# BUNDLE: A Reasoner for Probabilistic Ontologies

Fabrizio Riguzzi[1], Elena Bellodi[2], Evelina Lamma[2], and 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.** Representing uncertain information is very important for modeling real world domains. Recently, the DISPONTE semantics has been proposed for probabilistic description logics. In DISPONTE, the axioms of a knowledge base can be annotated with a set of variables and a real number between 0 and 1. This real number represents the probability of each version of the axiom in which the specified variables are instantiated. In this paper we present the algorithm BUNDLE for computing the probability of queries from DISPONTE knowledge bases that follow the $\mathcal{ALC}$ semantics. BUNDLE exploits an underlying DL reasoner, such as Pellet, that is able to return explanations for queries. The explanations are encoded in a Binary Decision Diagram from which the probability of the query is computed. The experiments performed by applying BUNDLE to probabilistic knowledge bases show that it can handle ontologies of realistic size and is competitive with the system PRONTO for the probabilistic description logic P-$\mathcal{SHIQ}$(D).

## 1 Introduction

Representing uncertain information is of foremost importance in order to effectively model real world domains. This has been fully recognized in the field of Description Logics (DLs) where many proposals have appeared on the combination of probability theory and DLs [17].

In general, the integration of probability with logic has been much studied lately, with many different proposals [30]. In the field of Logic Programming, the distribution semantics [25] has emerged as one of the most effective approaches.

In [24] we applied this approach to DLs obtaining DISPONTE for "DIstribution Semantics for Probabilistic ONTologiEs" (Spanish for "get ready"). The idea is to annotate axioms of a theory with a probability and assume that each axiom is independent of the others. DISPONTE allows two types of probabilistic axiom: an epistemic type, that represents a degree of belief in the axiom as a whole, and a statistical type, that considers the populations to which the axiom is applied. Statistical probabilistic axioms allow the representation of partial concept overlapping and knowledge about random individuals of populations.

The DISPONTE semantics differs from previous proposals because it extends the language minimally and provides a unified framework for representing different types of probabilistic knowledge, from an epistemic to a statistical type. Moreover, it allows to seamlessly represents probabilistic assertional and terminological knowledge.

In this paper we present the algorithm BUNDLE for "Binary decision diagrams for Uncertain reasoNing on Description Logic thEories", that performs inference over DISPONTE DLs that follow the $\mathcal{ALC}$ semantics. BUNDLE exploits an underlying reasoner such as Pellet [29] that returns explanations for queries. BUNDLE uses the inference techniques developed for probabilistic logic programs under the distribution semantics, in particular Binary Decision Diagrams, for computing the probability of queries from a covering set of explanations. We applied BUNDLE to various real world datasets and we found that it is able to handle domains of significant size. Moreover, we compared it with Pronto [14], a system for inference in the probabilistic description logic P-$\mathcal{SHIQ}$(D). The results show that BUNDLE is faster than PRONTO for knowledge bases of the same size. The paper is organized as follows. Section 2 introduces Description Logics and Section 3 illustrates DISPONTE while Section 4 describes BUNDLE. Section 5 discusses related work and Section 6 shows the results of experiments with BUNDLE. 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 [1,2] 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 sub-language of OWL 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 $\mathcal{ALC}$ [28], since the version of BUNDLE here presented considers this DL language.

Let $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{I}$ be sets of *atomic concepts*, *roles* and *individuals*, respectively.

*Concepts* are defined by induction as follows. Each $A \in \mathbf{A}$ is a concept and $\bot$ and $\top$ are concepts. If $C$, $C1$ and $C2$ are concepts and $R \in \mathbf{R}$, then $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$ and $\neg C$ are concepts, as well as $\exists R.C$ and $\forall R.C$.

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* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

A knowledge base $\mathcal{K}$ 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 semantics of DLs can be given equivalently by transforming a DL knowledge base into a predicate logic theory and then using the model-theoretic semantics of the resulting theory. A translation of $\mathcal{ALC}$ into First Order Logic is given in the following. The translation uses two functions $\pi_x$ and $\pi_y$ that map concept expressions to logical formulas, where $\pi_x$ is given by

$$\begin{aligned}
\pi_x(A) &= A(x) \\
\pi_x(\neg C) &= \neg \pi_x(C) \\
\pi_x(C \sqcap D) &= \pi_x(C) \wedge \pi_x(D) \\
\pi_x(C \sqcup D) &= \pi_x(C) \vee \pi_x(D) \\
\pi_x(\exists R.C) &= \exists y.R(x,y) \wedge \pi_y(C) \\
\pi_x(\forall R.C) &= \forall y.R(x,y) \rightarrow \pi_y(C)
\end{aligned}$$

and $\pi_y$ is obtained from $\pi_x$ by replacing $x$ with $y$ and vice-versa.

Table 1 shows the translation of each axiom of $\mathcal{ALC}$ knowledge bases.

| Axiom | Translation |
|---|---|
| $C \sqsubseteq D$ | $\forall v_1.\pi_x(C) \rightarrow \pi_x(D)$ |
| $a : C$ | $\pi_x(C)(a)$ |
| $(a,b) : R$ | $R(a,b)$ |
| $a = b$ | $a = b$ |
| $a \neq b$ | $a \neq b$ |

**Table 1.** Translation of $\mathcal{ALC}$ axioms into predicate logic.

A query over a knowledge base is usually an axiom for which we want to test the entailment from the knowledge base. 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 unsatisfiability of the concept $C \sqcap \neg D$.

A DL enjoys the *finite model property* [6] if a knowledge base that has an arbitrary, possibly infinite model, also has a finite one. For example, $\mathcal{ALC}$ has the finite model property while $\mathcal{SHOIN}(D)$, the basis of OWL DL, doesn't. If the DL enjoys the *finite model property*, the domain $\Delta^{\mathcal{I}}$ can be assumed finite and so also $\mathbf{I}$.

Given a predicate logic formula $F$, a *substitution* $\theta$ is a set of pairs $x/a$ where $x$ is a variable universally quantified in the outermost quantifier in $F$ and $a \in \mathbf{I}$. The application of $\theta$ to $F$, indicated by $F\theta$, is called an *instantiation of* $F$ and is obtained by replacing $x$ with $a$ in $F$ and by removing $x$ from the external quantification for every pair $x/a$ in $\theta$.

## 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 [25,26], Independent Choice Logic [20], Logic Programs with Annotated Disjunctions (LPADs) [32] and ProbLog [8].

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. The joint distribution of the query $Q$ and the worlds is considered: $P(Q, w)$. This can be expressed with the probability chain rule as $P(Q|w)P(w)$. Given a world $w$, the probability of a query is 1 if the query is entailed and 0 otherwise, so $P(Q|w) = 1$ if $w \models Q$ and 0 otherwise. We can thus obtain the probability of the query by marginalizing the joint distribution as $P(Q) = \sum_w P(Q, w) = \sum_w P(Q|w)P(w) = \sum_{w:w\models Q} P(w)$. The distribution semantics was applied successfully in many domains [8,26,3] and various inference and learning algorithms are available for it [13,22,4].

In DISPONTE, a *probabilistic knowledge base* $\mathcal{K}$ is a set of certain and probabilistic axioms. *Certain axioms* take the form of regular DL axioms. *Probabilistic axioms* take the form

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

where $p$ is a real number in $[0, 1]$, $Var$ can be empty or the symbol $x$ and it specifies the "type" of probability and $E$ is a DL axiom.

In order to give a semantics to such probabilistic knowledge bases, we consider their translation into predicate logic. The idea of DISPONTE is to associate independent Boolean random variables to (instantiations of) the formulas in predicate logic that are obtained from the translation of the axioms. By assigning values to every random variable we obtain a *world*, the set of predicate logic formulas whose random variable is assigned to 1.

In particular, we fix a set of individuals $\mathbf{I}$ and we assume it is finite. We give here the description of DISPONTE for description logics that enjoy the finite model property. For those that do not, the semantics has to be given slightly differently, following the semantics for probabilistic logic programming languages that include function symbols given in [25,26,20].

To obtain a world $w$, we include every formula from a certain axiom. For each probabilistic axiom, we generate all the substitutions for the variables of the equivalent predicate logic formula that are indicated in the subscript. The variables are replaced with elements of $\mathbf{I}$. For each instantiated formula we decide whether or not to include it in $w$. In this way we obtain a predicate logic theory which can be assigned a model-theoretic semantics. A query is entailed by a world if it is true in every model of the world.

If $Var$ is empty, the probability $p$ can be interpreted as an *epistemic probability*, i.e., as the degree of our belief in axiom $E$, while if $Var$ is equal to $x$, $p$ can be interpreted as a *statistical probability*, i.e., as information regarding random individuals from the domain.

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

$$p :: C \sqsubseteq D \tag{2}$$

represents the fact that we believe in the truth of $C \sqsubseteq D$ with probability $p$. $Var$ can take value $x$ in a probabilistic concept inclusion axiom of the form

$$p ::_x C \sqsubseteq D \tag{3}$$

This axiom means that a random individual of class $C$ has probability $p$ of belonging to $D$, thus representing the statistical information that a fraction $p$ of the individuals of $C$ belongs to $D$. In this way, the overlap between $C$ and $D$ is quantified by the probability $p$. We can highlight the difference between the two axioms by observing that, if two individuals belong to class $C$, the probability that they both belong to $D$ according to (2) is $p$, since $p$ represents the truth of the formula as a whole, while according to (3) is $p \cdot p$, since each individual of $C$ has probability $p$ of belonging to class $D$ and the two events are independent.

The statement that with 90% probability birds fly for example can be expressed as $0.9 :: Bird \sqsubseteq Flies$. If we want to compute the probability of flying of a bird, this axiom and $0.9 ::_x Bird \sqsubseteq Flies$ give the same result. For two birds, the probability of both flying will be $0.9 \cdot 0.9 = 0.81$ with the second axiom and still 0.9 with first.

Let us now give the formal definition of DISPONTE. An *atomic choice* is a triple $(F_i, \theta_j, k)$ where $F_i$ is the formula obtained by translating the $i$th probabilistic axiom in predicate logic following Table 1, $\theta_j$ is a substitution and $k \in \{0, 1\}$. $k$ indicates whether $F_i\theta_j$ is chosen to be included in a world ($k = 1$) or not ($k = 0$). $\theta_j$ instantiates the variable indicated in the $Var$ subscript of the $i$th probabilistic axiom.

A *composite choice* $\kappa$ is a consistent set of atomic choices, i.e., $(F_i, \theta_j, k) \in \kappa, (F_i, \theta_j, m) \in \kappa \Rightarrow k = m$ (only one decision for each formula). The probability of composite choice $\kappa$ is $P(\kappa) = \prod_{(F_i,\theta_j,1)\in\kappa} p_i \prod_{(F_i,\theta_j,0)\in\kappa}(1 - p_i)$, where $p_i$ is the probability associated to axiom $F_i$. A *selection* $\sigma$ is a composite choice that contains an atomic choice $(F_i, \theta_j, k)$ for every instantiation $F_i\theta_j$ of every probabilistic axiom of the theory. Let us indicate with $\mathcal{S}_\mathcal{K}$ the set of all selections. Since the set of individuals is finite, each selection is finite and so is $\mathcal{S}_\mathcal{K}$. A selection $\sigma$ identifies a theory $w_\sigma$ called a *world* in this way: $w_\sigma = \mathcal{C} \cup \{F_i\theta_j | (F_i, \theta_j, 1) \in \sigma\}$ where $\mathcal{C}$ is the set of the translations in predicate logic of certain axioms. Let us indicate with $\mathcal{W}_\mathcal{K}$ the set of all worlds. The probability of a world $w_\sigma$ is $P(w_\sigma) = P(\sigma) = \prod_{(F_i,\theta_j,1)\in\sigma} p_i \prod_{(F_i,\theta_j,0)\in\sigma}(1 - p_i)$. $P(w_\sigma)$ 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:

$$P(Q) = \sum_{w \in \mathcal{W}_\mathcal{K}} P(Q, w) = \sum_{w \in \mathcal{W}_\mathcal{K}} P(Q|w)p(w) = \sum_{w \in \mathcal{W}_\mathcal{K} : w \models Q} P(w) \tag{4}$$

However, 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 instantiated probabilistic axioms, a different approach is followed in which explanations for queries are found.

A composite choice $\kappa$ is an *explanation* for a query $Q$ if $Q$ is entailed by every world of $\omega_\kappa$, where $\omega_\kappa = \{w_\sigma | \sigma \in \mathcal{S}_\mathcal{K}, \sigma \supseteq \kappa\}$ 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 $w \in \mathcal{W}_\mathcal{K}$ in which $Q$ is entailed is such that $w \in \omega_K$.

We can associate with every instantiated formula $F_i\theta_j$ a Boolean random variable $X_{ij}$. An atomic choice $(F_i, \theta_j, 1)$ then corresponds to $X_{ij}$ assuming value 1 and $(F_i, \theta_j, 0)$ corresponds to $X_{ij}$ assuming value 0. The variables $\mathbf{X} = \{X_{ij} | F_i \in K, \theta_j \text{ is a substitution}\}$ are pairwise independent and the probability that $X_{ij}$ 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 with an assignment of $\mathbf{X}$ for which the following Boolean function takes value 1:

$$f_K(\mathbf{X}) = \bigvee_{\kappa \in K} \bigwedge_{(F_i, \theta_j, 1) \in \kappa} X_{ij} \bigwedge_{(F_i, \theta_j, 0) \in \kappa} \overline{X_{ij}} \tag{5}$$

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. The problem of computing $P(f_K(\mathbf{X}))$ for a DNF was shown to be #P-hard (see e.g. [21]). The class #P [31] describes counting problems associated with decision problems in NP. More formally, #P is the class of function problems of the form "compute $f(x)$", where $f$ is the number of accepting paths of a nondeterministic Turing machine running in polynomial time.

To solve the problem, we can apply *knowledge compilation* to the propositional formula $f_K(\mathbf{X})$ [7] in order to translate it to a target language that allows to compute the probability in polynomial time. A target language that was found to give good performances is the one of Binary Decision Diagrams (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. Given values for all the variables, a BDD can be used to compute the value of the formula by traversing the graph starting from the root, following the edges corresponding to the variables values and returning the value associated to the leaf that is reached.

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 $P(f_K(\mathbf{X})) = P(X)P(f_K^X(\mathbf{X})) + (1 - P(X))P(f_K^{\overline{X}}(\mathbf{X}))$. In other words, BDDs make the explanations mutually incompatible. Figure 1 shows the function PROB that implements the dynamic programming algorithm of [8] for computing 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 sub-diagrams, 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 [5] and several heuristic techniques are used in practice by highly efficient software packages such as CUDD. Alternative methods involve learning variable order from examples [9].

```
 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:         P_1 ←PROB(child_1(node))
 9:         P_0 ←PROB(child_0(node))
10:         return P(X) · P_1 + (1 − P(X)) · P_0
11:     end if
12: end function
```

**Fig. 1.** Function that computes the probability of a formula encoded as a BDD.

Let us discuss some examples.

*Example 1.* The following knowledge base is inspired by the ontology called `people+pets` proposed in [19]:

$$\exists hasAnimal.Pet \sqsubseteq NatureLover$$

$$(kevin, \textit{fluffy}) : hasAnimal$$
$$(kevin, tom) : hasAnimal$$
$$0.4 :: \textit{fluffy} : Cat$$
$$0.3 :: tom : Cat$$
$$0.6 :: Cat \sqsubseteq Pet$$

The knowledge base 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 cats are pets with a certain probability. The predicate logic formulas (without external quantifiers) equivalent to the probabilistic axioms are $F_1 = Cat(\textit{fluffy})$, $F_2 = Cat(tom)$ and $F_3 = Cat(x) \rightarrow Pet(x)$. A covering set of explanations for the query axiom $Q = kevin : NatureLover$ is $K = \{\kappa_1, \kappa_2\}$ where $\kappa_1 = \{(F_1, \emptyset, 1), (F_3, \emptyset, 1)\}$ and $\kappa_2 = \{(F_2, \emptyset, 1), (F_3, \emptyset, 1)\}$.

If we associate the random variables $X_{11}$ with $(F_1, \emptyset)$, $X_{21}$ with $(F_2, \emptyset)$ and $X_{31}$ with $(F_3, \emptyset)$, the BDD associated with the set $K$ of explanations is shown in Figure 2. By applying algorithm in Figure 1 we get

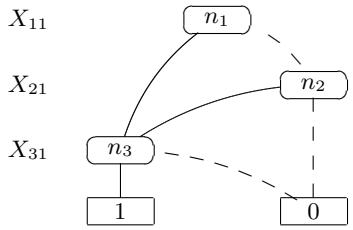



**Fig. 2.** BDD for Example 1.

$$\text{PROB}(n_3) = 0.6 \cdot 1 + 0.4 \cdot 0 = 0.6$$
$$\text{PROB}(n_2) = 0.4 \cdot 0.6 + 0.6 \cdot 0 = 0.24$$
$$\text{PROB}(n_1) = 0.3 \cdot 0.6 + 0.7 \cdot 0.24 = 0.348$$

so $P(Q) = \text{PROB}(n_1) = 0.348$.

*Example 2.* If we replace the axiom $0.6 :: Cat \sqsubseteq Pet$ in Example 1 with $0.6 ::_x Cat \sqsubseteq Pet$, we are expressing the knowledge that 60% of cats are pets. In this case the query would have the explanations $K = \{\kappa_1, \kappa_2\}$ where $\kappa_1 = \{(F_1, \emptyset, 1), (F_3, \{x/\textit{fluffy}\}, 1)\}$ and $\kappa_2 = \{(F_2, \emptyset, 1), (F_3, \{x/tom\}, 1)\}$. The probability in this case raises to $P(Q) = 0.3768$.

## 4 BUNDLE

BUNDLE (Binary decision diagrams for Uncertain reasoNing on Description Logic thEories) computes the probability of queries from a probabilistic knowl-

edge base that follows the DISPONTE semantics. It first finds a covering set of explanations for the query and then builds a BDD for computing the probability of the query.

The problem of finding explanations for a query has been investigated by various authors [27,11,12,10]. For example, Pellet finds explanations by using a tableau algorithm [11]. A *tableau* is a graph where each node $a$ 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 $\mathcal{L}(\langle a, b\rangle)$ to which the couple $(a, b)$ belongs. Pellet repeatedly applies 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. $\{C, \neg C\} \subseteq \mathcal{L}(a)$.

The transformation rule for disjunction is non-deterministic, i.e., it generates 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*. Each expansion rule of Pellet updates a *tracing function* $\tau$ as well, which associates sets of axioms with labels of nodes and edges. The tracing function $\tau$ 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 $\tau$ is initialized as the empty set for all elements of its domain except for $\tau(C, a)$ and $\tau(R, \langle a, b\rangle)$ to which the values $\{a : C\}$ and $\{(a, b) : R\}$ are assigned if $a : C$ and $(a, b) : R$ are in the ABox respectively.

In BUNDLE we are interested in instantiated explanations that entail an axiom. An *instantiated explanation* is a finite set $\mathcal{F} = \{(F_1, \theta_1), \ldots, (F_n, \theta_n)\}$ where $F_1, \ldots, F_n$ are axioms and $\theta_1, \ldots, \theta_n$ are substitutions. Thus BUNDLE modifies the tableau tracing function of Pellet to return a set of pairs (axiom, substitution) instead of a set of axioms. The application of the expansion rules now stores, together with information regarding concepts and roles, also information concerning individuals involved in the expansion rules, which will be returned at the end of the derivation process together with the axioms. Figure 3 shows the tableau expansion rules of BUNDLE for the case of $\mathcal{ALC}$.

BUNDLE finds a covering set of explanations using Pellet's strategy: it first finds a single explanation and then finds the others by iteratively removing from the theory an axiom belonging to an explanation and looking for a new explanation.

BUNDLE first finds a covering set of explanations for the query using the modified version of Pellet and then builds the BDD representing them from

$\rightarrow$ *CE*: **if** $(C \sqsubseteq D) \in K$ **thenif** $(\neg C \sqcup D) \notin \mathcal{L}(a)$, **then** $\mathcal{L}(a) = \mathcal{L}(a) \cup \{\neg C \sqcup D\}$
$\tau(\neg C \sqcup D, a) := \{(C \sqsubseteq D, a)\}$
$\rightarrow \sqcap$: **if** $(C_1 \sqcap C_2) \in \mathcal{L}(a)$ **then if** $\{C_1, C_2\} \not\subseteq \mathcal{L}(a)$, **then** $\mathcal{L}(a) = \mathcal{L}(a) \cup \{C_1 \sqcap C_2\}$
$\tau(C_i, a) := \tau((C_1 \sqcap C_2), a)$
$\rightarrow \sqcup$: **if** $(C_1 \sqcup C_2) \in \mathcal{L}(a)$ **then if** $\{C_1, C_2\} \cap \mathcal{L}(a) = \emptyset$, **then**
Generate graphs $G_i := G$ for each $i \in \{1, 2\}$, $\mathcal{L}(a) = \mathcal{L}(a) \cup \{C_i\}$ for each $i \in \{1, 2\}$
$\tau(C_i, a) := \tau((C_1 \sqcup C_2), a)$
$\rightarrow \exists$: **if** $\exists S.C \in \mathcal{L}(a)$ **then if** $a$ has no S-neighbor $b$ with $C \in \mathcal{L}(b)$,**then**
create new node $b$, $\mathcal{L}(b) = \{C\}$, $\mathcal{L}(\langle a, b\rangle) = \{S\}$,
$\tau(C, b) := \tau((\exists S.C), a)$, $\tau(S, \langle a, b\rangle) := \tau((\exists S.C), a)$
$\rightarrow \forall$: **if** $\forall(S, C) \in \mathcal{L}(a)$, there is an $S$-neighbor $b$ of $a$ **then if** $C \notin \mathcal{L}(b)$, **then** $\mathcal{L}(b) = \mathcal{L}(b) \cup \{C\}$
$\tau(C, b) := \tau(\forall S.C, a) \cup \tau(S, \langle a, b\rangle)$

**Fig. 3.** BUNDLE tableau expansion rules.

which it computes the probability. BUNDLE, shown in Figure 4, first builds a data structure $PMap$ that associates each DL axiom $E$ with a set of couples $(Var, p)$, one for each probabilistic axiom $p ::_{Var} E$ in the knowledge base $\mathcal{K}$. Then BUNDLE finds the explanations and initializes the array $VarAxAnn$ for storing the triples $(Axiom, \theta, Prob)$ associated with a Boolean variable in the BDD. It 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 of individual explanations. At the end, it computes the probability by calling the dynamic programming algorithm that visits the BDD. To manipulate BDDs, we use 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. If a query has a larger number of explanations, then the probability that is computed is a lower bound of the true probability.

## 5 Related Work

There are many works that propose approaches for combining probability and description logics. We refer to [24] for the relationships with DISPONTE. We discuss here only P-$\mathcal{SHIQ}$(D) proposed in [16] because it is equipped with a reasoner, PRONTO [14]. P-$\mathcal{SHIQ}$(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-$\mathcal{SHIQ}$(D) is based on Nilsson's probabilistic logic [18] 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 knowledge base $K$ 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 $K$.

```
1: function BUNDLE(𝒦, C)
2:     Input: 𝒦 (the knowledge base)
3:     Input: C (the concept to be tested for unsatisfiability)
4:     Output: the probability of the unsatisfiability of C w.r.t. 𝒦
5:     Build Map PMap from DL axioms to sets of couples (Var, probability)
6:     Explanations ←GETEXPLANATIONS(C, 𝒦, ∅)
7:     Initialize VarAxAnn to empty ▷ VarAxAnn: array of triples (Axiom, θ, Prob)
8:     BDD ←BDDZERO
9:     for all Explanation ∈ Explanations do
10:        BDDE ←BDDONE
11:        for all (Ax, θ) ∈ Explanation do
12:            (Var, p) ← PMap(Ax)
13:            Scan VarAxAnn looking for (Ax, θ)
14:            if !found then
15:                Add to VarAxAnn a new cell containing (Ax, θ, p)
16:            end if
17:            Let i be the position of (Ax, θ, p) in VarAxAnn
18:            B ← BDDGETITHVAR(i)
19:            BDDE ←BDDAND(BDDE,B)
20:        end for
21:        BDD ←BDDOR(BDD,BDDE)
22:    end for
23:    return PROB(BDD)          ▷ VarAxAnn is used to compute P(X) in PROB
24: end function
```

**Fig. 4.** Function BUNDLE: computation of the probability of a concept $C$ given the knowledge base $\mathcal{K}$.

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 knowledge base composed of $C(a) \geq 0.4$ and $C(b) \geq 0.5$. The distribution semantics allows 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.

## 6 Experiments

We evaluate the performances of BUNDLE in two different ways. In the first, following [15], we ran BUNDLE and the publicly available version of PRONTO to answer queries to increasingly complex ontologies obtained by randomly sampling axioms from a large probabilistic ontology for breast cancer risk assessment (BRCA) and compared their results. The central idea behind the design

of the ontology was to reduce risk assessment to probabilistic entailment in P-$\mathcal{SHIQ}$(D). The BRCA ontology is divided into two parts: a certain part and a probabilistic part. The probabilistic part contains conditional constraints 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 $[l, u]$".

The tests were defined by randomly sampling axioms from the probabilistic part of this ontology. In particular, each test case has been generated by sampling a subset of conditional constraints of the probabilistic part and adding these constraints to the certain part. So each sample was a probabilistic knowledge base with the full certain part of the BRCA ontology and a subset of the probabilistic constraints. We varied the number of these constraints from 9 to 15, and, for each number, we repeatedly sampled ontologies and tested them for consistency. We stopped sampling when we obtained 100 consistent ontologies for each number of constraints.

In order to generate a query, an individual $a$ is added to the ontology. $a$ is randomly assigned to each class that appears in the sampled conditional constraints with probability 0.6. If the class is composite, as for example *PostmenopausalWomanTakingTestosterone*, $a$ is assigned to the component classes rather than to the composite one. In the example above, $a$ will be added to *PostmenopausalWoman* and *WomanTakingTestosterone*.

The ontologies are then translated into DISPONTE by replacing the constraint $(D|C)[l, u]$ with the axiom $u ::_x C \sqsubseteq D$. For instance, the statement that an average woman has up to 12.3% chance of developing breast cancer in her lifetime is expressed by

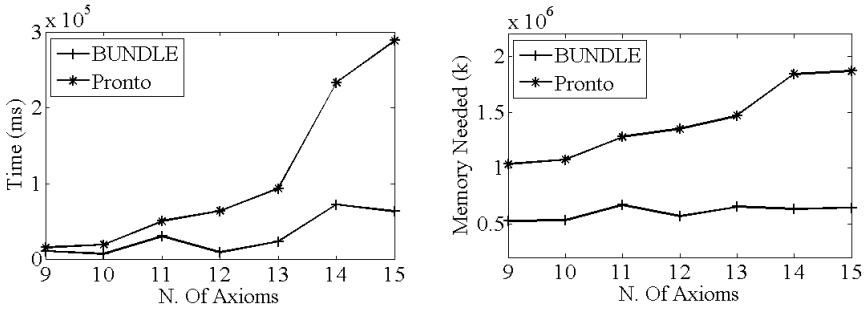
$$(WomanUnderAbsoluteBRCRisk|Woman)[0, 0.123]$$

is translated into

$$0.123 ::_x WomanUnderAbsoluteBRCRisk \sqsubseteq Woman$$

For each ontology we perform the query $a : C$ where the class $C$ is randomly selected among those that represent women under increased and lifetime risk such as *WomanUnderLifetimeBRCRisk* and *WomanUnderStronglyIncreasedBRCRisk*. We then applied both BUNDLE and PRONTO to each generated test and we measured the execution time and the memory used. Figure 5(a) shows the execution time averaged over the 100 knowledge bases as a function of the number of axioms and, similarly, Figure 5(b) shows the average amount of memory used. As one can see, execution times was similar for small knowledge bases, but the difference between the two reasoners rapidly increases for larger knowledge bases. The memory usage for BUNDLE is always less than 53 percent that of PRONTO.

In the second approach we used BUNDLE to compute lower bounds on the probability of queries to three larger real world datasets. The first one models the carcinogenesis prediction of cells, the second one is an extract from DBPedia and the third is an extract from `education.data.gov.uk` that contains information




(a) Execution times (ms).

(b) Memory used (Kb).

**Fig. 5.** Comparison between BUNDLE and PRONTO.

about the school system in the United Kingdom. All queries were concept membership axioms $ind : C$ where $ind$ is an individual and $C$ is a class contained in the ontology. To choose the class $C$ for each individual $ind$ we randomly selected a class to which $ind$ belongs in the ontology, so that each query has at least one explanation. To obtain a probabilistic ontology, all axioms were considered as statistical probabilistic axioms. We set to 10 the maximum number of explanations for each query. In Table 2 we report for each dataset the number of axioms contained in the ontology, the average number of explanations for each query, the number of queries executed, the average time in milliseconds that BUNDLE took for building the BDD and the average time for the overall execution of a query. In particular, the number of explanations for the Carcinogenesis dataset is on average between 50 and 100 because the ontology is very complex. In order to find all the explanations of a query w.r.t. this dataset, BUNDLE took on average about 2 hours. Therefore we did several tests to find a good compromise between the computation time and the approximation of the probability of the query, in order to choose a value that allowed BUNDLE to answer a query in a relatively short time with a small approximation of the final probability. After these tests, we choose 10 as value of maximum value of explanations to find.

These tests show that BUNDLE is able to scale to ontologies of realistic size and that the most expensive operation of the algorithm is the computation of explanations whle the construction of the BDD is usually cheaper.

| Dataset | n. axioms | n. expl | n. queries | BDD time (ms) | total time (ms) |
|---|---|---|---|---|---|
| Carcinogenesis | 74226 | 10 | 335 | 383.2 | 60766 |
| DBPedia | 3757 | 2.3 | 3607 | 128.2 | 41597 |
| educational.data.gov.uk | 5545 | 1.9 | 5723 | 101.7 | 45651 |

**Table 2.** Results of the experiments.

## 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 random ontologies of increasing complexity regarding breast cancer risk assessment and on other real world ontologies. BUNDLE is also used in the system EDGE appearing in this volume [23] for learning the parameters of DISPONTE ontologies.

In the future, we plan to apply this approach to different DLs, such as $\mathcal{SHOIN}(\mathbf{D})$ that is the semantics on which OWL DL is based.

## References


1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: Handbook of knowledge representation, chap. 3, pp. 135–179. Elsevier (2008)
3. Bellodi, E., Riguzzi, F.: Experimentation of an expectation maximization algorithm for probabilistic logic programs. Intelligenza Artificiale 8(1), 3–18 (2012)
4. Bellodi, E., Riguzzi, F.: Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intel. Data Anal. 17(2), 343–363 (2013)
5. Bollig, B., Wegener, I.: Improving the variable ordering of OBDDs is NP-complete. IEEE Trans. Computers 45(9), 993–1002 (1996)
6. Calvanese, D., Lenzerini, M.: On the interaction between isa and cardinality constraints. In: International Conference on Data Engineering. pp. 204–213. IEEE Computer Society (1994)
7. Darwiche, A., Marquis, P.: A knowledge compilation map. J. Artif. Intell. Res. 17, 229–264 (2002)
8. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: International Joint Conference on Artificial Intelligence. pp. 2462–2467 (2007)
9. Grumberg, O., Livne, S., Markovitch, S.: Learning to order BDD variables in verification. J. Artif. Intell. Res. 18, 83–116 (2003)
10. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in [OWL ontologies. In: International Conference on Scalable Uncertainty Management. LNCS, vol. 5785, pp. 124–137. Springer (2009)
11. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. Ph.D. thesis, The Graduate School of the University of Maryland (2006)
12. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: International Semantic Web Conference. LNCS, vol. 4825, pp. 267–280. Springer (2007)
13. Kimmig, A., Demoen, B., De Raedt, L., Costa, V.S., Rocha, R.: On the implementation of the probabilistic logic programming language ProbLog. Theor. Prac. Log. Prog. 11(2-3), 235–262 (2011)
14. Klinov, P.: Pronto: A non-monotonic probabilistic description logic reasoner. In: European Semantic Web Conference. LNCS, vol. 5021, pp. 822–826. Springer (2008)

15. Klinov, P., Parsia, B.: Optimization and evaluation of reasoning in probabilistic description logic: Towards a systematic approach. In: International Semantic Web Conference. LNCS, vol. 5318, pp. 213–228. Springer (2008)
16. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Int. 172(6-7), 852–883 (2008)
17. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. J. Web Sem. 6(4), 291–308 (2008)
18. Nilsson, N.J.: Probabilistic logic. Artif. Intell. 28(1), 71–87 (1986)
19. Patel-Schneider, P, F., Horrocks, I., Bechhofer, S.: Tutorial on OWL (2003)
20. Poole, D.: The Independent Choice Logic for modelling multiple agents under uncertainty. Artif. Intell. 94(1-2), 7–56 (1997)
21. Rauzy, A., Châtelet, E., Dutuit, Y., Bérenguer, C.: A practical comparison of methods to assess sum-of-products. Reliability Engineering and System Safety 79(1), 33–42 (2003)
22. Riguzzi, F.: Extended semantics and inference for the Independent Choice Logic. Log. J. IGPL 17(6), 589–629 (2009)
23. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Parameter learning for probabilistic ontologies. In: International Conference on Web Reasoning and Rule Systems. LNCS, Springer (2013)
24. Riguzzi, F., Lamma, E., Bellodi, E., Zese, R.: Epistemic and statistical probabilistic ontologies. In: Uncertainty Reasoning for the Semantic Web. CEUR Workshop Proceedings, vol. 900, pp. 3–14. Sun SITE Central Europe (2012)
25. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: International Conference on Logic Programming. pp. 715–729. MIT Press (1995)
26. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. Artif. Intell. Res. 15, 391–454 (2001)
27. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: International Joint Conference on Artificial Intelligence. pp. 355–362. Morgan Kaufmann (2003)
28. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intell. 48(1), 1–26 (1991)
29. Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2), 51–53 (2007)
30. Straccia, U.: Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In: International Summer School on Reasoning Web. LNCS, vol. 5224, pp. 54–103. Springer (2008)
31. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comp. 8(3), 410–421 (1979)
32. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: International Conference on Logic Programming. LNCS, vol. 3131, pp. 195–209. Springer (2004)