Reasoning with Probabilistic Ontologies*

Fabrizio Riguzzi(1), Elena Bellodi(2), Evelina Lamma(2), Riccardo Zese(2)

(1) Dipartimento di Matematica e Informatica – University of Ferrara
    Via Saragat 1, I-44122, Ferrara, Italy
(2) Dipartimento di Ingegneria – University of Ferrara
    Via Saragat 1, I-44122, Ferrara, Italy

{fabrizio.riguzzi, elena.bellodi, evelina.lamma, riccardo.zese}@unife.it

Abstract

Modeling real world domains requires ever more frequently to represent uncertain information. The DISPONTE semantics for probabilistic description logics allows to annotate axioms of a knowledge base with a value that represents their probability. In this paper we discuss approaches for performing inference from probabilistic ontologies following the DISPONTE semantics. We present the algorithm BUNDLE for computing the probability of queries. BUNDLE exploits an underlying Description Logic reasoner, such as Pellet, in order to find explanations for a query. These are then encoded in a Binary Decision Diagram that is used for computing the probability of the query.

1 Introduction

Representing uncertain information is of foremost importance in order to effectively model real world domains. Many authors studied the integration of probability with logic [Straccia, 2008] and in particular with Description Logics (DLs) [Łukasiewicz and Straccia, 2008]. In the field of Logic Programming, the distribution semantics [Sato, 1995] has emerged as one of the most effective approaches. Following this line, in [Riguzzi et al., 2012] we presented DISPONTE for “Distribution Semantics for Probabilistic Ontologies” (Spanish for “get ready”) which applies the distribution semantics to DLs. The main idea is to annotate axioms of a theory with a probability and assume that each axiom is independent of the others. A DISPONTE knowledge base (KB) defines a probability distribution over regular KBs (worlds) and the probability of a query is obtained from the joint probability of the worlds and the query.

In [Riguzzi et al., 2013] we presented the system BUNDLE for “Binary decision diagrams for Uncertain reasoning on Description Logic theories”. BUNDLE exploits an underlying ontology reasoner, such as Pellet [Sirin et al., 2007], for performing inference over DISPONTE DLs and uses the inference techniques developed for probabilistic logic programs under the distribution semantics, in particular Binary Decision Diagrams (BDDs), for computing the probability of queries. BUNDLE first finds the set of the explanations for the query by means of Pellet and then encodes them into a BDD from which the probability of the query can be computed in time linear in the size of the diagram.

In this paper we update and review [Riguzzi et al., 2013] by providing more information about BUNDLE, discussing the complexity of the algorithm and presenting further experiments. The complexity results imply that inference in DISPONTE is intractable in the worst case, thus limiting the general applicability of BUNDLE. However the experimentation shows that in practice BUNDLE is able to handle domains of significant size. The paper is organized as follows. Section 2 introduces DLs and Section 3 illustrates DISPONTE. Section 4 describes BUNDLE and discusses its complexity while Section 5 presents related work. Section 6 illustrates the results of scalability experiments with BUNDLE and, finally, Section 7 concludes the paper.

2 Description Logics

Description Logics (DLs) are knowledge representation formalisms that possess nice computational properties such as decidability and/or low complexity, see [Baader et al., 2003; 2008] for excellent introductions. DLs are particularly useful for representing ontologies and have been adopted as the basis of the Semantic Web. For example, the OWL DL sublanguage of OWL 1 is based on the $SHOIN(D)$ DL. While DLs can be translated into predicate logic, they are usually represented using a syntax based on concepts and roles. A concept corresponds to a set of individuals of the domain while a role corresponds to a set of couples of individuals of the domain. In the rest of this Section we concentrate on $SHOIN(D)$.

Let $A$, $R$ and $I$ be sets of atomic concepts, atomic roles and individuals, respectively. A role is either an atomic role $R \in R$ or the inverse $R^-$ of an atomic role $R \in R$. We use $R^\top$ to denote the set of all inverses of roles in $R$. Concepts are defined by induction as follows. Each $A \in A$, $\bot$ and $\top$ are concepts and if $a \in I$, then $\{a\}$ is a concept called a nominal. If $C$, $C_1$ and $C_2$ are concepts and $R \in R \cup R^-$, then $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$ and $\neg C$ are concepts, as well as $\exists R.C$ and $\forall R.C$ and $\geq n R$ and $\leq n R$ for an integer $n \geq 0$. 

---

*This paper was invited for submission to the Best Papers From Sister Conferences Track, based on a paper that appeared in the 7th International Conference on Web Reasoning and Rule Systems (RR 2013).
An RBox \( \mathcal{R} \) consists of a finite set of transitivity axioms \( \text{Trans}(\mathcal{R}) \), where \( R \in \mathbb{R} \), and role inclusion axioms \( R \sqsubseteq S \), where \( R, S \in \mathbb{R} \). A TBox \( \mathcal{T} \) is a finite set of concept inclusion axioms \( C \sqsubseteq D \), where \( C \) and \( D \) are concepts. We use \( C \equiv D \) to abbreviate \( C \sqsubseteq D \) and \( D \sqsubseteq C \). An ABox \( \mathcal{A} \) is a finite set of concept membership axioms \( a : C \), role membership axioms \( (a, b) : R \), equality axioms \( a = b \), and inequality axioms \( a \neq b \), where \( C \) is a concept, \( R \in \mathbb{R} \) and \( a, b \in I \). A knowledge base \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \) consists of a TBox \( \mathcal{T} \), an RBox \( \mathcal{R} \) and an ABox \( \mathcal{A} \).

A KB \( \mathcal{K} \) is usually assigned a semantics in terms of set-theoretic interpretations and models of the form \( \llbracket \mathcal{I} \rrbracket = (\Delta^2, \tau^2, a^T, (C_1 \sqcap C_2)^T = C_1^T \sqcap C_2^T, (C_1 \sqcup C_2)^T = C_1^T \sqcup C_2^T, (\neg C)^T = \Delta^2 \setminus C^2, (\forall R.C)^T = \{ x \in \Delta^2 \mid R^T(x) \subseteq C^2 \}, (\exists R.C)^T = \{ x \in \Delta^2 \mid R^T(x) \cap C^2 \neq \emptyset \}, (\geq n)R^T = \{ x \in \Delta^2 \mid |R^T(x)| \geq n \}, (\leq n)R^T = \{ x \in \Delta^2 \mid |R^T(x)| \leq n \} \) to each \( C \). The mapping \( \tau^2 \) is extended to all concepts (where \( R^2(x) = \{ y \mid (x, y) \in R^2 \} \) and \( \#X \) denotes the cardinality of the set \( X \)).

\( \text{SHOIN}(\mathbb{D}) \) allows the use of datatype roles which map an individual to an element of a datatype (integers, floats, etc). Then new concept definitions involving datatype roles are added that mirror those involving roles introduced above. We also assume that we have predicates over the datatypes.

A query over a KB is usually an axiom for which we want to test the entailment from the KB. The entailment test may be reduced to checking the unsatisfiability of a concept in the KB, i.e., the emptiness of the concept. For example, the entailment of the axiom \( C \sqsubseteq D \) may be tested by checking the unsatisfiability of the concept \( C \sqcap \neg D \). A DL is decidable if the problem of checking the satisfiability of a KB is decidable. In particular, \( \text{SHOIN}(\mathbb{D}) \) is decidable iff there are no number restrictions on roles which are transitive or have transitive subroles.

### 3 The DISPONTE Semantics for Probabilistic Description Logics

DISPONTE applies the distribution semantics to probabilistic DL theories. The distribution semantics underlies many probabilistic logic programming languages such as PRISM [Sato, 1995; Sato and Kameya, 2001], Independent Choice Logic [Poole, 1997], Logic Programs with Annotated Disjunctions (LPADs) [Vennekens et al., 2004] and ProbLog [De Raedt et al., 2007]. A program in one of these languages defines a probability distribution over normal logic programs called worlds. Each language has its own ways of specifying the distribution but all offer the possibility of specifying alternatives in clauses. The probability of a world is obtained by multiplying the probabilities associated to each alternative as these are considered independent of each other. This gives a probability distribution \( P(w) \) over the worlds. This is extended to the joint probability of the worlds and the query and the probability of the latter is obtained by marginalization.

The distribution semantics was applied successfully in many domains [De Raedt et al., 2007; Sato and Kameya, 2001; Bellodi and Riguzzi, 2012] and various inference and learning algorithms are available for it [Kimmig et al., 2011; Riguzzi, 2009; Bellodi and Riguzzi, 2013; 2015].

In DISPONTE, a probabilistic knowledge base \( \mathcal{K} \) is a set of probabilistic axioms that take the form

\[ p : E \]  

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 probabilistic axioms. By assigning values to every random variable we obtain a world, the set of axioms whose random variable is assigned to 1. 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 \( C(a) \). The statement that Tweety flies with probability 0.9 can be expressed as 0.9 :: \textit{tweety} : Flies. A probabilistic concept inclusion axiom of the form \( p : C \sqsubseteq D \) represents the fact that we believe in the truth of \( C \sqsubseteq D \) with probability \( p \). Note that this means that \( C \) is a subclass of \( D \) with probability \( p \), not that each individual of \( C \) has probability \( p \) of belonging to \( D \).

Let us now give the formal definition of DISPONTE. An atomic choice is a pair \( (E_i, k) \) where \( E_i \) is the \( i \)th 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 \( \kappa \) is a consistent set of atomic choices, i.e., \( (E_i, k) \in \kappa, (E_i, m) \in \kappa \Rightarrow k = m \) (only one decision for each formula). The probability of composite choice \( \kappa \) is \( P(\kappa) = \prod_{(E_i,k) \in \kappa} p_i \prod_{(E_i,0) \in \kappa} (1-p_i) \), where \( p_i \) is the probability associated with axiom \( E_i \). A selection \( \sigma \) is a composite choice that contains an atomic choice \( (E_i, k) \) for every probabilistic axiom of the theory. A selection identifies a theory \( w_\sigma \) called a world in this way: \( w_\sigma = \{ E_i | (E_i, 1) \in \sigma \} \). Let us indicate with \( \mathcal{S}_K \) the set of all selections and with \( \mathcal{W}_K \) the set of all worlds. The probability of a world \( w_\sigma \) is \( P(w_\sigma) = P(\sigma) = \prod_{(E_i,1) \in \sigma} p_i \prod_{(E_i,0) \in \sigma} (1-p_i) \). The probability \( P(w) \) is a probability distribution over worlds, i.e., \( \sum_{w \in \mathcal{W}_K} P(w) = 1 \). 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:

\[ P(Q) = \sum_{w \in \mathcal{W}_K} P(Q|w) \]  

\[ = \sum_{w \in \mathcal{W}_K} P(Q|w)P(w) \]  

\[ = \sum_{w \in \mathcal{W}_K:w \models Q} P(w) \]
The KB indicates that the individuals that own an animal which is a pet are nature lovers and that Kevin owns the animals fluffy and tom. Moreover, we believe in the fact that fluffy and tom are cats and that the class of cats is a subclass of the class of pets with a certain probability. This KB has eight worlds and the query axiom $Q = \text{kevin : NatureLover}$ is true in three of them, those corresponding to the following selections:

- $\{(5), 1\}$
- $\{(5), 0\}$
- $\{(5), 1\}$

so the probability is

$$P(Q) = 0.4 \cdot 0.7 \cdot 0.6 + 0.6 \cdot 0.3 \cdot 0.6 + 0.4 \cdot 0.3 \cdot 0.6 = 0.348.$$ 

4 BUNDLE

BUNDLE (“Binary decision diagrams for Uncertain reasoning on Description Logic theories”) computes the probability of queries from a probabilistic KB that follows the DISPONTE semantics. Using (4) to compute the probability of a query is not practical as it involves generating all possible worlds. Since their number is exponential in the number of probabilistic axioms, we follow a different approach in which explanations for queries are found. From the set of explanations, BUNDLE builds a Binary Decision Diagram (BDD) for computing the probability of the query. In order to discuss the approach, we first need to introduce some definitions.

A composite choice $\kappa$ is an explanation for a query $Q$ if $Q$ is entailed by every world of $\omega_\kappa$, where $\omega_\kappa = \{w_\sigma | \sigma \in S_\kappa, \sigma \supseteq \kappa\}$ is the set of worlds compatible with $\kappa$. We also define the set of worlds identified by a set of composite choices $K$ as $\omega_K = \bigcup_{\kappa \in K} \omega_\kappa$. A set of composite choices $K$ is covering with respect to $Q$ if every world $w \in \omega_K$ in which $Q$ is entailed is such that $w \in \omega_\kappa$.

We can associate a Boolean random variable $X_i$ to probabilistic axiom $E_i$. An atomic choice $(E_i, k)$ then corresponds to $X_i$ assuming value $k$. The variables $X = \{X_i | (E_i, k) \in \kappa, \kappa \in K\}$ are pairwise independent and the probability that $X_i$ takes value 1 is $p_i$, the probability associated with the $i$th axiom.

Given a covering set of explanations $K$ for a query $Q$, each world where the query is true corresponds to an assignment of $X$ for which the following Boolean function takes value 1:

$$f_K(X) = \bigvee_{\kappa \in K} \bigwedge_{(E_i, 1) \in \kappa} X_i \bigwedge_{(E_i, 0) \in \kappa} \bar{X_i}$$

Thus we can compute the probability of $Q$ by computing the probability that $f_K(X)$ takes value 1. This formula is in Disjunctive Normal Form (DNF) but we can’t compute $P(f_K(X))$ by summing the probability of each individual explanation because the different explanations may not be mutually disjoint. To solve the problem, we can apply knowledge compilation to the propositional formula $f_K(X)$ [Darwiche and Marquis, 2002] in order to translate it into a target language that allows the computation of the probability in polynomial time. A target language that was found to give good performances is the one of BDD.

A BDD for a function of Boolean variables is a rooted graph that has one level for each Boolean variable. A node $n$ in a BDD has two children: one corresponding to the 1 value of the variable associated with $n$, indicated with $\text{child}_1(n)$, and one corresponding to the 0 value of the variable, indicated with $\text{child}_0(n)$. The leaves store either 0 or 1.

A BDD performs a Shannon expansion of the Boolean formula $f_K(X)$, so that if $X$ is the variable associated to the root level of a BDD, the formula $f_K(X)$ can be represented as $f_K(X) = X \land f_X^1(X) \lor \overline{X} \land f_X^0(X)$ where $f_X^1(X)$ (or $f_X^0(X)$) is the formula obtained by $f_K(X)$ by setting $X$ to 1 (0). Now the two disjuncts are mutually exclusive and the probability of $f_K(X)$ can be computed as $P(f_K(X)) = P(X)P(f_X^1(X)) + (1 - P(X))P(f_X^0(X))$.

Figure 1 shows the function PROB that implements the dynamic programming algorithm of [De Raedt et al., 2007] for computing the probability of a formula encoded as a BDD.

```
1: function PROB(node)
2:   Input: a BDD node
3:   Output: the probability of the Boolean function associated to the node
4: if node is a terminal then
5:   return value(node)  // value(node) is 0 or 1
6: else
7:   let X be v(node)  // v(node) is the variable associated to node
8:   P1 ← PROB(child1(node))
9:   P0 ← PROB(child0(node))
10: return P(X) · P1 + (1 - P(X)) · P0
11: end if
12: end function
```

Figure 1: Function that computes the probability of a formula encoded as a BDD.
The problem of finding explanations for a query has been in accordance with the semantics.

If we associate sets of axioms with labels of nodes and edges, each edge \((a, b)\) in the graph is labeled with the set of rules \(L(a, b)\) to which the couple \((a, b)\) belongs. Each edge \((a, b)\) in the graph is labeled with the set of rules \(L(a, b)\) to which the variable \((a, b)\) belongs. Tableau algorithms repeatedly apply a set of consistency preserving tableau expansion rules 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 present for an example couple \((C, a)\) where \(C\) and \(\neg C\) are present in the label of a node, i.e., \(\{C, \neg C\} \subseteq L(a)\).

Some expansion rules are non-deterministic, i.e., they generate a set of tableauaux. 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 \textit{unsatisfiable} as no model can be found. Otherwise, any one clash-free tableau in \(T\) represents a possible model for the concept and the algorithm returns \textit{satisfiable}. In Pellet each expansion rule updates as well a \textit{tracing function} \(\tau\), which associates sets of axioms with labels of nodes and edges. Function \(\tau\) maps couples (concept, individual) or (role, couple of individuals) to a fragment of \(K\). For example, \(\tau(C, a) (\tau(R, (a, b)))\) is the set of axioms needed to explain the addition of \(C\) (\(R\)) to the label of \(a\) (\((a, b)\)). The function \(\tau\) is initialized by assigning the values \(\{a : C\}\) and \(\{(a, b) : R\}\) to \(\tau(C, a)\) and \(\tau(R, (a, b))\) if \(a : C\) and \(a, b : R\) are in the ABox respectively. \(\tau\) is initialized as the empty set for all the other elements of its domain.

In order to find a covering set of explanations, Pellet first finds a single explanation and then iteratively removes from the theory an axiom belonging to an explanation and looks for a new explanation.

BUNDLE finds a covering set of explanations for the query using Pellet and then builds a BDD from which it computes the probability. BUNDLE, shown in Figure 3, first builds a data structure \(PMap\) that associates each DL axiom \(E\) with its probability \(p\). Then BUNDLE finds the explanations and initializes the array \(VarAxAnn\) that stores in the \(r\)th cell the pair \((Ax, Prob)\) associated to the \(r\)th Boolean variable of the BDD. BUNDLE builds the BDD with a cycle over the set of explanations: for each explanation, it builds the BDD representing the conjunction of the random variables associated to the atomic choices and then computes the disjunction of the BDDs for individual explanations. At the end, it computes the probability by calling the dynamic programming algorithm that visits the BDD.

1: \function{BUNDLE}(\(K, C\))
2: \hspace{1em} \text{Input:} \(K\) the knowledge base
3: \hspace{1em} \text{Input:} \(C\) the concept to be tested for unsatisfiability
4: \hspace{1em} \text{Output:} the probability of the unsatisfiability of \(C\) w.r.t. \(K\)
5: \hspace{1em} \text{Build Map} \(PMap\) from DL axioms to probabilities
6: \hspace{1em} \text{Find a covering set of explanations} \textit{Explanations} with Pellet
7: \hspace{1em} \text{Initialize} \(VarAxAnn\) to empty \(\triangleright VarAxAnn\): array of pairs \((Ax, Prob)\)
8: \hspace{1em} \(BDD \leftarrow BDD\text{ZERO}\)
9: \hspace{1em} \text{for all} \(\text{Explanation} \in \textit{Explanations} \text{do}\)
10: \hspace{1em} \(BDD \leftarrow BDD\text{ONE}\)
11: \hspace{1em} \hspace{1em} \text{for all} \(Ax \in \text{Explanation} \text{do}\)
12: \hspace{1em} \hspace{1em} \hspace{1em} \(p \leftarrow PMap(Ax)\)
13: \hspace{1em} \hspace{1em} \hspace{1em} \text{Scan} \(VarAxAnn\) looking for \(Ax\)
14: \hspace{1em} \hspace{1em} \hspace{1em} \text{if} \(\text{found}\) \text{then}
15: \hspace{1em} \hspace{1em} \hspace{1em} \hspace{1em} \text{Add to} \(VarAxAnn\) \text{a new pair} \((Ax, p)\)
16: \hspace{1em} \hspace{1em} \hspace{1em} \text{end if}
17: \hspace{1em} \hspace{1em} \hspace{1em} \text{Let} \(i\) be the position of \((Ax, p)\) in \(VarAxAnn\)
18: \hspace{1em} \hspace{1em} \hspace{1em} \(B \leftarrow BDD\text{GETI}(\text{VAR}(i))\)
19: \hspace{1em} \hspace{1em} \hspace{1em} \(BDD \leftarrow BDD\text{AND}(BDD, B, BDD)\)
20: \hspace{1em} \hspace{1em} \hspace{1em} \text{end for}
21: \hspace{1em} \hspace{1em} \(BDD \leftarrow BDD\text{OR}(BDD, BDD)\)
22: \hspace{1em} \hspace{1em} \text{end for}
23: \hspace{1em} \text{return} \(\text{PROB}(BDD)\) \(\triangleright VarAxAnn\) is used to compute \(P(X)\) in \text{PROB}
24: \text{end function}

Figure 3: Function \textit{BUNDLE}: computation of the probability of a concept \(C\) given the knowledge base \(K\).

To manipulate BDDs, we use \textit{JavaBDD} that is an interface to a number of underlying BDD manipulation packages. As the underlying package we use CUDD.

BUNDLE has the possibility of setting a maximum number of explanations to be found. In this case the probability that

\[\text{http://vlsi.colorado.edu/~fabio/CUDD/}\]

\[\text{http://javabdd.sourceforge.net/}\]
is computed is a lower bound of the true probability.

4.1 Computational Complexity

[Jung and Lutz, 2012] considered the problem of computing the probability of conjunctive queries to probabilistic databases in the presence of an ontology. Probabilities can occur only in the ABox while the TBox is certain. In the case where each ABox assertion is associated with a Boolean random variable independent of all the others, they prove that only very simple conjunctive queries can be answered in PTime, while most queries are #P-hard when the ontology is a DL-Lite TBox and even when the ontology is an $\mathcal{EL}$ TBox. The setting considered by [Jung and Lutz, 2012] is subsumed by DISPONTE as it is equivalent to having probabilistic axioms in the ABox only of a DISPONTE KB. So the complexity result provides a lower bound for DISPONTE.

In order to investigate the complexity of BUNDLE, we can consider separately the two problems that it solves for answering a query: finding the set of explanations and computing the probability of the query.

The computational complexity of the first problem has been studied in a number of works [Peñaloza and Sertkaya, 2009; 2010]. In [Baader et al., 2007] the authors considered a very simple DL which allows only concept intersection and showed that there can be exponentially many explanations for it. Thus in case of a more expressive DL, such as $\text{SHOIN}(\mathcal{D})$, the number of explanations may be even larger. Corollary 15 in [Peñaloza and Sertkaya, 2010] shows that the problem of finding the covering set of explanations cannot be solved in output polynomial time for a sublogic of $\text{SHOIN}(\mathcal{D})$. The problem of computing the probability of a query from the explanations can be seen as computing the probability of a SUM-OF-PRODUCTS, which was shown to be #P-hard [Rauzy et al., 2003]. Given that the input of the SUM-OF-PRODUCTS problem is of at least exponential size in the worst case, this means that computing the probability of an axiom from a $\text{SHOIN}(\mathcal{D})$ KB is intractable.

However, the algorithms that have been proposed for solving the two problems were shown to be able to work on inputs of real-world size. For example, all explanations have been found for various entailments over many real-world ontologies within a few seconds [Kalyanpur, 2006; Kalyanpur et al., 2007]. As regards the SUM-OF-PRODUCTS problem, algorithms based on BDDs were able to solve problems with hundreds of thousands of variables (see e.g. the works on inference on probabilistic logic programs [De Raedt et al., 2007; Riguzzi, 2009; Kimmig et al., 2011; Riguzzi and Swift, 2011]). Methods for weighted model counting [Sang et al., 2005; Chavira and Darwiche, 2008] can also be used to solve the SUM-OF-PRODUCTS problem. Moreover, Section 6 shows that in practice we can compute the probability of entailments on KBs of real-world size with BUNDLE, too.

5 Related Work

Many works propose approaches for combining probability and description logics. We refer to [Riguzzi et al., 2015] for the relationships of some of them with DISPONTE. We discuss here only $P-\text{SHIQ}(\mathcal{D})$ proposed in [Łukasiewicz, 2008] because it is equipped with a reasoner, PRONTO [Klinov, 2008]. $P-\text{SHIQ}(\mathcal{D})$ uses probabilistic lexicographic entailment from probabilistic default reasoning and allows both terminological probabilistic knowledge as well as assertional probabilistic knowledge about instances of concepts and roles. $P-\text{SHIQ}(\mathcal{D})$ is based on Nilsson’s probabilistic logic [Nilsson, 1986] that defines probabilistic interpretations instead of a single probability distribution over theories. Each probabilistic interpretation $\Pr$ defines a probability distribution over the set of usual interpretations $\text{Int}$. The probability of a logical formula $F$ according to $\Pr$, denoted $\Pr(F)$, is the sum of all $\Pr(I)$ such that $I \in \text{Int}$ and $I \models F$. A probabilistic KB $K$ is a set of probabilistic formulas of the form $F \geq p$. A probabilistic interpretation $\Pr$ satisfies $F \geq p$ if $\Pr(F) \geq p$. A probabilistic interpretation $\Pr$ satisfies $F 
 p$ is a tight logical consequence of $K$ if $p$ is the infimum of $\Pr(F)$ subject to all models $\Pr$ of $K$.

Nilsson’s logic allows weaker conclusions than the distribution semantics: consider a probabilistic ontology composed of the axioms $0.4 :: a : C$ and $0.5 :: b : C$ and a probabilistic KB composed of $C(a) \geq 0.4$ and $C(b) \geq 0.5$. The distribution semantics allows us to say that $P(a : C \lor b : C) = 0.7$, while with Nilsson’s logic the lowest probability such that $\Pr(C(a) \lor C(b)) \geq p$ holds is $0.5$. This is due to the fact that in the distribution semantics the probabilistic axioms are considered as independent, which allows to make stronger conclusions. However, this does not restrict expressiveness as one can specify any joint probability distribution over the logical ground atoms interpreted as Boolean variables, possibly introducing new atoms if needed.

A different approach is shown in [Zese et al., 2013; 2014] where we presented TRILL, a tableau reasoner written in Prolog able to compute the probability of queries w.r.t. KBs that follow the DISPONTE semantics. Algorithms written in procedural languages have to implement a search strategy to explore the entire search space, while by exploiting Prolog’s backtracking facilities we can leave the management of the non-determinism to the Prolog language. TRILL is also available as a web application at http://trill.lamping.unife.it.

6 Experiments

In [Riguzzi et al., 2015] we evaluate the performances of BUNDLE with several experiments. We report here the most significant ones, performed using the methodology proposed in [Klinov and Parsia, 2011] for evaluating PRONTO. More tests can be found in [Riguzzi et al., 2015]. The experiments have been performed on Linux machines with a 3.10 GHz Intel Xeon E5-2687W with 2GB memory allotted to Java.

In [Klinov and Parsia, 2011] the authors considered three different datasets: an extract from the Cell ontology, an extract from the NCI Thesaurus and an extract from the Teleost anatomy. The Cell ontology represents cell types of the prokaryotic, fungal, and eukaryotic organisms. The NCI ontology is an extract from the NCI Thesaurus that describes human anatomy. The Teleost_anatomy

---

3 http://cellontology.org/
4 http://ncit.nci.nih.gov/
5 http://phenoscape.org/wiki/Teleost_Anatomy_Ontology
for short) ontology is a multi-species anatomy ontology for teleost fishes.

For each of these KBs, they created four versions of increasing size containing 250, 500, 750 and 1,000 new probabilistic conditional constraints of $P$-SHIQ for short) ontology is a multi-species anatomy ontology for teleost fishes.

Table 1: Average execution time for the queries to the Cell, TST and NCI KBs and number of queries terminated with a time-out.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size of the Probabilistic TBox</th>
<th>0</th>
<th>250</th>
<th>500</th>
<th>750</th>
<th>1,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>time (s)</td>
<td>0.76</td>
<td>2.84</td>
<td>3.88</td>
<td>3.94</td>
<td>4.53</td>
</tr>
<tr>
<td></td>
<td>time-out</td>
<td>0</td>
<td>28</td>
<td>39</td>
<td>50</td>
<td>55</td>
</tr>
<tr>
<td>TST</td>
<td>time (s)</td>
<td>2.11</td>
<td>8.87</td>
<td>31.80</td>
<td>33.82</td>
<td>36.33</td>
</tr>
<tr>
<td></td>
<td>time-out</td>
<td>0</td>
<td>7</td>
<td>32</td>
<td>32</td>
<td>44</td>
</tr>
<tr>
<td>NCI</td>
<td>time (s)</td>
<td>3.02</td>
<td>11.37</td>
<td>11.37</td>
<td>16.37</td>
<td>24.90</td>
</tr>
<tr>
<td></td>
<td>time-out</td>
<td>0</td>
<td>1</td>
<td>24</td>
<td>23</td>
<td>36</td>
</tr>
</tbody>
</table>

In Table 1 we report, for each version of the datasets, the number of queries that terminated with a time-out for BUNDLE and the average execution time computed on the queries that did not end with a time-out. The column corresponding to the dimension 0 of the probabilistic TBox show the average execution time for executing queries w.r.t. the non-probabilistic version of the KBs.

These tests show that BUNDLE is able to scale to ontologies of realistic size. Moreover, BUNDLE answers most queries in a few seconds. However, some queries have a very high complexity that causes BUNDLE to time-out, confirming the complexity results. In these cases, since the time-out is reached during the computation of the explanations, limiting the number of explanations can provide a lower bound on the probability that becomes tighter as more explanations are allowed.

7 Conclusions

BUNDLE computes the probability of queries from uncertain DL knowledge bases following the DISPONTE semantics. BUNDLE is available for download from http://sites.unife.it/ml/bundle together with the datasets used in the experiments. BUNDLE has been tested on ontologies of increasing complexity in various domains. BUNDLE is also used in the system EDGE for learning the parameters and LEAP for learning the structure [Riguzzi et al., 2014] of DISPONTE ontologies.

In the future we plan to investigate approaches for improving the scalability of BUNDLE, and of the systems based on

References


