\mathfrak{A}''$. The generated assertions in this process are added to the to the Abox $\mathcal{A}$. In the next step, only the consistent Aboxes are selected and the other inconsistent Aboxes are not considered for the next steps.

In the next step, the *Interpret* function is called to determine the new agenda $\mathfrak{A}''$, the new interpretation $NewI$ and the Abox differences Δ_1 and Δ_2 for additions and omissions among $CurrentI$ and $NewI$. Afterwards, the $CurrentI$ is set to the $NewI$ and the MI_Agent function communicates the Abox differences Δ_1 and Δ_2 to the partners. Additionally, the Tbox $\mathcal{T}$, the set of forward chaining rules $\mathcal{FR}$, the set of backward chaining rules $\mathcal{BR}$, and the set of weighted rules $\mathcal{WR}$ can be learnt by the *Learn* function. The termination condition of the MI_Agent function is that the *Die*() function is true.

Note that the MI_Agent waits at the function call *extractObservations*(Q) if $Q = \emptyset$.

After presenting the above algorithms, the mentioned unanswered questions can be discussed. A reason for performing the interpretation process and explaining the fiat assertions is that the probability of $P(\mathcal{A}, \mathcal{A}', \mathcal{R}, \mathcal{WR}, \mathcal{T})$ will increase through the interpretation process. In other words, by explaining the observations the agent's belief to the percepts will increase. This shows a new perspective for performing the interpretation process.

The answer to the question whether there is any measure for stopping the interpretation process, is indeed positive. This is expressed by $\mid p_i - p_{i-1} \mid < \frac{\epsilon}{i}$ which is the termination condition Ξ of the algorithm. The reason for selecting $\frac{\epsilon}{i}$ and not ϵ as the upper limit for the termination condition is to terminate the oscillation behaviour of the results. In other words, the precision interval is tightened step by step during the interpretation process. The function *manage_agenda* is explained Section 6. Before, we dicuss an example for interpreting a single video shot in Section 4, and a scene in Section 5.

4 Video Shot Interpretation

One of the main innovation introduced in the previous section, namely the introduction of a probabilistic preference measure to control the space of possible interpretations, is exemplified here using examples inspired by the environmental domain used in the project CASAM.

We have to mention that this example is not constructed to show the possible branchings through the interpretation process. The purpose of this example is to show how the probabilities of the most probable world of observations $P(\mathcal{A}_0, \mathcal{A}, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ behave during the interpretation process.

At the beginning of this example, the **signature** of the knowledge base is presented. The set of all concept names **CN** is divided into two disjoint sets **Events** and **PhysicalThings** such that

CN $=$ **Events** $\cup$ **PhysicalThings** where these two sets are defined as follows:
Events $= \{\mathit{CarEntry}, \mathit{EnvConference}, \mathit{EnvProt}, \mathit{HealthProt}\}$
PhysicalThings $= \{\mathit{Car}, \mathit{DoorSlam}, \mathit{Building}, \mathit{Environment}, \mathit{Agency}\}$
$\mathit{EnvConference}$, $\mathit{EnvProt}$ and $\mathit{HealthProt}$ denote respectively environmental conference, environmental protection and health protection.

The set of role names **RN** is defined as follows:

$$\mathbf{RN} = \{\mathit{Causes}, \mathit{OccursAt}, \mathit{HasAgency}, \mathit{HasTopic}, \mathit{HasSubject}, \mathit{HasObject}, \mathit{HasEffect}, \mathit{HasSubEvent}, \mathit{HasLocation}\}$$

In the following, the set of individual names **IN** is given:
IN $= \{C_1, DS_1, ES_1, Ind_{42}, Ind_{43}, Ind_{44}, Ind_{45}, Ind_{46}, Ind_{47}, Ind_{48}\}$

Note that the notations in this example are based on Alchemy notations i.e. the instance-, concept- and role names begin with capital letters. In the following, the set of forward chaining rules $\mathcal{FR}$ is defined:
$\mathcal{FR} = \{\forall x\ \ \mathit{CarEntry}(x) \rightarrow \exists y\ \ \mathit{Building}(y), \mathit{OccursAt}(x,y),$
$\forall x\ \ \mathit{EnvConference}(x) \rightarrow \exists y\ \ \mathit{Environment}(y), \mathit{HasTopic}(x,y),$
$\forall x\ \ \mathit{EnvProt}(x) \rightarrow \exists y\ \ \mathit{Agency}(y), \mathit{HasAgency}(x,y)\}$
Similarly, the set of backward chaining rules $\mathcal{BR}$ is given as follows:
$\mathcal{BR} = \{\mathit{Causes}(x,y) \leftarrow \mathit{CarEntry}(z), \mathit{HasObject}(z,x), \mathit{HasEffect}(z,y), \mathit{Car}(x), \mathit{DoorSlam}(y),$
$\mathit{OccursAt}(x,y) \leftarrow \mathit{EnvConference}(z), \mathit{HasSubEvent}(z,x), \mathit{HasLocation}(z,y), \mathit{CarEntry}(x), \mathit{Building}(y),$
$\mathit{HasTopic}(x,y) \leftarrow \mathit{EnvProt}(z), \mathit{HasSubEvent}(z,x), \mathit{HasObject}(z,y), \mathit{EnvConference}(x), \mathit{Environment}(y),$
$\mathit{HasAgency}(x,y) \leftarrow \mathit{HealthProt}(z), \mathit{HasObject}(z,x), \mathit{HasSubject}(z,y), \mathit{EnvProt}(x), \mathit{Agency}(y)\}$

In the following, a set of weighted rules $\mathcal{WR}$ is defined where all rules have the same high weight equal to 5:
$\mathcal{WR} = \{5\ \forall x,y,z\ \mathit{CarEntry}(z) \wedge \mathit{HasObject}(z,x) \wedge \mathit{HasEffect}(z,y) \rightarrow \mathit{Car}(x) \wedge \mathit{DoorSlam}(y) \wedge \mathit{Causes}(x,y),$
$5\ \forall x,y,z\ \mathit{EnvConference}(z) \wedge \mathit{HasSubEvent}(z,x) \wedge \mathit{HasLocation}(z,y) \rightarrow \mathit{CarEntry}(x) \wedge \mathit{Building}(y) \wedge \mathit{OccursAt}(x,y),$
$5\ \forall x,y,z\ \ \mathit{EnvProt}(z) \wedge \mathit{HasSubEvent}(z,x) \wedge \mathit{HasObject}(z,y) \ \rightarrow\ \mathit{EnvConference}(x) \wedge \mathit{Environment}(y) \wedge \mathit{HasTopic}(x,y),$
$5\ \forall x,y,z\ \mathit{HealthProt}(z) \wedge \mathit{HasObject}(z,x) \wedge \mathit{HasSubject}(z,y) \rightarrow \mathit{EnvProt}(x) \wedge \mathit{Agency}(y) \wedge \mathit{HasAgency}(x,y)\}$

Note that the weighted rules $\mathcal{WR}$ and their weights can be learnt by the machine learning component. The selected initial value for ϵ in this example is 0.05. In the following, Δ_1 and Δ_2 denote respectively the set of assertions hypothesized by a forward chaining rule and the set of assertions generated by a backward chaining rule at each interpretation level.

Let us assume that the media interpretation agent receives the following weighted Abox $\mathcal{A}$:
$\mathcal{A} = \{1.3\ \mathit{Car}(C_1), 1.2\ \mathit{DoorSlam}(DS_1), -0.3\ \mathit{EngineSound}(ES_1), \mathit{Causes}(C_1, DS_1)\}$
The first applied operation to A is the MAP function which returns the bit vector $W = \langle 1,1,0,1 \rangle$. This vector is composed of positive and negative events (bits). By applying the Select function to W and the input Abox $\mathcal{A}$, the assertions from the input Abox $\mathcal{A}$ are selected that correspond to the positive events in W. Additionally, the assigned weights to the positive assertions are also taken from the input Abox $\mathcal{A}$. In the following, Abox A_0 is depicted which contains the positive assertions:
$\mathcal{A}_0 = \{1.3\ \mathit{Car}(C_1), 1.2\ \mathit{DoorSlam}(DS_1), \mathit{Causes}(C_1, DS_1)\}$
At this step, $p_0 = P(\mathcal{A}_0, \mathcal{A}_0, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.755$. Since no appropriate forward chaining rule from $\mathcal{FR}$ is applicable to Abox $\mathcal{A}_0$, $\Delta_1 = \emptyset$ and as a result $\mathcal{A}_0 = \mathcal{A}_0 \cup \emptyset$. The next step is the performance of *backward_chain* function where the next backward chaining rule from $\mathcal{BR}$ can be applied to Abox $\mathcal{A}_0$:
$\mathit{Causes}(x,y) \leftarrow \mathit{CarEntry}(z), \mathit{HasObject}(z,x), \mathit{HasEffect}(z,y), \mathit{Car}(x), \mathit{DoorSlam}(y)$

Consequently, by applying the above rule the next set of assertions is hypothesized:
$\Delta_2 = \{\mathit{CarEntry}(Ind_{42}), \mathit{HasObject}(Ind_{42}, C_1), \mathit{HasEffect}(Ind_{42}, DS_1)\}$

which are considered as strict assertions. Consequently, $\mathcal{A}_1$ is defined as follows: $\mathcal{A}_1 = \mathcal{A}_0 \cup \Delta_2$.
In the above Abox, $p_1 = P(\mathcal{A}_0, \mathcal{A}_1, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.993$. As it can be seen, $p_1 > p_0$ i.e. $P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ increases by adding the new hypothesized assertions. This shows that the new assertions are considered as additional support. The termination condition of the algorithm is not

fulfilled therefore the algorithm continues processing. At this level, it is still not known whether Abox $\mathcal{A}_1$ can be considered as the final interpretation Abox. Thus, this process is continued with another level. Consider the next forward chaining rule:

$\forall x \;\; CarEntry(x) \rightarrow \exists y \;\; Building(y), OccursAt(x,y)$

By applying the above rule, the next set of assertions is generated namely:

$\Delta_1 = \{Building(Ind_{43}), OccursAt(Ind_{42}, Ind_{43})\}$

The new generated assertions are also considered as strict assertions. In the following, the expanded Abox $\mathcal{A}_1$ is defined as follows: $\mathcal{A}_1 = \mathcal{A}_1 \cup \Delta_1$.

Let us assume the next backward chaining rule from $\mathcal{BR}$:

$OccursAt(x,y) \leftarrow EnvConference(z), HasSubEvent(z,x), HasLocation(z,y), CarEntry(x), Building(y)$

Consequently, by applying the above abduction rule the next set of assertions is hypothesized:

$\Delta_2 = \{EnvConference(Ind_{44}), HasSubEvent(Ind_{44}, Ind_{42}), HasLocation(Ind_{44}, Ind_{43})\}$

which are considered as strict assertions. Consequently, $A_2 = A_1 \cup \Delta_2$.

In the above Abox, $p_2 = P(\mathcal{A}_0, \mathcal{A}_2, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.988$. As it can be seen, $p_2 < p_1$ i.e. $P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ decreases slightly by adding the new hypothesized assertions. Since the termination condition of the algorithm is fulfilled, Abox $\mathcal{A}_1$ can be considered as the final interpretation Abox. To realize how the further behaviour of the probabilities is, this process is continued. Consider the next forward chaining rule from $\mathcal{FR}$:

$\forall x \;\; EnvConference(x) \rightarrow \exists y \;\; Environment(y), HasTopic(x,y)$

By applying the above rule, new assertions are generated.

$\Delta_1 = \{Environment(Ind_{45}), HasTopic(Ind_{44}, Ind_{45})\}$

In the following, the expanded Abox $\mathcal{A}_2$ is defined: $\mathcal{A}_2 = \mathcal{A}_2 \cup \Delta_1$.
Consider the next backward chaining rule from $\mathcal{BR}$:

$HasTopic(x,y) \leftarrow EnvProt(z), HasSubEvent(z,x), HasObject(z,y), EnvConference(x), Environment(y)$
By applying the above abduction rule, the following set of assertions is hypothesized:

$\Delta_2 = \{EnvProt(Ind_{46}), HasSubEvent(Ind_{46}, Ind_{44}), HasObject(Ind_{46}, Ind_{45})\}$

which are considered as strict assertions. In the following, $\mathcal{A}_3$ is defined as follows $A_3 = A_2 \cup \Delta_2$.
In the above Abox $\mathcal{A}_3$, $p_3 = P(\mathcal{A}_0, \mathcal{A}_3, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.99$. As it can be seen, $p_3 > p_2$, i.e.
$P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ increases slightly by adding the new hypothesized assertions.

Consider the next forward chaining rule:

$\forall x \;\; EnvProt(x) \rightarrow \exists y \;\; Agency(y), HasAgency(x,y)$

By applying the above rule, the next assertions are generated:

$\Delta_1 = \{Agency(Ind_{47}), HasAgency(Ind_{46}, Ind_{47})\}$

As a result, the expanded Abox $\mathcal{A}_3$ is presented as follows: $\mathcal{A}_3 = \mathcal{A}_3 \cup \Delta_1$.

Let us consider the next backward chaining rule from $\mathcal{BR}$:

$HasAgency(x,y) \leftarrow HealthProt(z), HasObject(z,x), HasSubject(z,y), EnvProt(x), Agency(y)$

Consequently, new assertions are hypothesized by applying the above abduction rule, namely:

$\Delta_2 = \{HealthProt(Ind_{48}), HasObject(Ind_{48}, Ind_{46}), HasSubject(Ind_{48}, Ind_{47})\}$

which are considered as strict assertions. Consequently, $\mathcal{A}_4$ is defined as follows: $A_4 = A_3 \cup \Delta_2$.

In the above Abox, $p_4 = P(\mathcal{A}_0, \mathcal{A}_4, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.985$. As it can be seen, $p_4 < p_3$, i.e.
$P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ decreases slightly by adding the new hypothesized assertions.

Evaluation of the Results:

The determined probability values $P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$ of this example are summarized in the next table which shows clearly the behaviour of the probabilities stepwise after performing the interpretation process:

i	*Abox* $\mathcal{A}_i$	$p_i = P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$
0	$\mathcal{A}_0$	$p_0 = 0.755$
1	$\mathcal{A}_1$	$p_1 = 0.993$
2	$\mathcal{A}_2$	$p_2 = 0.988$
3	$\mathcal{A}_3$	$p_3 = 0.99$
4	$\mathcal{A}_4$	$p_4 = 0.985$

Table 1. Summary of the probability values

In the above table, variable i denotes the successive levels of the interpretation process. In this example, the interpretation process is consecutively performed four times. As it can be seen, through the first interpretation level the probability p_1 increases strongly in comparison to p_0. By performing the second, third and the forth interpretation levels, the probability values decrease slightly in comparison to p_1. This means no significant changes can be seen in the results. In other words, the determination of $\mathcal{A}_3$ and $\mathcal{A}_4$ were not required at all. But the determination of $\mathcal{A}_2$ was required to realize the slight difference $|p_2 - p_1| < \frac{\epsilon}{2}$. Consequently, Abox $\mathcal{A}_1$ is considered as the final interpretation Abox.

5 Preference-based Scene Interpretation

In this example, we discuss how the video shot interpretation process can be performed by considering the results of two consecutive video shots. For the interpretation of each video shot we require information about the previous video shots otherwise the interpretation process does not work correctly. The question is which assertions have to be considered from the previous video shots. As it was discussed in this paper we would like to consider the assertions from the previous video shots which increase $P(\mathcal{A}_0, \mathcal{A}_i, \mathcal{R}, \mathcal{WR}, \mathcal{T})$. At the beginning of this example, the signature of the knowledge base is presented. The set of the concept names **CN** is divided into two disjoint sets **Events** and **PhysicalThings** which are described as follows:

Events $= \{CarEntry, CarExit, CarRide\}$

PhysicalThings $= \{Car, DoorSlam\}$

Additionally, the set of the role names **RN** and the set of the individual names **IN** are represented as follows:

RN $= \{Causes, HasObject, HasEffect, Before, HasStartEvent, HasEndEvent\}$

IN $= \{C_1, C_2, DS_1, DS_2, Ind_{41}, Ind_{42}, Ind_{44}\}$

The Tbox $\mathcal{T}$ contains the axiom $CarEntry \sqsubseteq \neg CarExit$. In the following, the set of forward chaining rules $\mathcal{FR}$ is given:

$\mathcal{FR} = \{$

$\forall x, xl, y, yl, w, z\ AudioSeg(x), HasSegLoc(x, xl), VideoSeg(y), HasSegLoc(y, yl), IsSmaller(xl, yl),$
$Depicts(x, w), Depicts(y, z), CarEntry(w), CarEntry(z) \rightarrow Before(z, w),$

$\forall x, xl, y, yl, w, z\ AudioSeg(x), HasSegLoc(x, xl), VideoSeg(y), HasSegLoc(y, yl), IsSmaller(xl, yl),$
$Depicts(x, w), Depicts(y, z), CarEntry(w), CarExit(z) \rightarrow Before(z, w),$

$\forall x, xl, y, yl, w, z\ AudioSeg(x), HasSegLoc(x, xl), VideoSeg(y), HasSegLoc(y, yl), IsSmaller(xl, yl),$
$Depicts(x, w), Depicts(y, z), CarExit(w), CarEntry(z) \rightarrow Before(z, w),$

$\forall x, xl, y, yl, w, z\ AudioSeg(x), HasSegLoc(x, xl), VideoSeg(y), HasSegLoc(y, yl), IsSmaller(xl, yl),$
$Depicts(x, w), Depicts(y, z), CarExit(w), CarExit(z) \rightarrow Before(z, w)\}$

where $AudioSeg$, $HasSegLoc$ and $VideoSeg$ denote $AudioSegment$, $HasSegmentLocator$ and $VideoSegment$ respectively. Note that the concepts and roles in $\mathcal{FR}$ which are not given in **CN** and **RN** appear only in the multimedia content ontology. The multimedia content ontology determines the structure of the multimedia document. Additionally, it determines whether the concpts are originated from video, audio or text. The above rules mean that the concept assertion $CarEntry$ or $CarExit$ from the first shot appear chronologically before the concept assertion $CarEntry$ or $CarExit$ from the second video shot. The set of backward chaining rules $\mathcal{BR}$ is presented as follows:

$$\begin{aligned}\mathcal{BR} = \{&Causes(x,y) \leftarrow CarEntry(z), HasObject(z,x), HasEffect(z,y), Car(x), DoorSlam(y),\\ &Causes(x,y) \leftarrow CarExit(z), HasObject(z,x), HasEffect(z,y), Car(x), DoorSlam(y),\\ &Before(x,y) \leftarrow CarRide(z), HasStartEvent(z,x), HasEndEvent(z,y), CarEntry(x), CarExit(y)\}\end{aligned}$$

Additionally, the set of weighted rules is defined as follows:

$$\begin{aligned}\mathcal{WR} = \{&5\ \forall x,y,z\ CarEntry(z) \wedge HasObject(z,x) \wedge HasEffect(z,y) \Rightarrow Car(x) \wedge DoorSlam(y) \wedge Causes(x,y),\\ &5\ \forall x,y,z\ CarExit(z) \wedge HasObject(z,x) \wedge HasEffect(z,y) \Rightarrow Car(x) \wedge DoorSlam(y) \wedge Causes(x,y),\\ &5\ \forall x,y,z,k,m\ \ CarRide(z) \wedge HasStartEvent(z,x) \wedge HasEndEvent(z,y) \wedge HasObject(x,k) \wedge\\ &\qquad HasObject(y,m) \Rightarrow CarEntry(x) \wedge CarExit(y) \wedge Car(k) \wedge Car(m) \wedge k = m\}\end{aligned}$$

Consider the next figure as the first video shot of a video:

Let us assume that the analysis results of the first video shot represented in the Abox $\mathcal{A}_1$ are sent to the queue Q:

$\mathcal{A}_1 = \{1.3\ Car(C_1), 1.2\ DoorSlam(DS_1), Causes(C_1, DS_1)\}$

For the interpretation of the first video shot, we will call the function $MI\_Agent(Q, Partners, Die, (\mathcal{T}, \mathcal{A}_0), \mathcal{FR}, \mathcal{BR}, \mathcal{WR}, \epsilon)$. At the beginning of this function, there are initializations for some variables, namely $CurrentI = \emptyset$ and $\mathfrak{A}'' = \{\emptyset\}$. Afterwards extracting observations from the queue Q is performed, which leads to $\Gamma = \mathcal{A}_1$. Determination of the most probable world $W = \langle 1, 1, 1\rangle$ is performed in the next step and selecting the positive assertions and their related weights determines $\Gamma' = \Gamma$. At this step, $\mathcal{A} = \emptyset$ since $\mathfrak{A}'' = \{\emptyset\}$. Additionally, $\mathcal{A}_0 = \emptyset$. Consequently, $MAP(\Gamma', \mathcal{WR}, \mathcal{T}) = W$ and $Select(W, \Gamma') = \Gamma'$. $forward\_Chain(\Sigma, \mathcal{FR}, \Gamma') = \emptyset$ since there is no forward chaining rule applicable to Γ'. $\mathfrak{A}' = \Gamma'$. The $Interpret(\mathfrak{A}', CurrentI, \Gamma', \mathcal{T}, \mathcal{FR}, \mathcal{BR}, \mathcal{WR} \cup \Gamma, \epsilon)$ is called in the next step which determines $p_0 = P(\Gamma', \Gamma', \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.733$. The $Interpret$ function calls CAE function which returns $\mathfrak{A}' = \{\Gamma' \cup \Delta_1, \Gamma' \cup \Delta_2\}$ where the two possible explanations Δ' and Δ'' are defined as follows:

$\Delta_1 = \{CarEntry(Ind_{41}), HasObject(Ind_{41}, C_1), HasEffect(Ind_{41}, DS_1)\}$

$\Delta_2 = \{CarExit(Ind_{41}), HasObject(Ind_{41}, C_1), HasEffect(Ind_{41}, DS_1)\}$

Each of the above interpretation Aboxes have scoring values:

$p_1 = P(\Gamma', \Gamma' \cup \Delta_1, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.941$ and $p_1 = P(\Gamma', \Gamma' \cup \Delta_2, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.935$. $NewI = \Gamma' \cup \Delta_1$ since this is the interpretation Abox with the maximum scoring value. The termination condition is not fulfilled since $p_1 - p_0 = 0.208 > 0.05$. The Abox difference for additions is defined as follows: $\Delta_{add} = NewI - CurrentI = NewI - \emptyset = NewI$. Simiarly, $\Delta_{omi} = \emptyset$ is the Abox difference for omissions. The CAE function returns $NewI$, $\mathfrak{A}'$ and the Abox differences Δ_{add} and Δ_{omi} to the $Interpret$ function. Consider the next figure depicts the second video shot:

Assume that the analysis results of the second video shot given in the next Abox are sent to the queue Q:

$\mathcal{A}_2 = \{1.3\ Car(C_2), 1.2\ DoorSlam(DS_2), Causes(C_2, DS_2)\}$

Similarly, for the interpretation of the second video shot we will call the function $MI\_Agent(Q, Partners, Die, (\mathcal{T}, \mathcal{A}_0), \mathcal{FR}, \mathcal{BR}, \mathcal{WR}, \epsilon)$. The observation extracttion process from Q leads to $\Gamma = \mathcal{A}_2$. Afterwards, the most probable world $W = \langle 1, 1, 1\rangle$ is determined and applying $Select$ function on W gives $\Gamma' = \mathcal{A}_2$.

Consider $\mathcal{A} \in \mathfrak{A}''$ where $\mathfrak{A}'' = \{\mathcal{A}_1 \cup \Delta_1, \mathcal{A}_1 \cup \Delta_2\}$. $\Gamma' \cup \mathcal{A} = \{\mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_1, \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_2\}$. Applying $MAP(\Gamma' \cup \mathcal{A}, \mathcal{WR}, \mathcal{T})$ gives $W = \langle 1, \ldots, 1\rangle$ and applying the $Select(W, \Gamma' \cup \mathcal{A})$ function gives $\{\mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_1, \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_2\}$. Since no forward chaining rule is applicable to the above set and this set contains consistent Aboxes $\mathfrak{A}' = \{\mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_1, \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_2\}$. In the next step, the function $Interpret(\mathfrak{A}', CurrentI, \Gamma', \mathcal{T}, \mathcal{FR}, \mathcal{BR}, \mathcal{WR} \cup \Gamma, \epsilon)$ is called which determines $P(\Gamma', \Gamma', \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.733$. Afterwards, the CAE function is called which determines the next exaplanations:

$\Delta_3 = \{CarEntry(Ind_{42}), HasObject(Ind_{41}, C_2), HasEffect(Ind_{41}, DS_2)\}$

$\Delta_4 = \{CarExit(Ind_{42}), HasObject(Ind_{41}, C_2), HasEffect(Ind_{41}, DS_2)\}$

The CAE function generates the following agenda which contains all possible interpretation Aboxes: $\{I_1, I_2, I_3, I_4\}$

where:

$I_1 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_1 \cup \Delta_3 \qquad I_2 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_1 \cup \Delta_4$
$I_3 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_2 \cup \Delta_3 \qquad I_4 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_2 \cup \Delta_4$

Afterwards applies the forward chaining rules on the above agenda. A new assertion $Before(Ind_{41}, Ind_{42})$ is generated and added to the four interpretation Aboxes. In the following, the possible four interpretation Aboxes are given:

$I_1 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_1 \cup \Delta_3 \cup \{Before(Ind_{41}, Ind_{42})\}$
$I_2 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_1 \cup \Delta_4 \cup \{Before(Ind_{41}, Ind_{42})\}$
$I_3 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_2 \cup \Delta_3 \cup \{Before(Ind_{41}, Ind_{42})\}$
$I_4 = \mathcal{A}_2 \cup \mathcal{A}_1 \cup \Delta_2 \cup \Delta_4 \cup \{Before(Ind_{41}, Ind_{42})\}$

Afterwards the backward chaining rule is applied which generates the following set only for the interpretation Abox I_2:

$\Delta = \{CarRide(Ind_{44}), HasStartEvent(Ind_{44}, Ind_{41}), HasEndEvent(Ind_{44}, Ind_{42})\}$

Consequently $I_2 = I_2 \cup \Delta$. The interpretation Aboxes have the next scoring values:

$P(\mathcal{A}_1 \cup \mathcal{A}_2, I_1, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.964$
$P(\mathcal{A}_1 \cup \mathcal{A}_2, I_2, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.978$
$P(\mathcal{A}_1 \cup \mathcal{A}_2, I_3, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.952$
$P(\mathcal{A}_1 \cup \mathcal{A}_2, I_4, \mathcal{R}, \mathcal{WR}, \mathcal{T}) = 0.959$

The above values show that the interpretation Abox I_2 has a higher scoring value than the other interpretation Aboxes. Therefore the final interpretation Abox is $NewI = I_2$. The Abox differences for additions and omissions are defined as follows:

$\Delta_{add} = \mathcal{A}_2 \cup \Delta_4 \cup \Delta \cup \{Before(Ind_{41}, Ind_{42})\} \qquad \Delta_{omi} = \emptyset$

For the next interpretation steps the agenda can continue with I_2 and eliminate the other interpretation Aboxes since this Abox has a higher scoring value.

6 Manage Agenda

In this section, we introduce some techniques which improves the performance of the agenda.

- Elimination of the interpretation Aboxes: This technique is applied if there are multiple interpretation Aboxes with different scoring values where one of the Aboxes has a higher scoring value. At this step, we can select this Abox, eliminate the remaining interpretation Aboxes and continue the interpretation process with the selected Abox.
- Combining the interpretation Aboxes: Consider the interpretation Aboxes $I_1, \ldots, I_n$. In order to determine the final interpretation Abox, the MAP process can be applied to the union of all interpretation Aboxes $I_1 \cup \ldots \cup I_n$.
- Shrinking the interpretation Aboxes: By applying this technique, we can decide which assertions from the previous video shots have to be considered for the interpretation process of the following video shots since considering all assertions of the previous video shots will slow down the interpretation process. We believe that only the high level concept assertions from the previous video shots play an important role and not the low level concept assertions.

7 Summary

For multimedia interpretation, a semantically well-founded formalization is required. In accordance with previous work, in CASAM a well-founded abduction-based approach is pursued. Extending previous work, abduction is controlled by probabilistic knowledge, and it is done in terms of first-order logic. Rather than merely using abduction for computing explanation with which observations are entailed, the approach presented in this paper, uses a probabilistic logic to motivate the explanation endeavor by increasing the belief in the observations. Hence, there exists a certain utility for an agent for the computational resources it spends for generating explanations. Thus, we have presented a first attempt to more appropriately model a media interpretation agent.

References

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press.

[Castano et al., 2008] Castano, S., Espinosa, S., Ferrara, A., Karkaletsis, V., Kaya, A., Möller, R., Montanelli, S., Petasis, G., and Wessel, M. (2008). Multimedia interpretation for dynamic ontology evolution. In *Journal of Logic and Computation.* Oxford University Press.

[Domingos and Richardson, 2007] Domingos, P. and Richardson, M. (2007). Markov logic: A unifying framework for statistical relational learning. In Getoor, L. and Taskar, B., editors, *Introduction to Statistical Relational Learning*, pages 339–371. Cambridge, MA: MIT Press.

[Gries and Möller, 2010] Gries, O. and Möller, R. (2010). Gibbs sampling in probabilistic description logics with deterministic dependencies. In *Proc. of the First International Workshop on Uncertainty in Description Logics, Edinburgh.*

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA.

Relationships between Probabilistic Description and First-Order Logics

Pavel Klinov and Bijan Parsia

School of Computer Science
University of Manchester, United Kingdom
{pklinov|bparsia}@cs.man.ac.uk

Abstract This paper analyzes the probabilistic description logic P-$\mathcal{SHIQ}$ as a fragment of first-order probabilistic logic (FOPL). P-$\mathcal{SHIQ}$ was suggested as a language that is capable of representing and reasoning about different kinds of uncertainty in ontologies, namely generic probabilistic relationships between concepts and probabilistic facts about individuals. However, some semantic properties of P-$\mathcal{SHIQ}$ have been unclear which raised concerns regarding whether it could be used for representing probabilistic ontologies. In this paper we provide an insight into its semantics by translating P-$\mathcal{SHIQ}$ into FOPL with a specific semantics based on possible worlds. From that reduction, we show that some of the restrictions of P-$\mathcal{SHIQ}$ are fundamental and sketch alternative semantic foundations for a probabilistic description logic.

1 Introduction and Motivation

One common complaint about description logic (DL) based ontology languages, such as the Web Ontology Language (OWL), is they fail to support non-classical uncertainty, in particular, probability. One answer to this complaint is the P-$\mathcal{SH}$ family of logics which allow for the incorporation of probabilistic formulae as an extension of the familiar and widely used $\mathcal{SH}$ DLs [1] [2]. Unlike Bayesian extensions to DLs and OWL, the P-$\mathcal{SH}$ family consists of proper extensions to the syntax and semantics of the underlying logic and inference services. These logics are also decidable, generally of the same worst case complexity as the base logic, and can be implemented on top of existing DL reasoners.

However, there are several issues with the P-$\mathcal{SH}$ family both from an expressivity and from a theoretical point of view. First, it has not been fully clear how it actually combines statistical and subjective probabilities. Second, probabilistic ABoxes have a number of strong restrictions (including no support of roles assertions between probabilistic individuals and only one probabilistic individual per ABox).

Often, insight into a DL (and associated extensions and reasoning techniques) has followed by considering its standard first-order translation, that is, in considering it as a fragment of first order-logics. In this paper, we attempt to apply this methodology to the P-$\mathcal{SH}$ family by considering them as fragments of a first-order logic extended with various forms of probability (FOPL). We show that we can understand P-$\mathcal{SH}$ logics as fragments of FOPL and explain its limitations on the basis of the known properties of FOPL with semantics based on possible worlds. Finally, we sketch another fragment of FOPL which has different semantics and allows lifting of the P-$\mathcal{SH}$ restrictions.

2 Preliminaries

P-$\mathcal{SHIQ}$ We consider a particular representative of the P-$\mathcal{SH}$ family, named P-$\mathcal{SHIQ}$, whose syntactic constructs include those of $\mathcal{SHIQ}$ together with *conditional constraints*. Constraints are expressions of the form $(D|C)[l,u]$ where D, C are $\mathcal{SHIQ}$ concept expressions (called *conclusion* and *evidence* respectively) and $[l,u] \subseteq [0,1]$.

A probabilistic TBox (PTBox) is a 2-tuple $PT = (\mathcal{O}, \mathcal{P})$ where $\mathcal{O}$ is a classical DL ontology and P is a finite set of conditional constraints. Informally, a PTBox axiom $(D|C)[l,u]$ means that "if a randomly chosen individual belongs to C, its probability of belonging to D is in $[l,u]$". A probabilistic ABox (PABox) is a finite set of conditional constraints pertaining to a single probabilistic individual o. Set of all probabilistic individuals is denoted as $I_{\mathcal{P}}$. A probabilistic ontology $PO = (\mathcal{O}, \mathcal{P}, (\mathcal{P}_o)_{o \in I_P})$ is a combination of one PTBox and a set of PABoxes, one for each probabilistic individual.

The semantics of P-$\mathcal{SHIQ}$ is standardly explained in terms of the notion of a *possible world* which is defined with respect to a set of basic concepts Φ [2]. A possible world I is a set of DL concepts from Φ such that $\{a : C | C \in I\} \cup \{a : \neg C | C \notin I\}$ is satisfiable for a fresh individual a (in other words, possible worlds correspond to *realizable* concept types). The set of all possible worlds with respect to Φ is denoted as $\mathcal{I}_{\Phi}$. A world I satisfies a concept C denoted as $I \models C$ if $C \in I$. Satisfiability of basic concepts is inductively extended to concept expressions as usual.

A world I is said to be a *model* of a DL axiom η denoted as $I \models \eta$ if $\eta \cup \{a : C | C \in I\} \cup \{a : \neg C | C \notin I\}$ is satisfiable for a fresh individual a. A world I is a model of a classical DL knowledge base $\mathcal{O}$ denoted as $I \models \mathcal{O}$ if it is a model of all axioms of $\mathcal{O}$. Existence of such world is equivalent to the standard satisfiability in DL [2].

A probabilistic interpretation Pr is a discrete probability distribution over $\mathcal{I}_{\Phi}$. Pr is said to *satisfy* a DL knowledge base $\mathcal{O}$ denoted as $Pr \models KB$ iff $\forall I \in \mathcal{I}_{\Phi}, Pr(I) > 0 \Rightarrow I \models KB$. The probability of a concept C, denoted as $Pr(C)$, is defined as $\sum_{I \models C} Pr(I)$. $Pr(D|C)$ is used as an abbreviation for $Pr(C \sqcap D)/Pr(C)$ given $Pr(C) > 0$. A probabilistic interpretation Pr satisfies a conditional constraint $(D|C)[l,u]$, denoted as $Pr \models (D|C)[l,u]$, iff $Pr(C) = 0$ or $Pr(D|C) \in [l,u]$. Pr satisfies a set of conditional constraints F iff it satisfies each of the constraints. A PTBox $PT = (\mathcal{O}, \mathcal{P})$ is called *satisfiable* iff there exists an interpretation that satisfies $\mathcal{O} \cup \mathcal{P}$. Logical entailment is defined in a standard way [2][1].

First-Order Probabilistic Logic FOPL$_2$ is a probabilistic generalization of first-order logic aimed at capturing belief statements (the subscript 2 stands for the Type 2 semantics [3]), like "the probability that Tweety (a particular bird) flies is over 90%". It is very expressive allowing to attach probabilities to arbitrary first-order formulas. Its representational and computational properties have been thoroughly investigated, and the results of these investigations are applicable to its fragments.

The *syntax* of FOPL$_2$ is defined as follows [3]: assume a first-order alphabet Φ of function and predicate names, and a countable set of object variables X^o. *Object*

[1] P-$\mathcal{SHIQ}$, as it is presented in [2], is a non-monotonic formalism. However, we consider only its monotonic basis in this paper. Our position is that it must be clarified first, before proceeding to non-monotonic machinery, such as lexicographic entailment, built on its top.

terms are formed by closing X^o off under function application as usual. In addition, the language contains *field terms*, which range over reals (with 0 and 1 being distinguished constants) and probability terms of the form $w(\phi)$, where ϕ is a first-order formula. Field terms are closed off under applications of functions $\times, +$ on reals (the denotation $w(\phi|\psi) \leq t$ is the abbreviation of $w(\phi \wedge \psi) \leq t \times w(\psi)$). Then FOPL$_2$ formulas are defined as follows:

- If P is an n-ary predicate name in Φ and $t_1, \ldots, t_n$ are object terms, then $P(t_1, \ldots, t_n)$ is an atomic formula.
- If t_1, t_2 are field terms, then $t_1 \leq t_2$, $t_1 \geq t_2$, $t_1 < t_2$, $t_1 > t_2$, $t_1 = t_2$ are atomic formulas. Standard relationships between (in)equality relations are assumed.
- If ϕ, ψ are formulas and $x \in X^o$, then $\phi \wedge \psi$, $\phi \vee \psi$, $\forall(x)\phi$, $\exists(x)\phi$, $\neg\phi$ are formulas. Standard relationships between logical connectives and quantifiers are assumed.

A *probabilistic interpretation* (Type 2 probability structure in [3]) M is a tuple (D, S, π, μ), where D is a domain, S is a set of states, π is a function $S \times \Phi \rightarrow \Phi_D$ (where Φ_D is a set of predicates and functions over D) which preserves arity, and μ is a probability distribution over S. M together with a state s and a valuation v associates each object term o with an element of D ($[o]^{(M,s,v)} \in D$) and each field term f with a real number. (M, s, v) associates formulas with truth values (we write $(M, s, v) \models \phi$ if ϕ is true in (M, s, v)) as follows:

- $(M, s, v) \models P(x)$ if $v(x) \in \pi(s, P)$.
- $(M, s, v) \models t_1 < t_2$ if $[t_1]^{M,s,v} < [t_2]^{(M,s,v)}$.
- $(M, s, v) \models \forall(x)\phi$ if $(M, s, v[x/d]) \models \phi$ for all $d \in D$.

Other formulas, e.g. $\phi \wedge \psi$, $\neg\psi$, $t_1 = t_2$, etc. are defined as usual. It remains to define the mapping for the probability terms of the form $w(\phi)$: $[w(\phi)]^{(M,s,v)} = \mu\{s' \in S|(M, s', v) \models \phi\}$. As usual, a FOPL$_2$ formula is called *satisfiable* if there exists a tuple (M, s, v) in which the formula is true.

Note that, although FOPL$_2$ does not impose any restrictions on the set S (i.e. it can be any set over which a probability distribution can be defined). However, it is natural to associate states with possible interpretations of symbols in Φ over D (see [4]). Then the model structure can be simplified to (D, S, μ) since the interpretations are implicitly encoded in the states.

3 Mapping between P-$\mathcal{SHIQ}$ and FOPL$_2$

This section presents a mapping between P-$\mathcal{SHIQ}$ and FOPL$_2$. For brevity we will limit our attention to $\mathcal{ALC}$ concepts (calling the resulting logic P-$\mathcal{ALC}$) as the translation can be easily extended to more expressive DLs. We will show that it preserves entailments so that P-$\mathcal{SHIQ}$ can be viewed as a fragment of FOPL$_2$.

Basic Translation We define the injective function κ to be the mapping of syntactic constructs of P-$\mathcal{ALC}$ to FOPL$_2$[2]. It is a superset of the standard translation of $\mathcal{ALC}$ into

Table 1. Translation of P-$\mathcal{ALC}$ formulae into FOPL$_2$

P-$\mathcal{ALC}$	**FOPL$_2$**
$\kappa(A, var)$	$A(var)$
$\kappa(\neg C, var)$	$\neg(\kappa(C, var))$
$\kappa(R, var, var')$	$R(var, var')$
$\kappa(C \sqcap D, var)$	$\kappa(C, var) \wedge \kappa(D, var)$
$\kappa(C \sqcup D, var)$	$\kappa(C, var) \vee \kappa(D, var)$
$\kappa(\forall R.C, var)$	$\forall(var')(R(var, var') \rightarrow \kappa(C, var'))$
$\kappa(\exists R.C, var)$	$\exists(var')(R(var, var') \wedge \kappa(C, var'))$
$\kappa(a : C)$	$\kappa(C, x)[a/x]$
$\kappa((a, b) : R)$	$R[a/x, b/y]$
$\kappa(C \sqsubseteq D, x)$	$\forall(x)(\kappa(C, x) \rightarrow \kappa(D, x))$
$\kappa((B\|A)[l, u], x)$	$l \leq w(B(r)\|A(r)) \leq u$

FOL [5] (in the Table 3 A, B stand for concept names, R for a role name, C, D for concepts, r for a fresh constant, $var \in \{x, y\}; var' = x$ if $var = y$ and y if $var = x$).

This function transforms a P-$\mathcal{ALC}$ PTBox into a FOPL$_2$ theory. The most important thing is that it translates *generic* PTBox constraints into *ground* probabilistic formulas for a fresh constant r. The implications of this will be discussed in Section 4.

Faithfulness We next show that this translation is faithful by establishing correspondence between models in P-$\mathcal{ALC}$ and FOPL$_2$. Observe, that in contrast to [6], here we consider the natural choice of states in Type 2 model structure in which they correspond to first-order models of the knowledge base.

Theorem 1. *Let $PT = (\mathcal{O}, \mathcal{P})$ be a PTBox in P-$\mathcal{ALC}$ and $F = \{\kappa(\phi)|\phi \in \mathcal{O} \cup \mathcal{P}\}$ be the corresponding FOPL$_2$ theory. Then for every P-$\mathcal{ALC}$ model Pr of PT there exists a corresponding Type 2 structure $M = (D, S, \mu)$ such that 1) $M \models \kappa(\phi)$ for all $\phi \in \mathcal{O}$ and 2) $M \models l \leq w(B(r)|A(r)) \leq u$ for all conditional constraints $(B|A)[l, u]$ in $\mathcal{P}$, and vice versa, where κ is defined according to Table 1.*

Proof. We prove only ($\Rightarrow$). Let $Pr : \mathcal{I}_\Phi \rightarrow [0, 1]$ be a model of PT. Pr satisfies classical ontology $\mathcal{O}$ so there exists a classical model $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ of $\mathcal{O}$. We first extend $\Delta^\mathcal{I}$ to ensure that all possible worlds are realizable over it (one possibility is to take the disjoint union of all realizations)[3]. Then we construct a Type 2 structure (D, S, μ) as follows: let $D = \Delta^\mathcal{I}$ and S be the set of all interpretations of predicate names (translations of concept and roles names) and r over $\Delta^\mathcal{I}$ that satisfy classical formulas in F. S must be non-empty: let $s_\mathcal{I}$ be such that $s_\mathcal{I}(P) = \kappa^{-1}(P)^\mathcal{I}$ for all predicate names and $s_\mathcal{I}(r)$ is an arbitrary domain element. Since r is a fresh constant and κ encompasses a standard and faithful translation from $\mathcal{ALC}$ to FOL, $s_\mathcal{I}$ is a model of all classical formulas in F and therefore *1)* holds.

[2] For a possible world $I = \{C_i\}$ we use the notation $\kappa(I)$ to denote the set $\{\kappa(C_i)\}$

[3] This is only possible if the DL does not allow for nominals.

The rest is to define a probability distribution μ that satisfies probabilistic formulas in F. Recall that Pr is probability distribution over the set of (possible worlds). We define a function σ which maps each world $I = \{C_i\}$ to a set of states $\sigma(I) \subseteq S$ as follows: $\sigma(I) = \{s | s \models \kappa(I)(r)\}$. Then let $\mu(\sigma(I)) = Pr(I)$ for all possible worlds. It is not hard to see that μ is a probability distribution as it mimics the probability distribution Pr. Finally, (D, S, μ) satisfies all formulas of the kind $l \leq w(B(r)|A(r)) \leq u$ in F because $\mu(B(r) \wedge A(r)) = Pr(B \sqcap A), \mu(A(r)) = Pr(A))$ (by construction, e.g., $\mu(A(r)) = \mu(\{s|s \models A(r)\}) = \sum_{Ct_i \models A} \mu\{\sigma(Ct_i)\} = \sum_{Ct_i \models A} Pr(Ct_i) = Pr(A))$ and $Pr \models (B|A)[l, u]$ and therefore *2)* holds. $\square$

Theorem 1 implies that the translation preserves satisfiability and entailments.

Translation of PABoxes One particularly odd restriction of P-$\mathcal{SHIQ}$ is that PABoxes cannot be combined into a single set of formulas. This is so because PABox constraints are modeled as generic constraints and the information about the individual is present only on a meta-level (as a label of the PABox). Therefore, to extend our translation to PABoxes we either have to translate them into a corresponding disjoint set of labeled FOPL$_2$ theories or make special arrangements to faithfully translate them into a combined FOPL$_2$ theory. We opt for the latter because it will let us get rid of any meta-logical aspects and help analyze a P-$\mathcal{SHIQ}$ ontology as a standard FOPL$_2$ theory.

Since PABoxes in P-$\mathcal{SHIQ}$ are isolated from each other, the translation should preserve that isolation. The most obvious way to prevent any interaction between sets of formulas in a single logical theory is to make their signatures disjoint. However, the translation should not only respect disjointness of PABoxes but also preserve their interaction with PTBox and the classical part of the ontology (see Example 1).

Example 1. Consider the following PTBox: $\mathcal{P} = \{(FlyingObject|Bird)[0.9, 1],$ $(FlyingObject|\neg Bird)[0, 0.5]\}$ and two PABoxes: $\mathcal{P}_{Tweety} = \{(Bird|\top)[1, 1]\}$, $\mathcal{P}_{Sam} = \{(\neg Bird|\top)[1, 1]\}$. Obviously, if these sets of axioms are translated and combined into a single FOPL$_2$ theory then it will contain a conflicting pair of formulas $\{w(Bird(r)) \geq 0.9, w(Bird(r)) \leq 0.5\} \subseteq F$. This inconsistency can be avoided by introducing fresh first-order predicates for every PABox: $\{w(Bird_{Tweety}(r)) \geq 0.9,$ $w(Bird_{Sam}(r)) \leq 0.5\}$. However, this would break any connection between PTBox and PABox axioms, for example, prevent the following entailments:
$\{w(FlyingObject_{Tweety}(r)) \geq 0.9, w(FlyingObject_{Sam}(r)) \leq 0.5\}$.

One can faithfully extend the translation to PABoxes by introducing fresh concept names to *relativize* each TBox and PTBox axiom for every probabilistic individual to avoid inconsistencies. The transformation will consist of the following steps[4]:

– Firstly, we transform a P-$\mathcal{ALC}$ ontology $PO = (\mathcal{O}, \mathcal{P}, (\mathcal{P}_o))$ into a set of PTBoxes $\{(\mathcal{O}, \mathcal{P} \cup \mathcal{P}_o)\} \cup \{(\mathcal{O}, \mathcal{P})\}$. Informally, we create a copy PTBox for every probabilistic individual (PT_o) and make them isolated from each other. Now, instead of one PTBox and a set of PABoxes we have just a set of PTBoxes. This step preserves probabilistic entailments in the following sense: $PO \models (B|A)[l, u]$ iff $(\mathcal{O}, \mathcal{P}) \models (B|A)[l, u]$ and $PO \models (B|A)[l, u]$ for o iff $PT_o \models (B|A)[l, u]$.

[4] Full example is available at http://www.cs.man.ac.uk/~klinovp/research/pshiq/example.pdf.

- Secondly, we transform every PTBox PT_o into PT'_o by renaming every concept name C into C_o in all TBox axioms and conditional constraints. It is easy to see that $PT_o \models C \sqsubseteq D$ iff $PT'_o \models C_o \sqsubseteq D_o$ and $PT_o \models (B|A)[l,u]$ iff $PT'_o \models (B_o|A_o)[l,u]$. Intuitively, we have created a fresh copy of each PTBox to guard against possible conflicts between PABox constraints for different probabilistic individuals. Signatures of PT'_o are pairwise disjoint and denoted as Σ_o.
- Next, we union all PT'_o with disjoint signatures (including the original $PT = (\mathcal{O}, \mathcal{P})$) into a single unified PTBox $PT_U = \bigcup_{o \in I_p} PT_o \cup PT$ with signature $\Sigma_U = \bigcup_{o \in I_p} \Sigma_o \cup \Sigma$.
- Finally we can apply the previously presented faithful translation to PT_U and obtain a single FOPL$_2$ theory which corresponds to the original P-$\mathcal{ALC}$ ontology.

A necessary condition for faithfulness of this transformation is that the original isolation of PABoxes is preserved by creating fresh copies of PTBoxes. In particular, this means that the unified PTBox cannot entail any subsumption relation between concept expressions C_{o_1} and C_{o_2} defined over disjoint signatures except of the case when one of them is either $\top$ or $\bot$. If this is false, for example, if $PT_U \models C_{o_1} \sqsubseteq C_{o_2}$ then the following PABox constraints represented as $(C_{o_1}|\top)[1,1]$ and $(C_{o_2}|\top)[0,0]$ will be mutually inconsistent in PT_U (but they were consistent in the original P-$\mathcal{ALC}$ because they belonged to different PABoxes isolated from each other). This condition is formalized in the following lemma (whose proof is omitted for brevity):

Lemma 1. *Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be copies of a satisfiable $\mathcal{ALC}$ ontology $\mathcal{O}$ with disjoint signatures Σ_1 and Σ_2, and $\mathcal{O}_U$ be the union of $\mathcal{O}_1$ and $\mathcal{O}_2$. Then for any concept expressions C_1, C_2 over Σ_1 and Σ_2 respectively such that $\mathcal{O}_1 \not\models C_1 \sqsubseteq \bot$ and $\mathcal{O}_1 \not\models \top \sqsubseteq C_2$, $\mathcal{O}_U \not\models C_1 \sqsubseteq C_2$.*

Now we can obtain the main result:

Theorem 2. *Let $PO = (\mathcal{O}, \mathcal{P}, (\mathcal{P}_o))$ be a P-$\mathcal{ALC}$ ontology and F be a FOPL$_2$ theory obtained by combining PABoxes and translating the resulting PTBox into FOPL. Then for every P-$\mathcal{ALC}$ model Pr_o of $PT_o = (\mathcal{O}, \mathcal{P} \cup \mathcal{P}_o)$ for every probabilistic individual o there exists a corresponding Type 2 structure $M = (D, S, \mu)$ such that:*

1. *$M \models \kappa(\phi)$ for all $\phi \in \mathcal{O}$,*
2. *$M \models l \leq w(B(r)|A(r)) \leq u$ for all conditional constraints $(B|A)[l,u]$ in $\mathcal{P}$,*
3. *$M \models l \leq w(B_o(r)|A_o(r)) \leq u$ for all conditional constraints $(B|A)[l,u]$ in $\mathcal{P}_o$,*

and vice versa, where κ is defined according to Table 1.

Proof. Due to Theorem 1 it suffices to show that the steps 1-3 of the transformation preserve probabilistic models. This can be done by establishing a correspondence between possible worlds of each PT_o and PT_U. Since there are no subsumptions between concept expressions over signatures of different PTBoxes (see Lemma 1), each possible world I_o in PT_o corresponds to a finite set of possible worlds of PT_U defined as: $\sigma(I_o) = \{I_U | C_{i_o} \in I_U \text{ iff } C_i \in I_o\}$ (each C_{i_o} is a new concept name for C_i introduced on step 2). Then, a probability distribution over all possible worlds in PT_U can be defined as $Pr_U(I_U) = Pr_o(I_o)/|\sigma(I_o)|$. It follows that for any concept C over Σ_o,

$Pr_o(C)$ is equal to $Pr_U(C_o)$ where C_o is the correspondingly renamed concept. Therefore, $Pr_U \models (B_o|A_o)[l, u]$ if $Pr_o \models (B|A)[l, u]$. The reverse direction can be proved along the same lines (i.e., $Pr_o(I_o)$ can be defined as $\sum_{I_U \in \sigma(I_o)} Pr_U(I_U)$).

4 Discussion

The main conclusion following from the presented translation is that P-$\mathcal{SHIQ}$ all PTBox statements express *degrees of belief* (i.e. subjective probabilities) about a single, yet unnamed, individual. This is not an easily expected outcome because the variable-free syntax of P-$\mathcal{SHIQ}$ may give a misleading impression that PTBox constraints correspond to universally quantified formulas of some sort. The fact that probabilistic individuals are not translated to corresponding constants in FOPL$_2$ (in contrast to classical individuals) is also not a trivial outcome. Both these features of P-$\mathcal{SHIQ}$ have important implications, but before moving to them, let us consider another, perhaps more naturally looking translation and explain why it is not faithful.

It may well appear that conditional constraints in P-$\mathcal{SHIQ}$ should be interpreted as implicitly universally quantified formulas analogously to probabilistic logic programming. That way, $(B|A)[l, u]$ corresponds to $\forall x(l \leq w(B(x)|A(x)) \leq u)$. However, the standard behavior of the universal quantifier is incompatible with the P-$\mathcal{SHIQ}$ semantics in which classical and probabilistic individuals are separated. For example, the PTBox $(\{a : \neg A\}, \{(A|\top)[1, 1]\})$ is satisfiable although the corresponding FOPL theory $\{\neg A(a), \forall x(w(A(x)) = 1)\}$ is not.

There is a possibility to interpret conditional constraints in P-$\mathcal{SHIQ}$ as closed quantified formulas, but this requires a non-standard quantifier which makes the variable act as a random designator. This idea dates back to Cheeseman who originally proposed to use formulas of the form $\forall x.pr[B(x)|A(x)][l, u]$ to capture statistical knowledge [7]. In fact, the fresh constant r used in our translation plays the role of such non-standard quantifier. However, as pointed out by several authors (see especially [3] [8] [9]), such formulas *cannot* serve as representations of statistical assertions because their interpretations are not based on proportions of domain elements[5].

Unfortunately, using Type 2 semantics to interpret different kinds of probabilities complicates not only the representation of statistics but also the combination of statistical assertions with probabilistic statements about specific individuals (degrees of belief). In particular, this requires modeling of PABox constraints in P-$\mathcal{SHIQ}$ as generic PTBox statements with information about individuals presenting only on a meta-level. This is the reason why PABox statements do not correspond to ground probabilistic formulas in FOPL$_2$. If they did, then there would be no connection between a “statistical” statement $(FlyingObject|Bird)[0.9, 1]$ (represented in FOPL$_2$ as $(0.9 \leq flyingobject(r)|bird(r)) \leq 1)$ and a belief statement $(tweety : Bird)[1, 1]$ (represented as $1 \leq w(bird(tweety) \leq 1)$ since beliefs about r cannot affect beliefs about $tweety$. Therefore $(tweety : Bird)[1, 1]$ is effectively modeled as $(Bird|\top)[1, 1]$

[5] We must mention that P-$\mathcal{SHIQ}$ could, in principle, be translated to FOPL with domain-based semantics by employing a known translation between domain-based probability and possible-world-based probability (see [10] for details). However, this will solve no issues with P-$\mathcal{SHIQ}$ as it will still behave as FOPL$_2$ with single probabilistic individual.

(or as $1 \leq w(bird(r)) \leq 1$ in FOPL$_2$) with the individual name $tweety$ lifted at the meta-level to serve as a label for the corresponding PABox.

However, this introduces other problems which are responsible for the limitations of P-$\mathcal{SHIQ}$. Since PABox constraints expressing probabilistic knowledge about different probabilistic individuals *must* be isolated from each other, there appears to be no straightforward way of combining them. In particular, this prohibits representation of classical or probabilistic role assertions between different probabilistic individuals or, in other words, the logic does not support probabilistic relational structures[6]. Thus, it can be concluded that, in essence, P-$\mathcal{SHIQ}$ is closer to a propositional probabilistic logic rather than to a full-fledged probabilistic first-order formalism.

The problems mentioned above cannot be solved simply by adopting an appropriate semantics for representing statistics, such as Type 1 semantics in which probability distributions are defined over the interpretation domain. Such an attempt has been made by Giugno and Lukasiewicz in the early paper on P-$\mathcal{SHOQ}$ [1]. In that logic probabilistic concept membership assertions were represented using nominals, for example, $(C|\{a\})[0.5, 1]$. Unfortunately, as proved by Halpern, closed first-order formulas can only have probability 0 or 1 in any Type 1 probabilistic model (see Lemma 2.3 in [3]) so the representation is unsatisfactory. It is not hard to see that the probability of $(C|\{a\})$, equivalent to $\frac{Pr(C \sqcap \{a\})}{Pr(\{a\})}$, is 0 if $a^{\mathcal{I}} \notin C^{\mathcal{I}}$ or 1 if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ if Pr is defined over $\Delta^{\mathcal{I}}$.

All the features and limitations explained above are by no means unique to P-$\mathcal{SHIQ}$. They have been discovered and studied for first-order logics by a number of authors who claimed that neither domain-based nor possible world-based semantics *by itself* is suitable for representation and reasoning about different kinds of probabilities. However, their proper combination (called Type 3 semantics [3]) has the required potential. The corresponding logic (FOPL$_3$) is free of any limitations described above, is completely axiomatizable for a range of interesting fragments (e.g., logics with bounded model property such as $\mathcal{ALC}$), and can be used for defining probabilistic DLs.

5 Probabilistic Description Logic with Combined Semantics

In this section we briefly outline the syntax and semantics of the extended probabilistic DL for representation and reasoning about different kinds of probabilities. The language corresponds to the DL fragment of FOPL$_3$ with the principle of direct inference [9]. We loosely call it P-$\mathcal{DL}$ (where $\mathcal{DL}$ stands for a subset of $\mathcal{SROIQ}$).

Syntax Analogously to P-$\mathcal{SHIQ}$ the syntax of P-$\mathcal{DL}$ is based on conditional constraints. However, we distinguish between statistical constraints and belief constraints by providing different syntactic constructs for each. *Statistical conditional constraints* are expressions of the form $(D|C)_{stat}[l, u]$ where D, C are concept expressions. *Belief constraints* are expressions of the form $(\phi)_{belief}[l, u]$ or $(\psi|\phi)_{belief}[l, u]$ where ψ, ϕ are ABox assertions. We define PTBox to be a set of statistical constraints, and PABox to

[6] Allowing nominals in the classical part of the language lets us express probabilistic roles $R(a, b)[l, u]$ as $(\exists R.\{b\}|\top)[l, u]$ for a [2]. However, this is still very restrictive because there cannot be a PABox for b (in other words, b cannot be a probabilistic individual).

be a set of belief constraints. An ontology in P-$\mathcal{DL}$ is a triple $(\mathcal{O}, \mathcal{P}_{stat}, \mathcal{P}_{belief})$ where $\mathcal{O}$ is a $\mathcal{DL}$ ontology, $\mathcal{P}_{stat}$ is a PTBox and $\mathcal{P}_{belief}$ is a PABox.

Semantics Both types of conditional constraints are interpreted using the Type 3 structure $M = (\Delta, S, Pr_{stat}, Pr_{belief})$ [3]. Here Δ is a non-empty domain, S is a set of states that correspond to interpretations of concept, role and individual names over Δ, Pr_{stat} is a probability distribution over Δ, and Pr_{belief} is a probability distribution over S. For a state $s \in S$ we use $s(C)$ (resp. $s(R), s(a)$) to express the interpretation of a concept C (resp. role R and individual a) in s. For an axiom η we write $s \models \eta$ if η is satisfied by the corresponding interpretation. Such combined structure is used to interpret both statistical and belief statements respectively in the following way:

– Statistical probability of a concept C in M in a state s (denoted as $C^{(M,s)}$) is equal to $Pr_{stat}\{d \in \Delta | d \in s(C)\}$. $(D|C)^{M,s}$ is an abbreviation of $\frac{(D \sqcap C)^{(M,s)}}{C^{(M,s)}}$.
– Subjective probability of an ABox assertion ϕ (denoted as ϕ^M is equal to $Pr_{belief}\{s \in S | s \models \phi\}$. $(\psi|\phi)^M$ is an abbreviation of $\frac{(\psi \sqcap \phi)^M}{\phi^M}$.
– M satisfies a statistical constraint $(D|C)_{stat}[l, u]$ if $\forall s \in S, (D|C)^{(M,s)} \in [l, u]$.
– M satisfies a belief constraint $(\psi|\phi)_{belief}[l, u]$ if $(\psi|\phi)^M \in [l, u]$.

Direct Inference FOPL$_3$ provides means for representing and reasoning about different kinds of probabilities but, as it stands, it does not support any relationship between them. However, in most scenarios, e.g., in actuarial reasoning, it is desirable to infer subjective beliefs from available classical and statistical knowledge. Such reasoning is often called *direct inference* and it can be supported in FOPL$_3$ and its fragments.

The main idea behind direct inference, that goes back to Reichenbach's reference class reasoning [11], is to consider every individual to be a *typical* representative of the *smallest* class of objects which it belongs to and for which *reliable* statistics is available. For example, the probability that Tweety flies should be equal to the probability that a randomly taken object, having the same set of properties as Tweety, flies. There are a few proposed ways to implement this idea, one of which we sketch below.

One can capture the notion of typicality directly by equating the degree of belief in a ground formula to the *expectation* of the statistical probability of its *randomized* version given the rest of classical and statistical formulas, as proposed in [9]. Randomization is replacement of all constants in ground formulas by fresh variables. Expectation of a field term f is a rigid (i.e. not depending on a state) term defined as $E(f)^M = \sum_{s \in S} Pr_{belief}(s) \times [f]^{(M,s)}$. The expectation operator and conditioning on statistical formulae are only used on the semantic, not syntactic, level of P-$\mathcal{DL}$.

Consider the example. Let $\{(Fly|Bird)[0.9, 1]\}$ be PTBox and $Bird(tweety)$ be an ABox axiom. Then the degree of belief in $Bird(tweety)$ is within the bounds of $E(bird(v)|0.9 \leq w(fly(v)|bird(v)) \geq 1)$, where v is a fresh constant introduced by randomization. The resulting interval is $[0.9, 1]$, as expected. Note that P-$\mathcal{DL}$ will probably require a non-monotonic mechanism similar to P-$\mathcal{SHIQ}$ to handle situations when an explicitly specified subjective probability is different from the computed via direct inference (e.g., when the individuals in question are *not* typical).

Direct inference via randomization serves the same purpose as P-$\mathcal{SHIQ}$'s way of combining PTBox and PABox constraints (in that sense P-$\mathcal{SHIQ}$ can be thought of as an implementable, non-monotonic *approximation* of FOPL$_3$). However, it is considerably less restrictive because it does not require representing PABox statements as universal PTBox constraints. Since all belief statements about particular individual are ground formulas with proper constants (like $tweety$), they can be combined in a single theory. Thus the representation supports arbitrary relational structures involving different probabilistic individuals and does not force unnatural separation of PABoxes. It is also possible to make the assumption that a pair of individuals are typical thus enabling the inference of probabilistic role assertions. Finally, it supports smooth integration of classical knowledge (i.e. ABox axioms) and beliefs about the same individual while P-$\mathcal{SHIQ}$ requires separation between classical and probabilistic individuals.

6 Conclusion

In this paper we have presented a new look at the probabilistic DL P-$\mathcal{SHIQ}$ as a fragment of probabilistic first-order logic. We gave a translation of P-$\mathcal{SHIQ}$ knowledge bases into FOPL$_2$ theories and proved its faithfulness. This brought an extra insight into P-$\mathcal{SHIQ}$, most importantly, into its limitations. It appears that the major restriction, namely the lack of support of relational structure for probabilistic individuals, is caused by attempt to use the possible world based semantics for different kinds of probabilities. This makes the probabilistic component of P-$\mathcal{SHIQ}$ essentially propositional (i.e. all probabilistic statements relate to a single constant r). We sketched how a more direct fragment of FOPL, which we called P-$\mathcal{DL}$, could overcome these limitations while still retaining the ability to combine probabilities of different sorts. Future investigations include decidability, implementability, and modelling applicability of P-$\mathcal{DL}$.

References

1. Giugno, R., Lukasiewicz, T.: P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In: JELIA. (2002) 86–97
2. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. **172(6-7)** (2008) 852–883
3. Halpern, J.Y.: An analysis of first-order logics of probability. Artif Intell. **46** (1990) 311–350
4. Koller, D., Halpern, J.Y.: Irrelevance and conditioning in first-order probabilistic logic. In: Advances in Artificial Intelligence Conference. (1996) 569–576
5. Baader, F., Calvanese, D., McGuiness, D., Nardi, D., Patel-Schneider, P.F.: Description Logic Handbook. Cambridge University Press (2003)
6. Klinov, P., Parsia, B., Sattler, U.: On correspondences between probabilistic first-order and description logics. In: Description Logic Workshop. (2009)
7. Cheeseman, P.: An inquiry into computer understanding. Comp. Intell. **4** (1988) 58–66
8. Bacchus, F.: Lp, a logic for representing and reasoning with statistical knowledge. Comp. Intell. **6** (1990) 209–231
9. Bacchus, F.: Representing and reasoning with probabilistic knowledge. MIT Press (1990)
10. Abadi, M., Halpern, J.Y.: Decidability and expressiveness for first-order logics of probability. Inform. and Comp. **112**(1) (1994) 1–36
11. Reichenbach, H.: Theory of Probability. University of California Press, Berkeley (1949)

Probabilistic Logic Encoding of Spatial Domains

P. Santos[1], B. Hummel[2], V. Fenelon[3,1], and F. G. Cozman[3]

[1]FEI, São Paulo, Brazil
[2]University Karlsruhe, Germany
[3]Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil

Abstract. This paper presents a formalisation of a spatial domain in terms of a qualitative spatial reasoning formalism, encoded in a probabilistic description logic. The QSR formalism chosen is a subset of a cardinal direction calculus and the probabilistic description logic used has the relational structures of the well-known $\mathcal{ALC}$ language, allied with the inference methods of Bayesian Networks. We consider a scenario consisting of a road navigated by an experimental vehicle equipped with three on-board sensors: a digital map, a GPS and a video camera. This paper presents experiments where the proposed formalism is used to answer queries about driving directions, lanes and vehicles.

1 Introduction

Much work in computer vision in the 70's and 80's had as a goal the development of high-level vision, whereby the numerical (or quantitative) processing feeds a symbolical (or qualitative) level of knowledge from where an agent is capable of interpreting the world, and acting in accordance to this interpretation. These early attempts were frustrated by the non-existence (at the time) of efficient algorithms for dealing with uncertainty, of tractable knowledge representation formalisms and also by the rudimentary stage of image-processing algorithms.

Since then, important advancements in Artificial Intelligence (AI) suggest that we may be at the stage of bridging the gap between AI and Computer Vision. The present paper is related to three of these advancements: *Bayesian Networks* [1], which are graphical representations of domain variables that provide efficient probabilistic inference methods; *Description Logics* (DLs) [15], a family of formalisms (sub-sets of first-order logic) that have a positive trade-off between expressivity and complexity; and qualitative spatial reasoning (QSR) [25], that are formalisms for representing and reasoning about space.

The purpose of this paper is to investigate the use of a qualitative spatial reasoning formalism, encoded on a probabilistic description logic, to answer queries about a traffic scenario. The QSR formalism chosen is a sub-set of a cardinal direction calculus [8, 9] and the probabilistic DL used is $\textsc{cr}\mathcal{ALC}$ [23, 28], which has the relational structures of the description logic $\mathcal{ALC}$ [4], allied with the inference methods of Bayesian Networks.

2 Literature Overview

Qualitative spatial reasoning: The aim of QSR is the logical formalisation of knowledge from elementary spatial entities, such as spatial regions, line segments, cardinal directions, and so forth [13, 25]. These formalisms provide the basic machinery for a system to represent and reason about spatial entities on a more abstract level than quantitative methods [25].

Relevant to the present work are the developments of spatial formalisms for computer vision and robotics. The first proposal for a logic-based interpretation of images is described in [2], where image interpretation is reduced to a constraint satisfaction problem on a set of axioms; [3] proposes a system that generates descriptions of aerial images, which more recently received a descriptive logic enhancement [24].

A spatial system based on spatio-temporal histories for scene interpretation was investigated in [14], which was inspired on an earlier proposal for learning event models from visual information [10]. Recently, [16] proposes a system that uses multiple spatio-temporal histories in order to evaluate an image sequence. A logic formalisation of the viewpoint of a mobile agent was presented in [11], and was further explored in the interpretation of scenes within a mobile robotics scenario in [21, 31]. In [29], functional and geometric properties of roads and intersections could be inferred using an expressive, deterministic, DL in combination with on-board vehicle sensors.

Probabilistic Description Logics: Description Logics (DLs) are fragments of first-order logics originated in the 1970s as a means to provide a formal account of frames and semantic networks. Description logics are based on *concepts* , which represent sets of individuals (such as Plant or Animal); and *roles*, which denote binary relations between individuals, such as fatherOf or friendOf. Set intersection, union and complement are usual operators found in DLs, as well as some forms of quantification. A key feature of most description logics is that their inference is decidable [15].

In recent years there have been an increasing interest in the combination of probabilistic reasoning and logics (and with description logics in particular) [22, 27, 20]. This combination is not only motivated by pure theoretical interest, but it is very relevant from an application standpoint in order to equip a reasoning system with relational inferences capable of making also probabilistic assessments.

In [5, 6] a number of distinct probabilistic logics were proposed where probabilities were defined over subsets of domain elements. These logics have difficulty in handling probabilistic assertions over individuals, as statistical information over the domain does not imply information about individuals; this is known as the *direct inference* problem [7]. The direct inference problem is solved in [18] by adopting probabilities only on assertions. An alternative way around the direct inference problem is to assign probabilities to subsets of interpretations, as assumed in [17, 26] and is also at the kernel of the probabilistic DL we use here.

3 The credal $\mathcal{ALC}$

The credal $\mathcal{ALC}$ (CR$\mathcal{ALC}$) [28] is a probabilistic extension of the $\mathcal{ALC}$ description logic [4]. The basic vocabulary of $\mathcal{ALC}$ contains individuals, concepts (sets of individuals) and roles (binary relations of individuals). Given two concepts C and D, they can be combined to form new concepts from *conjunction* ($C \sqcap D$), *disjunction* ($C \sqcup D$), *negation* ($\neg C$), *existential* restriction ($\exists r.C$) and *value restriction* ($\forall r.C$). A concept *inclusion*, $C \sqsubseteq D$, indicates that the concept D contains the concept C and a *definition*, $C \equiv D$, indicates that the concepts C and D are identical. The set of inclusions and definitions constitute a *terminology*. In general, a terminology is constrained to be acyclic, i.e., no concept can refer to itself in inclusions or definitions.

The semantics of $\mathcal{ALC}$ is defined by a domain $\mathcal{D}$ and an interpretation function $\mathcal{I}$, which maps: each individual to a domain element; each concept to a sub-set of $\mathcal{D}$; and, each role to a binary relation $\mathcal{D} \times \mathcal{D}$, such that the following holds: $\mathcal{I}(C \sqcap D) = \mathcal{I}(C) \cap \mathcal{I}(D); \mathcal{I}(C \sqcup D) = \mathcal{I}(C) \cup \mathcal{I}(D); \mathcal{I}(\neg C) = \mathcal{D} \backslash \mathcal{I}(C); \mathcal{I}(\exists r.C) = \{x \in \mathcal{D} | \exists y : (x, y) \in \mathcal{I}(r) \wedge y \in \mathcal{I}(C)\}; \mathcal{I}(\forall r.C) = \{x \in \mathcal{D} | \forall y : (x, y) \in \mathcal{I}(r) \rightarrow y \in \mathcal{I}(C)\}$. An inclusion $C \sqsubseteq D$ holds if and only if $\mathcal{I}(C) \subseteq \mathcal{I}(D)$, and a definition $C \equiv D$ holds if and only if $\mathcal{I}(D) = \mathcal{I}(D)$ (e.g. $\mathsf{C} \sqsubseteq (\exists\ \mathsf{hasSibling.Woman}) \sqcap (\forall \mathsf{buys.}(\mathsf{Fish} \sqcup \mathsf{Fruit}))$ indicates that C contains only individuals who have sisters and buy fruits or fishes).

In the probabilistic version of $\mathcal{ALC}$ (CR$\mathcal{ALC}$), on the left hand side of inclusions/ definitions only concepts may appear. Given a concept name C, a concept D and a role name r, the following probabilistic assessments are possible:

$$P(C) \in [\underline{\alpha}, \overline{\alpha}], \tag{1}$$

$$P(C|D) \in [\underline{\alpha}, \overline{\alpha}], \tag{2}$$

$$P(r) \in [\underline{\beta}, \overline{\beta}]. \tag{3}$$

We write $P(C|D) = \underline{\alpha}$ when $\underline{\alpha} = \overline{\alpha}$, $P(C|D) \geq \underline{\alpha}$ when $\underline{\alpha} < \overline{\alpha} = 1$, and so on. In order to guarantee acyclicity, no concept is allowed to use itself in deterministic (or probabilistic) inclusions and definitions.

The semantics of CR$\mathcal{ALC}$ is based on probabilities over interpretations so that the direct inference problem can be avoided. In other words, probabilistic values are assigned to the set of all interpretations. The semantics of Formula (1) is, thus: for any $x \in \mathcal{D}$, the probability that x belongs to the interpretation of C is in $[\underline{\alpha}, \overline{\alpha}]$. That is,

$$\forall x \in \mathcal{D} : P\Big(\Big\{ \mathcal{I} : x \in \mathcal{I}(C) \Big\} \Big) \in [\underline{\alpha}, \overline{\alpha}].$$

Informally, the semantics can be represented as $\forall x \in \mathcal{D} : P(C(x)) \in [\underline{\alpha}, \overline{\alpha}]$. The semantics of Expressions (2) and (3) is then:

$$\begin{aligned} &\forall\ x \in \mathcal{D}\ :\ P(C(x)|D(x)) \in [\underline{\alpha}, \overline{\alpha}], \\ &\forall\ (x, y) \in \mathcal{D} \times \mathcal{D}\ :\ P(r(x, y)) \in [\underline{\beta}, \overline{\beta}]. \end{aligned}$$

Given a finite domain, a set of sentences in CR$\mathcal{ALC}$ specifies probabilities over all instantiated concepts and roles. In general, a set of probabilities is specified by a set of sentences; a few assumptions guarantee that a single probability distribution is specified by a set of sentences: unique-names, point-probabilities on assessments, rigidity of names [28]. So, a finite domain and a set of sentences specify a unique Bayesian network over the instantiated concepts and roles. To compute the probability of a particular instantiated concept or role, one can generate this Bayesian network and then perform probabilistic inference in the network. Because the domains we deal with in this paper are small, we follow this propositionalisation strategy in our examples. For large domains it may be impractical to explicitly generate a Bayesian network. In this case, approximate algorithms can be used and, in particular, algorithms based on variational methods have been developed with success [28].

4 Cardinal Direction Calculus

The cardinal direction calculus (CDC) [8] is a formalism for reasoning about cardinal directions between spatial objects. The major reasoning task that CDC is concerned with is to infer the direction between two objects A and C, from the known directions between A and (another object) B and between B and C. The basic part of the calculus has nine relations: equal (eq), north (n), east (e), west (w), south (s), northwest (nw), northeast (ne), southeast (se) and southwest (sw).

We define a CDC inspired on the formulation given in [9], where spatial objects are points in a two-dimensional space and the cardinal directions between two objects A and B are defined as the two projections of the straight line from A to B: one on the axis South-North and the other on the axis West-East.

In order to make clear that we are not dealing with global cardinal directions (while also taking inspiration of the dynamic nature of a traffic scenes), this paper we assume that each road defines its local cardinal direction system, whereby the directions *"Down-Up"* instead of *"South-North"* goes from the origin of the road towards its end, following the road's centre line. In other words, the *"Down-Up"* direction between two objects A and B *on the road* are defined as the projection of the line from A to B on the road's centre line. The *"East-West"* direction, refer as *right-left*, is defined at every point of the road as the continuous orthogonal line to the tangent of the centre line at that point. Figure 1 shows an example of this local CDC.

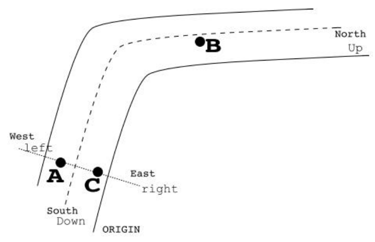


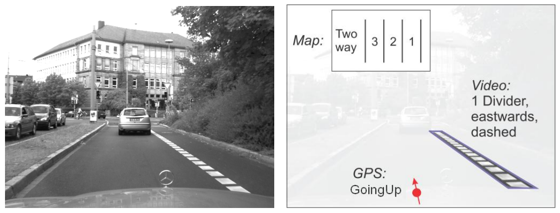
Fig. 1. The local cardinal system for roads: A is south of B and west of C

5 The CR$\mathcal{ALC}$ encoding of a traffic scenario

This section presents a formalisation in CR$\mathcal{ALC}$ of a road traffic domain where incomplete sensor data and domain knowledge can be jointly exploited to solve functional lane recognition tasks. Let *ego-road* and *ego-lane* denote, respectively, the road and the particular lane on which a vehicle is driving. The scenario chosen is composed of a road, where each of its lanes has either the direction *going up* or the direction *going down*. Dividing every pair of adjacent lanes is either a *dashed divider* or a *solid divider*. The scenario also contains an experimental vehicle equipped with three on-board sensors: a digital map, a GPS and a video camera. The task of the formalism is to estimate the following functional properties of the ego-road using on-board vehicle sensors:

– Which lane corresponds to the ego-lane? This task is derived from the fact that current differential GPS receivers are able to reliably determine a vehicle's ego-road, but not its ego-lane (e.g. [19]).
– Which driving direction does each lane permit, "going up" or "'going down"?

Extending these tasks in order to allow queries about turning directions and multiple traffic actors should be straightforward once the above issues are solved.



(a) (b)

Fig. 2. (a) original scene and (b) Example for sensor input from on-board camera, digital map, and map-matched GPS from scene (a).

Sensor Input: The sensors input available to solve that task are:

– *Video-based divider marking recognition:* recognises lane divider markings on the right of the vehicle and classifies them into either dashed or solid divider lines. Hit and false alarm rate of the recognition task, and the confusion table of the classification task, are given in Tables 1(a) and 1(b), respectively.
– *Map-matched GPS position*: retrieves the current road from a digital map and provides the vehicle's driving direction on that road segment. The algorithm proposed in [19] has been shown to be accurate under batch-processing.
– *Digital navigation map*: provides the classification of the road into either one-way or two-way traffic. Table 1(c) is a confusion table for this classification task.

It is worth pointing out that tables 1(a) and 1(c) are based on comparing the algorithm's outcomes with ground truth [29], whereas the data in Table 1(b) was estimated. A typical sensor input is sketched in Figure 2(b), that shows the information obtained by the sensors on the situation in Figure 2(a).

Table 1. Sensor model. In the confusion tables (b) and (c), columns denote ground truth and rows denote estimates.

(a) Video: Divider Recognition

Hit rate	.51
FA rate	.23

(b) Video: Divider Classification

	Solid	Dashed
Solid	.80	.067
Dashed	.20	.933

(c) Digital map: Road Classification

	Oneway	Twoway
Oneway	.99	.01
Twoway	.01	.99

Road Building Regulations: A taxonomy of concepts and roles relevant to the traffic task is set up, in which the concept Lane is defined by the two primitives GoingUp and GoingDown, the concept Divider is defined as the union of DashedDivider and

SolidDivider, and Vehicle is either on a one-way road (OnOneWayRoad) or on a two-way road (OnTwoWayRoad):

$$\mathsf{Lane} \equiv \mathsf{GoingUp} \sqcup \mathsf{GoingDown}$$
$$\mathsf{Divider} \equiv \mathsf{DashedDivider} \sqcup \mathsf{SolidDivider}$$
$$\mathsf{Vehicle} \equiv \mathsf{OnOneWayRoad} \sqcup \mathsf{OnTwoWayRoad}.$$

In Formulae (4)–(7) and (9) we use the abbreviation $\mathsf{disjoint}(\mathsf{t}_1, \mathsf{t}_2, \ldots, \mathsf{t}_n)$ to represent the set of statements about pairwise disjoint terms, i.e., $\mathsf{t}_i \sqsubseteq \neg \mathsf{t}_j \; \forall i, j \in 1, ..., n, i \neq j$.

$$\mathsf{disjoint}(\mathsf{Vehicle, Divider, Lane}) \tag{4}$$
$$\mathsf{disjoint}(\mathsf{GoingUp, GoingDown}) \tag{5}$$
$$\mathsf{disjoint}(\mathsf{DashedDivider, SolidDivider}) \tag{6}$$
$$\mathsf{disjoint}(\mathsf{OnOneWayRoad, OnTwoWayRoad}). \tag{7}$$

The taxonomy of roles consists of CDC relations only. Out of the nine cardinal directions, only three are relevant to the task at hand right (ri), left (le), since the domain does not have cross-roads, and equal (eq):

$$\mathsf{cdc} \equiv \mathsf{ri} \sqcup \mathsf{le} \sqcup \mathsf{eq} \tag{8}$$
$$\mathsf{disjoint}(\mathsf{ri, le, eq}). \tag{9}$$

Next, a set of hard constraints about road building regulations are formulated, making use of the concepts and roles introduced before.The Formulae (10) and (11) formalise the semantics of right-handed traffic: to the right of a lane allowing for traffic *going up* the road (with respect to the road's egocentric coordinate system) there must only be lanes allowing for "going up" traffic, and to the left of traffic *going down* the road there must only be "going down" lanes.

$$\mathsf{GoingUp} \sqsubseteq \forall \mathsf{ri}.(\mathsf{GoingUp} \sqcup \neg \mathsf{Lane}) \tag{10}$$
$$\mathsf{GoingDown} \sqsubseteq \forall \mathsf{le}.(\mathsf{GoingDown} \sqcup \neg \mathsf{Lane}). \tag{11}$$

Formulae (12) and (13) refer to the dividers function, which may be distinct in different countries; these axioms holds for right-handed traffic. A dashed divider divides two lanes, a solid divider either marks the road border or it separates roads with opposing driving directions. And, the axiom states that a two-way road has traffic in both directions (Formula (14)).

$$\mathsf{DashedDivider} \sqsubseteq \exists \mathsf{ri}.\mathsf{Lane} \sqcap \exists \mathsf{le}.\mathsf{Lane} \tag{12}$$
$$\mathsf{SolidDivider} \sqsubseteq \neg \exists \mathsf{ri}.\mathsf{Lane} \sqcup \neg \exists \mathsf{le}.\mathsf{Lane} \sqcup (\exists \mathsf{cdc}.\mathsf{GoingUp} \sqcap \exists \mathsf{cdc}.\mathsf{GoingDown}) \tag{13}$$
$$\mathsf{OnTwoWayRoad} \sqsubseteq \exists \mathsf{cdc}.\mathsf{GoingUp} \sqcap \exists \mathsf{cdc}.\mathsf{GoingDown}. \tag{14}$$

Sensor Model: Concepts are added to represent all probabilistic inputs from sensors: SensedOnOneWayRoad, SensedOnTwoWayRoad and SensedDivider, that can be either SensedDashedDivider or SensedSolidDivider. The confusion tables (Tables 1(a)–1(c)) show joint probabilities of an event and its detection by a sensor, and those conditional probabilities are formulated as a set of axioms:

$$P(\mathsf{DashedDivider}|\mathsf{SensedDashedDivider}) = 0.93$$
$$P(\mathsf{SolidDivider}|\mathsf{SensedDashedDivider}) = 0.07$$
$$P(\mathsf{DashedDivider}|\mathsf{SensedSolidDivider}) = 0.20$$
$$P(\mathsf{SolidDivider}|\mathsf{SensedSolidDivider}) = 0.80.$$

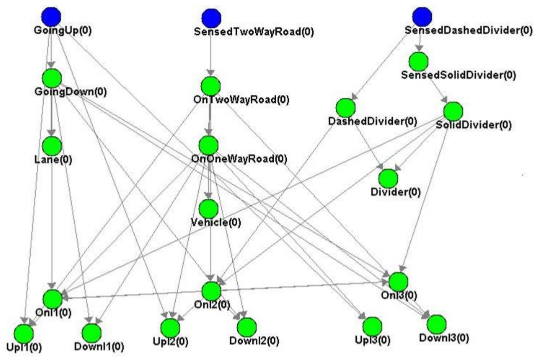


Fig. 3. Bayesian Network representing a traffic domain.

6 Coding and running the scenario

The formalisation presented in the previous section is within the basic definitions of CR$\mathcal{ALC}$. However, the original role hierarchies are not within the scope of $\mathcal{ALC}$ (and, consequently, not within CR$\mathcal{ALC}$). Therefore, we could not represent directly Formulae 8 and 9. Instead, the spatial information about the domain was implicit in the definitions of each of the lanes and their possible directions. This information was included by grounding the descriptions $\exists$eq.Lane ("there is a vehicle in the lane") to new concepts Onl1, Onl2 and Onl3 ("the vehicle is on the lane li, for $i \in 1, 2, 3$"). An analogous idea was used with respect to the lane directions, where Upli and Downli (for $i \in 1, 2, 3$) where used to represent that the lane li is a going up (resp. down) lane. By merging the roles taken on by the individuals l1, l2, and l3 into concepts Onli, Upli and Downli, it was possible to represent the Bayesian network for only one individual, the vehicle, and not for {l1, l2, l3 and ν}. Our solution for representing formulae such as disjoint(A,B,C) was to include probabilistic statements, such as $P(A|B) = 0$ and $P(C|A \vee B) = 0$.

Given the formalisation presented in Section 5 (and the consideration above), the system generated automatically the Bayesian network for only one individual represented in Figure 3, where the nodes in blue are observed variables, i.e. sensors' states. It is now possible to answer the queries specified in Section 5, which correspond to the following:

1. $\text{argmax}_{\text{li}} P((v : \text{Onli}))$, i.e. li is the lane with maximum probability of being the vehicle's (v) ego-lane .
2. $\forall i : P(\text{li} : \text{GoingUp})$, i.e. for each lane li, the probability of being a GoingUp lane.

Consulting the network in Figure 3 for all of the eight possible states of the three sensors, we obtained the answers presented in Tables 2 and 3 for the queries 1 and 2 respectively. In these tables we used the abbreviations $STWR$ for SensedOnTwoWayRoad and SDD for SensedDashedDivider. Table 2 shows probable lane on which the vehicle v is driving ($\text{argmax}_{li} P((v, \text{Onli}))$), given the evidences, represented on the first three columns. The first line of the table, for instance, represents the state where the sensor obtained GoingDown, vehicle on a *one way road* and a *solid divider*. Given these evidences the node Onli with the highest probability was Onl3. This case is shown in Figure 4(c).

Table 2. Answer to query 1: the probability on the *ego-lane* given the evidence A (expressed on the first three columns)

GPS	map	video	$\text{argmax}_{li} P((v : \mathsf{Onli}\|A))$
GoingUp	STWR	SDD	
0	0	0	l3
0	0	1	l1 $\vee$ l2
0	1	0	l2
0	1	1	l3
1	0	0	l1
1	0	1	l2 $\vee$ l3
1	1	0	l1
1	1	1	l2

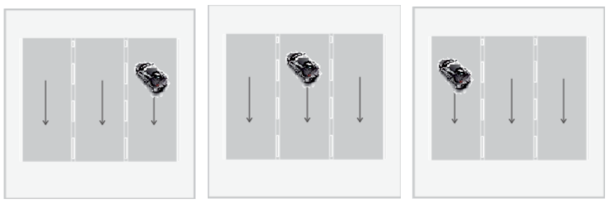

(a) (b) (c)

Fig. 4. Examples of three traffic situations, where the vehicle is an one way road and going down.

Table 3 represents the probabilities of each of the li lanes be a GoingDown lane, given the evidences on the first three columns (the probability of GoingUp is the complement of the values stated in the table). Take for instance the first line, the highest probability for l1, l2 and l3 is GoingDown, which is consistent with the evidences GoingDown for the vehicle, and SensedOnOneWayRoad. Similarly for the remainder sensor states represented in the table.

Table 3. Answer to query 2: the probability for the *lane's driving direction* given the evidence A (expressed on the first three columns)

GPS	map	video	l1	l2	l3
GoingUp	STWR	SDD	$P($l1:GoingDown$\|A)$	$P($l2:GoingDown$\|A)$	$P($l3:GoingDown$\|A)$
0	0	0	0.99	0.99	1.00
0	0	1	0.99	0.99	1.00
0	1	0	0.01	0.76	1.00
0	1	1	0.01	0.95	1.00
1	0	0	0.00	0.01	0.01
1	0	1	0.00	0.00	0.01
1	1	0	0.00	0.61	0.99
1	1	1	0.00	0.09	0.99

7 Conclusion

The representation of QSR systems into description logics is a recent endeavour [12, 30]. The major difficulty of this task is the representation of transitive relations, which are fundamental pieces of spatial knowledge. In particular, [12] presents undecidability results of various $\mathcal{ALC}$ extensions that allow composition-based role inclusion axioms, such as $\mathsf{A} \sqsubseteq \mathsf{B} \sqcap \mathsf{R}_1 \cup \ldots \cup \mathsf{R}_n$ [30]. Decidability of description logic representations of spatial formalisms were proved in [30] for a combination of $\mathcal{ALC}$ with a decidable constraint system (called $\mathcal{ALC}(C)$, where C is the constraint system). The investigation of probabilistic extensions of $\mathcal{ALC}(C)$, and whether decidability is maintained, is an interesting issue for future research.

In this paper we investigated the formalisation of a spatial domain into a probabilistic extension of a basic description logic, CR$\mathcal{ALC}$. In this formalisation we were capable of using the expressivity of a relational formalism (the description logic $\mathcal{ALC}$), with the treatment of uncertainty provided by Bayesian networks, with which sensor model was encoded. To the best of our knowledge, this paper presented the first principled approach on sensor modelling in a logic language.

This work was successful in showing that the expected queries were consequences of the formalisation of the assumed domain. Given this initial success we conjecture that there is a suitable extension of CR$\mathcal{ALC}$ capable of representing (and reasoning about) spatial domains from any qualitative spatial reasoning system. The development of this formalism is a task of future research

Acknowledgements
This work has been partially supported by Fapesp Project 2008/03995-5 (LogProb). Paulo Santos acknowledges travel support from CAPES and CNPq, Brazil. Valquiria Fenelon is a graduate student sponsored by CAPES, Brazil. Fabio Cozman acknowledges support from CNPq, Brazil.

References

1. J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
2. R. Reiter e A. Mackworth. A logical framework for depiction and image interpretation. *Artificial Intelligence*, 41(2):125–155, 1989.
3. T. Matsuyama e V. S. Hwang. *SIGMA: A Knowledge-Based Image Understanding System*. Plenum Press, New York, U.S., 1990.
4. M. Schmidt-Schauss e G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.
5. M. Jaeger. Probabilistic reasoning in terminological logics. In *Principles of Knowledge Representation (KR)*, p. 461–472, 1994.
6. F. Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *International ACM Conference on Research and Development in Information Retrieval*, p. 122–130, Dublin, Ireland, 1994. Springer-Verlag.
7. F. Bacchus, A. Grove, J. Y. Halpern, e D. Koller. From statistical knowledge bases to degrees of belief. *Artificial Intelligence*, 87:75–143, 1996.
8. A. U. Frank. Qualitative spatial reasoning: Cardinal directions as an example. *International Journal of Geographical Information Science*, 10(3):269–290, 1996.
9. G. Ligozat. Reasoning about cardinal directions. *J. Vis. Lang. Comput.*, 9(1):23–44, 1998.

10. J. Fernyhough, A. G. Cohn, e D. C. Hogg. Constructing qualitative event models automatically from video input. *Image and Vision Computing*, 18:81–103, 2000.
11. D. Randell, M. Witkowski, e M. Shanahan. From images to bodies: Modeling and exploiting spatial occlusion and motion parallax. In *Proc. of IJCAI*, p. 57–63, Seattle, U.S., 2001.
12. M. Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In *Description Logics*, volume 49 of *CEUR Workshop Proceedings*, 2001.
13. A. G. Cohn e S. M. Hazarika. Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 46(1-2):1–29, 2001.
14. S. M. Hazarika e A. G. Cohn. Abducing qualitative spatio-temporal histories from partial observations. In *Proc. of KR*, p. 14–25, Toulouse, France, 2002.
15. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, e P.F. Patel-Schneider. *Description Logic Handbook*. Cambridge University Press, 2002.
16. B. Bennett, A. Cohn, e D. Magee. Enforcing global spatio-temporal consistency to enhance reliability of moving object tracking and classification. *Künstliche Intelligenz*, 2:32–35, 2005.
17. P. C. G. da Costa e K. B. Laskey. Of Klingons and starships: Bayesian logic for the 23rd century. In *Conference on Uncertainty in Artificial Intelligence*, 2005.
18. M. Dürig e T. Studer. Probabilistic ABox reasoning: preliminary results. In *Description Logics*, p. 104–111, 2005.
19. B. Hummel. *Dynamic and Mobile GIS: Investigating Changes in Space and Time*, chapter Map Matching for Vehicle Guidance. CRC Press, 2006.
20. B. Milch e S. Russell. First-order probabilistic languages: into the unknown. In *International Conference on Inductive Logic Programming*, 2007.
21. P. Santos. Reasoning about depth and motion from an observer's viewpoint. *Spatial Cognition and Computation*, 7(2):133–178, 2007.
22. A. Nikolov, V. Uren, E. Motta, e A. de Roeck. Using the Dempster-Shafer theory of evidence to resolve ABox inconsistencies. In *Workshop on Uncertainty Reasoning for the Semantic Web*, 2007.
23. F. G. Cozman e R. B. Polastro. Loopy propagation in a probabilistic description logic. In *SUM*, p. 120–133, 2008.
24. B. Neumann e R. Möller. On scene interpretation with description logics. *Image Vision Comput.*, 26(1):82–101, 2008.
25. A. G. Cohn e J. Renz. Qualitative spatial representation and reasoning. In *Handbook of Knowledge Representation*, p. 551–596. Elsevier, 2008.
26. C. D'Amato, N. Fanizzi, e T. Lukasiewicz. Tractable reasoning with Bayesian description logics. In *SUM*, p. 146–159, Berlin, Heidelberg, 2008. Springer-Verlag.
27. C. P. de Campos, F. G. Cozman, e J. E. O. Luna. Assembling a consistent set of sentences in relational probabilistic logic with stochastic indepence. *Journal of Applied Logic: Combining Probability and Logic*, 7:137–154, 2009.
28. F. G. Cozman e R. Polastro. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In *Proc. of the Conference on Uncertainty in Artificial Intelligence*, 2009.
29. B. Hummel. *Description Logic for Intersection Understanding at the Example of Urban Road Intersections*. Phd Thesis, April 2009.
30. M. Cristani e N. Gabrielli. Practical issues of description logics for spatial reasoning. In *Proc. of the AAAI Spring Symposium Benchmarking of Qualitative Spatial and Temporal Reasoning Systems*, p. 5–10, 2009.
31. P. Santos, H. Dee, e V. Fenelon. Qualitative robot localisation using information from cast shadows. In *Proc. of IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

Gibbs Sampling in Probabilistic Description Logics with Deterministic Dependencies

Oliver Gries and Ralf Möller

Hamburg University of Technology[⋆]
21073 Hamburg, Germany

Abstract. In many applications there is interest in representing both probabilistic and deterministic dependencies. This is especially the case in applications using Description Logics (DLs), where ontology engineering usually is based on strict knowledge, while there is also the need to represent uncertainty. We introduce a Markovian style of probabilistic reasoning in first-order logic known as Markov logic and investigate the opportunities for restricting this formalism to DLs. In particular, we show that Gibbs sampling with deterministic dependencies specified in an appropriate fragment remains correct, i.e., probability estimates approximate the correct probabilities. We propose a Gibbs sampling method incorporating deterministic dependencies and conclude that this incorporation can speed up Gibbs sampling significantly.

1 Introduction

While probabilistic languages often represent uncertain evidence and exceptions, there is also the need to further represent deterministic knowledge. As a tradeoff, there are probabilistic formalisms that allow for the representation of near-deterministic knowledge, i.e., knowledge represented with probabilities approximating 0 or 1. However, since knowledge representation with logic in its origin is based on strict formulas, we believe it is important to consider the perspective in which uncertainty is an additional feature to deterministic knowledge. This perspective is especially suited for Description Logics (DLs) [3], where ontology engineering usually is based on specifying strict taxonomies with strict disjointness and strict domain and range restrictions on roles, and where the tradeoff between complexity and expressivity is well-studied.

Gibbs sampling [2, 4, 5] is a Markov chain Monte Carlo (MCMC) method to estimate a conditional probability distribution by generating samples from a simpler distribution. Since Markov chains constructed with Gibbs sampling are known to be regular, estimates of probabilities are known to be correct in the long run. Though this method is often used in practice, it is insufficient for logic in the presence of deterministic dependencies, since in this case the correct flow of the

[⋆] This work is supported by the European Union project CASAM (FP7-217061).

chain usually is broken down. Similarly, Gibbs sampling with near-determinism involves transitions with probabilities approximating 0, leading to unacceptably long convergence times (cf. [7]). In order to solve this problem, in [7] the MC-SAT algorithm is proposed. MC-SAT samples new states with respect to a set of auxiliary variables. However, while MC-SAT is a powerful method to compute conditional probabilities, it is restricted to near-determinism.

In this paper, we propose a formalism based on Markov logic [6] and show that by restricting the deterministic part of the knowledge to a fragment of $\mathcal{ALH}$, Markov chains constructed with Gibbs sampling remain regular. To the best of our knowledge, until now there has not been a language discovered allowing for global deterministic dependencies in Gibbs sampling. We present a Gibbs sampling method incorporating these dependencies and conclude that this incorporation can speed up Gibbs sampling significantly.

In Sect. 2, we introduce probabilistic knowledge representation and Markov networks. In Sect. 3, the formalism of Markov logic [6] is presented and extended to incorporate deterministic dependencies. Further, we define the language $\mathcal{ALH}^-$ for the representation of determinism. Then, in Sect. 4, we introduce to problems with Gibbs sampling in knowledge representation. In Sect. 5, we propose a Gibbs sampling algorithm incorporating deterministic dependencies in $\mathcal{ALH}^-$. Finally, in Sect. 6 we summarize the results.

2 Preliminaries

For the representation of probabilistic and deterministic knowledge, we will focus on Description Logics (DLs). We assume the reader to be familiar with the syntax and semantics of DLs [3] and first-order logic [4].

2.1 Probabilistic Knowledge Representation

The basic notion of probabilistic knowledge representation formalisms is the so-called *random experiment*. A *random variable* X is a function assigning a value to the result of a random experiment. In the sequel, we will use only Boolean random variables with values 1 or 0 (*true* or *false*, respectively).

Let $\boldsymbol{X} = \{X_1, ..., X_n\}$ be the ordered set of all random variables of a random experiment. An *event* $\boldsymbol{X} = \boldsymbol{x}$ is an assignment $X_1 = x_1, ..., X_n = x_n$ to all random variables, and a certain vector of values $\boldsymbol{x}$ is referred to as a *possible world*. If it is clear from the context, we write x as an abbreviation for $X = true$ and $\neg x$ as an abbreviation for $X = false$. A possible world can be associated with a probability $P(\boldsymbol{X} = \boldsymbol{x}) = p$, where p is a real value in $[0, 1]$.[1]

A *distribution* $\mathbf{P}(X)$ of a random variable X is a mapping from the domain of X to probability values in $[0, 1]$ such that the values of X sum up to 1. Distributions can be defined for (ordered) sets of random variables as well. A mapping $\mathbf{P}(X_1, \ldots, X_k)$ from the domain of a set of random variables to the

[1] We assume the reader to be familiar with Kolmogorov's axioms of probability.

k-dimensional cross product of $[0,1]$ such that all combinations of values sum up to 1 is called *joint distribution*. A *full joint distribution* $\mathbf{P}(X_1,\ldots,X_n)$ is a joint distribution where all random variables of a random experiment are involved. Let $\Omega = \{\boldsymbol{x}_1,...,\boldsymbol{x}_r\}$ be the set of all possible worlds. In order to specify a full joint distribution $\mathbf{P}(X_1,\ldots,X_n)$, probabilities $P(\boldsymbol{X} = \boldsymbol{x}_i)$ for all $\boldsymbol{x}_i$ must be given such that $\sum_{i=1}^{r} P(\boldsymbol{X} = \boldsymbol{x}_i) = 1$. The expression $\mathbf{P}(X_1,\ldots,X_m,x_{m+1},\ldots,x_l)$ denotes an m-dimensional distribution where the values of $X_{m+1},...,X_l$ have been fixed. In slight misuse of notation, we sometimes write $\boldsymbol{e}$ for these fixed values.

The *conditional probability distribution of X given evidence $\boldsymbol{e}$* is defined by $\mathbf{P}(X \mid \boldsymbol{e}) = \frac{\mathbf{P}(X,\boldsymbol{e})}{P(\boldsymbol{e})} = \alpha\,\mathbf{P}(X,\boldsymbol{e}) = \alpha\,{<}P(x,\boldsymbol{e}),P(\neg x,\boldsymbol{e}){>}$, where α is a normalizing constant. Note that $\mathbf{P}(X \mid \boldsymbol{e})$ is only defined, if $P(\boldsymbol{e}) > 0$.

The most common query types in probabilistic knowledge representation are the *conditional probability query*, i.e., the computation of $\mathbf{P}(X \mid \boldsymbol{e})$ and the *maximum a posteriori (MAP) query*, where the objective is to find the most likely assignment of values to random variables $X_1,...,X_m$ given evidence $\boldsymbol{e}$, i.e., the computation of $\mathit{argmax}_{X_1,...,X_m}\mathbf{P}(X_1,...,X_m \mid \boldsymbol{e})$. In this paper, we will focus on the former query type.

2.2 Markov Networks

Since the representation of $\mathbf{P}(X_1,\ldots,X_n)$ in principle requires the specification of 2^n probability values, there is interest in formalisms with a less complex representation. The main idea to achieve a more compact representation is that one can exploit independence assumptions. Usually, this is the case in graph-based formalisms, where the representation of the full joint probability distribution can be decomposed into different factors. Besides Bayesian networks, the most important graph-based formalism is the formalism of Markov networks [1,2].

A *Markov network graph* is a tuple $\boldsymbol{G} = (\boldsymbol{X},\boldsymbol{E})$, where $\boldsymbol{X} = \{X_1,...,X_n\}$ is a set of nodes corresponding to the random variables of the domain and $\boldsymbol{E}$ is a set of undirected edges (X_i,X_j), $i \neq j$, between these nodes. A *clique* C is a subgraph of G, whose nodes are all adjacent to each other. Let $\boldsymbol{X}_C$ be the set of nodes contained in C. A clique C is called maximal, if there is no other clique C_i with $\boldsymbol{X}_C \subset \boldsymbol{X}_{C_i}$. Further, $\boldsymbol{C} = \{C_1,...,C_m\}$ is a set of cliques of $\boldsymbol{G}$ consisting of all nodes of $\boldsymbol{X}$, i.e., $\boldsymbol{X}_{C_1} \cup ... \cup \boldsymbol{X}_{C_m} = \boldsymbol{X}$.

A *Markov network* $\mathcal{M} = (\boldsymbol{G},\boldsymbol{F})$ consists of a Markov network graph $\boldsymbol{G}$ and a set $\boldsymbol{F}$ which is comprised of non-negative real-valued functions f_i for each clique $C_i, i = 1,...,m$ in $\boldsymbol{G}$. A full joint probability distribution to be expressed can be decomposed into factors f_i (cf. [1,2]) such that

$$P(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z}\prod_{i}^{m} f_i(\boldsymbol{x}_{C_i}) \tag{1}$$

where Z is a normalizing constant summing over the products in (1) for all possible worlds $\boldsymbol{x}$ ensuring that $\sum_{k=1}^{r} P(\boldsymbol{X} = \boldsymbol{x}_k) = 1$. Note that each f_i does only depend on the values of random variables corresponding to its clique C_i.

3 Markov Description Logics

3.1 Markov Logic

The formalism of Markov logic [6] provides a means to combine the formalism of Markov networks with the expressivity of first-order logic. A knowledge base in Markov logic is called a *Markov logic network* $MLN = (\mathcal{F}, \mathcal{W})$. It consists of a multiset of first-order formulas $\mathcal{F} = \{F_1, ..., F_p\}$ and a multiset of real number weights $\mathcal{W} = \{w_1, ..., w_p\}$ such that each w_i is associated to F_i. For simplicity, we use the notation of a set of weighted formulas $w_i\, F_i$. An example Markov logic network is $MLN_1 = \{4\, \forall x\, P(x) \rightarrow Q(x), 1.1\, P(i)\}$, where i is a constant.

Let $\Gamma = \{c_1, ..., c_s\}$ be the set of all constants mentioned in $\mathcal{F}$. A *grounding* of a formula F_i is a substitution of all variables in the matrix of F_i with constants from Γ (this corresponds to a domain closure).[2] A Markov logic network MLN can be converted to a (finite) set of weighted ground clauses $Cl = \{cl_1, ..., cl_m\}$. Each atom appearing in Cl is referred to as a *ground atom*. The set of all these ground atoms corresponds to a set of Boolean random variables $\boldsymbol{X} = \{X_1, ..., X_n\}$. Consequently, for each MLN with a fixed set of constants Γ, there is a set of possible worlds $\boldsymbol{x}$. When a world $\boldsymbol{x}$ does not satisfy a formula, the idea is to ensure that this world is less probable rather than impossible as in first-order logic.

For each MLN there is a corresponding Markov network $\mathcal{M} = (\boldsymbol{G}, \boldsymbol{F})$, with $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{E}_{MLN})$, where $\boldsymbol{E}_{MLN}$ is the set of pairs of ground atoms (X_i, X_j) appearing together in at least one cl_i and a function $f_i \in \boldsymbol{F}$ for each $cl_i \in Cl$. Note that each weighted ground clause cl_i corresponds to a (not necessarily maximal) clique C_i. In notation slightly differently from [6], we specify the full joint distribution of a MLN with (1), where

$$f_i(\boldsymbol{x}_{C_i}) = \begin{cases} exp(w_i), & \text{if } \boldsymbol{x}_{C_i} \text{ satisfies } cl_i \\ \quad 1, & \text{otherwise} \end{cases} \tag{2}$$

Note that $\prod_{i=1}^{m} exp(w_i) = \prod_{i=1}^{m} exp(ln\, f_i(\boldsymbol{x}_{C_i})) = exp(-\sum_{i=1}^{m} -ln\, f_i(\boldsymbol{x}_{C_i}))$, where the last term often is used in the context of statistical physics with $-ln\, f_i(\boldsymbol{x}_{C_i})$ called an energy function [2, 5]. The advantage of using exp in Markov logic is that $P(\boldsymbol{X} = \boldsymbol{x}) > 0$ and that it is possible to specify $w \in \mathbb{R}$.

3.2 Exploiting DLs: Incorporating Deterministic Constraints

There is often interest to represent a domain with both deterministic and probabilistic dependencies. While in [6] deterministic dependencies are approximated by assigning large weights to formulas, we propose to incorporate deterministic constraints to Markov logic in the context of DLs.

Definition 1. *A Markov DL knowledge base KB_M is a tuple $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is comprised of sets $\mathcal{T}_{det}$ and $\mathcal{T}_w$ of deterministic resp. weighted axioms and $\mathcal{A}$ is comprised of sets $\mathcal{A}_{det}$ and $\mathcal{A}_w$ of deterministic resp. weighted assertions.*

[2] An existentially quantified formula is replaced by a disjunction of its groundings.

Under consideration of a domain closure, corresponding Markov networks are similar to the ones introduced in Sect. 3.1. However, in Markov DL knowledge bases $Cl = \{cl_1, ..., cl_m\}$ is the set of weighted and deterministic ground clauses. We specify the full joint probability distribution of KB_{M} with (1), where

$$f_i(\boldsymbol{x}_{C_i}) = \begin{cases} exp(w_i), & \text{if } cl_i \text{ is weighted and } \boldsymbol{x}_{C_i} \text{ satisfies } cl_i \\ 0, & \text{if } cl_i \text{ is deterministic and } \boldsymbol{x}_{C_i} \text{ does not satisfy } cl_i \\ 1, & \text{otherwise.} \end{cases} \quad (3)$$

An advantage of having both deterministic and probabilistic dependencies is that initial ontology engineering is done as usual with standard reasoning support and with the possibility to add weighted axioms and weighted assertions on top of the strict fundament. Since lots of possible worlds do not have to be considered because their probability is known to be 0, probabilistic reasoning can be significantly faster.

The language for representing $\mathcal{T}_w$ and $\mathcal{A}_w$ can be any DL in which it is reasonable to assume a fixed set of constants $\Gamma = \{c_1, ..., c_s\}$ such that it is possible to compute a finite set of weighted ground clauses. For the representation of deterministic knowledge, in this paper we use the language $\mathcal{ALH}^-$, a rather simple DL without an assertional component (i.e., we are not allowing for deterministic assertions). An *$\mathcal{ALH}^-$ knowledge base KB* consists of a set $\mathcal{T}$ of terminological axioms as depicted in Table 1, where A, B are atomic concepts and R, S are atomic roles with the additional restriction that equivalences as well as terminological cycles are not allowed. The semantics is defined as usual.

Table 1. Terminological axioms in $\mathcal{ALH}^-$

$A \sqsubseteq B$	concept inclusion	$\exists R.\top \sqsubseteq A$	domain restriction on roles
$A \sqsubseteq \neg B$	concept disjointness	$\top \sqsubseteq \forall R.A$	range restriction on roles
$R \sqsubseteq S$	role inclusion		

Example 1. Let $KB_M = (\{Lynx \sqsubseteq Animal, Animal \sqsubseteq \neg Plant\}, \{1.1\ Lynx(i), 0.6\ Plant(i)\})$, where $\mathcal{T}_w = \mathcal{A}_{det} = \{\}$. Under domain closure there are 2^3 possible worlds $\boldsymbol{x}_k$ (where e.g. *Lynx* is abbreviated with L):

$$\begin{array}{ll} \boldsymbol{x}_1 = <L(i), A(i), P(i)> & \boldsymbol{x}_5 = <\neg L(i), A(i), P(i)> \\ \boldsymbol{x}_2 = <L(i), A(i), \neg P(i)> & \boldsymbol{x}_6 = <\neg L(i), A(i), \neg P(i)> \\ \boldsymbol{x}_3 = <L(i), \neg A(i), P(i)> & \boldsymbol{x}_7 = <\neg L(i), \neg A(i), P(i)> \\ \boldsymbol{x}_4 = <L(i), \neg A(i), \neg P(i)> & \boldsymbol{x}_8 = <\neg L(i), \neg A(i), \neg P(i)> \end{array}$$

The full joint probability distribution is specified with respect to the set $Cl = \{\neg Lynx(i) \vee Animal(i),\ \neg Animal(i) \vee \neg Plant(i),\ 1.1\ Lynx(i), 0.6\ Plant(i)\}$. The normalizing constant $Z = 0 + exp(1.1) + 0 + 0 + 0 + 1 + exp(0.6) + 1 \approx 6.83$ such that e.g. $P(\boldsymbol{X} = \boldsymbol{x}_7) \approx \frac{exp(0.6)}{6.83} \approx 0.267$. In order to keep the example simple, we do not consider roles (but the general structure would be the same).

4 Gibbs Sampling for Reasoning about Knowledge

Answering conditional probability queries $\mathbf{P}(X \mid \boldsymbol{e})$ by simply applying the full joint probability distribution is intractable [4]. *Sampling* or *Monte Carlo*-algorithms avoid this problem by generating samples from a probability distribution which is much easier to compute. The objective is to approximate $\mathbf{P}(X \mid \boldsymbol{e})$. Sampling is like "coin flipping": Random numbers in $[0, 1]$ are generated and a variable X_i is assigned *true* resp. *false*, if the corresponding number is lower resp. greater than the respective probability in the distribution of X_i.

A *Markov chain (of length k)* is a sequence of states $\boldsymbol{x}_1, ..., \boldsymbol{x}_k$ (a state corresponds to a possible world) where each successing state only depends on the current state. *Markov chain Monte Carlo (MCMC)* algorithms [2,4] are a powerful class of sampling algorithms walking through the state space $\Omega = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_r\}$. With respect to an arbitrary order, each non-evidence variable $X_i, i = 1, ..., m$ is sampled. This process is repeated N-times such that $k = m \cdot N$. After $k-1$ steps, the fractions of the number of states visited with $X = x$ resp. $X = \neg x$ are taken as the estimated probabilities of $\mathbf{P}(X \mid \boldsymbol{e})$.

Let $\overline{\boldsymbol{x}_i}$ be an assignment to $X_1, ..., X_{i-1}, X_{i+1}, ..., X_n$. *Gibbs sampling* [2,4,5] is a special case of MCMC, where the probability of a transition $T(\boldsymbol{x}_t \rightarrow \boldsymbol{x}_{t+1})$ is $T((x_i, \overline{\boldsymbol{x}_i}) \rightarrow (x'_i, \overline{\boldsymbol{x}_i})) = P(x'_i \mid \overline{\boldsymbol{x}_i})$. In graph-based formalisms such as Markov networks, Gibbs sampling can be optimized by exploiting the graph structure: Let $C_1, ..., C_s$ be all cliques containing the node X_i. The *Markov boundary* of X_i is the set of its neighbours $MB(X_i) = \boldsymbol{X}_{C_1} \cup \ldots \cup \boldsymbol{X}_{C_s} \setminus \{X_i\}$. If the values of $MB(X_i)$ are known, denoted $mb(X_i)$, it shields X_i from influences of all other nodes in $\boldsymbol{G}$, i.e., $\mathbf{P}(X_i \mid x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = \mathbf{P}(X_i \mid mb(X_i)) = \alpha < P(x_i, mb(X_i)), P(\neg x_i, mb(X_i)) >$.

A Markov chain is *regular* if there is a number v such that for all pairs of states $\boldsymbol{x}_i, \boldsymbol{x}_j$ the probability of getting from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ in v steps is greater than 0 [2]. If a Markov chain is regular (in Markov networks if $f_i > 0$ for all $f_i \in \boldsymbol{F}$) the probability distribution of the states converges to a unique *stationary distribution* π. Finally, in the case of Gibbs Sampling, $\pi(\boldsymbol{x}_i)$ is known to be equal to $P(\boldsymbol{x}_i \mid \boldsymbol{e})$. Markov chains through Gibbs Sampling and deterministic dependencies usually are not regular and can break down the state graph into disconnected regions:

Example 2. Consider two ground atoms X_1, X_2 with the deterministic constraint $X_1 \equiv X_2$. The state graph for applying Gibbs sampling is depicted in Fig. 1. If the chain starts with $(0, 0)$, it will never reach $(1, 1)$. Consequently, $\pi(0, 0) = 1$ and $\pi(1, 1) = 0$, though there is no information of any preference.

Similarly, Gibbs sampling with near-determinism involves transitions with probabilities approximating 0, leading to unacceptably long convergence times (cf. [7]). Thus, since (near-)determinism is required in many applications, Gibbs sampling in general is not sufficient for reasoning about knowledge.

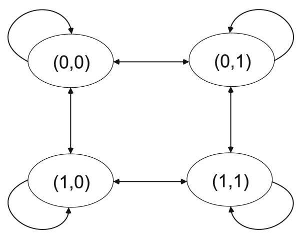


Fig. 1. Gibbs sampling state graph of two Boolean random variables

5 Gibbs Sampling with Deterministic Dependencies

Definition 2. *Let $\boldsymbol{D}$ be a set of deterministic dependencies. A Markov chain is regular with respect to $\boldsymbol{D}$ if there is a number v such that for all pairs of states $\boldsymbol{x}_i, \boldsymbol{x}_j$ both satisfying $\boldsymbol{D}$, the probability of getting from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ in v steps is greater than 0.*

A Markov chain being regular with respect to $\boldsymbol{D}$ has a unique stationary distribution π, if the initial state x_1 satisfies $\boldsymbol{D}$, i.e., $P(\boldsymbol{X} = \boldsymbol{x_1}) > 0$. The difference to regular Markov chains defined in the previous section is that there are states $\boldsymbol{x}_i$ with $P(\boldsymbol{X} = \boldsymbol{x}_i) = 0$ that are simply not stepped into.

As can be seen from Example 2, the regularity of the Markov chain is broken, if X_1 and X_2 are constrained to be equal. This is also the case if they are constrained to be different. We will now show that it is possible to run a Markov chain constructed with Gibbs sampling that is regular with respect to deterministic constraints in $\mathcal{ALH}^-$.

Theorem 1. *Let KB_M be a Markov DL knowledge base where $\mathcal{T}_{det}$ is represented with $\mathcal{ALH}^-$ and $\mathcal{A}_{det}$ is empty. Then, a Markov chain constructed with Gibbs sampling is regular with respect to $\mathcal{T}_{det}$.*

Proof. In order to prove the regularity, it is important to figure out states satisfying or falsifying $\mathcal{T}_{det}$. Ground clauses derived from disjointness axioms are of the form $\neg At_1 \vee \neg At_2$, and ground clauses derived from all other axioms in $\mathcal{ALH}^-$ are of the form $\neg At_1 \vee At_2$, where At_1 and At_2 are ground atoms. Therefore, clauses derived from $\mathcal{T}_{det}$ are falsified only if At_1 is assigned 1 (*true*) and At_2 assigned 0 (*false*) or if both At_1 and At_2 are assigned 1. Since in $\mathcal{ALH}^-$ there are no cycles, an order $X_1 < ... < X_n$ can be defined for all ground atoms such that for all ground clauses $\neg At_1 \vee At_2$ no ground atom At_2 is lower in the order than At_1. Atoms only mentioned in clauses $\neg At_1 \vee \neg At_2$ can be added arbitrarily. Then, it is possible to construct a tree, where each node respects the defined order and corresponds to a state $(x_1, ..., x_n)$, and where the root node is a state where all atoms are assigned with 0, i.e., $(0, ..., 0)$. For every 0 in $(0, ..., 0)$, there is exactly one child node where the corresponding 0 is changed to 1. Then, for every 0 preceding the leftmost 1 in a node, there is also exactly one child node where the corresponding 0 is changed to 1. After constructing all children nodes, the tree contains exactly 2^n nodes (i.e., one node for each state). This tree has the following property: If a node represents a state satisfying $\mathcal{T}_{det}$, then its parent node also represents a state satisfying $\mathcal{T}_{det}$. Since in $\mathcal{ALH}^-$ the state $(0, ..., 0)$ satisfies $\mathcal{T}_{det}$, for each node it holds that if it satisfies $\mathcal{T}_{det}$ then there is a path from this node to $(0, ..., 0)$ satisfying $\mathcal{T}_{det}$. $\square$

We will now specify a Gibbs sampling algorithm with deterministic dependencies in $\mathcal{ALH}^-$. Instead of answering conditional probability queries $\mathbf{P}(X \mid \boldsymbol{e})$, with this algorithm it is possible to answer probability queries $\mathbf{P}(X)$ conditioned on the ground clauses obtained from $\mathcal{T}_{det}$.

Our objective is to exploit the fact that – due to restrictions in $\mathcal{T}_{det}$ – there are a lot of state transition possibilities where the bit-flip probability (the probability that a random variable will change its value) is 0, i.e., a lot of cases in which one does not need to sample at all. We propose to assign an integer γ_i to each random variable X_i. γ_i is initially 0 and is increased whenever a bit-flip occurs for a variable $X_j, i \neq j$ that restricts the bit-flip probability of X_i to be 0 ("set"), and γ_i is decreased whenever this specific restriction does not hold any more ("release"). As long as $\gamma_i > 0$, X_i is not sampled and we say that X_i is "blocked". Settings $(+1)$ and releases (-1) are depicted in Table 2 for groundings of all $\mathcal{ALH}^-$-axioms with individuals i, j. Consider e.g. the first row in the second column of Table 2: If the ground atom $A(i)$ is 0 in $\boldsymbol{x}_t$ and is 1 in $\boldsymbol{x}_{t+1}$ then $B(i)$ is known to be 1 (otherwise $\boldsymbol{x}_{t+1}$ would violate this constraint) and $\gamma_{B(i)}$

Table 2. Setting and releasing blockings due to $\mathcal{ALH}^-$-axioms

	$A(i)_t \rightarrow$	$A(i)_{t+1}$	$\gamma_{B(i)}$	$B(i)_t \rightarrow$	$B(i)_{t+1}$	$\gamma_{A(i)}$
$A \sqsubseteq B$	0	1	$+1$	0	1	-1
	1	0	-1	1	0	$+1$

	$A(i)_t \rightarrow$	$A(i)_{t+1}$	$\gamma_{B(i)}$	$B(i)_t \rightarrow$	$B(i)_{t+1}$	$\gamma_{A(i)}$
$A \sqsubseteq \neg B$	0	1	$+1$	0	1	$+1$
	1	0	-1	1	0	-1

	$R(i,j)_t \rightarrow$	$R(i,j)_{t+1}$	$\gamma_{S(i,j)}$	$S(i,j)_t \rightarrow$	$S(i,j)_{t+1}$	$\gamma_{R(i,j)}$
$R \sqsubseteq S$	0	1	$+1$	0	1	-1
	1	0	-1	1	0	$+1$

	$R(i,j)_t \rightarrow$	$R(i,j)_{t+1}$	$\gamma_{A(i)}$	$A(i)_t \rightarrow$	$A(i)_{t+1}$	$\gamma_{R(i,j^*)}$
$\exists R.\top \sqsubseteq A$	0	1	$+1$	0	1	-1
	1	0	-1	1	0	$+1$

	$R(i,j)_t \rightarrow$	$R(i,j)_{t+1}$	$\gamma_{A(j)}$	$A(j)_t \rightarrow$	$A(j)_{t+1}$	$\gamma_{R(i^*,j)}$
$\top \sqsubseteq \forall R.A$	0	1	$+1$	0	1	-1
	1	0	-1	1	0	$+1$

is increased, i.e., $B(i)$ is blocked by $A(i)$. Individuals with * indicate that every individual of the closed domain has to be considered separately.

The Markov chain has to start with a state $\boldsymbol{x}_1$ satisfying $\mathcal{T}_{_{det}}$. Such a state can be found with MaxWalkSat [9], a local search algorithm for the weighted satisfiability problem. All deterministic clauses are assigned with a weight greater than the sum l of all weights of clauses obtained from $\mathcal{T}_{_w} \cup \mathcal{A}_{_w}$. Then, a state violating clauses of total weight l or less satisfies $\mathcal{T}_{_{det}}$ (cf. [9]). The state $(0, ...0)$ is known to satisfy $\mathcal{T}_{_{det}}$ such that the chain could also start from this node, but MaxWalkSat will find a node that will be (near to) the mode of the distribution such that the chain will approximate faster. Since all transitions to states not satisfying $\mathcal{T}_{_{det}}$ have probability 0, the whole chain does only involve states satisfying $\mathcal{T}_{_{det}}$. In other words, it is guaranteed that $P(mb(X_i)) > 0$ such that the transition distribution $\mathbf{P}(X_i \mid mb(X_i))$ is defined. Given $KB_{_M}$, this distribution is computed with (1) where functions f_i are defined according to (3).

The Gibbs-$\mathcal{ALH}^-$-algorithm is depicted in Fig. 2. If the value of X_i is flipped, the method *setRelease* is called with parameters X_i, the old and new value of X_i, x_i resp. x'_i, and the set Cl_i of deterministic ground clauses containing X_i. This method sets and releases blockings as specified in Table 2. At the beginning of the process, it has to be ensured that all initial blockings are set. This is done by calling *set*, a function similar to *setRelease*, but only increasing γ_i. Then, the settings and releases depicted in Table 2 ensure that for each X_i, $\gamma_i > 0$ if and only if either $P(x_i \mid mb(X_i)) = 0$ or $P(\neg x_i \mid mb(X_i)) = 0$. Finally, after $n \cdot N$ samples, the Boolean vector $\mathbf{N}[X]$ of counts over X is normalized by α and the result represents the estimated distribution of $\mathbf{P}(X)$. The method can further be optimized by restricting $\boldsymbol{X}$ to a set of ground atoms assumed to be relevant for the query answer.

```
Algorithm Gibbs-$\mathcal{ALH}^-(KB_{_M}, X, N)$
Output An estimate of $\mathbf{P}(X)$
    local variables:
        $\mathbf{N}[X]$, a vector of counts over $X$
        $Cl$, a set of all (weighted) ground clauses obtained from $KB_{_M}$
        $\boldsymbol{X} = \{X_1, ..., X_n\}$, the set of all random variables
        $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_n\}$, a set of integers initially assigned with 0
        $\boldsymbol{x} = (x_1, ..., x_n)$, an initial state satisfying $\mathcal{T}_{_{det}}$
    for i = 1 to $n$ do if $(\gamma_i = 0)$ set($X_i$, $\neg x_i$, $x_i$, $Cl_i$);
    for j = 1 to $N$ do
        for i = 1 to $n$ do
            if $(\gamma_i = 0)$
                sample the value $x'_i$ of $X_i$ in $\boldsymbol{x}$ from $\mathbf{P}(X_i \mid mb(X_i))$;
                if $(x_i \neq x'_i)$ setRelease($X_i$, $x_i$, $x'_i$, $Cl_i$);
            $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where $x$ is the value of $X$ in $\boldsymbol{x}$
    return $\alpha$ <$\mathbf{N}[1], \mathbf{N}[0]$>
```

Fig. 2. The Gibbs-$\mathcal{ALH}^-$ algorithm

6 Conclusion

We have shown that deterministic dependencies in $\mathcal{ALH}^-$ retain the regularity of Markov chains constructed with Gibbs sampling. Further, we proposed a method incorporating these dependencies. Since lots of redundant samples are not generated by this method, we conclude that significant efficiency is gained. This is in contrast to previous results, where Gibbs sampling with (near-)determinism is known to give poor results. While $\mathcal{T}_{det}$ is specified with $\mathcal{ALH}^-$, the sets $\mathcal{T}_w$ and $\mathcal{A}_w$ of weighted axioms resp. weighted assertions can e.g. be specified in the expressive DL $\mathcal{SHQ}$. In [7] it is shown that Gibbs sampling slows down more and more when clauses are assigned with weights beyond 4. Note that a world not satisfying a ground clause with weight 4 is as probable as a world not satisifying e^3 ground clauses with weight 1. Thus, for ground clauses not intended to represent deterministic knowledge, a weight around 4 usually will suit the requirements of a knowledge engineer. However, often, in addition to axioms in $\mathcal{ALH}^-$, there is also the need to represent deterministic assertions in $\mathcal{A}_{det}$ as well as deterministic functional- and transitive roles. Further research is required, in order to also incorporate these dependencies. Another open task is the implementation of the proposed method in order to compare its runtime performance with other approaches estimating probability distributions under consideration of (near-)deterministic knowledge (such as MC-SAT [7] or SampleSearch [8]).

References

1. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, CA, 1988.
2. Koller, D., Levy, A., Pfeffer, A., Getoor, L., Taskar, B.: Graphical Models in a Nutshell. In: Introduction to Statistical Relational Learning, pages 13–55, Cambridge, MA: MIT Press, 2007.
3. Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, January 2003.
4. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (Second Edition) Prentice Hall, 2003.
5. Geman, S., Geman, D.: Stochastic Relaxation, Gibbs Distribution and Bayesian Restoration of Images. IEE Transactions on Pattern Analysis and Machine Intelligence 6, pages 721–741, 1984.
6. Domingos, P., Richardson, M.: Markov Logic: A Unifying Framework for Statistical Relational Learning. In: Introduction to Statistical Relational Learning, pages 339–371, Cambridge, MA: MIT Press, 2007.
7. Poon, H., Domingos, P.: Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In Proceedings of AAAI-06, Boston, Massachusetts, July 2006.
8. Gogate, V., Dechter, R.: SampleSearch: Importance Sampling in presence of Determinism. ICS technical report (Submitted), July 2009.
9. Jiang, Y., Kautz, H., Selman, B.: Solving Problems with Hard and Soft Constraints using a Stochastic Algorithm for MAX-SAT. In First International Joint Workshop on Artificial Intelligence and Operations Research, 1995.

Towards Approximative Most Specific Concepts by Completion for $\mathcal{EL}$ with Subjective Probabilities

Rafael Peñaloza and Anni-Yasmin Turhan

TU Dresden, Germany, email: *last name*@tcs.inf.tu-dresden.de

1 Introduction

In Description Logics the inference *most specific concept* (msc) constructs a concept description that generalizes an individual into a concept description. For the Description Logic $\mathcal{EL}$ the msc needs not exist [1], if computed with respect to general $\mathcal{EL}$-TBoxes. However, it is still possible to find a concept description that is the msc up to a fixed role-depth. In this paper we present a practical approach for computing the role-depth bounded msc, based on the polynomial-time completion algorithm for $\mathcal{EL}$. We extend this method to a simple probabilistic variant of $\mathcal{EL}$ that can express subjective probabilities and that was recently introduced in [6]. The probabilistic DL that we use, called Prob-$\mathcal{EL}_c^{01}$, allows only a fairly limited use of uncertainty. More precisely, it is only possible to express that a concept *may* hold ($P_{>0}C$), or that it holds *almost surely* ($P_{=1}C$). Despite its limited expressivity, this logic is interesting due to its nice algorithmic properties; as shown in [6], subsumption can be decided in polynomial time and instance checking can be performed in polynomial time as well.

Many practical applications that need to represent probabilistic information, such as medical applications or context-aware applications, need to characterize that observations only hold with certain probability. Furthermore, these applications face the problem that information from different sources does not coincide or that different diagnoses yield differing results. These applications need to "integrate" differing observations for the same state of affairs. A way to determine what the different information sources agree upon is to represent this information as ABox individuals and to find a common generalization of these individuals. A description of such a generalization of a group of ABox individuals can be obtained by applying the so-called *bottom-up approach* for constructing knowledge bases [4]. In this approach a set of individuals is generalized into a single concept description by first generating the msc of each concept and then apply the least common subsumer (lcs) to the set of obtained concept descriptions to extract their commonalities.

The second step, i.e., a computation procedure for the approximative lcs has been investigated for $\mathcal{EL}$ and Prob-$\mathcal{EL}_c^{01}$ in [8]. In this paper we present a similar procedure for the msc. We devise a practical algorithm for computing the msc up to a certain role-depth for $\mathcal{EL}$ and Prob-$\mathcal{EL}_c^{01}$. The so-called $k\text{-}msc$ obtained by the algorithm is still a generalization of the input, but not necessarily the least one – in this sense it is only an approximation of the msc. Moreover, our algorithms are based upon the completion algorithms for $\mathcal{EL}$ and Prob-$\mathcal{EL}_c^{01}$ and thus can be easily implemented on top of reasoners of these DLs. Due to space limitations the proofs can be found in [7].

2 $\mathcal{EL}$ and Prob-$\mathcal{EL}$

We introduce the DL $\mathcal{EL}$ and its probabilistic variant Prob-$\mathcal{EL}_c^{01}$. Let N_I, N_C and N_R be disjoint sets of *individual-*, *concept-* and *role names*, respectively. *Prob-$\mathcal{EL}_c^{01}$-concept descriptions* are built using the syntax rule

$$C ::= \top \mid A \mid C \sqcap D \mid \exists r.C \mid P_{>0}C \mid P_{=1}C,$$

where $A \in N_C$, and $r \in N_R$. $\mathcal{EL}$-concept descriptions are Prob-$\mathcal{EL}_c^{01}$-concept description that do not contain the constructors $P_{>0}$ or $P_{=1}$.

A *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. An $\mathcal{EL}$- (Prob-$\mathcal{EL}_c^{01}$-)TBox is a finite set of *concept inclusions* (CIs) of the form $C \sqsubseteq D$, where C, D are $\mathcal{EL}$- (Prob-$\mathcal{EL}_c^{01}$-)concept descriptions. An $\mathcal{EL}$-ABox is a set of assertions of the form $C(a), r(a,b)$, where C is an $\mathcal{EL}$-concept description, $r \in N_R$, and $a, b \in N_I$. A Prob-$\mathcal{EL}_c^{01}$-ABox is a set of assertions of the form $C(a), r(a,b), P_{>0}r(a,b), P_{=1}r(a,b)$, where C is a Prob-$\mathcal{EL}_c^{01}$-concept description, $r \in N_R$, and $a, b \in N_I$.

The semantics of $\mathcal{EL}$ is defined by means of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names, subsets of $\Delta^{\mathcal{I}}$ to concepts and elements of $\Delta^{\mathcal{I}}$ to individual names. For a more detailed description of this semantics, see [3].

An interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; it *satisfies* an assertion $C(a)$ ($r(a,b)$), denoted as $\mathcal{I} \models C(a)$ ($\mathcal{I} \models r(a,b)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$). It is a *model* of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies all CIs in $\mathcal{T}$ and all assertions in $\mathcal{A}$.

The semantics of Prob-$\mathcal{EL}_c^{01}$ generalizes the semantics of $\mathcal{EL}$. A *probabilistic interpretation* is of the form

$$\mathcal{I} = (\Delta^{\mathcal{I}}, W, (\mathcal{I}_w)_{w \in W}, \mu),$$

where $\Delta^{\mathcal{I}}$ is the (non-empty) *domain*, W is a set of *worlds*, μ is a discrete probability distribution on W, and for each world $w \in W$, $\mathcal{I}_w$ is a classical $\mathcal{EL}$ interpretation with domain $\Delta^{\mathcal{I}}$, where $a^{\mathcal{I}_w} = a^{\mathcal{I}_{w'}}$ for all $a \in N_I$, $w, w' \in W$. The probability that a given element of the domain $d \in \Delta^{\mathcal{I}}$ belongs to the interpretation of a concept name A is

$$p_d^{\mathcal{I}}(A) := \mu(\{w \in W \mid d \in A^{\mathcal{I}_w}\}).$$

The functions $\mathcal{I}_w$ and $p_d^{\mathcal{I}}$ are extended to complex concepts in the usual way for the classical $\mathcal{EL}$ constructors, where the extension to the new constructors P_* is defined as

$$(P_{>0}C)^{\mathcal{I}_w} := \{d \in \Delta^{\mathcal{I}} \mid p_d^{\mathcal{I}}(C) > 0\}, \qquad (P_{=1}C)^{\mathcal{I}_w} := \{d \in \Delta^{\mathcal{I}} \mid p_d^{\mathcal{I}}(C) = 1\}.$$

A probabilistic interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ if for every $w \in W$ it holds that $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$. It is a *model* of a TBox $\mathcal{T}$ if it satisfies all concept inclusions in $\mathcal{T}$. Let C, D be two Prob-$\mathcal{EL}_c^{01}$ concepts and $\mathcal{T}$ a TBox. We say that C is *subsumed* by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$. The probabilistic interpretation $\mathcal{I}$ *satisfies* the assertion $P_{>0}r(a,b)$ if $\mu(\{w \in W \mid I_w \models r(a,b)\}) > 0$, and analogously for $P_{=1}r(a,b)$. $\mathcal{I}$ *satisfies* the ABox $\mathcal{A}$ if there is a $w \in W$ such that $\mathcal{I}_w \models \mathcal{A}$.

Finally, an individual $a \in N_I$ is an *instance* of a concept description C w.r.t. $\mathcal{K}$ ($\mathcal{K} \models C(a)$) if $\mathcal{I} \models C(a)$ for all models $\mathcal{I}$ of $\mathcal{K}$. The *ABox realization problem* is to compute for each individual a in $\mathcal{A}$ the set of named concepts from $\mathcal{K}$ that have a as an instance and that are least (w.r.t. $\sqsubseteq$). In this paper we are interested in computing most specific concepts.

Definition 1 (most specific concept). *Let $\mathcal{L}$ be a DL, $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) be a $\mathcal{L}$-KB. The* most specific concept *(msc) of an individual a from $\mathcal{A}$ is the $\mathcal{L}$-concept description C s. t.*

1. *$\mathcal{K} \models C(a)$, and*
2. *for each $\mathcal{L}$-concept description D holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

The msc depends on the DL in use. For the DLs with conjunction as concept constructor the msc is, if it exists, unique up to equivalence. Thus it is justified to speak of *the* msc.

3 Completion-based Instance Checking Algorithms

Now we briefly sketch the completion algorithms for instance checking in $\mathcal{EL}$ [2] and Prob-$\mathcal{EL}_c^{01}$ [6].

3.1 Completion Algorithms for $\mathcal{EL}$

Assume we want to test for an $\mathcal{EL}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ whether $\mathcal{K} \models D(a)$ holds. The completion algorithm first augments the knowledge base by introducing a concept name for the complex concept description D from the instance check, i.e., it sets $\mathcal{K} = (\mathcal{T} \cup \{A_q \equiv D\}, \mathcal{A})$, where A_q is a new concept name not appearing in $\mathcal{K}$. The instance checking algorithm for $\mathcal{EL}$ works on normalized knowledge bases. The normalization is done in two steps: first the ABox is transformed into a simple ABox. An ABox is a *simple ABox*, if it only contains concept names in concept assertions. An $\mathcal{EL}$-ABox $\mathcal{A}$ can be transformed into a simple ABox by first replacing each complex assertion $C(A)$ in $\mathcal{A}$ by $A(a)$ with a fresh name A and, second, introduce $A \equiv C$ in the TBox.

After this step the TBox is normalized. For a concept description C let $\mathsf{CN}(C)$ denote the set of all concept names and $\mathsf{RN}(C)$ denote the set of all role names that appear in C. The *signature of a concept description* C (denoted $\mathsf{sig}(C)$) is $\mathsf{CN}(C) \cup \mathsf{RN}(C)$. Similarly, the set of concept (role) names that appear in a TBox are denoted by $\mathsf{CN}(\mathcal{T})$ ($\mathsf{RN}(\mathcal{T})$). The *signature of a TBox* $\mathcal{T}$ (denoted $\mathsf{sig}(\mathcal{T})$) is $\mathsf{CN}(\mathcal{T}) \cup \mathsf{RN}(\mathcal{T})$. The *signature of an ABox* $\mathcal{A}$ (denoted $\mathsf{sig}(\mathcal{A})$) is the set of concept (role / individual) names $\mathsf{CN}(\mathcal{A})$ ($\mathsf{RN}(\mathcal{A})$/$\mathsf{IN}(\mathcal{A})$ resp.) that appear in $\mathcal{A}$. The signature of a KB $\mathcal{K}$ = ($\mathcal{T}$, $\mathcal{A}$) (denoted $\mathsf{sig}(\mathcal{K})$) is $\mathsf{sig}(\mathcal{T}) \cup \mathsf{sig}(\mathcal{A})$.

Now, an $\mathcal{EL}$-TBox $\mathcal{T}$ is in *normal form* if all concept axioms have one of the following forms, where $C_1, C_2 \in \mathsf{sig}(\mathcal{T})$ and $D \in \mathsf{sig}(\mathcal{T}) \cup \{\bot\}$:

$$C_1 \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C_1 \sqsubseteq \exists r.C_2 \quad \text{or} \quad \exists r.C_1 \sqsubseteq D.$$

Any $\mathcal{EL}$-TBox can be transformed into normal form by introducing new concept names and by applying the normalization rules displayed in Figure 1 exhaustively. These rules replace the GCI on the left-hand side of the rules with the set of GCIs on the right-hand

$$\begin{array}{ll}
\textbf{NF1} & C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\ \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E\ \} \\
\textbf{NF2} & \exists r.\hat{C} \sqsubseteq D \longrightarrow \{\ \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D\ \} \\
\textbf{NF3} & \hat{C} \sqsubseteq \hat{D} \longrightarrow \{\ \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}\ \} \\
\textbf{NF4} & B \sqsubseteq \exists r.\hat{C} \longrightarrow \{\ B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}\ \} \\
\textbf{NF5} & B \sqsubseteq C \sqcap D \longrightarrow \{\ B \sqsubseteq C, B \sqsubseteq D\ \}
\end{array}$$

where $\hat{C}, \hat{D} \notin \mathsf{BC}_{\mathcal{T}}$ and A is a new concept name.

Fig. 1. $\mathcal{EL}$ normalization rules (from [2])

side. Clearly, for a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ the signature of $\mathcal{A}$ may be changed only during the first of the two normalization steps and the signature of $\mathcal{T}$ may be extended during both of them. The normalization of the KB can be done in linear time.

The completion algorithm for instance checking is based on the one for classifying $\mathcal{EL}$-TBoxes introduced in [2]. The completion algorithm constructs a representation of the minimal model of $\mathcal{K}$. Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be a normalized $\mathcal{EL}$-KB, i.e., with a simple ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ in normal form. The completion algorithm works on four kinds of *completion sets*: $S(a), S(a, r), S(C)$ and $S(C, r)$ for each $a \in \mathsf{IN}(\mathcal{A})$, $C \in \mathsf{CN}(\mathcal{K})$ and $r \in \mathsf{RN}(\mathcal{K})$. The completion sets contain concept names from $\mathsf{CN}(\mathcal{K})$. Intuitively, the completion rules make implicit subsumption and instance relationships explicit in the following sense:

– $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
– $D \in S(C, r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.
– $D \in S(a)$ implies that a is an instance of D w.r.t. $\mathcal{K}$,
– $D \in S(a, r)$ implies that a is an instance of $\exists r.D$ w.r.t. $\mathcal{K}$.

$\mathsf{S}_{\mathcal{K}}$ denotes the set of all completion sets of a normalized $\mathcal{K}$. The completion sets are initialized for each $a \in \mathsf{IN}(\mathcal{A})$ and each $C \in \mathsf{CN}(\mathcal{K})$ as follows:

– $S(C) := \{C, \top\}$ for each $C \in \mathsf{CN}(\mathcal{K})$,
– $S(C, r) := \emptyset$ for each $r \in \mathsf{RN}(\mathcal{K})$,
– $S(a) := \{C \in \mathsf{CN}(\mathcal{A}) \mid C(a) \text{ appears in } \mathcal{A}\} \cup \{\top\}$, and
– $S(a, r) := \{b \in \mathsf{IN}(\mathcal{A}) \mid r(a, b) \text{ appears in } \mathcal{A}\}$ for each $r \in \mathsf{RN}(\mathcal{K})$.

Then these sets are extended by applying the completion rules shown in Figure 2 until no more rule applies. In these rules X and Y can refer to concept or individual names, while C, C_1, C_2 and D are concept names and r is a role name. After the completion has terminated, the following relations hold between an individual a, a role r and named concepts A and B:

– subsumption relation between A and B from $\mathcal{K}$ holds iff $B \in S(A)$
– instance relation between a and B from $\mathcal{K}$ holds iff $B \in S(a)$,

which has been shown in [2]. To decide the initial query: $\mathcal{K} \models D(a)$, one has to test now, whether A_q appears in $S(a)$. In fact, instance queries for all individuals and all named concepts from the KB can be answered now; the completion algorithm does not only perform one instance check, but complete ABox realization. The completion algorithm runs in polynomial time in size of the knowledge base.

CR1 If $C \in S(X)$, $C \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR2 If $C_1, C_2 \in S(X)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S(X)$
then $S(X) := S(X) \cup \{D\}$

CR3 If $C \in S(X)$, $C \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S(X, r)$
then $S(X, r) := S(X, r) \cup \{D\}$

CR4 If $Y \in S(X, r)$, $C \in S(Y)$, $\exists r.C \sqsubseteq D \in \mathcal{T}$, and
$D \notin S(X)$ then $S(X) := S(X) \cup \{D\}$

Fig. 2. $\mathcal{EL}$ completion rules

3.2 Completion Algorithms for Prob-$\mathcal{EL}$

To describe the completion algorithm for Prob-$\mathcal{EL}$, we need the notion of basic concepts. The set $\mathsf{BC}_{\mathcal{T}}$ of Prob-$\mathcal{EL}_c^{01}$ *basic concepts* for a KB $\mathcal{K}$ is the smallest set that contains (1) $\top$, (2) all concept names used in $\mathcal{K}$, and (3) all concepts of the form $P_* A$, where A is a concept name in $\mathcal{K}$. A Prob-$\mathcal{EL}_c^{01}$-TBox $\mathcal{T}$ is in *normal form* if all its axioms are of one of the following forms

$$C \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C \sqsubseteq \exists r.A, \quad \exists r.A \sqsubseteq D,$$

where $C, C_1, C_2, D \in \mathsf{BC}_{\mathcal{T}}$ and A is a concept name. The normalization rules in Figure 1 can also be used to transform a Prob-$\mathcal{EL}_c^{01}$-TBox into this extended notion of normal form. We further assume that for all assertions $C(a)$ in the ABox $\mathcal{A}$, C is a concept name. We denote as $\mathcal{P}_0^{\mathcal{T}}$, $\mathcal{P}_1^{\mathcal{T}}$ and $\mathcal{R}_0^{\mathcal{T}}$ the set of all concepts of the form $P_{>0}A$, $P_{=1}A$, and $P_{>0}r(a, b)$ respectively, occurring in a normalized knowledge base $\mathcal{K}$.

The completion algorithm for Prob-$\mathcal{EL}_c^{01}$ follows the same idea as the algorithm for $\mathcal{EL}$, but uses several completion sets to deal with the information of what needs to be satisfied in the different worlds of a model. We define the set of worlds $V := \{0, \varepsilon, 1\} \cup \mathcal{P}_0^{\mathcal{T}} \cup \mathcal{R}_0^{\mathcal{T}}$, where the probability distribution μ assigns a probability of 0 to the world 0, and the uniform probability $1/(|V| - 1)$ to all other worlds. For each individual name a, concept name A, role name r and world v, we store the completion sets $S_0(A, v), S_\varepsilon(A, v), S_0(A, r, v)$, $S_\varepsilon(A, r, v)$, $S(a, v)$, and $S(a, r, v)$.

The algorithm initializes the sets as follows for every $A \in \mathsf{BC}_{\mathcal{T}}, r \in \mathsf{RN}(\mathcal{K})$, and $a \in \mathsf{IN}(\mathcal{A})$:

– $S_0(A, 0) = \{\top, A\}$ and $S_0(A, v) = \{\top\}$ for all $v \in V \setminus \{0\}$,
– $S_\varepsilon(A, \varepsilon) = \{\top, A\}$ and $S_\varepsilon(A, v) = \{\top\}$ for all $v \in V \setminus \{\varepsilon\}$,
– $S(a, 0) = \{\top\} \cup \{A \mid A(a) \in \mathcal{A}\}$, $S(a, v) = \{\top\}$ for all $v \neq 0$,
– $S_0(A, r, v) = S_\varepsilon(A, r, v) = \emptyset$ for all $v \in V$, $S(a, r, v) = \emptyset$ for $v \neq 0$,
– $S(a, r, 0) = \{b \in \mathsf{IN}(\mathcal{A}) \mid r(a, b) \in \mathcal{A}\}$.

These sets are then extended by exhaustively applying the rules shown in Figure 3, where X ranges over $\mathsf{BC}_{\mathcal{T}} \cup \mathsf{IN}(\mathcal{A})$, $S_*(X, v)$ stands for $S(X, v)$ if X is an individual and for $S_0(X, v), S_\varepsilon(X, v)$ if $X \in \mathsf{BC}_{\mathcal{T}}$, and $\gamma : V \to \{0, \varepsilon\}$ is defined by $\gamma(0) = 0$, and $\gamma(v) = \varepsilon$ for all $v \in V \setminus \{0\}$.

PR1 If $C' \in S_*(X, v)$, $C' \sqsubseteq D \in \mathcal{T}$, and $D \notin S_*(X, v)$
then $S_*(X, v) := S_*(X, v) \cup \{D\}$

PR2 If $C_1, C_2 \in S_*(X, v)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S_*(X, v)$
then $S_*(X, v) := S_*(X, v) \cup \{D\}$

PR3 If $C' \in S_*(X, v)$, $C' \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S_*(X, r, v)$
then $S_*(X, r, v) := S_*(X, r, v) \cup \{D\}$

PR4 If $D \in S_*(X, r, v)$, $D' \in S_{\gamma(v)}(D, \gamma(v))$, $\exists r.D' \sqsubseteq E \in \mathcal{T}$,
and $E \notin S_*(X, v)$ then $S_*(X, v) := S_*(X, v) \cup \{E\}$

PR5 If $P_{>0}A \in S_*(X, v)$, and $A \notin S_*(X, P_{>0}A)$
then $S_*(X, P_{>0}A) := S_*(X, P_{>0}A) \cup \{A\}$

PR6 If $P_{=1}A \in S_*(X, v)$, $v \neq 0$, and $A \notin S_*(X, v)$
then $S_*(X, v) := S_*(X, v) \cup \{A\}$

PR7 If $A \in S_*(X, v)$, $v \neq 0$, $P_{>0}A \in \mathcal{P}_0^{\mathcal{T}}$, and $P_{>0}A \notin S_*(X, v')$
then $S_*(X, v') := S_*(X, v') \cup \{P_{>0}A\}$

PR8 If $A \in S_*(X, 1)$, $P_{=1}A \in \mathcal{P}_1^{\mathcal{T}}$, and $P_{=1}A \notin S_*(X, v)$
then $S_*(X, v) := S_*(X, v) \cup \{P_{=1}A\}$

PR9 If $r(a, b) \in \mathcal{A}, C \in S(b, 0), \exists r.C \sqsubseteq D \in \mathcal{T}$,
and $D \notin S(a, 0)$ then $S(a, 0) := S(a, 0) \cup \{D\}$

PR10 If $P_{>0}r(a, b) \in \mathcal{A}$, $C \in S(b, P_{>0}r(a, b)), \exists r.C \sqsubseteq D \in \mathcal{T}$,
and $D \notin S(a, P_{>0}r(a, b))$
then $S(a, P_{>0}r(a, b)) := S(a, P_{>0}r(a, b)) \cup \{D\}$

PR11 If $P_{=1}r(a, b) \in \mathcal{A}$, $C \in S(b, v)$ with $v \neq 0$, $\exists r.C \sqsubseteq D \in \mathcal{T}$
and $D \notin S(a, v)$ then $S(a, v) := S(a, v) \cup \{D\}$

Fig. 3. Prob-$\mathcal{EL}_c^{01}$ completion rules

This algorithm terminates in polynomial time. After termination, the completion sets store all the information necessary to decide subsumption of concept names, as well as checking whether an individual is an instance of a given concept name [6]. For the former decision, it holds that for every pair A, B of concept names: $B \in S_0(A, 0)$ iff $A \sqsubseteq_{\mathcal{K}} B$. In the case of instance checking, we have that $\mathcal{K} \models A(a)$ iff $A \in S(a, 0)$.

4 Computing the k-MSC using Completion

The msc was first investigated for $\mathcal{EL}$-concept descriptions and w.r.t. unfoldable TBoxes and possibly cyclic ABoxes in [5]. It was shown that the msc does not need to exists for cyclic ABoxes. Consider the ABox $\mathcal{A} = \{r(a, a), C(a)\}$. The msc of a is then

$$C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \cdots$$

and cannot be expressed by a finite concept description. For cyclic TBoxes it has been shown in [1] that the msc does not need to exists even if the ABox is acyclic.

To avoid infinite nestings in presence of cyclic ABoxes it was proposed in [5] to limit the role-depth of the concept description to be computed. This limitation yields an approximation of the msc, which is still a concept description with the input individual as an instance, but it does not need to be the least one (w.r.t. $\sqsubseteq$) with this property. We follow this idea to compute approximative msc also in presence of general TBoxes.

The *role-depth* of a concept description C (denoted $rd(C)$) is the maximal number of nested quantifiers of C. Now we can define the msc with limited role-depth for $\mathcal{EL}$.

Definition 2 (role-depth bounded $\mathcal{EL}$-msc). *Let $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) be an $\mathcal{EL}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then the $\mathcal{EL}$-concept description C is the* role-depth bounded $\mathcal{EL}$-most specific concept *of a w.r.t. $\mathcal{K}$ and role-depth k (written k-$msc_{\mathcal{K}}(a)$) iff*

1. *$rd(C) \leq k$,*
2. *$\mathcal{K} \models C(a)$, and*
3. *for all $\mathcal{EL}$-concept descriptions E with $rd(E) \leq k$ holds: $\mathcal{K} \models E(a)$ implies $C \sqsubseteq_{\mathcal{T}} E$.*

Please note that in case the exact msc has a role-depth less than k the role-depth bounded msc is the exact msc.

4.1 Computing the k-msc in $\mathcal{EL}$ by completion

The computation of the msc relies on a characterization of the instance relation. While in earlier works this was given by homomorphism [5] or simulations [1] between graph representations of the knowledge base and the concept in question, we use the completion algorithm as such a characterization. Furthermore, we construct the msc by traversing the completion sets to "collect" the msc. More precisely, the set of completion sets encodes a graph structure, where the sets $S(X)$ are the nodes and the sets $S(X, r)$ encode the edges. Traversing this graph structure, one can construct an $\mathcal{EL}$-concept. To obtain a finite concept in the presence of cyclic ABoxes or TBoxes one has to limit the role-depth of the concept to be obtained.

Definition 3 (traversal concept). *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ be its normalized form, $\mathsf{S}_{\mathcal{K}}$ the completion set obtained from $\mathcal{K}$ and $k \in \mathbb{N}$. Then the* traversal concept of a named concept *A (denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(A)$) with $\mathsf{sig}(A) \subseteq \mathsf{sig}(\mathcal{K}'')$ is the concept obtained from executing the procedure call* traversal-concept-c*(A, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 1.*

The traversal concept of an individual *a (denoted k-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$) with $a \subseteq \mathsf{sig}(\mathcal{K})$ is the concept description obtained from executing the procedure call* traversal-concept-i*(a, $\mathsf{S}_{\mathcal{K}}$, k) shown in Algorithm 1.*

The idea is that the traversal concept of an individual yields its msc. However, the traversal concept contains names from $\mathsf{sig}(\mathcal{K}'') \setminus \mathsf{sig}(\mathcal{K})$, i.e., concept names that were introduced during normalization – we call this kind of concept names *normalization names* in the following. The returned msc should be formulated w.r.t. the signature of the original KB, thus the normalization names need to be removed or replaced.

Algorithm 1 Computation of a role-depth bounded $\mathcal{EL}$-msc.

```
Procedure k-msc $(a, \mathcal{K}, k)$
Input: $a$: individual from $\mathcal{K}$; $\mathcal{K}$ =($\mathcal{T}$, $\mathcal{A}$) an $\mathcal{EL}$-KB; $k \in \mathbb{N}$
Output: role-depth bounded $\mathcal{EL}$-msc of $a$ w.r.t. $\mathcal{K}$ and $k$.
1: ($\mathcal{T}'$, $\mathcal{A}'$) := simplify-ABox($\mathcal{T}$, $\mathcal{A}$)
2: $\mathcal{K}''$ := (normalize($\mathcal{T}'$), $\mathcal{A}'$)
3: $\mathsf{S}_{\mathcal{K}}$ := apply-completion-rules($\mathcal{K}$)
4: return Remove-normalization-names ( traversal-concept-i$(a, \mathsf{S}_{\mathcal{K}}, k)$)

Procedure traversal-concept-i ($a$, S, $k$)
Input: $a$: individual name from $\mathcal{K}$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth traversal concept (w.r.t. $\mathcal{K}$) and $k$.
1: if $k = 0$ then return $\bigsqcap_{A \in S(a)} A$
2: else return $\bigsqcap_{A \in \mathsf{S}(a)} A \sqcap$
       $\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{A \in \mathsf{CN}(\mathcal{K}'') \cap S(a,r)} \exists r.$ traversal-concept-c $(A, \mathsf{S}, k-1) \sqcap$
       $\bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{b \in \mathsf{IN}(\mathcal{K}'') \cap S(a,r)} \exists r.$ traversal-concept-i $(b, \mathsf{S}, k-1)$
3: end if

Procedure traversal-concept-c ($A$, S, $k$)
Input: $A$: concept name from $\mathcal{K}''$; S: set of completion sets; $k \in \mathbb{N}$
Output: role-depth bounded traversal concept.
1: if $k = 0$ then return $\bigsqcap_{B \in S(A)} B$
2: else return $\bigsqcap_{B \in S(A)} B \sqcap \bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \bigsqcap_{B \in S(A,r)} \exists r.$traversal-concept-c $(B, \mathsf{S}, k-1)$
3: end if
```

Lemma 1. *Let $\mathcal{K}$ be an $\mathcal{EL}$-KB, $\mathcal{K}''$ its normalized version, $\mathsf{S}_{\mathcal{K}}$ be the set of completion sets obtained for $\mathcal{K}$, $k \in \mathbb{N}$ a natural number and $a \in \mathsf{IN}(\mathcal{K})$. Furthermore let $C = k$-$\mathsf{C}_{\mathsf{S}_{\mathcal{K}}}(a)$ and $\widehat{C}$ be obtained from C by removing the normalization names. Then*

$$\mathcal{K}'' \models C(a) \textit{ iff } \mathcal{K} \models \widehat{C}(a).$$

This lemma guarantees that removing the normalization names from the traversal concept preserves the instance relationships. Intuitively, this lemma holds since the construction of the traversal concept conjoins exhaustively all named subsumers and all subsuming existential restrictions to a normalization name up to the role-depth bound. Thus removing the normalization name does not change the extension of the conjunction. The proof can be found in [7]. We are now ready to devise a computation algorithm for the role-depth bounded msc: procedure k-msc as displayed in Algorithm 1.

The procedure k-msc has an individual a from a knowledge base $\mathcal{K}$, the knowledge base $\mathcal{K}$ itself and number k for the role depth-bound as parameter. It first performs the two normalization steps on $\mathcal{K}$, then applies the completion rules from Figure 2 to the normalized KB $\mathcal{K}''$ and stores the set of completion sets in $\mathsf{S}_{\mathcal{K}}$. Afterwards it computes the traversal-concept of a from $\mathsf{S}_{\mathcal{K}}$ w.r.t. role-depth bound k. In a post-processing step it applies Remove-normalization-names to the traversal concept.

Obviously, the concept description returned from the procedure $\mathsf{k\text{-}msc}$ has a role-depth less or equal to k. Thus the first condition of Definition 2 is fulfilled. We prove next that the concept description obtained from $\mathsf{k\text{-}msc}$ fulfills the second condition from Definition 2.

Lemma 2. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. If $C = \mathsf{k\text{-}msc}(a, \mathcal{K}, k)$, then $\mathcal{K} \models C(a)$.*

The claim can be shown by induction on k. Each name in C is from a completion set of (1) an individual or (2) a concept, which is connected via existential restrictions to an individual. The full proof can be found in [7].

Lemma 3. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. If $C = \mathsf{k\text{-}msc}(a, \mathcal{K}, k)$, then for all $\mathcal{EL}$-concept descriptions E with $rd(E) \leq k$ holds: $\mathcal{K} \models E(a)$ implies $C \sqsubseteq_{\mathcal{T}} E$.*

Again, the full proof can be found in [7]. The two lemmas yield the correctness of the overall procedure.

Theorem 1. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$-KB and a an individual in $\mathcal{A}$ and $k \in \mathbb{N}$. Then $\mathsf{k\text{-}msc}(a, \mathcal{K}, k) \equiv k\text{-}msc_{\mathcal{K}}(a)$.*

The $k\text{-}msc$ can grow exponential in the size of the knowledge base.

4.2 Most specific concept in Prob-$\mathcal{EL}^{01}_{c}$

In order to compute the msc, we simply accumulate all concepts to which the individual a belongs, given the information in the completion sets. This process needs to be done recursively in order to account for both, the successors of a explicitly encoded in the ABox, and the nesting of existential restrictions masked by normalization names. In the following we use the abbreviation $S^{>0}(a, r) := \bigcup_{v \in V \setminus \{0\}} S(a, r, v)$. We then define $\mathsf{traversal\text{-}concept\text{-}i}(a, \mathsf{S}, k)$ as

$$\begin{aligned}\bigsqcap_{B \in S(a,0)} B \sqcap \bigsqcap_{r \in \mathsf{RN}(\mathcal{K}'')} \Big(& \bigsqcap_{r(a,b) \in \mathcal{K}''} \exists r.\mathsf{traversal\text{-}concept\text{-}i}(b, \mathsf{S}, k-1) \sqcap \\ & \bigsqcap_{B \in \mathsf{CN}(\mathcal{K}'') \cap S(a,r,0)} \exists r.\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k-1) \sqcap \\ & \bigsqcap_{B \in \mathsf{CN}(\mathcal{K}'') \cap S(a,r,1)} P_{=1}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k-1)) \sqcap \\ & \bigsqcap_{B \in \mathsf{CN}(\mathcal{K}'') \cap S^{>0}(a,r)} P_{>0}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k-1))\Big),\end{aligned}$$

where $\mathsf{traversal\text{-}concept\text{-}c}(B, \mathsf{S}, k+1)$ is

$$\begin{aligned}\bigsqcap_{C \in S_0(B,0)} B \sqcap \bigsqcap_{r \in \mathsf{RN}} \Big(& \bigsqcap_{C \in S_0(B,r,0)} \exists r.\mathsf{traversal\text{-}concept\text{-}c}(C, \mathsf{S}, k) \sqcap \\ & \bigsqcap_{C \in S_0(B,r,1)} P_{=1}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(C, \mathsf{S}, k)) \sqcap \\ & \bigsqcap_{C \in S_0^{>0}(B,r)} P_{>0}(\exists r.\mathsf{traversal\text{-}concept\text{-}c}(C, \mathsf{S}, k))\Big)\end{aligned}$$

and traversal-concept-c$(B, \mathsf{S}, 0) = \bigsqcap_{C \in S_0(B,0)} B$. Once the traversal concept has been computed, it is possible to remove all normalization names preserving the instance relation, which gives us the msc in the original signature of $\mathcal{K}$. The proof can be found in [7].

Theorem 2. *Let $\mathcal{K}$ a Prob-$\mathcal{EL}_c^{01}$-knowledge base, $a \in \mathsf{IN}(\mathcal{A})$, and $k \in \mathbb{N}$; then* Remove-normalization-names(traversal-concept-i$(a, \mathsf{S}, k)) \equiv k\text{-}msc_{\mathcal{K}}(a)$.

5 Conclusions

In this paper we have presented a practical method for computing the role-depth bounded msc of $\mathcal{EL}$ concepts w.r.t. a general TBox. Our approach is based on the completion sets that are computed during realization of a KB. Thus, any of the available implementations of the $\mathcal{EL}$ completion algorithm can be easily extended to an implementation of the (approximative) msc computation algorithm. We also showed that the same idea can be adapted for the computation of the msc in the probabilistic DL Prob-$\mathcal{EL}_c^{01}$.

Together with the completion-based computation of role-depth limited (least) common subsumers given in [8] these results complete the bottom-up approach for general $\mathcal{EL}$- and Prob-$\mathcal{EL}_c^{01}$-KBs. This approach yields a practical method to compute commonalities for differing observations regarding individuals. To the best of our knowledge this has not been investigated for DLs that can express uncertainty.

References

1. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proc. of IJCAI'03*, pages 325–330. Morgan Kaufmann, 2003.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI'05*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. CUP., 2003.
4. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In T. Dean, editor, *Proc. of IJCAI'99*, pages 96–101, Stockholm, Sweden, 1999. Morgan Kaufmann.
5. R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, 15(1):47–59, 2002.
6. C. Lutz and L. Schröder. Probabilistic description logics for subjective probabilities. In F. Lin and U. Sattler, editors, *Proc. of KR'10*, 2010.
7. R. Peñaloza and A.-Y. Turhan. Completion-based computation of most specific concepts with limited role-depth for $\mathcal{EL}$ and prob-$\mathcal{EL}^{01}$. LTCS-Report LTCS-10-03, Chair f. Autom. Theory, Inst. f. Theor. C. S. TU Dresden, Germany, 2010.
8. R. Peñaloza and A.-Y. Turhan. Role-depth bounded least common subsumers by completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes. In V. Haarslev, D. Toman, and G. Weddell, editors, *Proc. of the 2010 Description Logic Workshop (DL'10)*, 2010.

Compatibility Formalization Between PR-OWL and OWL

Rommel Novaes Carvalho, Kathryn Laskey, and Paulo Costa

Department of Systems Engineering & Operations Research
School of Information Technology and Engineering
George Mason University, Fairfax, VA 22030, USA
rommel.carvalho@gmail.com
{klaskey,pcosta}@gmu.edu
http://seor.gmu.edu

Abstract. As stated in [5], a major design goal for PR-OWL was to attain compatibility with OWL. However, this goal has been only partially achieved as yet, primarily due to several key issues not fully addressed in the original work. This paper describes several important issues of compatibility between PR-OWL and OWL, and suggests approaches to deal with them. To illustrate the issues and how they can be addressed, we use procurement fraud as an example application domain [2]. First, we describe the lack of mapping between PR-OWL random variables (RVs) and the concepts defined in OWL, and then show how this mapping can be done. Second, we describe PR-OWL's lack of compatibility with existing types already present in OWL, and then show how every type defined in PR-OWL can be directly mapped to concepts already present in OWL.

Key words: OWL, PR-OWL, MEBN, probabilistic ontology, semantic web, compatibility

1 Introduction

The Semantic Web (SW) is predicated upon radical notions of information sharing, which include [1]: (i) the Anyone can say Anything about Any topic (AAA) requirement; (ii) the open world assumption, i.e. there may exist more information of which we are not aware, and (iii) nonunique naming, meaning that different people can assign different names to the same concept. The Semantic Web (SW) differs from the document web in that it is intended to provide not only information sharing, but also knowledge synergy. We call such an environment characterized a Radical Information Sharing (RIS) environment. While the SW promises great power and flexibility, RIS environments present fundamental challenges, and can lead to chaos, disagreement and conflict.

The challenge facing SW architects is therefore to avoid the natural chaos to which RIS environments are prone, and move to a state characterized by information sharing, cooperation and collaboration. According to [1], one solution to this challenge lies in modeling. Modeling is the process of organizing information for community use. Modeling supports information sharing in four ways: (1) It provides a framework for human communication; (2) it provides a means for explaining conclusions; (3) it provides a basis for formalization and automation of reasoning; and (4) it provides a structure for managing varying viewpoints.

There is an immense variety of modeling approaches. Different approaches and processes are supported by different modeling languages. One of special interest to this research is the Web Ontology Language (OWL) [18, 9]. OWL was developed with the aim of enabling achievement of the full SW potential. According to [18] OWL is intended for use when the information contained in documents needs to be processed by applications, as opposed to situations in which the content need only be presented to humans. OWL can be used to explicitly and formally represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology.

One of the first definitions of ontology in the context of the Semantic Web was given by Thomas Gruber [10].

> An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For Artificial Intelligence (AI) systems, what "exists" is that which can be represented. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly.

In the past few years, as the Semantic Web community has developed standards and more complex use cases, the need for principled approaches for representing and reasoning under uncertainty has received increasing appreciation. As a consequence, the World Wide Web Consortium (W3C) created the Uncertainty Reasoning for the World Wide Web Incubator Group (URW3-XG) in 2007 to identify requirements for reasoning with and representing uncertain information in the World Wide Web. The work of the URW3-XG provided an important beginning for characterizing the range of uncertainty that affects reasoning on the scale of the World Wide Web, and the issues to be considered in designing a standard representation of that uncertainty. However, the work to date likely falls short of what would be needed to charter an effort to develop that representation. A candidate representation for uncertainty reasoning in the semantic web is Probabilistic OWL (PR-OWL) [5], an OWL upper ontology for representing probabilistic ontologies based on Multi-Entity Bayesian Networks (MEBN) [15].

As stated in [5], a major design goal for PR-OWL was to attain compatibility with OWL. However, this goal has been only partially achieved as yet, due to several key issues not fully addressed in the original work. First, there is no mapping in PR-OWL to properties of OWL. Second, although PR-OWL has the concept of meta-entities, which allows the definition of complex types, it lacks compatibility with existing types already present in OWL.

These problems have been noted in the literature [20]:

> PR-OWL does not provide a proper integration of the formalism of MEBN and the logical basis of OWL on the meta level. More specifically, as the connection between a statement in PR-OWL and a statement in OWL is not formalized, it is unclear how to perform the integration of ontologies that contain statements of both formalisms.

This paper is structured as follows. Section 2 briefly describes PR-OWL and its underlying logic, MEBN. Section 3 presents PR-OWL's lack of mapping to OWL and our suggested solution to the problem. Section 4 presents the lack of compatibility between types in OWL and PR-OWL and describes how they could be integrated. Finally, Section 5 presents the conclusion of the proposed extension to PR-OWL language.

2 PR-OWL and MEBN Logic

Ontologies are becoming increasingly popular as a means to ensure formal semantic support for knowledge sharing [3, 4, 7, 8, 13, 22]. Representing and reasoning with uncertainty is becoming recognized as an essential capability in many domains. The naïve approach of simply annotating ontologies with numerical probabilities is inadequate, because it cannot capture complex relational probabilistic dependencies. More expressive representation formalisms are needed [16].

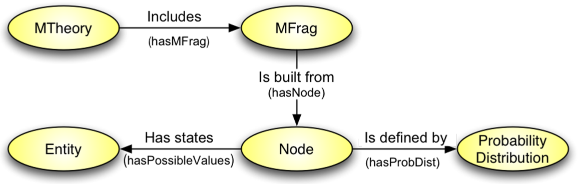


Fig. 1. PR-OWL main concepts.

Probabilistic Ontologies [5, 6] have been proposed as a more expressive formalism for representing knowledge in domains characterized by uncertainty. The PR-OWL probabilistic ontology language [5, 6] has its logical basis in Multi-Entity Bayesian Networks (MEBN), an extension of Bayesian networks (BNs) to achieve first-order expressive power [14, 15]. MEBN represents knowledge as a collection of MEBN Fragments (MFrags), which are organized into MEBN Theories (MTheories). Figure 1 presents the main concepts needed to define an MTheory in PR-OWL. In the diagram, the ellipses represent the general classes, while the arcs represent the main relationships among the classes.

An MFrag contains random variables (RVs) and a fragment graph representing dependencies among these RVs. An MFrag is a template for a fragment of a

Bayesian network. It is instantiated by binding its arguments to domain entity identifiers to create instances of its RVs. There are three kinds of RV: context, resident and input. Context RVs represent conditions that must be satisfied for the distributions represented in the MFrag to apply. Input nodes represent RVs that may influence the distributions defined in the MFrag, but whose distributions are defined in other MFrags. Distributions for resident RV instances are defined in the MFrag. Distributions for resident RVs are defined by specifying local distributions conditioned on the values of the instances of their parents in the fragment graph.

A set of MFrags represents a joint distribution over instances of its random variables. MEBN provides a compact way to represent repeated structure in a BN. An advantage of MEBN is that there is no fixed limit on the number of RV instances, and the random variable instances can be dynamically instantiated as needed.

An MTheory is a set of MFrags that satisfies conditions of consistency ensuring the existence of a unique joint probability distribution over its random variable instances.

To apply an MTheory to reason about particular scenarios, one needs to provide the system with specific information about the individual entity instances involved in the scenario. Throughout the remainder of the paper, we use procurement fraud as an example application domain [2]. On receipt of information about a particular procurement scenario, Bayesian inference can be used both to answer specific questions of interest (e.g., how likely is it that a particular procurement is being directed to a specific enterprise?) and to refine the MTheory (e.g., each new investigation provides additional statistical data about relevant indicators for a given category of fraud). Bayesian inference is used to perform both problem specific inference and learning in a sound, logically coherent manner (for more details see [15, 17]).

3 Mapping PR-OWL Random Variables to OWL Concepts

Suppose we have an OWL ontology for the public procurement domain. The ontology defines concepts such as procurement, winner of a procurement, members of a committee responsible for a procurement, etc. Figure 2 shows a light-weight ontology for this domain represented in Unified Modeling Language (UML) [21].

Now, imagine we want to define some uncertain relations about this domain, e.g. it is common to identify a front for an enterprise by looking at his/her income and the value of a procurement the enterprise he/she represents won, meaning, if the enterprise won a procurement of millions of dollars, but the responsible person for this enterprise makes less than 10 thousand dollars a year, it is likely that this person is a front. Figure 3 shows this probabilistic relation defined using PR-OWL in an open-source tool for probabilistic reasoning, UnBBayes.

As expected, we would need to ensure some conditions were met in order to make assertions about this probabilistic relationship. One of these conditions

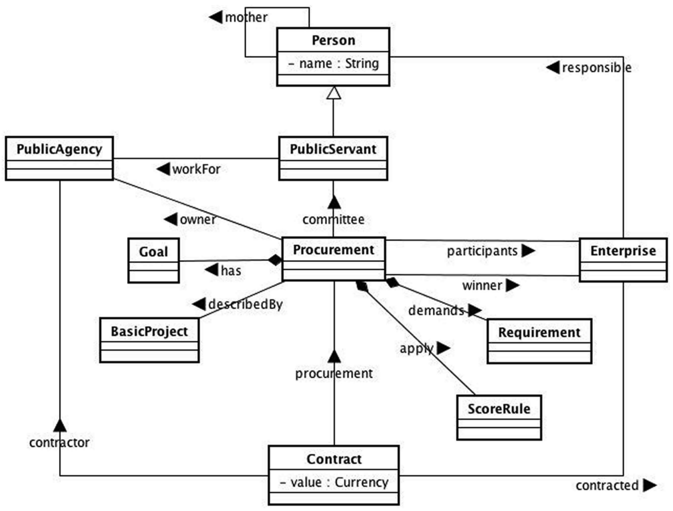


Fig. 2. A class diagram for the procurement domain.

is that the person we are trying to determine as a possible front has to be responsible for the enterprise we are analyzing.

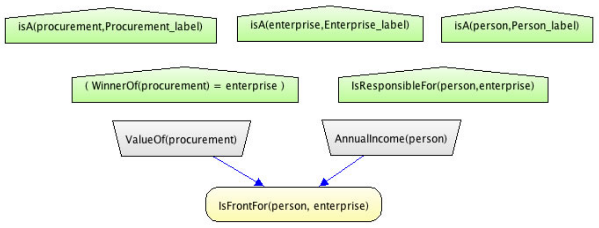

isA(procurement,Procurement_label)
isA(enterprise,Enterprise_label)
isA(person,Person_label)
(WinnerOf(procurement) = enterprise)
IsResponsibleFor(person,enterprise)
ValueOf(procurement)
AnnualIncome(person)
IsFrontFor(person, enterprise)

Fig. 3. Front of an Enterprise MFrag.

It is natural to think that the data we have about this domain would be associated with the ontological markups defined in OWL. In other words, our database would have instances of persons and enterprises, and these instances would be linked to their semantic meaning defined in the OWL ontology.

Accessing this information should be trivial once the definitions in the ontology were made available and permission was granted to retrieve data from the database. However, this can only be achieved by developing a link between PR-OWL random variables (RVs) and the concepts defined in OWL. In its current state, though, the relations defined in PR-OWL are not formally linked to the relevant concepts in the OWL procurement ontology. That is, the relation

IsResponsibleFor should be linked to the OWL concepts representing persons and enterprises.

From this simple example, it is clear that every probabilistic definition involving a concept must keep a reference to its semantic definition. In other words, full compatibility with OWL requires modifications to PR-OWL that guarantee the preservation of OWL's semantics.

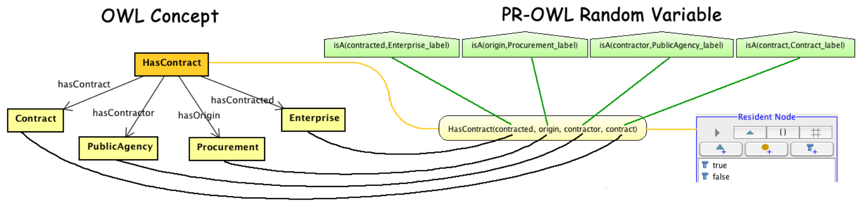

OWL Concept
PR-OWL Random Variable
HasContract
hasContract
hasContractor
hasOrigin
hasContracted
Contract
PublicAgency
Procurement
Enterprise
isA(contracted,Enterprise_label)
isA(origin,Procurement_label)
isA(contractor,PublicAgency_label)
isA(contract,Contract_label)
HasContract(contracted, origin, contractor, contract)
Resident Node
true
false

Fig. 4. Ternary relation mapping between OWL and PR-OWL.

Figure 4 shows a suggested approach to map concepts in OWL to random variables in PR-OWL. In this case, the relation is represented by a class (see [11] for details on how to define n−ary relations in OWL) named *HasContract* which represents a 4-ary relation that relates a contract that has a public agency as a contractor, has its origin in a procurement, and has an enterprise contracted. This relation is mapped as a predicate because in this example it is possible to have more than one contracted enterprise for the same contract.

The idea is to keep a reference to the main relation (OWL concept) when creating its probabilistic definition (PR-OWL random variable). In this case, the random variable *HasContract* is a function that defines the probabilistic characteristics of the concept *HasContract*, in this case a class that has a role of a relation as explained above. As it can be seen on Figure 4 the range of the random variable is a boolean.

However, it is not enough to map a PR-OWL random variable to an OWL concept whose probabilistic characteristics are being defined. It is also necessary to map the arguments of the random variable to their respective classes or data types in OWL. In this example we have that: the argument *contract* is mapped to the class *Contract*, which is the range of the property *hasContract*; the argument *contracted* is mapped to the class *Enterprise*, which is the range of the property *hasContracted*; the argument *origin* is mapped to the class *Procurement*, which is the range of the property *hasOrigin*; and the the argument *contractor* is mapped to the class *PublicAgency*, which is the range of the property *hasContractor*.

Finally, in First Order Logic (FOL) the range of any n-ary predicate is a boolean. However, due to the lack of n-ary relations in OWL, this predicate was modeled in PR-OWL as a class. Therefore, it has no defined range. In fact, the only possible value of a class is an instantiation. So there is one last mapping to be done.

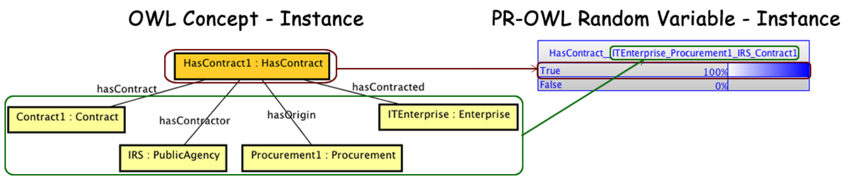

OWL Concept - Instance
PR-OWL Random Variable - Instance
HasContract1 : HasContract
hasContract
hasContractor
hasOrigin
hasContracted
Contract1 : Contract
IRS : PublicAgency
Procurement1 : Procurement
ITEnterprise : Enterprise
HasContract_ITEnterprise_Procurement1_IRS_Contract1
True 100%
False 0%

Fig. 5. Ternary relation instance mapping between OWL and PR-OWL.

We need to map the existence of an instance to a random variable with value true. This is a one−to−one mapping. I.e., there is only one RV that describes the uncertainty of an OWL instance and there is only one OWL instance that describes the semantics of a RV. If there is no such instance, then the value of the RV is either false (if we are assuming a closed world) or unknown (if we are assuming an open world). The fact is that we need to have an extra parameter in our RV to state to which instance of the class *HasContract* it is related to.

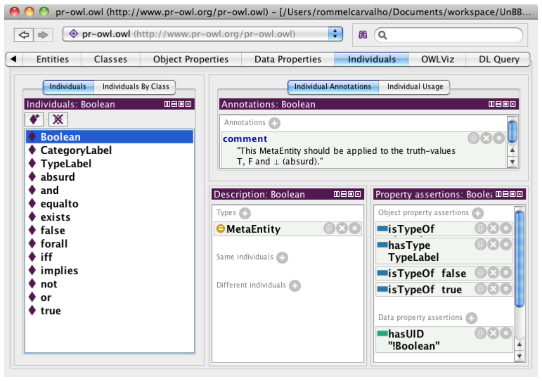
So, once the random variable is instantiated as a node in a Bayesian network for a specific situation, it is necessary to maintain the mapping we had in our PR-OWL random variable. Figure 5 shows a suggestion for how to perform this mapping. In this example, as we have an OWL assertion that *HasContract1* is a predicate that states that the *Contract1* contract had origin in *Procurement1*, has *IRS* as its contractor, and has contracted *ITEnterprise*, we have the PR-OWL counterpart stating that the node *HasContract__ITEnterprise_-Procurement1_IRS_Contract1* has the state *True* with probability 100%. The actual mapping between the OWL instance and the node is kept because the node is in fact an instance of the random variable defined in PR-OWL, which in turn is an OWL class. As the PR-OWL random variable has the mapping, so does its instance.

The mapping described in this section provides the basis for a formal definition of consistency between a PR-OWL probabilistic ontology and an OWL ontology, in which rules in the OWL ontology correspond to probability one assertions in the PR-OWL ontology. A formal notion of consistency can lead to development of consistency checking algorithms.

4 Extending PR-OWL to Use OWL's Types

One of the main concerns when developing OWL [12] was to keep the same semantics of its predecessors, RDF and XML, which meant reusing all the concepts already defined in those languages, including primitive types, such as string, boolean, decimal, etc. On the other hand, PR-OWL does not make use of the primitive types used in OWL. For instance, PR-OWL defines *Boolean* as an individual of the class *MetaEntity*, as shown in Figure 6, but does not keep any relation to the boolean type used in OWL.

If we wanted to define a continuous random variable for the annual income of a person in PR-OWL, we would need to define the real numbers, even though



pr-owl.owl (http://www.pr-owl.org/pr-owl.owl) – [/Users/rommelcarvalho/Documents/workspace/UnBB...

pr-owl.owl (http://www.pr-owl.org/pr-owl.owl)

Individuals: Boolean

- Boolean
- CategoryLabel
- TypeLabel
- absurd
- and
- equalto
- exists
- false
- forall
- iff
- implies
- not
- or
- true

Annotations: Boolean

comment
"This MetaEntity should be applied to the truth-values T, F and ⊥ (absurd)."

Description: Boolean

MetaEntity

Property assertions: Boole

isTypeOf
hasType TypeLabel
isTypeOf false
isTypeOf true

hasUID "!Boolean"

Fig. 6. Boolean individual defined in PR-OWL.

they are already defined in OWL. Moreover, concepts that use this primitive type in OWL would not be understood in PR-OWL, in other words, they lack compatibility as far as primitive types are concerned.

Figure 7 shows the different types of entities defined in PR-OWL. A possible approach to keep OWL's semantics is to avoid defining new types of entities and use what is already available in OWL. For instance, the class *ObjectEntity* can be substituted by the OWL class *Thing*, after all, according to [5] *ObjectEntity* aggregates the MEBN entities that are real world concepts of interest in a domain. They are akin to objects in Object-Oriented (OO) models and to frames in frame-based knowledge systems. In other words, they are nothing more than OWL classes.

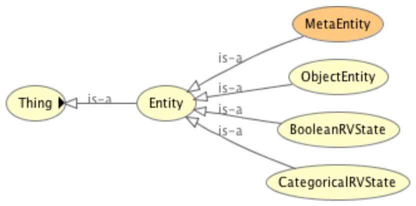


Fig. 7. The different types of entities defined in PR-OWL.

According to [5] *CategoricalRVState* is used to represent a list of mutually exclusive, collectively exhaustive states, which in turn are possible states of random variables, represented by nodes in PR-OWL. Therefore, it can be replaced by *DataOneOf* if it needs to enumerate data types or *ObjectOneOf* if it needs to enumerate objects. These concepts allow the enumeration of literals and individuals, respectively (see [19] for more details).

BooleanRVState can be replaced by the boolean data type present in OWL. Finally, the *MetaEntity* class, which includes all the entities that convey specific definitions about entities (e.g. typelabels that name the possible types of entities), can be eliminated since all other entities were replaced by a concept already present in OWL.

5 Conclusion

We described the main issues with PR-OWL probabilistic ontology language with respect to its compatibility with the OWL ontology language and presented possible approaches to deal with these issues.

The first issue described was the lack of mapping between PR-OWL random variables (RVs) and the concepts defined in OWL. In its current state, though, the relations defined in PR-OWL are not formally linked to concepts in OWL. We have shown through an example how this mapping can be done.

The second issue described was that PR-OWL does not make use of the primitive types used in OWL, as OWL did with respect to RDF and XML. For this reason, concepts already defined in one language must be redefined in the other. We have shown that every type defined in PR-OWL can be directly mapped to concepts already present in OWL without any loss of generality.

This paper has provided qualitative descriptions and examples of how to deal with these compatibility issues. We are currently working on formalizing these qualitative descriptions and on modifying PR-OWL's syntax and semantics to incorporate the approaches presented here.

Acknowledgments. The authors would like to thank the Brazilian Office of the Comptroller General (CGU) for their active support since 2008 and for providing the human resources necessary to conduct this research.

References

1. Dean Allemang and James A. Hendler. *Semantic web for the working ontologist.* Morgan Kaufmann, 2008.
2. Rommel Novaes Carvalho, Kathryn B. Laskey, Paulo C. G. Costa, Marcelo Ladeira, Laecio Lima Santos, and Shou Matsumoto. Probabilistic ontology and knowledge fusion for procurement fraud detection in brazil. In *Proceedings of the 5th Uncertainty Reasoning for the Semantic Web (URSW 2009) on the 8th International Semantic Web Conference (ISWC 2009)*, Chantilly, Virginia, USA, October 2009.
3. Huajun Chen and Zhaohui Wu. On Case-Based knowledge sharing in semantic web. In *Tools with Artificial Intelligence, IEEE International Conference on*, volume 0, page 200, Los Alamitos, CA, USA, 2003. IEEE Computer Society.
4. Huajun Chen, Zhaohui Wu, and Jiefeng Xu. KB-Grid: enabling knowledge sharing on the semantic web. In *Challenges of Large Applications in Distributed Environments, International Workshop on*, volume 0, page 70, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

5. Paulo C. G Costa. *Bayesian Semantics for the Semantic Web.* PhD, George Mason University, July 2005. Brazilian Air Force.
6. Paulo Cesar Costa, Kathryn B. Laskey, and Kenneth J. Laskey. PR-OWL: a bayesian ontology language for the semantic web. In *Uncertainty Reasoning for the Semantic Web I: ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers*, pages 88–107. Springer-Verlag, 2008.
7. P.C.G. Costa, Kuo-Chu Chang, K.B. Laskey, and Rommel Novaes Carvalho. A multi-disciplinary approach to high level fusion in predictive situational awareness. In *Proceedings of the 12th International Conference on Information Fusion*, pages 248–255, Seattle, Washington, USA, July 2009.
8. A.-S. Dadzie, R. Bhagdev, A. Chakravarthy, S. Chapman, J. Iria, V. Lanfranchi, J. Magalhes, D. Petrelli, and F. Ciravegna. Applying semantic web technologies to knowledge sharing in aerospace engineering. *Journal of Intelligent Manufacturing*, 20(5):611–623, 2008.
9. W3C OWL Working Group. OWL 2 web ontology language document overview. http://www.w3.org/TR/2009/PR-owl2-overview-20090922/, September 2009.
10. Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.
11. Patrick Hayes and Alan Rector. Defining n-ary relations on the semantic web. http://www.w3.org/TR/swbp-n-aryRelations/, 2006.
12. Ian Horrocks, Peter F Patel-Schneider, and Frank Van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *JOURNAL OF WEB SEMANTICS*, 1:2003, 2003.
13. Nicholas J. Kings and John Davies. Semantic web for knowledge sharing. In *Semantic Knowledge Management*, pages 103–111. 2009.
14. Kathryn B. Laskey and Paulo C. G. Costa. Of starships and klingons: Bayesian logic for the 23rd century. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, Arlington, Virginia, USA, 2005. AUAI Press.
15. Kathryn Blackmond Laskey. MEBN: a language for first-order bayesian knowledge bases. *Artif. Intell.*, 172(2-3):140–178, 2008.
16. K.B. Laskey, P. Costa, and T. Janssen. Probabilistic ontologies for knowledge fusion. In *Information Fusion, 2008 11th International Conference on*, pages 1–8, 2008.
17. Suzanne Mahoney and Kathryn B. Laskey. Constructing situation specific belief networks. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, San Francisco, CA, 1998. Morgan Kaufmann.
18. Deborah L. McGuinness and Frank Van Harmelen. OWL web ontology language overview. http://www.w3.org/TR/owl-features/, February 2004.
19. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language structural specification and Functional-Style syntax. http://www.w3.org/TR/owl2-syntax/, October 2009.
20. Livia Predoiu and Heiner Stuckenschmidt. Probabilistic extensions of semantic web languages - a survey. In *The Semantic Web for Knowledge and Data Management: Technologies and Practices.* Idea Group Inc, 2008.
21. James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual.* Addison-Wesley Professional, 1999.
22. Galina V Veres, Trung Dong Huynh, Mark S Nixon, Paul R Smart, and Nigel R Shadbolt. The military knowledge information fusion via semantic web technologies. Technical report, 2006.

Pronto: A Practical Probabilistic Description Logic Reasoner

Pavel Klinov and Bijan Parsia

School of Computer Science
University of Manchester, United Kingdom
{pklinov,bparsia}@cs.man.ac.uk

Abstract. This paper presents a system description of Pronto — the first probabilistic Description Logic reasoner capable of processing knowledge bases containing about a thousand of probabilistic axioms. We describe the design and architecture of the reasoner with an emphasis on the components that implement algorithms which are crucial for achieving such level of scalability. Finally, we present the results of the experimental evaluation of Pronto's performance on series of propositional and non-propositional probabilistic knowledge bases.

1 Introduction

There are many proposed formalisms for combining Description Logics (DLs) with various sorts of uncertainty, although, to our knowledge, none have been used for a production ontology. We believe that this is due to two reasons: 1) there is comparatively little knowledge about how to use these formalisms effectively (or even, which are best suited for what purposes) and 2) there is a severe lack of tooling, in particular, there have been no sufficiently effective reasoners.

This paper describes our work on the second problem. We present Pronto — the reasoner for the probabilistic extension of DL $\mathcal{SHIQ}$ (named P-$\mathcal{SHIQ}$) [1]. This logic can be viewed either as a generalization of the Nilsson's propositional probabilistic logic [2] or as a fragment of first-order probabilistic logic of Halpern and Bacchus [3] [4] (with certain non-monotonic extensions). One attractive feature of these probabilistic logics is that they allow modelers to declaratively describe their uncertain knowledge without fully specifying any probability distribution (in contrast to, for example, Bayesian networks). They are also proper generalizations of their classical counterparts which, in the case of P-$\mathcal{SHIQ}$, means that modelers can take an existing $\mathcal{SHIQ}$ ontology and add probabilistic axioms to capture uncertain, such as statistical, relationships.

In spite of their attractive features Nilsson-style logics have been criticized, partly for the intractability of probabilistic inference. Reasoning procedures are typically implemented via reduction to linear programming but it is well known that corresponding linear programs are exponentially large so the scalability is very limited. Over the last two decades there have been several attempts to overcome that issue in the propositional case which led to some promising results, such as solving the probabilistic satisfiability problem (PSAT) for 800-1000 formulas [5]. It has been unclear whether the methods used to solve large propositional PSATs can be directly applied to PSATs in probabilistic DLs.

To the best of our knowledge, Pronto is the first reasoner for a Nilsson-style probabilistic DL which scalability is comparable (and often better) than scalability of propositional solvers. In particular, it can solve propositional PSATs of the same size as them but can also effectively deal with KBs with non-propositional classical knowledge, such as large $\mathcal{SHIQ}$ terminologies. We present experimental results which show that the level of scalability is comparable in the propositional and non-propositional cases. In addition, Pronto implements all the standard reasoning services for P-$\mathcal{SHIQ}$ as well as useful extra services, in particular, finding *all* minimal unsatisfiable fragments of a KB which is crucial for analyzing large bodies of conflicting probabilistic knowledge.

2 Preliminaries

P-$\mathcal{SHIQ}$ [1] is a probabilistic generalization of the DL $\mathcal{SHIQ}$ [6]. It supports probabilistic subsumptions between arbitrary $\mathcal{SHIQ}$ concepts and a certain class of probabilistic concept assertions (but no form of probabilistic role assertions). Any $\mathcal{SHIQ}$ ontology can be used as a basis for a P-$\mathcal{SHIQ}$ ontology which facilitates transition from classical to probabilistic ontological models. Finally, it combines probabilistic and default reasoning. This allows for a consistent treatment of exceptional individuals and subconcepts.

P-$\mathcal{SHIQ}$ is extends the syntax of $\mathcal{SHIQ}$ with *conditional constraints*, that is, expressions of the form $(D|C)[l, u]$ where C and D are arbitrary $\mathcal{SHIQ}$ concept expressions. Conditional constraints can be used for representing uncertainty in both terminological (TBox) and assertional (ABox) knowledge. A probabilistic TBox (PTBox) is a 2-tuple $PT = (\mathcal{T}, \mathcal{P})$ where $\mathcal{T}$ is a $\mathcal{SHIQ}$ TBox (sometimes called *the classical part*) and $\mathcal{P}$ is a finite set of default conditional constraints (or *probabilistic part*). Informally, a PTBox axiom $(D|C)[l, u]$ means that "generally, if a *randomly* chosen individual belongs to C, its probability of belonging to D is in $[l, u]$". A probabilistic ABox (PABox) is a finite set of strict conditional constraints pertaining to a *concrete* probabilistic individual o. A knowledge base (ontology) in P-$\mathcal{SHIQ}$ is a combination of a PTBox and a collection of PABoxes (one for each probabilistic individual).

P-$\mathcal{SHIQ}$ semantics is based on probability distributions over *possible worlds*, where each possible world is a subset of probabilistically relevant concepts Φ (i.e. concepts used to define conditional constraints). Informally, each world can be thought of as a concept type for a randomly chosen individual. A world is *possible* if there exists an individual that is an instance of all concepts in the world (i.e. the concept type is *realizable*). A KB is satisfiable if there exists a probability distribution that satisfies all conditional constraints.

Standard reasoning tasks in P-$\mathcal{SHIQ}$ include PSAT, tight logical entailment (TLogEnt), and tight lexicographic entailment (TLexEnt). The first two tasks are probabilistic counterparts of classical satisfiability and entailment problems in DL. In contrast, TLexEnt is a *non-monotonic* reasoning task which is reducible to the logical entailment from the largest, conflict-free fragments of the KB.

See [1] for a formal presentation of P-$\mathcal{SHIQ}$ semantics, reasoning procedures and complexity results.

3 Probabilistic Satisfiability Algorithm

In this section we briefly sketch the novel PSAT algorithm implemented in Pronto (see Section 6 for its differences from the previously developed methods). For clarity we will consider a special case of PSAT where the PTBox is of the form $PT = (\mathcal{T}, \{(C_i|\top)[p_i, p_i]\})$ (i.e. all probabilistic statements are unconditional constraints with point-valued probabilities). It is straightforward, but technically awkward, to generalize the procedure to handle conditional interval statements.

A PTBox $PT = (\mathcal{T}, \{(C_i|\top)[p_i, p_i]\})$ is satisfiable iff the following linear program admits a solution:

$$\begin{aligned} &max \sum_{I \in I_\Phi} x_I \\ &s.t. \sum_{C_i \in I} x_I = p_i, \text{ for each } (C_i|\top)[p_i, p_i] \in \mathcal{P} \\ &\quad\;\; \sum_{I \in I_\Phi} x_I = 1 \text{ and all } x_I \geq 0 \end{aligned} \tag{1}$$

where I_Φ is the set of all possible worlds for the set of concepts Φ in $\mathcal{T}$.

Let A denote the matrix of linear coefficients in (1). At every step of the simplex algorithm, A is represented as a combination (B, N) where B and N are the submatrices of the *basic* and *non-basic* variables, respectively. Values of non-basic variables are fixed to zero, and the solver proceeds by replacing one basic variable by a non-basic one until the optimal solution is found. The index of the non-basic column is determined according to the following expression [5]:

$$j \in \{1, \ldots, |N|\} \text{ s.t. } c_j - u^T A^j \text{ is minimal} \tag{2}$$

where c_j is the objective coefficient for the new variable (it is always equal to 1 in (1)) and u^T is the current dual solution of (1).

As the size of N is exponential in $|\Phi|$, one should compute (2) without examining all columns in N. This is done using the column generation technique in which (2) is treated as an optimization problem with the following objective function:

$$min \; (1 - \sum_{i=1}^{m+1} u_i a_i^j), \; A^j = (a_i^j) \in \{0, 1\}^{m+1} \tag{3}$$

Since columns in (1) correspond to possible worlds, $a_i^j = 1$ means that $C_i \in I_j$ while $a_i^j = 0$ means that $\neg C_i \in I_j$, where I_j is the possible world corresponding to the column A^j. Thus it is possible to represent I_j as a conjunctive $\mathcal{SHIQ}$ concept expression as follows (we call the correspondence function η):

$$I_j = \eta(A^j) = \bigsqcap X_i, \text{ where } X_i = \begin{cases} C_i, & a_i^j = 1 \\ \neg C_i, & a_i^j = 0 \end{cases} \tag{4}$$

The critical step is to formulate linear constraints for (3) such that every solution corresponds to a concept expression that is satisfiable w.r.t. $\mathcal{T}$, i.e. a possible world. In the propositional case, where each C_i is a clause, this can be done by employing a well known formulation of SAT as a mixed-integer linear program [7]. For example, if $C_i = x_{i1} \vee \neg x_{i2} \vee x_{i3}$ then (3) will have the constraint $a_i = x_{i1} + (1 - x_{i2}) + x_{i3}$ where all variables are binary.

In the case of an expressive language, such as $\mathcal{SHIQ}$, there appears to be no easy way of determining a set of constraints H for (3) such that its set of solutions in one-to-one correspondence with $\mathcal{I}_\Phi$ (in particular, it is important to ensure that, if A^j is a solution then $\mathcal{T} \nvDash \eta(A^j) \sqsubseteq \bot$, i.e. H faithfully captures the TBox $\mathcal{T}$). Instead, Pronto implements a novel *hybrid*, *iterative* procedure to compute H which can be summarized as follows:

```
   Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$, current dual solution $u^T$ of (1)
   Output: New column $A^j$ or null
 1 Initialize (3) using $u^T$, $H \leftarrow \emptyset$
 2 while $A^j \neq$ null do
 3     $A^j \leftarrow$ current optimal solution of (3)
 4     if $A^j \neq$ null then
 5         if satisfiable($\eta(A^j), \mathcal{T}$) then
 6             return $A^j$
 7         else
 8             add constraints to $H$ that prohibit $A^j$
 9         end
10     end
11 end
12 return null;
```

Algorithm 1: Hybrid iterative column generation algorithm

The key steps are 5 and 8. On step 5 the algorithm invokes a $\mathcal{SHIQ}$ SAT solver (in our case, Pellet) to determine if the computed column corresponds to a *possible* world. If yes, the column is returned. If no, the current set of constraints H is augmented on step 8 to exclude A^j from the set of solutions to (3). The algorithm is called "hybrid" because it combines invocations of simplex and $\mathcal{SHIQ}$ solvers and "iterative" because it iteratively tightens the set of solutions to (3) until either a valid column is found or provably no such column exists.

The actual implementation is considerably more involved, in particular, because it is important to minimize the number of calls to the $\mathcal{SHIQ}$ solver.

Therefore the algorithm tries to learn the set H as quickly as possible. Such details, as well as the description of step 8, are beyond the scope of this paper.

4 Architecture

Pronto has layered architecture presented on Figure 1. Each layer has one or more components which invoke other components at the same or lower levels (but not the other way around).

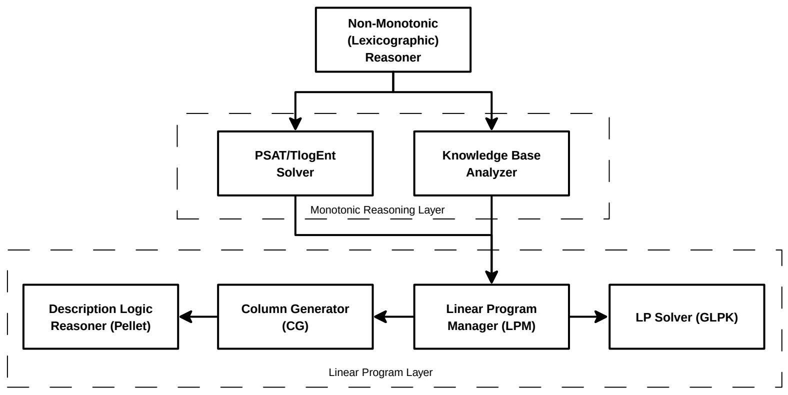

Non-Monotonic (Lexicographic) Reasoner
PSAT/TlogEnt Solver
Knowledge Base Analyzer
Monotonic Reasoning Layer
Description Logic Reasoner (Pellet)
Column Generator (CG)
Linear Program Manager (LPM)
LP Solver (GLPK)
Linear Program Layer

Fig. 1: Architecture of Pronto

Linear Program Layer The components at the lowermost level are responsible for managing linear programs which are optimized in order to solve PSAT or TLogEnt problems. As mentioned earlier, these linear programs usually have exponentially many variables so it is futile to try to represent them explicitly. Therefore, the main function of the components, namely, the linear program manager (LPM) and the column generator (CG), is to generate partial linear systems (1) which have the same optimal objective values as their complete versions.

The LPM is responsible for producing the initial version of the linear program (1), incorporating each new column into it, and checking the optimality (i.e. stopping) criteria. It interacts with the simplex solver, such as GLPK, which solves the current program (1) and returns its primal and dual solutions. The latter is supplied to the CG component.

The CG component implements Algorithm 1. It maintains the binary linear program 3, accepts the dual values u^T, and interacts with Pellet in order to produce improving columns which are then returned back to the LPM. This component implements a number of optimizations such as tuning the binary program (3), learning and re-using constraints H that reflect the structure of the TBox $\mathcal{T}$, and others.

Monotonic Probabilistic Reasoning Layer The components on the next layer use the underlying linear programs to perform monotonic reasoning, i.e. solve PSAT and TLogEnt, and analyze unsatisfiable probabilistic KBs. The first two tasks are straightforward. They amount to checking if a linear system generated by the components of the lower layer admits a solution (PSAT) or solving it to optimality (TLogEnt).

The analysis of an unsatisfiable probabilistic KB is a problem of finding all minimal unsatisfiable subsets of the KB where minimality is defined with respect to the set inclusion. This is essential for 1) computing all maximal satisfiable fragments of the KB during non-monotonic (lexicographic) reasoning, and 2) computing explanations for the results of probabilistic reasoning.

On the linear system level this analysis is equivalent to discovering all irreducible infeasible subsystems (IIS) which can be exponentially many [8]. The task of finding all IISes is somewhat complicated by the fact that the linear systems are never complete, so their set of IISes may not be the same as for the complete system (despite that their objective values are optimal). Therefore the analyzer has to repeatedly invoke the LP layer components to enrich the system with new columns after finding each new IIS. Apart of that the analysis follows the classical model-based diagnosis methodology based on hitting set trees [9].

Non-Monotonic Probabilistic Reasoning Layer The uppermost layer consists of a single component: the lexicographic reasoner. It implements the TLexEnt algorithm which relies on the KB analyzer and the TLogEnt reasoner. TLexEnt is equivalent to solving TLogEnt for all lexicographically minimal subsets of the KB [1]. The latter are computed in three phases: 1) KB analyzer computes the structure called *conflict graph* which represents conflicts between pieces of probabilistic knowledge, 2) the graph is used to rank conflicting statements by specificity, and 3) conflicts are resolved by preferring more specific statements to less specific (the resulting conflict-free fragments of the KB are lexicographically minimal). The last phase may fail if equally specific statements happen to be in conflict. In that case the reasoner reports probabilistic inconsistency (which is different from probabilistic unsatisfiability, see [1] for details).

5 Experimental Evaluation

We have conducted two experiments to demonstrate that Pronto is practical to use on ontologies of realistic size. Both experiments evaluate the performance of solving PSAT (since all other reasoning tasks are reducible to it). KBs in the first experiments are randomly generated sets of propositional conditional constraints with no classical part. KBs in the second experiment are randomly generated conditional constraints with TBoxes from real-life ontologies represented in expressive DLs: the GeoSkills ontology and the SWEET Process ontology, both taken from the TONES repository[1]. There does not seem to be an easy way to "propositionalize" these ontologies in order to use propositional PSAT solvers.

The results, which are presented in Table 1, were averaged over 10 PSAT instances solved for each size. In the table n stands for the number of concepts in the classical part of the KB and m — for the number of conditional constraints. We used a conventional PC with 2GHz CPU and 2GB RAM.

Table 1: Performance on random propositional and non-propositional probabilistic KBs

Propositional KBs			GeoSkills ($\mathcal{ALCHOIN}$)			Process ($\mathcal{ALCHOF}$)		
n	m	Time (s)	n	m	Time (s)	n	m	Time (s)
50	100	3	603	100	20	1537	100	5
100	200	12	603	200	38	1537	200	30
250	500	30	603	500	151	1537	500	92
500	1000	291	603	1000	332	1537	1000	176

The results show that Pronto performs comparably to the state-of-the-art propositional PSAT solvers on propositional KBs [5][2]. However, it can also handle probabilistic KBs of the same size defined over highly expressive DL ontologies without significant loss of performance. In fact, expressive TBoxes may improve the performance because they tend to shrink the space of all potential columns (possible worlds) by constraining models. This can explain why the probabilistic extension of the Process ontology is sometimes easier than random propositional KBs of the same size. We are now in the process of investigating this and other complexity factors through a more comprehensive and systematic evaluation. It is anticipated that understanding of such factors will help to develop corresponding optimizations.

6 Related Work

To the best of our knowledge there are no other reasoners for Nilsson-style probabilistic DLs. One exception is [10] but that system does not implement any technique to reduce the size of the linear programs and is, therefore, limited to 10-15 probabilistic statements.

Among other tools the most closely related are propositional PSAT solvers which also use the column generation method to cope with the size of the linear systems [5] [11]. However, the main difference between Pronto and those tools lies in the optimization problem (3) used to produce columns. They can encode the

[1] http://owl.cs.manchester.ac.uk/repository/

[2] We must mention that our problem generation methodology is slightly different. Hansen and Perron experimented with unconditional constraints and point-valued probabilities while we generated mixtures of conditional and unconditional statements with intervals which, as we believe, are more useful for practical modeling.

entire structure of the *propositional* KB in the set of constraints H for (3) while Pronto employs a $\mathcal{SHIQ}$ reasoner to compute such set iteratively. One important consequence is that Pronto can be used for a Nilsson-style probabilistic extension to *any* logic for which a SAT solver is available.

7 Summary

The primary conclusion of this work is that Pronto is *practical* to use for probabilistic ontologies of a realistic size. PSAT and the various entailment problems for P-$\mathcal{SHIQ}$ are EXPTIME-complete, so, as with any logic in the $\mathcal{SH}$ family, practicality, efficiency, and scalability claims must be carefully qualified. However, Pronto handles P-$\mathcal{SHIQ}$ ontologies which are comparable in size to, for example, various handcrafted Bayesian networks[3], which gives good reasons to believe that it will prove practical for future probabilistic ontologies.

More generally, our work also shows that P-$\mathcal{SHIQ}$, as well as other Nilsson-style extensions to DLs, can be *practical* (in terms of reasoning complexity) probabilistic ontology languages. In a certain sense they are no less practical than $\mathcal{SHIQ}$. Although there will certainly be P-$\mathcal{SHIQ}$ ontologies that will defeat the current version of Pronto, this is also the case with any NP-hard logic. However, now it makes sense for modelers to experiment with P-$\mathcal{SHIQ}$ and report troublesome ontologies, so that tool developers can adapt to new challenges.

References

1. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence **172(6-7)** (2008) 852–883
2. Nilsson, N.J.: Probabilistic logic. Artificial Intelligence **28**(1) (1986) 71–87
3. Halpern, J.Y.: An analysis of first-order logics of probability. Artificial Intelligence **46** (1990) 311–350
4. Bacchus, F.: Representing and reasoning with probabilistic knowledge. MIT Press (1990)
5. Hansen, P., Perron, S.: Merging the local and global approaches to probabilistic satisfiability. Int. Journal of Approximate Reasoning **47**(2) (2008) 125–140
6. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Journal of the IGPL **8**(3) (2000)
7. Hooker, J.N.: Quantitative approach to logical reasoning. Decision Support Systems **4** (1988) 45–69
8. Gleeson, J., Ryan, J.: Identifying minimally infeasible subsystems of inequalities. INFORMS Journal on Computing **2**(1) (1990) 61–63
9. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **32** (1987) 57–95
10. Näth, T.H., Möller, R.: ContraBovemRufum: A system for probabilistic lexicographic entailment. In: Description Logics. (2008)
11. de Souza Andrade, P.S., da Rocha, J.C.F., Couto, D.P., da Costa Teves, A., Cozman, F.G.: A toolset for propositional probabilistic logic. In: Encontro Nacional de Inteligencia Artificial. (2007) 1371–1380

[3] See, in particular, the repository at http://genie.sis.pitt.edu/networks.html

Using f-$\mathcal{SHIN}$ to represent objects: an aid to visual grasping

Nicola Vitucci, Mario Arrigoni Neri, and Giuseppina Gini

Politecnico di Milano - Dipartimento di Elettronica e Informazione
Via Ponzio 34/5, 20133 Milano, Italy
{vitucci,arrigoni,gini}@elet.polimi.it

Abstract. Description Logics (DLs) are nowadays used to face a variety of problems. When dealing with numerical data coming from the real world, however, the use of traditional logics results in a loss of useful information that can be otherwise exploited using more expressive logics. Fuzzy extensions of traditional DLs, being able to represent vague concepts, are well suited to reason on such objects.
In this paper we present an architecture for the automatic building and querying of a fuzzy ontology related to the representation of objects in terms of their composing parts. Our approach mainly aims to face the problem of visual grasping, which is of wide interest in the robotics field.

1 Introduction

The decomposition of an object in parts has been recognized as an important problem in artificial intelligence: it is considered both as a human-like way of reasoning on objects [1] and as a good way to reduce complexity in tasks like object recognition [14]. Apart of the actual image decomposition phase, a major issue is constituted by the semantic description of the extracted features and their mutual relationships. Due to the vagueness affecting real world data, some tolerance should be taken into account when formally representing the structure of an object; this is a reason to take advantage of novel tools as fuzzy DLs [11].

Fuzzy DLs extend crisp DLs by adding imprecision and vagueness in the reasoning process, thus giving some degrees of truth in place of binary answers as *yes* or *no*. Although the available fuzzy reasoners are not yet as powerful as their crisp counterparts, some interesting applications can be found. One of them lies in the robotics field, in which a symbolic representation of objects can improve the grasping capabilities of a robot by the use of some semantic information, regarding both the type of grasp itself and the structure of the object to be grasped.

To the best of our knowledge, the problem of semantic part decomposition is still an open problem and there are no tools available to automatically create a fuzzy ontology from raw concepts. The use of ontologies for object recognition has been investigated in some works as [4,5,6], but none of them makes explicitly use of fuzzy reasoning except for the creation of (crisp) descriptors as Very_high to be used in the classical way; furthermore, they rely on a previous phase of semantic annotation by domain experts, while we focus on the automatic generation of simple concepts, which are sufficient for our purposes.

There are some recent works in which fuzzy DLs are thoroughly used to reason on multimedia information (see [7,8,10,12]) but little advantage is taken from the expressiveness given by cardinality restrictions (when available). Generally speaking, this is due to the fact that, for scene understanding purposes, it is sufficient to know whether a kind of object is present or not (see [15,16]). On the other hand, for object recognition purposes, it is often necessary to be able to count the instances of each kind of recognized component.

In this paper, we show why we use f-$\mathcal{SHIN}$ [9] as the underlying DL for addressing this problem, then we describe an architecture for the automatic building of a (crisp) ontology and its use for object recognition via fuzzy ABox reasoning services; eventually, in the last section, we make some considerations and propose some future work. The architecture we propose here is still far from being considered complete, yet we were able to obtain some interesting results.

2 The f-$\mathcal{SHIN}$ logic

The f-$\mathcal{SHIN}$ logic is the fuzzy extension of the $\mathcal{SHIN}$ logic [9]. The main improvement of this extension with respect to its crisp version is the possibility to use assertions like $\mathsf{Concept}(p)[\geq 0.7]$, meaning that the individual p has a *minimum degree of participation* of 0.7 to the concept $\mathsf{Concept}$, or $\mathsf{role}(p,q)[\leq 0.3]$, meaning that the individuals p and q participate in the role role with a *maximum degree* of 0.3. The *greatest lower bound* (GLB) [11] is used to know "how much" an individual can be considered to belong to a certain class. A complete description of the f-$\mathcal{SHIN}$ logic can be found in [9].

For the f-$\mathcal{SHIN}$ logic there exists a reasoner called *FiRE*[1], while there exist other reasoners like *fuzzyDL*[2] which is based on the fuzzy extension of the $\mathcal{SHIF}$ logic. The reason why we chose to use FiRE as reasoner is, independently from the supported reasoning services, the high expressivity of the underlying f-$\mathcal{SHIN}$ logic as it supports cardinality restrictions; on the other hand, such a choice needs some functional blocks to be added to carry out operations like the definition of concepts in terms of membership functions.

3 Architecture

As anticipated in the previous section, due to the limitations of the reasoner, the whole architecture is complex and requires some functional elements to be split among different modules (e.g. the reasoner used on the definitions ontology is different from the one used on the objects ontology). The whole architecture of the system is depicted in Fig. 1.

[1] `http://www.image.ece.ntua.gr/~nsimou/FiRE/`

[2] `http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html`

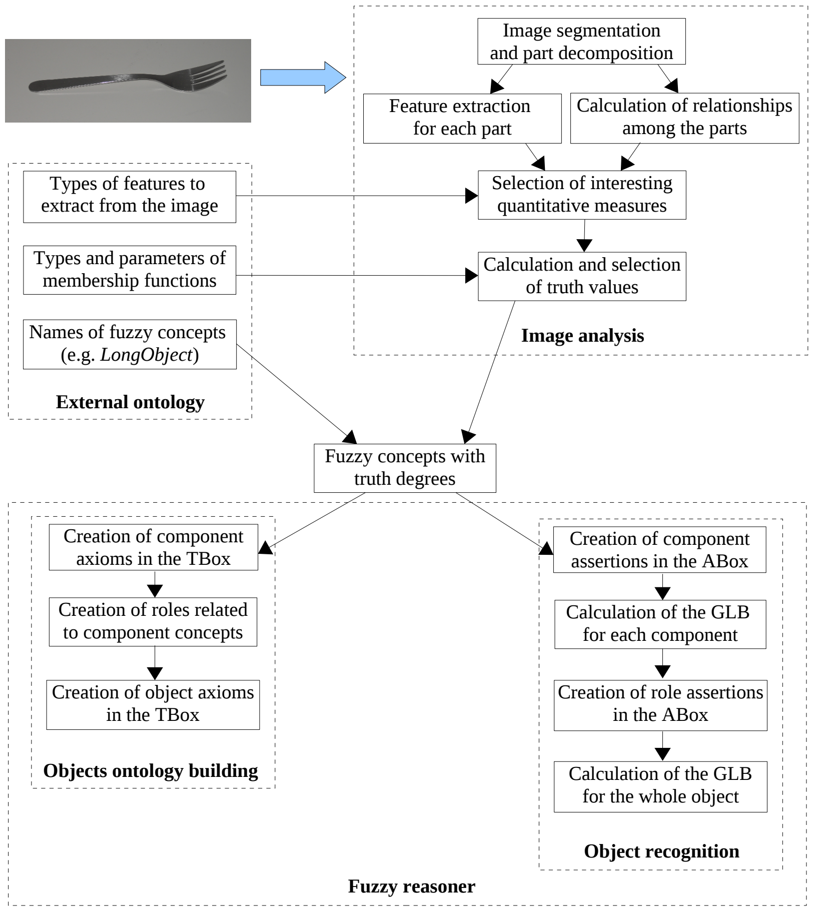


Image segmentation and part decomposition
Feature extraction for each part
Calculation of relationships among the parts
Selection of interesting quantitative measures
Calculation and selection of truth values
Image analysis
Types of features to extract from the image
Types and parameters of membership functions
Names of fuzzy concepts (e.g. LongObject)
External ontology
Fuzzy concepts with truth degrees
Creation of component axioms in the TBox
Creation of roles related to component concepts
Creation of object axioms in the TBox
Objects ontology building
Creation of component assertions in the ABox
Calculation of the GLB for each component
Creation of role assertions in the ABox
Calculation of the GLB for the whole object
Object recognition
Fuzzy reasoner

Fig. 1: The general architecture of the system

The "high level" information, which reflects the kind of knowledge that is to be extracted from the image, is encoded in the external ontology; the image analysis and the numerical calculations are performed with MATLAB$^{\text{TM}}$, while the intermediate steps are performed either in MATLAB$^{\text{TM}}$or in Java$^{\text{TM}}$. The FiRE reasoner is standalone, thus some steps are still to be carried out by hand.

As an example, we will model a fork in terms of its parts; thus, we will use the images shown in Fig. 2.

3.1 External ontology

The external ontology, also called the "definitions ontology", is used to specify the kinds of membership functions to be used as well as the kinds of features to be extracted from the objects found in the images (e.g. elongation, eccentricity, parallelism with respect to other objects and so on) and the meanings of concepts like LongObj and SmallObj in terms of membership functions.

Taking the ontology described in [2] as an example, we built a meta-ontology (based on the crisp logic $\mathcal{SHOIN}(\mathbf{D})$ with datatypes) in which the features to be extracted from the image are subclasses of the meta-class GeometricConcept and the kinds of membership functions to use are subclasses of the meta-class MembFunc. The ontology presented in [2] makes use of some "concrete" concepts like TrapezoidalConcreteFuzzyConcept and TriangularConcreteFuzzyConcept, each one having several properties defined as hasParameterX (where X stands for A, B, K1 etc.) depending on the parameters needed by the considered membership function; an individual tra1 represents a trapezoidal membership function with given parameters.

In our ontology, a concept like "a long object" is modeled as an individual longObject of meta-class Length which has, as its membership function, another individual longMF of a subclass of MembFunc with the function parameters given as datatype properties. By means of the Jena Ontology API[3] and the Pellet reasoner[4], information like the kind and the parameters of a membership function representing a concept related to the image is extracted to feed the image analysis module; thus, a SPARQL query like:

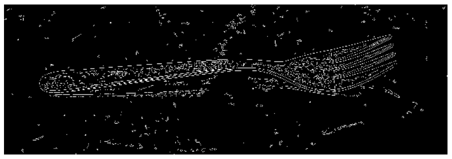
```
SELECT *
WHERE {
   ?x rdfs:subClassOf :GeometryConcept .
   ?y rdf:type ?x .
   ?y :hasMembershipFunction ?z .
   ?z rdf:type ?w .
   ?w rdf:subClassOf :MembershipFunction .
   FILTER (?w != :MembershipFunction) .
   ?z :hasParameter1 ?k1 .
   ?z :hasParameter2 ?k2 .
   OPTIONAL {?z :hasParameter3 ?k3} .
   OPTIONAL {?z :hasParameter4 ?k4}
}
```

is used to extract the individuals representing the actual fuzzy geometry concepts (e.g. LongObject) used in the objects ontology and their related membership functions data (e.g. a sigmoidal function with two parameters).

The ontology is built by a domain expert to reflect the physical characteristics of the robot, so that for example an object can be considered "long" with respect to the maximum aperture of the robot hand. Although a system of measurement has to be established, we now use only pixel measures.

[3] http://jena.sourceforge.net/
[4] http://clarkparsia.com/pellet/

(a) Original image


(b) Image after segmentation

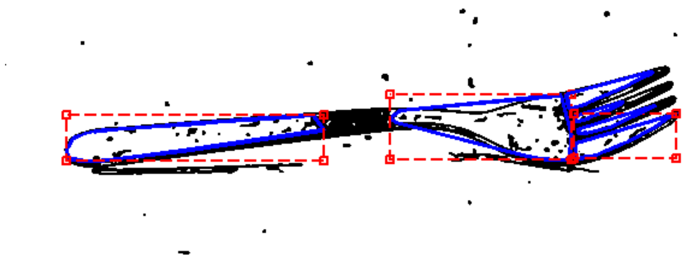

(c) Image after edge dilation and part decomposition (with three parts out of six put in evidence)

Fig. 2: Steps of the image analysis phase

3.2 Image analysis

In this phase, the original image is converted in a binary image after thresholding and edge recognition performed by Canny method [17] (Fig. 2b); the resulting edges are dilated, then the parts having an area over a threshold are selected (Fig. 2c). This segmentation and decomposition phase is actually non-robust, so that the use of fuzzy relationships can be better shown.

After the first phase, some features like the area, the length of the major axis of the ellipse having the same normalized second central moments as the selected region, and so on, are extracted from each found part (see Tab. 1 for some examples of extracted values); then, some quantitative characteristics are computed: for example, the measure of parallelness π, given α and β as the angles between the major axes of the two objects and the x axis of the image, is defined as $\pi = |\cos(\alpha - \beta)|$, while the distance between two parts, instead, is defined as the minimum distance between their convex hulls.

Using the definitions from the external ontology, for every part we calculate the degree of membership of each feature to its related membership functions. For example, for the feature "length" (i.e. the length of its major axis), the truth values for the functions "LongObj", "MediumLengthObj" and "ShortObj" are calculated; if a MediumLengthObj is associated to a generalized bell curve membership function with parameters $a = 240$, $b = 2.5$, $c = 600$ and the length of the major axis of the considered object is 456.61 pixels, the object will belong to the class MediumLengthObj with a truth degree $\mu = 0.93$.

Table 1: Examples of features extracted from the image

(a) Measures of parallelness between every pair of parts

	p1	p2	p3	p4	p5	p6
p1	1.00	0.97	0.97	0.97	0.97	0.97
p2	0.97	1.00	0.89	0.90	0.88	0.89
p3	0.97	0.89	1.00	0.99	0.99	1.00
p4	0.97	0.90	0.99	1.00	0.99	0.99
p5	0.97	0.88	0.99	0.99	1.00	0.99
p6	0.97	0.89	1.00	0.99	0.99	1.00

(b) Other features (area and lengths are in pixels)

	Area	Eccentricity	Major axis	Minor axis	Axes ratio
p1	14860	0.99	456.61	47.33	0.10
p2	12351	0.95	288.23	88.41	0.30
p3	2194	0.98	151.78	23.11	0.15
p4	500	0.99	93.09	8.81	0.09
p5	2617	0.98	181.07	25.79	0.14
p6	771	0.99	141.47	9.39	0.06

3.3 Objects ontology building

Using the results from the previous phase, and taking as a working hypothesis that all the found parts belong to the same object (i.e. there is just one object in the scene), for each part only the membership functions which give the highest truth value for each feature are selected; for example, if a part has a truth degree over a threshold for the membership function "MediumLengthObj", the concept MediumLengthObj is added to the concept representing that part in the fuzzy ontology. At the end, we obtain a concept like (for the sake of simplicity we list only some concepts and roles):

$$\mathsf{ObjClass1} \equiv \mathsf{MediumLengthObj} \sqcap \mathsf{SmallObj} \sqcap \geq 5\ \mathsf{parall} \sqcap \geq 1\ \mathsf{near} \sqcap \ldots$$

where ObjClass1 is the newly created concept related to the part which has been considered. A new fuzzy concept is created only if the current analyzed part does not belong to any existing concept, i.e. there is no concept that fully describes the part (it can be verified via the fuzzy reasoner). Since FiRE does not let us write fuzzy TBox axioms, the degrees of truth are discarded in this phase.

When there are no parts left, a role for each concept is created. For example, from the class ObjClass1 the role hasObjClass1 is created, so that the class Fork can be created using the previously found number of objects per class:

$$\mathsf{Fork} \equiv \geq 1\ \mathsf{hasObjClass1} \sqcap \geq 4\ \mathsf{hasObjClass2} \sqcap \geq 1\ \mathsf{hasObjClass3}$$

This is due to the fact that the f-$\mathcal{SHIN}$ logic lacks of the qualified cardinality restrictions, so a general hasPart role cannot be used. We use a "typographical" operation, yet the problem of role creation has been faced in [3]. For the sake of completeness, domain and range role axioms should be added to qualify the new roles introduced, but the used reasoner does not fully support them yet.

3.4 Object recognition

Once the objects ontology TBox has been built, it is possible to find whether an object, after it has been decomposed in parts, belongs to a class or not (i.e.

how much it can be considered to belong to the considered class with respect to a certain threshold); the image analysis steps are the same for the ontology building phase.

When for every part all the pertaining concepts and roles can be written in the ABox, the fuzzy reasoning is performed to find the GLB of that part belonging to a certain class; then, roles like $\mathsf{hasObjClass1}$ are created with the same value of the found GLBs and, at the end, the GLB of the main object is calculated.

This procedure can be applied to determine whether a specific kind of grasp can be performed or not on the selected object. For example, given the concept defined as (for the sake of simplicity using no roles):

$$\mathsf{GraspableByPinch} \equiv \mathsf{MediumLengthObj} \sqcap \mathsf{HighlyEccentricalObj}$$

representing objects that are graspable by a pinch grip, we can find which part of the object (if any) can be grasped this way via a subsumption check.

4 Conclusions

In this paper we have presented a possible architecture for the generation and the use of a fuzzy ontology for object recognition by means of objects decomposition in parts. We take advantage of the use of fuzzy cardinality restrictions which, to the best of our knowledge, have not been fully exploited in the current fuzzy DLs applications (e.g. multimedia retrieval). Our results are preliminar and prone to errors, partly due to limitations in the modules in use (e.g. the fuzzy reasoner is still experimental), partly due to the approximations induced by the use of a $\mathcal{SHIN}$ logic, while at least qualified cardinality restrictions would be needed.

As future work, we plan to take advantage of a more powerful fuzzy DL as it seems to be needed for object modeling purposes, so we will work on a more powerful reasoner and on a better integration between classical and fuzzy knowledge bases; furthermore, as we plan to use the system as an aid to the grasping task, we will add physical information (that can obtained via different sensors, e.g. haptic devices) and further information on the grasping types along with their quality measurements.

References

1. Biederman, I.: Recognition-by-components: A theory of human image understanding. Psychological Review **94** 2 (1987) 115-117
2. Bobillo, F., Straccia, U.: An OWL Ontology for Fuzzy OWL 2. Proceedings of the 18th International Symposium on Methodologies for Intelligent Systems (2009)
3. Haarslev, V., Lutz, C., Möller, R.: Foundations of spatioterminological reasoning with description logics. Proceedings of Sixth International Conference on Principles of Knowledge Representation and Reasoning (1998) 112–123
4. Hudelot, C.: Towards a cognitive vision platform for semantic image interpretation; Application to the recognition of biological organisms. PhD thesis. University of Nice Sophia Antipolis (2005)

5. Maillot, N.: Ontology based object learning and recognition. PhD thesis. University of Nice Sophia Antipolis (2005)
6. Hudelot, C., Atif, J., Bloch, I.: Fuzzy spatial relation ontology for image interpretation. In: Fuzzy Sets and Systems , **159** 15 (2008) 1929–1951
7. Stoilos, G., Stamou, G., Pan, J.Z., Simou, N., Tzouvaras, V.: Reasoning with the fuzzy description logic f-$\mathcal{SHIN}$: Theory, practice and applications. In P.C.G. da Costa et al. (eds): Uncertainty Reasoning for the Semantic Web I (2008) 262–281
8. Simou, N., Athanasiadis, T., Tzouvaras, V., Kollias, S.: Multimedia reasoning with f-$\mathcal{SHIN}$. Second International Workshop on Semantic Media Adaptation and Personalization (2007) 44–49
9. Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J.Z., Horrocks, I.: The fuzzy description logic f-$\mathcal{SHIN}$. International Workshop on Uncertainty Reasoning For the Semantic Web (2005)
10. Mylonas, P., Simou, N., Tzouvaras, V., Avrithis, Y.: Towards semantic multimedia indexing by classification and reasoning on textual metadata. Knowledge Acquisition from Multimedia Content Workshop (2007)
11. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in Description Logics for the Semantic Web. Journal of Web Semantics **6** 4 (2008) 291–308
12. Straccia, U.: Towards Spatial Reasoning in Fuzzy Description Logics. Proc. of the 2009 IEEE International Conference on Fuzzy Systems (2009)
13. Suh, I. H., Lim, G. H., Hwang, W., Suh, H., Choi, J.-H., Park, Y.-T.: Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence. IEEE Int. Conf. on Intelligent Robots and Systems (2007) 429–436
14. Wan, L.: Parts-based 2D shape decomposition by convex hull. IEEE International Conference on Shape Modeling and Applications (2009) 89–95
15. Dasiopoulou, S., Kompatsiaris, I., Strintzis, M.G.: Applying Fuzzy DLs in the extraction of image semantics. Journal of Data Semantics **14** (2009) 105–132
16. Meghini, C., Sebastiani, F., Straccia, U.: A model of multimedia information retrieval. Journal of ACM **48** 5 (2001) 909–970
17. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, **8** 6 (1986) 679-698