Reasoning with Probabilistic Logics

RICCARDO ZESE
Dipartimento di Ingegneria – University of Ferrara
Via Saragat 1, 44122, Ferrara, Italy
(*e-mail:* riccardo.zese@unife.it)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

The interest in the combination of probability with logics for modeling the world has rapidly increased in the last few years. One of the most effective approaches is the Distribution Semantics which was adopted by many logic programming languages and in Descripion Logics. In this paper, we illustrate the work we have done in this research field by presenting a probabilistic semantics for description logics and reasoning and learning algorithms. In particular, we present in detail the system TRILL^P, which computes the probability of queries w.r.t. probabilistic knowledge bases, which has been implemented in Prolog.

KEYWORDS: Probabilistic Description Logics, Probabilistic Reasoning, Tableau, Prolog, Semantic Web.

1 Introduction

In the last few years, many researchers tried to combine first-order logic and probability for modeling uncertain domains and performing inferece and learning. In the field of Probabilistic Logic Programming (PLP for short) many proposals have been presented. An effective and popular approach is the Distribution Semantics (Sato 1995), which underlies many PLP languages such as PRISM (Sato 1995; Sato and Kameya 2001), Independent Choice Logic (Poole 1997), Logic Programs with Annotated Disjunctions (Vennekens et al. 2004) and ProbLog (De Raedt et al. 2007). Along this line, many reserchers proposed to combine probability theory with Description Logics (DLs for short) (Lukasiewicz and Straccia 2008; Straccia 2008). DLs are at the basis of the Web Ontology Language (OWL for short), a family of knowledge representation formalisms used for modeling information of the Semantic Web (Hitzler et al. 2009). In (Riguzzi et al. 2012) we presented DISPONTE, a probabilistic semantics for DLs based on the distribution semantics that allows probabilistic assertional and terminological knowledge.

In order to allow inference over the information in the Semantic Web, many efficient DL reasoners, such as Pellet (Sirin et al. 2007), RacerPro (Haarslev et al. 2012) and HermiT (Shearer et al. 2008), have been developed. Despite the availability of many DL reasoners, the number of probabilistic reasoners is quite small. In (Riguzzi et al. 2013a) we presented BUNDLE, a reasoner based on Pellet that extends it by allowing to perform inference on DISPONTE theories. Most of the available DL reasoners, included BUNDLE, exploit procedural languages for implementing their reasoning algorithms. Nonetheless, some of

them use non-deterministic operators for doing inference. We implemented a reasoner, called TRILL (Zese et al. 2013), that exploits Prolog for managing the non-determinism. Then, we developed a new version of TRILL, called TRILL^P and we added in both versions the ability to manage DISPONTE knowledge bases (KBs for short) and computing the probability of a query given a probabilistic KB under the DISPONTE semantics.

Since a problem of probabilistic KBs is that the parameters are difficult to define, in (Riguzzi et al. 2013b) we presented EDGE that learns the parameters of a DISPONTE KB from the information available in the domain. Moreover, we are currently working on the extension of EDGE in order also to learn the structure of the probabilistic KB togheter with the parameters.

In the field of PLP, we are working at improving existing algorithms. We have considered lifted inference that allows to perform inference in a time that is polynomial in the variables' domain size. We applied lifted variable elimination, and GC-FOVE (Taghipour et al. 2013) in particular, to PLP and developed the algorithm LP^2.

The paper is organised as follows. Section 2 briefly introduces $\mathcal{ALC}$, while Section 3 presents the DISPONTE semantics. Section 4 defines the problem of finding explanations for a probabilistic query w.r.t. a given probabilistic KB. Section 5 presents TRILL and TRILL^P and Section 6 discusses related work. Section 7 shows experiments and section 8 discusses our achievements and future plans. Finally, Section 9 concludes the paper.

2 Description Logics

DLs are knowledge representation formalisms that are at the basis of the Semantic Web (Baader et al. 2003; Baader et al. 2008) and are used for modeling ontologies. They are represented using a syntax based on concepts, basically sets of individuals of the domain, and roles, sets of pairs of individuals of the domain. In this section, we recall the expressive description logic $\mathcal{ALC}$ (Schmidt-Schauß and Smolka 1991). We refer to (Lukasiewicz and Straccia 2008) for a detailed description of $\mathcal{SHOIN}(\mathbf{D})$ DL, that is at the basis of OWL DL.

Let $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{I}$ be sets of *atomic concepts*, *roles* and *individuals*. A *role* is an atomic role $R \in \mathbf{R}$. *Concepts* are defined by induction as follows. Each $C \in \mathbf{A}$, $\bot$ and $\top$ are concepts. If C, C_1 and C_2 are concepts and $R \in \mathbf{R}$, then $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$, $\neg C$, $\exists R.C$, and $\forall R.C$ are concepts. Let C, D be concepts, $R \in \mathbf{R}$ and $a, b \in \mathbf{I}$. An *ABox* $\mathcal{A}$ is a finite set of *concept membership axioms* $a : C$ and *role membership axioms* $(a, b) : R$, while a *TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$. $C \equiv D$ abbreviates $C \sqsubseteq D$ and $D \sqsubseteq C$.

A *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A KB $\mathcal{K}$ is assigned a semantics in terms of set-theoretic interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function* that assigns an element in $\Delta^{\mathcal{I}}$ to each $a \in \mathbf{I}$, a subset of $\Delta^{\mathcal{I}}$ to each $C \in \mathbf{A}$ and a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each $R \in \mathbf{R}$.

A query Q over a KB $\mathcal{K}$ is an axiom for which we want to test the entailment from the knowledge base, written $\mathcal{K} \models Q$. The entailment test may be reduced to checking the unsatisfiability of a concept in the knowledge base, i.e., the emptiness of the concept. For example, the entailment of the axiom $C \sqsubseteq D$ may be tested by checking the satisfiability of the concept $C \sqcap \neg D$.

3 The DISPONTE Semantics

DISPONTE (Riguzzi et al. 2012) applies the distribution semantics (Sato 1995) of probabilistic logic programming to DLs. A program following this semantics defines a probability distribution over normal logic programs called *worlds*. Then the distribution is extended to queries and the probability of a query is obtained by marginalizing the joint distribution of the query and the programs.

In DISPONTE, a *probabilistic knowledge base* $\mathcal{K}$ is a set of *certain axioms* or *probabilistic axioms* in which each axiom is independent evidence. Certain axioms take the form of regular DL axioms while probabilistic axioms are $p :: E$ where p is a real number in $[0, 1]$ and E is a DL axiom.

The idea of DISPONTE is to associate independent Boolean random variables to the probabilistic axioms. To obtain a *world*, we include every formula obtained from a certain axiom. For each probabilistic axiom, we decide whether to include it or not in w. A world therefore is a non probabilistic KB that can be assigned a semantics in the usual way. A query is entailed by a world if it is true in every model of the world.

The probability p can be interpreted as an *epistemic probability*, i.e., as the degree of our belief in axiom E. For example, a probabilistic concept membership axiom $p :: a : C$ means that we have degree of belief p in $C(a)$. A probabilistic concept inclusion axiom of the form $p :: C \sqsubseteq D$ represents our belief in the truth of $C \sqsubseteq D$ with probability p.

Formally, an *atomic choice* is a couple (E_i, k) where E_i is the ith probabilistic axiom and $k \in \{0, 1\}$. k indicates whether E_i is chosen to be included in a world ($k = 1$) or not ($k = 0$). A *composite choice* κ is a consistent set of atomic choices, i.e., $(E_i, k) \in \kappa, (E_i, m) \in \kappa$ implies $k = m$ (only one decision is taken for each formula). The probability of a composite choice κ is $P(\kappa) = \prod_{(E_i,1)\in\kappa} p_i \prod_{(E_i,0)\in\kappa}(1 - p_i)$, where p_i is the probability associated with axiom E_i. A *selection* σ is a total composite choice, i.e., it contains an atomic choice (E_i, k) for every probabilistic axiom of the probabilistic KB. A selection σ identifies a theory w_σ called a *world* in this way: $w_\sigma = \mathcal{C} \cup \{E_i | (E_i, 1) \in \sigma\}$ where $\mathcal{C}$ is the set of certain axioms. Let us indicate with $\mathcal{S}_\mathcal{K}$ the set of all selections and with $\mathcal{W}_\mathcal{K}$ the set of all worlds. The probability of a world w_σ is $P(w_\sigma) = P(\sigma) = \prod_{(E_i,1)\in\sigma} p_i \prod_{(E_i,0)\in\sigma}(1 - p_i)$. $P(w_\sigma)$ is a probability distribution over worlds, i.e., $\sum_{w\in\mathcal{W}_\mathcal{K}} P(w) = 1$.

We can now assign probabilities to queries. Given a world w, the probability of a query Q is defined as $P(Q|w) = 1$ if $w \models Q$ and 0 otherwise. The probability of a query can be defined by marginalizing the joint probability of the query and the worlds, i.e. $P(Q) = \sum_{w\in\mathcal{W}_\mathcal{K}} P(Q, w) = \sum_{w\in\mathcal{W}_\mathcal{K}} P(Q|w)p(w) = \sum_{w\in\mathcal{W}_\mathcal{K}: w\models Q} P(w)$.

Example 3.1
Consider the following KB, inspired by the `people+pets` ontology (Patel-Schneider et al. 2003):

$$0.5 :: \exists hasAnimal.Pet \sqsubseteq NatureLover \qquad 0.6 :: Cat \sqsubseteq Pet$$
$$(kevin, tom) : hasAnimal \qquad (kevin, fluffy) : hasAnimal \qquad tom : Cat \qquad fluffy : Cat$$

The KB indicates that the individuals that own an animal which is a pet are nature lovers with a 50% probability and that *kevin* has the animals *fluffy* and *tom*. Fluffy and *tom* are cats and cats are pets with probability 60%. We associate a Boolean variable to each axiom as follow $F_1 = \exists hasAnimal.Pet \sqsubseteq NatureLover$, $F_2 = (kevin, fluffy) : hasAnimal$, $F_3 = (kevin, tom) : hasAnimal$, $F_4 = fluffy : Cat$, $F_5 = tom : Cat$ and $F_6 = Cat \sqsubseteq Pet$.

The KB has four worlds and the query axiom $Q = kevin : NatureLover$ is true in one of them, the one corresponding to the selection $\{(F_1, 1), (F_2, 1)\}$. The probability of the query is $P(Q) = 0.5 \cdot 0.6 = 0.3$.

Example 3.2
Sometimes we have to combine knowledge from multiple, untrusted sources, each one with a different reliability. Consider a KB similar to the one of Example 3.1 but where we have a single cat, *fluffy*.

$$\exists hasAnimal.Pet \sqsubseteq NatureLover \qquad (kevin, fluffy) : hasAnimal \qquad Cat \sqsubseteq Pet$$

and there are two sources of information with different reliability that provide the information that *fluffy* is a cat. On one source the user has a degree of belief of 0.4, i.e., he thinks it is correct with a 40% probability, while on the other source he has a degree of belief 0.3. The user can reason on this knowledge by adding the following statements to his KB:

$$0.4 \;::\; fluffy : Cat \qquad 0.3 \;::\; fluffy : Cat$$

The two statements represent independent evidence on *fluffy* being a cat. We associate F_1 (F_2) to the first (second) probabilistic axiom.

The query axiom $Q = kevin : NatureLover$ is true in 3 out of the 4 worlds, those corresponding to the selections $\{\{(F_1, 1), (F_2, 1)\}, \{(F_1, 1), (F_2, 0)\}, \{(F_1, 0), (F_2, 1)\}\}$. So $P(Q) = 0.4 \cdot 0.3 + 0.4 \cdot 0.7 + 0.6 \cdot 0.3 = 0.58$. This is reasonable if the two sources can be considered as independent. In fact, the probability comes from the disjunction of two independent Boolean random variables with probabilities respectively 0.4 and 0.3: $P(Q) = P(X_1 \vee X_2) = P(X_1) + P(X_2) - P(X_1 \wedge X_2) = P(X_1) + P(X_2) - P(X_1)P(X_2) = 0.4 + 0.3 - 0.4 \cdot 0.3 = 0.58$

4 Querying KBs

Traditionally, a reasoning algorithm decides whether an axiom is entailed or not by a KB by refutation: the axiom E is entailed if $\neg E$ has no model in the KB. Besides deciding whether an axiom is entailed by a KB, we want to find also explanations for the axiom.

The problem of finding explanations for a query has been investigated by various authors (Schlobach and Cornet 2003; Kalyanpur et al. 2005; Kalyanpur 2006; Kalyanpur et al. 2007; Halaschek-Wiener et al. 2006). It was called *axiom pinpointing* in (Schlobach and Cornet 2003) and considered as a non-standard reasoning service useful for tracing derivations and debugging ontologies. In particular, in (Schlobach and Cornet 2003) the authors define *minimal axiom sets* (*MinAs* for short).

Definition 4.1 (*MinA*)
Let $\mathcal{K}$ be a knowledge base and Q an axiom that follows from it, i.e., $\mathcal{K} \models Q$. We call a set $M \subseteq \mathcal{K}$ a *minimal axiom set* or *MinA* for Q in $\mathcal{K}$ if $M \models Q$ and it is minimal w.r.t. set inclusion.

The problem of enumerating all MinAs is called MIN-A-ENUM. ALL-MINAS$(Q, \mathcal{K})$ is the set of all MinAs for query Q in knowledge base $\mathcal{K}$. Reasoners such as Pellet solve the MIN-A-ENUM problem by finding a single MinA using a tableau algorithm and then applying the *hitting set* (Reiter 1987) algorithm for finding all the others.

A *tableau* is a graph where each node represents an individual a and is labeled with the set of concepts $\mathcal{L}(a)$ it belongs to. Each edge $\langle a, b\rangle$ in the graph is labeled with the set of roles to which the couple (a, b) belongs. Then, a set of consistency preserving tableau expansion rules are repeatedly applied until a clash (i.e., a contradiction) is detected or a clash-free graph is found to which no more rules are applicable. A clash is for example a couple (C, a) where C and $\neg C$ are present in the label of a node, i.e. $C, \neg C \subseteq \mathcal{L}(a)$.

Some expansion rules are non-deterministic, i.e., they generate a finite set of tableaux. Thus the algorithm keeps a set of tableaux that is consistent if there is any tableau in

it that is consistent, i.e., that is clash-free. Each time a clash is detected in a tableau G, the algorithm stops applying rules to G. Once every tableau in T contains a clash or no more expansion rules can be applied to it, the algorithm terminates. If all the tableaux in the final set T contain a clash, the algorithm returns unsatisfiable as no model can be found. Otherwise, any one clash-free completion graph in T represents a possible model for the concept and the algorithm returns satisfiable. The hitting set algorithm is a black box method: it repeatedly removes an axiom from the KB and then computes again a MinA recording all the different MinAs so found.

MIN-A-ENUM is required to answer queries to KBs following the DISPONTE semantics. To compute the probability of a query, the explanations must be made mutually exclusive, so that the probability of each individual explanation is computed and summed with the others. This can be done by exploiting a splitting algorithm as shown in (Poole 2000). Alternatively, we can assign independent Boolean random variables to the axioms contained in the explanations and defining the Disjunctive Normal Form (DNF) Boolean formula f_K which models the set of explanations. Thus $f_K(\mathbf{X}) = \bigvee_{\kappa \in K} \bigwedge_{(E_i,1)} X_i \bigwedge_{(E_i,0)} \overline{X_i}$ where $\mathbf{X} = \{X_i | (E_i, k) \in \kappa, \kappa \in K\}$ is the set of Boolean random variables. We can now translate f_K to a Binary Decision Diagram (BDD), from which we can compute the probability of the query with a dynamic programming algorithm that is linear in the size of the BDD.

5 The algorithms TRILL and TRILL^P

TRILL (Zese et al. 2013) implements the tableau algorithm using Prolog. In this way, we do not have to implement a search strategy, such as the hitting set algorithm, because the management of the non-determinism is demanded to Prolog. TRILL takes as input an OWL DL ontology translated into Prolog facts by using the Thea2 library (Vassiliadis et al. 2009). For example, a subclass axiom $Cat \sqsubseteq Pet$ is translated into `subClass('Cat','Pet')` while for more complex axioms, Thea2 uses Prolog lists, so the axiom $NatureLover \equiv PetOwner \sqcup GardenOwner$ is translated into `equivalentClasses(['NatureLover',unionOf(['PetOwner','GardenOwner'])])`.

TRILL builds a tableau following the tableau algorithm. The non-deterministic rules are treated differently from the deterministic ones. While the latter ones are implemented by predicates that take as input a tableau and return a single tableau, the former ones are implemented by predicates that take as input a tableau but return a list of tableaux from which one is non-deterministically chosen. The computation of ALL-MINAS$(Q, \mathcal{K})$ is performed by simply calling `findall/3` over the tableau predicate.

A new version of TRILL, called TRILL^P, resolves the axiom pinpointing problem by computing a *pinpointing formula* (Baader and Peñaloza 2010a; Baader and Peñaloza 2010b) instead of a set of MinAs. To define the pinpointing formula we first have to associate a Boolean variable to each axiom of the KB $\mathcal{K}$. The pinpointing formula is a monotone Boolean formula on these variables. This formula compactly encodes the set of all MinAs. Let assume that each axiom E of a KB $\mathcal{K}$ is associated with the propositional variable $var(E)$. The set of all propositional variables is indicated with $var(\mathcal{K})$. A valuation ν of a monotone Boolean formula is the set of propositional variables that are true. For a valuation $\nu \subseteq var(\mathcal{K})$, let $\mathcal{K}_\nu := \{t \in \mathcal{K} | var(t) \in \nu\}$.

Definition 5.1 (*Pinpointing formula*)
Given a query Q and a KB $\mathcal{K}$, a monotone Boolean formula ϕ over $var(\mathcal{K})$ is called a *pinpointing formula* for Q if for every valuation $\nu \subset var(\mathcal{K})$ it holds that $\mathcal{K}_\nu \models Q$ if ν satisfies ϕ.

In Lemma 2.4 of (Baader and Peñaloza 2010b), the authors proved that we can obtain all MinAs from a pinpointing formula by transforming the formula into DNF and removing disjuncts implying other disjuncts. From this formula, the construction of BDD can be performed as for MinAs. For formal definitions see (Baader and Peñaloza 2010a; Baader and Peñaloza 2010b).

Example 5.1 (*Pinpointing formula*)
Consider the KB of Example 3.1 with the same association between Boolean variables and axioms. Let $Q = kevin : NatureLover$ be the query, then ALL-MINAS$(Q, \mathcal{K}) = \{\{F_2, F_4, F_6, F_1\}, \{F_3, F_5, F_6, F_1\}\}$, while the pinpointing formula is $((F_2 \wedge F_4) \vee (F_3 \wedge F_5)) \wedge F_6 \wedge F_1$.

In order to build the BDDs and compute the associated probabilities, TRILL and TRILL^P exploit a Prolog library of the `cplint` suite (Riguzzi 2009). The code of TRILL and TRILL^P is available at `https://sites.google.com/a/unife.it/ml/trill`.

6 Related Work

DL reasoners written in Prolog do not need to implement a backtracking algorithm but can exploit Prolog backtracking facilities for performing the search. This has been observed in various works. Beckert and Posegga (1995) proposed a tableau reasoner in Prolog for FOL based on variable-free semantic tableaux, but it is not tailored to DLs. Meissner (2004) presented the implementation of a Prolog reasoner for the DL $\mathcal{ALCN}$. Herchenröder (2006) improved it by implementing heuristic search techniques to reduce the running time. Faizi (2011) added to its work the possibility of returning explanations for queries w.r.t. $\mathcal{ALC}$ KBs. Hustadt et al. (2008) presented the KAON2 algorithm that exploits basic superposition, a refutational theorem proving method for FOL with equality and a new inference rule, called decomposition, to reduce a $\mathcal{SHIQ}$ KB into a disjunctive datalog program. Lukácsy and Szeredi (2009) presented DLog, that is an ABox reasoning algorithm for the $\mathcal{SHIQ}$ language. It allows to store the content of the ABox externally in a database and to answer ABox queries by transforming the KB into a Prolog program. TRILL and TRILL^P differ from these works for the considered DL and, in particular, from DLog for the capability of answering general queries.

Bruynooghe et al. (2010) presented FOProblog that is based on Problog, in which a program contains a set of *probabilistic facts*, i.e. facts annotated with probabilities, and a set of general clauses which can have positive and negative probabilistic facts in their body. Each fact is assumed to be probabilistically independent. It follows the distribution semantics and exploits BDDs to compute the probability of queries. FOProblog is a reasoner for FOL that is not tailored to DLs, so the algorithm could be suboptimal. It does not exploit a tableau algorithm and cannot manage probabilistic facts which are annotated with more than one probability value.

A different approach is the one of Ricca et al. (2009) that presented OntoDLV, a system for reasoning on logic-based ontology representation language, called OntoDLP.

OntoDLP is an extension of (disjunctive) ASP and can interoperate with OWL. OntoDLV rewrites the OWL KB into the OntoDLP language, can retrieve information directly from external OWL Ontologies and answers queries by using ASP.

BUNDLE (Riguzzi et al. 2013a) is a probabilistic reasoner that computes the probability of queries from probabilistic KBs that follow the DISPONTE semantics. It is based on Pellet and is completely written in Java. It exploits a modified version of Pellet for finding the ALL-MINAS set and then it translates it into a BDD from which it computes the probability of the query. Similarly to BUNDLE, PRONTO (Klinov 2008) is based on Pellet and performs inference on P-$\mathcal{SHIQ}$(D) KBs in which the probabilistic part contains *conditional contraints* of the form $(D|C)[l, u]$ that informally mean "generally, if an object belongs to C, then it belongs to D with a probability in the interval $[l, u]$". P-$\mathcal{SHIQ}$(D) (Lukasiewicz 2008) uses probabilistic lexicographic entailment from probabilistic default reasoning and allows both terminological and assertional probabilistic knowledge about instances of concepts and roles. P-$\mathcal{SHIQ}$(D) is based on Nilsson's probabilistic logic (Nilsson 1986) in which the probabilistic interpretation Pr defines a probability distribution over the set of interpretations Int instead of a probability distribution over theories. The probability of a logical formula F according to Pr, denoted $Pr(F)$, is the sum of all $Pr(I)$ such that $I \in Int$ and $I \models F$.

7 Experiments

We did several experiments in order to evaluate the performances of the algorithms we have implemented. Here we report a comparison between the performances of TRILL, TRILL^P and BUNDLE when computing probability for queries. We used four different knowledge bases of various complexity: 1) BRCA[1] models the risk factor of breast cancer; 2) an extract of the DBPedia[2] ontology obtained from Wikipedia; 3) Biopax level 3[3] models metabolic pathways; 4) Vicodi[4] contains information on European history.

For the tests, we used a version of the DBPedia and Biopax KBs without the ABox, a version of the BRCA with an ABox containing 1 individual and a version of Vicodi with an ABox containing 19 individuals. To each KB, we added 50 probabilistic axioms. For each datasets we randomly created 100 different queries. In particular, for the DBPedia and Biopax datasets we created 100 subclass-of queries while for the other KBs we created 80 subclass-of and 20 instance-of queries. For generating the subclass-of queries, we randomly selected two classes that are connected in the hierarchy of classes contained in the ontology, so that each query had at least one explanation. For the instance-of queries, we randomly selected an individual a and a class to which a belongs by following the hierarchy of the classes starting from the class to which a is instantiated in the KB.

Table 1 shows, for each ontology, the average number of different MinAs computed and the average time in seconds that TRILL, TRILL^P and BUNDLE took for answering the queries. In particular, the BRCA and the version of DBPedia that we used contain a large number of subclass axioms between complex concepts. These preliminary tests show that

[1] `http://www2.cs.man.ac.uk/~klinovp/pronto/brc/cancer_cc.owl`
[2] `http://dbpedia.org/`
[3] `http://www.biopax.org/`
[4] `http://www.vicodi.org/`

Dataset	avg. n. minAs	TRILL time (s)	TRILL^P time (s)	BUNDLE time (s)
BRCA	6.49	27.87	4.74	6.96
DBPedia	16.32	51.56	4.67	3.79
Biopax level 3	3.92	0.12	0.12	1.85
Vicodi	1.02	0.19	0.19	1.12

Table 1. *Average times for computing the probability of queries in seconds.*

both TRILL and TRILL^P performances can sometimes be better than BUNDLE, even if they lack all the optimizations that BUNDLE inherits from Pellet. This represents evidence that a Prolog implementation of a Semantic Web tableau reasoner is feasible and that may lead to a practical system. Moreover, TRILL^P presents an improvement of the execution time with respect to TRILL when more MinAs are present.

8 Open Issues and expected achievement

Our work aims at developing fast algorithms for performing inference over probabilistic DISPONTE semantics. Section 5 shows that TRILL and TRILL^P can compute the explanations for a query and its probability w.r.t. a $\mathcal{SHOIN}(\mathbf{D})$ and an $\mathcal{ALC}$ probbilistic KB respectively. For the future we plan to improve the performances of both algorithms.

We are also studying the problem of lifted inference for probabilistic logic programming using lifted variable elimination. We are adapting the Generalized Counting First Order Variable Elimination (GC-FOVE) algorithm presented in (Taghipour et al. 2013) to probabilistic logic programming under the distribution semantics. To this purpose, we are developing the system LP^2 that extends GC-FOVE by introducing two new operators, *heterogeneous sum* and *heterogeneous multiplication*. This work will be presented at the ICLP 2014 main conference.

A second line of research is the problem of learning the parameters and the structure of a DISPONTE KB. Along this line, in (Riguzzi et al. 2013b) we presented a learning algorithm, called EDGE, that learns the parameters by taking as input a DL theory and a number of examples that are usually concept assertions divided into positive and negative examples. EDGE first computes, for each example, the BDD encoding its explanations, then it executes an *Expectation-Maximization* (EM) algorithm, in which the functions Expactation and Maximization are repeatedly applied until the log-likelihood of the examples reaches a local maximum. Moreover, we are working on extending EDGE in order to learn also the structure of a DISPONTE KB togheter with the parameters by adapting the CELOE algorithm (Lehmann et al. 2011).

9 Conclusions

In this paper we presented two algorithms TRILL and TRILL^P for reasoning on DISPONTE KBs which are written in Prolog. The experiments show that Prolog is a viable language for implementing DL reasoning algorithms and that the performances of the two presented algorithms are comparable with those of a state-of-art reasoner.

10 Acknowledgements

This work was started by the Artificial Intelligence research group of the engineering department of the University of Ferrara. We would personally thank my colleagues and friends (in alphabetical order) Elena Bellodi, Evelina Lamma and Fabrizio Riguzzi.

References

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., Eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baader, F., Horrocks, I., and Sattler, U. 2008. Description logics. In *Handbook of knowledge representation*. Elsevier, Chapter 3, 135–179.

Baader, F. and Peñaloza, R. 2010a. Automata-based axiom pinpointing. *J. Autom. Reasoning 45,* 2, 91–129.

Baader, F. and Peñaloza, R. 2010b. Axiom pinpointing in general tableaux. *J. Log. Comput. 20,* 1, 5–34.

Beckert, B. and Posegga, J. 1995. leantap: Lean tableau-based deduction. *J. Autom. Reasoning 15,* 3, 339–358.

Bruynooghe, M., Mantadelis, T., Kimmig, A., Gutmann, B., Vennekens, J., Janssens, G., and Raedt, L. D. 2010. Problog technology for inference in a probabilistic first order logic. In *ECAI*. 719–724.

De Raedt, L., Kimmig, A., and Toivonen, H. 2007. ProbLog: A probabilistic Prolog and its application in link discovery. In *International Joint Conference on Artificial Intelligence*. 2462–2467.

Faizi, I. 2011. A Description Logic Prover in Prolog, Bachelor's thesis, Informatics Mathematical Modelling, Technical University of Denmark.

Haarslev, V., Hidde, K., Mller, R., and Wessel, M. 2012. The racerpro knowledge representation and reasoning system. *Semantic Web*, 267–277.

Halaschek-Wiener, C., Kalyanpur, A., and Parsia, B. 2006. Extending tableau tracing for ABox updates. Tech. rep., University of Maryland.

Herchenröder, T. 2006. Lightweight semantic web oriented reasoning in Prolog: Tableaux inference for description logics. M.S. thesis, School of Informatics, University of Edinburgh.

Hitzler, P., Krötzsch, M., and Rudolph, S. 2009. *Foundations of Semantic Web Technologies*. CRCPress.

Hustadt, U., Motik, B., and Sattler, U. 2008. Deciding expressive description logics in the framework of resolution. *Inf. Comput. 206,* 5, 579–601.

Kalyanpur, A. 2006. Debugging and repair of OWL ontologies. Ph.D. thesis, The Graduate School of the University of Maryland.

Kalyanpur, A., Parsia, B., Horridge, M., and Sirin, E. 2007. Finding all justifications of OWL DL entailments. In *ISWC*. LNCS, vol. 4825. Springer, 267–280.

Kalyanpur, A., Parsia, B., Sirin, E., and Hendler, J. A. 2005. Debugging unsatisfiable classes in OWL ontologies. *J. Web Sem. 3,* 4, 268–293.

Klinov, P. 2008. Pronto: A non-monotonic probabilistic description logic reasoner. In *European Semantic Web Conference*. LNCS, vol. 5021. Springer, 822–826.

Lehmann, J., Auer, S., Bhmann, L., and Tramp, S. 2011. Class expression learning for ontology engineering. *J. Web Sem.*, 71–81.

Lukácsy, G. and Szeredi, P. 2009. Efficient description logic reasoning in prolog: The dlog system. *TPLP 9,* 3, 343–414.

Lukasiewicz, T. 2008. Expressive probabilistic description logics. *Artif. Int. 172,* 6-7, 852–883.

Lukasiewicz, T. and Straccia, U. 2008. Managing uncertainty and vagueness in description logics for the semantic web. *J. Web Sem. 6,* 4, 291–308.

Meissner, A. 2004. An automated deduction system for description logic with alcn language. *Studia z Automatyki i Informatyki 28-29*, 91–110.

Nilsson, N. J. 1986. Probabilistic logic. *Artif. Intell. 28,* 1, 71–87.

Patel-Schneider, P, F., Horrocks, I., and Bechhofer, S. 2003. Tutorial on OWL.

Poole, D. 1997. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artif. Intell. 94,* 1-2, 7–56.

Poole, D. 2000. Abducing through negation as failure: stable models within the independent choice logic. *J. Log. Program. 44,* 1-3, 5–35.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artif. Intell. 32,* 1, 57–95.

Ricca, F., Gallucci, L., Schindlauer, R., Dell'Armi, T., Grasso, G., and Leone, N. 2009. Ontodlv: An asp-based system for enterprise ontologies. *J. Log. Comput. 19,* 4, 643–670.

Riguzzi, F. 2009. Extended semantics and inference for the Independent Choice Logic. *Log. J. IGPL 17,* 6, 589–629.

Riguzzi, F., Bellodi, E., Lamma, E., and Zese, R. 2013a. BUNDLE: A reasoner for probabilistic ontologies. In *RR 2013*, W. Faber and D. Lembo, Eds. LNCS, vol. 7994. Springer, 183–197.

Riguzzi, F., Bellodi, E., Lamma, E., and Zese, R. 2013b. Parameter learning for probabilistic ontologies. In *RR 2013*, W. Faber and D. Lembo, Eds. LNCS, vol. 7994. Springer, 183–197.

Riguzzi, F., Lamma, E., Bellodi, E., and Zese, R. 2012. Epistemic and statistical probabilistic ontologies. In *Uncertainty Reasoning for the Semantic Web*. CEUR Workshop Proceedings, vol. 900. Sun SITE Central Europe, 3–14.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *International Conference on Logic Programming*. MIT Press, 715–729.

Sato, T. and Kameya, Y. 2001. Parameter learning of logic programs for symbolic-statistical modeling. *J. Artif. Intell. Res. 15*, 391–454.

Schlobach, S. and Cornet, R. 2003. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI*. Morgan Kaufmann, 355–362.

Schmidt-Schauss, M. and Smolka, G. 1991. Attributive concept descriptions with complements. *Artificial Intelligence 48,* 1, 1–26.

Shearer, R., Motik, B., and Horrocks, I. 2008. Hermit: A highly-efficient owl reasoner. In *OWLED*.

Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., and Katz, Y. 2007. Pellet: A practical OWL-DL reasoner. *J. Web Sem. 5,* 2, 51–53.

Straccia, U. 2008. Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In *International Summer School on Reasoning Web*. LNCS, vol. 5224. Springer, 54–103.

Taghipour, N., Fierens, D., Davis, J., and Blockeel, H. 2013. Lifted variable elimination: Decoupling the operators from the constraint language. *J. Artif. Intell. Res. (JAIR) 47*, 393–439.

Vassiliadis, V., Wielemaker, J., and Mungall, C. 2009. Processing owl2 ontologies using thea: An application of logic programming. In *International Workshop on OWL: Experiences and Directions*. CEUR Workshop Proceedings, vol. 529. CEUR-WS.org.

Vennekens, J., Verbaeten, S., and Bruynooghe, M. 2004. Logic programs with annotated disjunctions. In *International Conference on Logic Programming*. LNCS, vol. 3131. Springer, 195–209.

Zese, R., Bellodi, E., Lamma, E., and Riguzzi, F. 2013. A description logics tableau reasoner in prolog. In *CILC*, D. Cantone and M. N. Asmundo, Eds. CEUR Workshop Proceedings, vol. 1068. CEUR-WS.org, 33–47.