Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

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 Descrition Logics (DLs) [Lukasiewicz 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 $\mathcal{SHOIN}(\mathbf{D})$ 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 $\mathcal{SHOIN}(\mathbf{D})$.

Let $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{I}$ be sets of *atomic concepts*, *atomic roles* and *individuals*, respectively. A *role* is either an atomic role $R \in \mathbf{R}$ or the inverse R^- of an atomic role $R \in \mathbf{R}$. We use $\mathbf{R}^-$ to denote the set of all inverses of roles in $\mathbf{R}$. *Concepts* are defined by induction as follows. Each $A \in \mathbf{A}$, $\perp$ and $\top$ are concepts and if $a \in \mathbf{I}$, then $\{a\}$ is a concept called a *nominal*. If C, C_1 and C_2 are concepts and $R \in \mathbf{R} \cup \mathbf{R}^-$, 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 $\geq nR$ and $\leq nR$ 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* $Trans(R)$, where $R \in \mathbf{R}$, and *role inclusion axioms* $R \sqsubseteq S$, where $R, S \in \mathbf{R} \cup \mathbf{R}^-$. 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, $R \in \mathbf{R}$ and $a, b \in \mathbf{I}$. 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 $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, 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 $R \in \mathbf{R}$. The mapping $\cdot^\mathcal{I}$ is extended to all concepts (where $R^\mathcal{I}(x) = \{y|(x, y) \in R^\mathcal{I}\}$ and $\#X$ denotes the cardinality of the set X) as:

$$
\begin{aligned}
(R^-)^\mathcal{I} &= \{(y, x)|(x, y) \in R^\mathcal{I}\} \\
\top^\mathcal{I} &= \Delta^\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset \\
\{a\}^\mathcal{I} &= \{a^\mathcal{I}\} \\
(C_1 \sqcap C_2)^\mathcal{I} &= C_1^\mathcal{I} \cap C_2^\mathcal{I} \\
(C_1 \sqcup C_2)^\mathcal{I} &= C_1^\mathcal{I} \cup C_2^\mathcal{I} \\
(\neg C)^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I} \\
(\forall R.C)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} | R^\mathcal{I}(x) \subseteq C^\mathcal{I}\} \\
(\exists R.C)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} | R^\mathcal{I}(x) \cap C^\mathcal{I} \neq \emptyset\} \\
(\geq nR)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} | \#R^\mathcal{I}(x) \geq n\} \\
(\leq nR)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} | \#R^\mathcal{I}(x) \leq n\}
\end{aligned}
$$

$\mathcal{SHOIN}(\mathbf{D})$ 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, $\mathcal{SHOIN}(\mathbf{D})$ 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

$$p :: E \tag{1}$$

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 $0.9 :: tweety : Flies$. 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 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., $(E_i, k) \in \kappa, (E_i, m) \in \kappa \Rightarrow k = m$ (only one decision for each formula). The probability of 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 composite choice that contains an atomic choice (E_i, k) for every probabilistic axiom of the theory. A selection σ identifies a theory w_σ called a *world* in this way: $w_\sigma = \{E_i|(E_i, 1) \in \sigma\}$. 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:

$$
\begin{aligned}
P(Q) &= \sum_{w\in\mathcal{W}_\mathcal{K}} P(Q, w) && (2) \\
&= \sum_{w\in\mathcal{W}_\mathcal{K}} P(Q|w)P(w) && (3) \\
&= \sum_{w\in\mathcal{W}_\mathcal{K}: w\models Q} P(w) && (4)
\end{aligned}
$$

Example 1 *The following KB is inspired by the ontology called* `people+pets` *proposed in [Patel-Schneider* et al.*, 2003]:*

$$\exists hasAnimal.Pet \sqsubseteq NatureLover$$
$$(kevin, fluffy) : hasAnimal$$
$$(kevin, tom) : hasAnimal$$
$$0.4 \;::\; fluffy : Cat \quad (5)$$
$$0.3 \;::\; tom : Cat \quad (6)$$
$$0.6 \;::\; Cat \sqsubseteq Pet \quad (7)$$

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 $Q = kevin : NatureLover$ is true in three of them, those corresponding to the following selections:

$$\{((5), 1), ((6), 0), ((7), 1)\}$$
$$\{((5), 0), ((6), 1), ((7), 1)\}$$
$$\{((5), 1), ((6), 1), ((7), 1)\}$$

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 κ is an *explanation* for a query Q if Q is entailed by every world of ω_κ, where $\omega_\kappa = \{w_\sigma | \sigma \in \mathcal{S}_\mathcal{K}, \sigma \supseteq \kappa\}$ is the set of worlds compatible with κ. 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 $w \in \mathcal{W}_\mathcal{K}$ in which Q is entailed is such that $w \in \omega_K$.

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 $\mathbf{X} = \{X_i | (E_i, k) \in \kappa, \kappa \in K\}$ are pairwise independent and the probability that X_i takes value 1 is p_i, the probability associated with the ith axiom.

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

$$f_K(\mathbf{X}) = \bigvee_{\kappa \in K} \bigwedge_{(E_i,1) \in \kappa} X_i \bigwedge_{(E_i,0) \in \kappa} \overline{X_i} \qquad (8)$$

Thus we can compute the probability of Q by computing the probability that $f_K(\mathbf{X})$ takes value 1. This formula is in Disjunctive Normal Form (DNF) but we can't compute $P(f_K(\mathbf{X}))$ 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 $f_K(\mathbf{X})$ [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 $child_1(n)$, and one corresponding to the 0 value of the variable, indicated with $child_0(n)$. The leaves store either 0 or 1.

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

function PROB($node$)
  Input: a BDD node
  Output: the probability of the Boolean function associated to the node
  if $node$ is a terminal **then**
    return $value(node)$ ▷ $value(node)$ is 0 or 1
  else
    let X be $v(node)$ ▷ $v(node)$ is the variable associated to $node$
    $P_1 \leftarrow$PROB($child_1(node)$)
    $P_0 \leftarrow$PROB($child_0(node)$)
    return $P(X) \cdot P_1 + (1 - P(X)) \cdot P_0$
  end if
end function

Figure 1: Function that computes the probability of a formula encoded as a BDD.

BDDs can be built by combining simpler BDDs using Boolean operators. While building BDDs, simplification operations can be applied that delete or merge nodes. Merging is performed when the diagram contains two identical subdiagrams, while deletion is performed when both arcs from a node point to the same node. In this way a reduced BDD is obtained, often with a much smaller number of nodes with respect to the original BDD. The size of the reduced BDD depends on the order of the variables: finding an optimal order is an NP-complete problem [Bollig and Wegener, 1996] and several heuristic techniques are used in practice by sophisti-

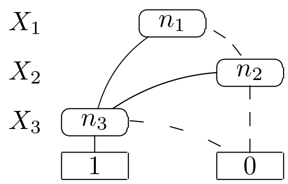

Figure 2: BDD for Example 1.

cated software packages such as CUDD[1]. Alternative methods involve learning variable order from examples [Grumberg *et al.*, 2003].

Example 2 *Let us consider the KB of example 1. A covering set of explanations for the query axiom* $Q = kevin : NatureLover$ *is* $K = \{\kappa_1, \kappa_2\}$ *where* $\kappa_1 = \{((5), 1), ((7), 1)\}$ *and* $\kappa_2 = \{((6), 1), ((7), 1)\}$.

If we associate the random variables X_1 *with* (5), X_2 *with* (6) *and* X_3 *with* (7), *the BDD associated to the set K of explanations is shown in Figure 2. By applying algorithm in Figure 1 we get*

$$\begin{aligned}\textsc{Prob}(n_3) &= 0.6 \cdot 1 + 0.4 \cdot 0 = 0.6\\ \textsc{Prob}(n_2) &= 0.4 \cdot 0.6 + 0.6 \cdot 0 = 0.24\\ \textsc{Prob}(n_1) &= 0.3 \cdot 0.6 + 0.7 \cdot 0.24 = 0.348\end{aligned}$$

in accordance with the semantics.

The problem of finding explanations for a query has been investigated by various authors [Schlobach and Cornet, 2003; Kalyanpur, 2006; Kalyanpur *et al.*, 2007; Horridge *et al.*, 2009]. For example, Pellet finds explanations by using a tableau algorithm [Kalyanpur, 2006]. A *tableau* is a graph where each node represents an individual a and is labeled with the set of concepts $\mathcal{L}(a)$ to which a belongs. Each edge $\langle a, b\rangle$ in the graph is labeled with the set of roles $\mathcal{L}(\langle a, b\rangle)$ to which the couple (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 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 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 tableau in T represents a possible model for the concept and the algorithm returns *satisfiable*. In Pellet each expansion rule updates as well a *tracing function* τ, which associates sets of axioms with labels of nodes and edges. Function τ maps couples (concept, individual) or (role, couple of individuals) to a fragment of $\mathcal{K}$. For example, $\tau(C, a)$ $(\tau(R, \langle a, b\rangle))$ is the set of axioms needed to explain the addition of C (R) to the label of a $(\langle a, b\rangle)$. The function τ is initialized by assigning the values $\{a : C\}$ and $\{(a, b) : R\}$ to $\tau(C, a)$ and $\tau(R, \langle a, b\rangle)$ if $a : C$ and $(a, b) : R$ are in the ABox respectively. τ 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 ith cell the pair $(Axiom, Prob)$ associated to the ith 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.

```
function BUNDLE(K, C)
    Input: K (the knowledge base)
    Input: C (the concept to be tested for unsatisfiability)
    Output: the probability of the unsatisfiability of C w.r.t. K
    Build Map PMap from DL axioms to probabilities
    Find a covering set of explanations Explanations with Pellet
    Initialize VarAxAnn to empty    ▷ VarAxAnn: array of pairs (Axiom, Prob)
    BDD ←BDDZero
    for all Explanation ∈ Explanations do
        BDDE ←BDDOne
        for all Ax ∈ Explanation do
            p ← PMap(Ax)
            Scan VarAxAnn looking for Ax
            if !found then
                Add to VarAxAnn a new pair (Ax, p)
            end if
            Let i be the position of (Ax, p) in VarAxAnn
            B ← BDDGetIthVar(i)
            BDDE ←BDDAnd(BDDE,B)
        end for
        BDD ←BDDOr(BDD,BDDE)
    end for
    return Prob(BDD)    ▷ VarAxAnn is used to compute P(X) in Prob
end function
```

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

To manipulate BDDs, we use JavaBDD[2] 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

[1] http://vlsi.colorado.edu/~fabio/CUDD/

[2] 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 $\mathcal{ELI}$ 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 $\mathcal{SHOIN}(\mathbf{D})$, 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 $\mathcal{SHOIN}(\mathbf{D})$. 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 $\mathcal{SHOIN}(\mathbf{D})$ 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-$\mathcal{SHIQ}(\mathbf{D})$ proposed in [Lukasiewicz, 2008] because it is equipped with a reasoner, PRONTO [Klinov, 2008]. P-$\mathcal{SHIQ}(\mathbf{D})$ 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-$\mathcal{SHIQ}(\mathbf{D})$ 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 Int. The probability of a logical formula F according to Pr, denoted $Pr(F)$, is the sum of all $Pr(I)$ such that $I \in Int$ 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 $F \geq p$ iff $Pr(F) \geq p$. $Pr(F) \geq p$ is a tight logical consequence of K iff 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 $P(a : C \vee b : C) = 0.7$, while with Nilsson's logic the lowest p such that $Pr(C(a) \vee C(b)) \geq p$ 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 random 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[3] ontology, an extract from the NCI Thesaurus[4] and an extract from the Teleost_anatomy[5] ontology. 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 (TST

[3]http://cellontology.org/

[4]http://ncit.nci.nih.gov/

[5]http://phenoscape.org/wiki/Teleost_Anatomy_Ontology

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

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

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-$\mathcal{SHIQ}(\mathbf{D})$ which are added to the publicly available non-probabilistic version of each ontology. Conditional constraints are of the form $(D|C)[l, u]$ and informally mean "generally, if an object belongs to C, then it belongs to D with a probability in the interval $[l, u]$". We converted these KBs into DISPONTE by replacing the constraint $(D|C)[l, u]$ with the axiom $u :: C \sqsubseteq D$ and we created a set of 100 different random subclass queries for each KB. For generating the queries we built the hierarchy of each KB and we randomly selected two classes connected in the hierarchy, so that each query has at least one explanation. We impose a time limit of 5 minutes for BUNDLE to answer each query.

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 it, by exploiting modern computing infrastructures with many cores and large main memories. Moreover, we plan to develop web interfaces to these systems along the lines of the one developed for TRILL.

References

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

[Baader *et al.*, 2007] F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic EL^{+}. In *Annual German Conference on AI*, volume 4667 of *LNCS*, pages 52–67. Springer, 2007.

[Baader *et al.*, 2008] F. Baader, I. Horrocks, and U. Sattler. Description logics. In *Handbook of knowledge representation*, chapter 3, pages 135–179. Elsevier, 2008.

[Bellodi and Riguzzi, 2012] E. Bellodi and F. Riguzzi. Experimentation of an expectation maximization algorithm for probabilistic logic programs. *Intelligenza Artificiale*, 8(1):3–18, 2012.

[Bellodi and Riguzzi, 2013] E. Bellodi and F. Riguzzi. Expectation Maximization over binary decision diagrams for probabilistic logic programs. *Intelligent Data Analysis*, 17(2):343–363, 2013.

[Bellodi and Riguzzi, 2015] E. Bellodi and F. Riguzzi. Structure learning of probabilistic logic programs by searching the clause space. *Theory and Practice of Logic Programming*, 15(2):169–212, 2015.

[Bollig and Wegener, 1996] B. Bollig and I. Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on Computers*, 45(9):993–1002, 1996.

[Chavira and Darwiche, 2008] M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.

[Darwiche and Marquis, 2002] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[De Raedt *et al.*, 2007] L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI 2007*, pages 2462–2467, 2007.

[Grumberg *et al.*, 2003] O. Grumberg, S. Livne, and S. Markovitch. Learning to order BDD variables in verification. *Journal of Artificial Intelligence Research*, 18:83–116, 2003.

[Horridge *et al.*, 2009] M. Horridge, B. Parsia, and U. Sattler. Explaining inconsistencies in OWL ontologies. In *SUM 2009*, volume 5785 of *LNCS*, pages 124–137. Springer, 2009.

[Jung and Lutz, 2012] J. C. Jung and C. Lutz. Ontology-based access to probabilistic data with OWL QL. In *ISWC 2012*, volume 7649 of *LNCS*, pages 182–197. Springer, 2012.

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

[Kalyanpur, 2006] A. Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, The Graduate School of the University of Maryland, 2006.

[Kimmig *et al.*, 2011] A. Kimmig, B. Demoen, L. De Raedt, V. Santos Costa, and R. Rocha. On the implementation of the probabilistic logic programming language ProbLog. *Theory and Practice of Logic Programming*, 11(2-3):235–262, 2011.

[Klinov and Parsia, 2011] P. Klinov and B. Parsia. A hybrid method for probabilistic satisfiability. In *CADE-23*, volume 6803 of *LNCS*, pages 354–368. Springer, 2011.

[Klinov, 2008] P. Klinov. Pronto: A non-monotonic probabilistic description logic reasoner. In *ESWC 2008*, volume 5021 of *LNCS*, pages 822–826. Springer, 2008.

[Lukasiewicz and Straccia, 2008] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308, 2008.

[Lukasiewicz, 2008] T. Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, 2008.

[Nilsson, 1986] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[Patel-Schneider *et al.*, 2003] F. Patel-Schneider, P, I. Horrocks, and S. Bechhofer. Tutorial on OWL, 2003.

[Peñaloza and Sertkaya, 2009] R. Peñaloza and B. Sertkaya. Axiom pinpointing is hard. In *DL 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[Peñaloza and Sertkaya, 2010] R. Peñaloza and B. Sertkaya. Complexity of axiom pinpointing in the DL-Lite family of description logics. In *ECAI 2010*, pages 29–34. IOS Press, 2010.

[Poole, 1997] D. Poole. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56, 1997.

[Rauzy *et al.*, 2003] A. Rauzy, E. Châtelet, Y. Dutuit, and C. Bérenguer. A practical comparison of methods to assess sum-of-products. *Reliability Engineering and System Safety*, 79(1):33–42, 2003.

[Riguzzi and Swift, 2011] F. Riguzzi and T. Swift. The PITA system: Tabling and answer subsumption for reasoning under uncertainty. *Theory and Practice of Logic Programming*, 11(Special Issue 4–5):433–449, 2011.

[Riguzzi *et al.*, 2012] F. Riguzzi, E. Lamma, E. Bellodi, and R. Zese. Epistemic and statistical probabilistic ontologies. In *URSW 2012*, volume 900 of *CEUR Workshop Proceedings*, pages 3–14. Sun SITE Central Europe, 2012.

[Riguzzi *et al.*, 2013] F. Riguzzi, E. Bellodi, E. Lamma, and R. Zese. BUNDLE: A reasoner for probabilistic ontologies. In *RR 2013*, volume 7994 of *LNCS*, pages 183–197. Springer, 2013.

[Riguzzi *et al.*, 2014] F. Riguzzi, E. Bellodi, E. Lamma, R. Zese, and G. Cota. Learning probabilistic description logics. In *URSW III*, volume 8816 of *LNCS*, pages 63–78. Springer, 2014.

[Riguzzi *et al.*, 2015] F. Riguzzi, E. Bellodi, E. Lamma, and R. Zese. Probabilistic description logics under the distribution semantics. *Semantic Web Journal*, 2015. DOI: 10.3233/SW-140154.

[Riguzzi, 2009] F. Riguzzi. Extended semantics and inference for the Independent Choice Logic. *Logic Journal of the IGPL*, 17(6):589–629, 2009.

[Sang *et al.*, 2005] T. Sang, P. Beame, and H. A. Kautz. Performing bayesian inference by weighted model counting. In *AAAI 2005 and IAAI 2005*, pages 475–482. AAAI Press / The MIT Press, 2005.

[Sato and Kameya, 2001] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[Sato, 1995] T. Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP 1995*, pages 715–729. MIT Press, 1995.

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

[Sirin *et al.*, 2007] E. Sirin, B. Parsia, B. Cuenca-Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[Straccia, 2008] U. Straccia. Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In *RW 2008*, volume 5224 of *LNCS*, pages 54–103. Springer, 2008.

[Vennekens *et al.*, 2004] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *ICLP 2004*, volume 3131 of *LNCS*, pages 195–209. Springer, 2004.

[Zese *et al.*, 2013] R. Zese, E. Bellodi, E. Lamma, and F. Riguzzi. A description logics tableau reasoner in prolog. In *CILC*, volume 1068 of *CEUR Workshop Proceedings*, pages 33–47. CEUR-WS.org, 2013.

[Zese *et al.*, 2014] Riccardo Zese, Elena Bellodi, Evelina Lamma, Fabrizio Riguzzi, and Fabiano Aguiari. Semantics and inference for probabilistic description logics. In *URSW III*, volume 8816 of *LNCS*, pages 79–99. Springer, 2014.