A Framework for Reasoning on Probabilistic Description Logics

Giuseppe Cota[1][0000−0002−3780−6265], Riccardo Zese[2][0000−0001−8352−6304], Elena Bellodi[2][0000−0002−3717−3779], Evelina Lamma[2][0000−0003−2747−4292], and Fabrizio Riguzzi[3][0000−0003−1654−9703]

[1] Dipartimento di Scienze Matematiche, Fisiche e Informatiche – University of Parma
Parco Area delle Scienze, 53/A, 43124 Parma
giuseppe.cota@unipr.it
[2] Dipartimento di Ingegneria – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
[3] Dipartimento di Matematica e Informatica – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy

Abstract. While there exist several reasoners for Description Logics, very few of them can cope with uncertainty. BUNDLE is an inference framework that can exploit several OWL (non-probabilistic) reasoners to perform inference over Probabilistic Description Logics.
In this chapter, we report the latest advances implemented in BUNDLE. In particular, BUNDLE can now interface with the reasoners of the TRILL system, thus providing a uniform method to execute probabilistic queries using different settings. BUNDLE can be easily extended and can be used either as a standalone desktop application or as a library in OWL API-based applications that need to reason over Probabilistic Description Logics.
The reasoning performance heavily depends on the reasoner and method used to compute the probability. We provide a comparison of the different reasoning settings on several datasets.

Keywords: Probabilistic Description Logic, Semantic Web, Reasoner

1 Introduction

The aim of the Semantic Web is to make information available in a form that is understandable and automatically manageable by machines. In order to realize this vision, the W3C has supported the development of a family of knowledge representation formalisms of increasing complexity for defining ontologies, called OWL (Web Ontology Languages), that are based on Description Logics (DLs). Many inference systems, generally called reasoners, have been proposed to reason upon these ontologies, such as Pellet [33], Hermit [32] and Fact++ [34].

Nonetheless, modeling real-world domains requires dealing with information that is incomplete or that comes from sources with different trust levels. This motivates the need for the management of uncertainty in the Semantic Web,

and many proposals have appeared for combining probability theory with OWL languages, or with the underlying DLs [24,17,21,10,23]. Among them, in [27,36] we introduced the DISPONTE semantics, which applies the distribution semantics [30] to DLs. Examples of systems that perform probabilistic logic inference under DISPONTE are BUNDLE [27,28,36], and TRILL [38,36,37]. The former is implemented in Java, the latter in Prolog.

To perform exact probabilistic inference over DISPONTE knowledge bases (KBs), it is necessary to perform either one of the following reasoning tasks: *justification finding* or *pinpointing formula extraction*. The former method consists of finding the covering set of justifications, i.e., the set of all justifications for the entailment of the query. In the latter, a monotonic boolean formula is built, which compactly represents the covering set of justifications. Both of these non-standard reasoning tasks can be performed by a non-probabilistic OWL reasoner. The first version of BUNDLE was able to execute justification finding by exploiting the Pellet reasoner only [33]. Then it was extended to exploit different non-probabilistic OWL reasoners and approaches for justification finding [7]. In particular, it embeds Pellet, Hermit, Fact++ and JFact as OWL reasoners, and three justification generators, namely GlassBox (only for Pellet), BlackBox and OWL Explanation.

In this chapter, we illustrate the state of the art of the BUNDLE framework. In particular, we present a newer version of BUNDLE which also interfaces with the probabilistic reasoners of the TRILL system, namely: (i) TRILL [38,36], which solves the justification finding problem, (ii) TRILL$^{\mathrm{P}}$ [38,36], which returns the pinpointing formula using the approach defined in [2,3], and (iii) TORNADO [37], which, similarly to TRILL$^{\mathrm{P}}$, returns the pinpointing formula, but the formula is represented in a way that can be directly used to compute the probability. In this way, the user can run probabilistic queries in a uniform way by using the preferred reasoner. In addition, BUNDLE can be easily extended by "plugging-in" a new reasoner or by including new concrete implementations of algorithms for justification finding/axiom pinpointing.

The performance of reasoning heavily depends on the reasoner and method used to compute the probability for a given query and it is of foremost importance for (distributed) probabilistic rule learning systems such as [29,8]. To evaluate the system, we performed several experiments on various real-world and synthetic datasets using different settings.

The chapter is organized as follows: Section 2 briefly introduces DLs, while Section 3 illustrates the problems of justification finding and pinpointing formula extraction. Section 4 and Section 5 present DISPONTE and the theoretical aspects of inference in DISPONTE KBs respectively. The description of BUNDLE is provided in Section 6. Finally, Section 7 shows the experimental evaluation and Section 8 concludes the paper.

2 Description Logics

An ontology describes the concepts of the domain of interest and their relations with a formalism that allows information to be processable by machines. The *Web Ontology Language* (OWL) is a knowledge representation language for authoring ontologies or knowledge bases. The latest version of this language, OWL 2 [35], is a W3C recommendation since 2012. In order to reach some computational properties, it is possible to define a sublanguage of OWL 2, also called *profile*, by applying syntactic restrictions.

Descriptions Logics (DLs) provide a logical formalism for knowledge representation. They are useful in all the domains where it is necessary to represent information and perform inference on it, such as software engineering, medical diagnosis, digital libraries, databases and Web-based informative systems. They possess nice computational properties such as decidability and (for some DLs) low complexity [4].

There are many different DL languages that differ in the constructs that are allowed for defining concepts (sets of individuals of the domain) and roles (sets of pairs of individuals). The $\mathcal{SROIQ}(\mathbf{D})$ DL is one of the most common fragments; it was introduced by Horrocks et al. in [14] and it is of particular importance because it is semantically equivalent to OWL 2.

Let us consider a set of *atomic concepts* $\mathbf{C}$, a set of *atomic roles* $\mathbf{R}$ and a set of individuals $\mathbf{I}$. A *role* could be an atomic role $R \in \mathbf{R}$, the inverse R^- of an atomic role $R \in \mathbf{R}$ or a complex role $R \circ S$. We use $\mathbf{R}^-$ to denote the set of all inverses of roles in $\mathbf{R}$. Each $A \in \mathbf{A}$, $\bot$ and $\top$ are concepts and if $a \in \mathbf{I}$, then $\{a\}$ is a concept called *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 $\exists R.C$, $\forall R.C$, $\exists R.\text{Self}$, $\geq nR.C$ and $\leq nR.C$ for an integer $n \geq 0$.

A *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ consists of a TBox $\mathcal{T}$, an RBox $\mathcal{R}$ and an ABox $\mathcal{A}$. An RBox $\mathcal{R}$ is a finite set of *transitivity axioms* $Trans(R)$, *role asimmetricity axioms* $Asy(R)$, *role disjointness axioms* $Dis(R, S)$, *role inclusion axioms* $R \sqsubseteq S$ and *role chain axioms* $R \circ P \sqsubseteq S$, where $R, P, S \in \mathbf{R} \cup \mathbf{R}^-$. A *TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, where C and D are concepts. An *ABox* $\mathcal{A}$ is a finite set of *concept membership axioms* $a : C$ and *role membership axioms* $(a, b) : R$, where C is a concept, $R \in \mathbf{R}$ and $a, b \in \mathbf{I}$.

A KB is usually assigned a semantics using interpretations 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 individual a, a subset of $\Delta^\mathcal{I}$ to each concept C and a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ to each role R. The mapping $\cdot^\mathcal{I}$ is extended to complex concepts as follows (where $R^\mathcal{I}(x, C) = \{y | \langle x, y \rangle \in R^\mathcal{I}, y \in C^\mathcal{I}\}$ and

$\#X$ denotes the cardinality of the set X):

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} & \bot^{\mathcal{I}} &= \emptyset \\
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} & (\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} & (C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \cap C^{\mathcal{I}} \neq \emptyset\} & (\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \subseteq C^{\mathcal{I}}\} \\
(\geq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \geq n\} & (\leq nR.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} | \#R^{\mathcal{I}}(x, C) \leq n\} \\
(R^-)^{\mathcal{I}} &= \{\langle y, x\rangle | \langle x, y\rangle \in R^{\mathcal{I}}\} & (R_1 \circ ... \circ R_n)^{\mathcal{I}} &= R_1^{\mathcal{I}} \circ ... \circ R_n^{\mathcal{I}} \\
(\exists R.\text{Self})^{\mathcal{I}} &= \{x | \langle x, x\rangle \in R^{\mathcal{I}}\}
\end{aligned}$$

$\mathcal{SROIQ}(\mathbf{D})$ also permits the definition of datatype roles, which connect an individual to an element of a datatype such as integers, floats, etc.

A query Q over a KB $\mathcal{K}$ is usually an axiom for which we want to test the entailment from the KB, written as $\mathcal{K} \models Q$.

Example 1. Consider the following KB "Crime and Punishment" containing 4 axioms α_i

$$\begin{aligned}
\alpha_1 &= \mathsf{Nihilist} \sqsubseteq \mathsf{GreatMan} & \alpha_2 &= \exists \mathsf{killed}.\top \sqsubseteq \mathsf{Nihilist} \\
\alpha_3 &= (\mathsf{raskolnikov}, \mathsf{alyona}) : \mathsf{killed} & \alpha_4 &= (\mathsf{raskolnikov}, \mathsf{lizaveta}) : \mathsf{killed}
\end{aligned}$$

This KB states that if you killed someone then you are a nihilist and whoever is a nihilist is a "great man" (TBox). It also states that Raskolnikov killed Alyona and Lizaveta (ABox). The KB entails the query $Q = \mathsf{raskolnikov} : \mathsf{GreatMan}$ (but are we sure about that?).

2.1 Decidability of $\mathcal{SROIQ}$

In order to ensure decidability of inferencing in $\mathcal{SROIQ}$ DL, three conditions must be met:

- no cardinality restrictions must be applied on transitive roles or on roles that have transitive subroles [16,15];
- the RBox must be regular [14];
- in the class expressions $\geq nR.C$, $\leq nR.C$ and $\exists R.\text{Self}$, and in the axioms $\exists R.\text{Self} \sqsubseteq \bot$ (also known as *role irreflexivity axiom* $Irr(R)$), $Asy(R)$ and $Dis(R)$, R must be *simple*.

Definition 1 (Regular RBox [14]).

- *An RBox is regular if there is a strict linear order $\prec$ on roles and the RBox contains only role chain axioms of the following forms:*

$$\begin{aligned}
&R \circ R \sqsubseteq R & &S^- \sqsubseteq R \\
&S_1 \circ S_2 \circ ... \circ S_n \sqsubseteq R & &R \circ S_1 \circ S_2 \circ ... \circ S_n \sqsubseteq R \\
&S_1 \circ S_2 \circ ... \circ S_n \circ R \sqsubseteq R
\end{aligned}$$

  where $S_i \prec R$ for all $i = 1, 2, \ldots, n$.

Definition 2 (Simple role in $\mathcal{SROIQ}$(D) [14]). *Given a role R, its simplicity is inductively defined as follows:*

- *R is simple if it does not occur on the right hand side of a role inclusion axiom in $\mathcal{R}$, i.e. there is no role chain axiom of the form: $S_1 \circ S_2 \circ \cdots \circ S_n \sqsubseteq R$;*
- *an inverse role R^- is simple if R is, and*
- *if R occurs on the right hand side of a role inclusion axiom in $\mathcal{R}$, then R is simple if, for each $S \sqsubseteq R$, S is a simple role.*

3 Justification Finding and Pinpointing Formula

3.1 Justification Finding

Here we discuss the problem of finding the covering set of justifications for a given query. This non-standard reasoning service is also known as *axiom pinpointing* [31] and it is useful for tracing derivations and debugging ontologies. This problem has been investigated by various authors [18,31,5,13]. A justification corresponds to an *explanation* for a query Q. An explanation is a subset of logical axioms $\mathcal{E}$ of a KB $\mathcal{K}$ such that $\mathcal{E} \models Q$, whereas a justification is an explanation such that it is minimal w.r.t. set inclusion. Formally, we say that an explanation $\mathcal{J} \subseteq \mathcal{K}$ is a justification if for all $\mathcal{J}' \subset \mathcal{J}$, $\mathcal{J}' \not\models Q$, i.e. $\mathcal{J}'$ is not an explanation for Q . The problem of enumerating all justifications that entail a given query is called axiom pinpointing or justification finding. *The set of all the justifications for the query Q is the covering set of justifications for Q*. Given a KB $\mathcal{K}$, the covering set of justifications for Q is denoted by $\textsc{All-Just}(Q, \mathcal{K})$.

Below, we provide the formal definitions of justification finding problem.

Definition 3 (Justification finding problem).
***Input**: A knowledge base $\mathcal{K}$, and an axiom Q such that $\mathcal{K} \models Q$.*
***Output**: The set $\textsc{All-Just}(Q, \mathcal{K})$ of all the justifications for Q in $\mathcal{K}$.*

Example 2. Consider the KB and the query $Q = \mathsf{raskolnikov} : \mathsf{GreatMan}$ of Example 1. $\textsc{All-Just}(Q, \mathcal{K}) = \{\ \{\alpha_1, \alpha_2, \alpha_3\},\ \{\alpha_1, \alpha_2, \alpha_4\}\ \}$.

There are two categories of algorithms for finding a single justification: glass-box [18] and black-box algorithms. The former category is reasoner-dependent, i.e. a glass-box algorithm implementation depends on a specific reasoner, whereas a black-box algorithm is reasoner-independent, i.e. it can be used with any reasoner. In both cases, we still need a reasoner to obtain a justification.

It is possible to incrementally compute all justifications for an entailment by using Reiter's Hitting Set Tree (HST) algorithm [26]. This algorithm repeatedly calls a glass-box or a black-box algorithm which builds a new justification. To avoid the extraction of already found justifications, at each iteration the extraction process is performed on a KB from which some axioms are removed by taking into account the previously found justifications. For instance, given a KB $\mathcal{K}$ and a query Q, if the justification $\mathcal{J} = \{\beta_1, \beta_2, \beta_3\}$ was found, where β_is are axioms, to avoid the generation of the same justification, the HST algorithm

tries to find a new justification on $\mathcal{K}' = \mathcal{K} \setminus \beta_1$. If no new justification is found the HST algorithm backtracks and tries to find another justification by removing other axioms from $\mathcal{J}$, one at a time.

3.2 Pinpointing Formula

Given a query, instead of finding the covering set of justifications, we can compute the *pinpointing formula*, which compactly represents the covering set of justifications, following the approaches proposed in [2,3].

To build a pinpointing formula, first we have to associate a unique propositional variable to every axiom E of the KB $\mathcal{K}$, indicated with $var(E)$. Let $var(\mathcal{K})$ be the set of all the propositional variables associated with axioms in $\mathcal{K}$, then the pinpointing formula is a *monotone Boolean formula* built using some or all of the variables in $var(\mathcal{K})$ and the conjunction and disjunction connectives. A valuation ν of a set of variables $var(\mathcal{K})$ is the set of propositional variables that are true, i.e., $\nu \subseteq var(\mathcal{K})$. For a valuation $\nu \subseteq var(\mathcal{K})$, let $\mathcal{K}_\nu := \{E \in \mathcal{K} | var(E) \in \nu\}$.

Definition 4 (Pinpointing formula). *Given a query Q and a KB $\mathcal{K}$, a monotone Boolean formula ϕ over $var(\mathcal{K})$ is called a* pinpointing formula *for Q if for every valuation $\nu \subseteq var(\mathcal{K})$ it holds that $\mathcal{K}_\nu \models Q$ iff ν satisfies ϕ.*

It is worth noting that a pinpointing formula could be directly generated from the covering set of justifications. However, this formula may not be the most compact.

Example 3. Consider the KB and the query $Q = \mathsf{raskolnikov} : \mathsf{GreatMan}$ of Example 1. If we associate a Boolean variable X_i with each axiom α_i, a pinpointing formula could be $X_1 \wedge X_2 \wedge (X_3 \vee X_4)$. Another (non-compact) pinpointing formula could be directly generated from the covering set of justifications: $(X_1 \wedge X_2 \wedge X_3) \vee (X_1 \wedge X_2 \wedge X_4)$.

4 Probabilistic Description Logics

DISPONTE [27,36] applies the distribution semantics [30] to Probabilistic Description Logic KBs.

In DISPONTE, a *probabilistic knowledge base* $\mathcal{K}$ is a set of certain axioms and probabilistic axioms. Certain *axioms* take the form of regular DL axioms, whereas *probabilistic axioms* take the form $p :: E$, where $p \in [0, 1]$ and E is a DL axiom. The probability p can be interpreted as an *epistemic probability*, i.e., as the degree of our belief in the truth of axiom E. Another interpretation is that p represents the trustworthiness level of the data source for the axiom E.

DISPONTE associates independent Boolean random variables to the DL axioms. The set of axioms that have the random variable assigned to 1 constitutes a *world*. The probability of a world w is computed by multiplying the probability p_i for each probabilistic axiom β_i included in the world by the probability $1 - p_i$ for each probabilistic axiom β_i not included in the world.

Below, we provide some formal definitions for DISPONTE.

Definition 5 (Atomic choice). *An atomic choice is a couple (β_i, k) where β_i is the ith probabilistic axiom and $k \in \{0, 1\}$. The variable k indicates whether β_i is chosen to be included in a world $(k = 1)$ or not $(k = 0)$.*

Definition 6 (Composite choice). *A composite choice κ is a consistent set of atomic choices, i.e., $(\beta_i, k) \in \kappa$, $(\beta_i, m) \in \kappa$ implies $k = m$ (only one decision is taken for each axiom).*

The probability of composite choice κ is: $P(\kappa) = \prod_{(\beta_i,1)\in\kappa} p_i \prod_{(\beta_i,0)\in\kappa}(1 - p_i)$, where p_i is the probability associated with axiom β_i, because the random variables associated with axioms are independent.

Definition 7 (Selection). *A selection σ is a total composite choice, i.e., it contains an atomic choice (β_i, k) for every probabilistic axiom of the theory. A selection σ identifies a theory w_σ called a world: $w_\sigma = \mathcal{C} \cup \{\beta_i | (\beta_i, 1) \in \sigma\}$, where $\mathcal{C}$ is the set of certain axioms.*

$P(w_\sigma)$ is a probability distribution over worlds. Let us indicate with $\mathcal{W}$ the set of all worlds. The probability of Q is [27]: $P(Q) = \sum_{w\in\mathcal{W}:w\models Q} P(w)$, i.e. the probability of the query is the sum of the probabilities of the worlds in which the query is true.

Example 4. Let us consider the knowledge base and the query $Q = \mathsf{raskolnikov} : \mathsf{GreatMan}$ of Example 1 where some of the axioms are probabilistic:

$\beta_1 = 0.2 :: \mathsf{Nihilist} \sqsubseteq \mathsf{GreatMan}$ $\quad \alpha_1 = \exists \mathsf{killed}.\top \sqsubseteq \mathsf{Nihilist}$

$\beta_2 = 0.6 :: (\mathsf{raskolnikov}, \mathsf{alyona}) : \mathsf{killed}$ $\quad \beta_3 = 0.7 :: (\mathsf{raskolnikov}, \mathsf{lizaveta}) : \mathsf{killed}$

Whoever is a nihilist is a "great man" with probability 0.2 (β_1) and Raskolnikov killed Alyona and Lizaveta with probability 0.6 and 0.7 respectively (β_2 and β_3). Moreover there is a certain axiom (α_1). The KB has eight worlds and Q is true in three of them, corresponding to the selections:

$$\{\ \{(\beta_1, 1), (\beta_2, 1), (\beta_3, 1)\},\ \{(\beta_1, 1), (\beta_2, 1), (\beta_3, 0)\},\ \{(\beta_1, 1), (\beta_2, 0), (\beta_3, 1)\}\ \}$$

The probability is $P(Q) = 0.2{\cdot}0.6{\cdot}0.7{+}0.2{\cdot}0.6{\cdot}(1{-}0.7){+}0.2{\cdot}(1{-}0.6){\cdot}0.7 = 0.176$.

5 Inference in Probabilistic Description Logics

It is often infeasible to find all the worlds where the query is true. To reduce reasoning time, inference algorithms find, instead, explanations for the query and then compute the probability of the query from them. Below we provide the definitions of DISPONTE explanations and justifications, which are tightly intertwined with the previous definitions of explanation and justification for the non-probabilistic case.

Definition 8 (DISPONTE Explanation). *A composite choice ϕ identifies a set of worlds $\omega_\phi = \{w_\sigma | \sigma \in \mathcal{S}, \sigma \supseteq \phi\}$, where $\mathcal{S}$ is the set of all selections. We say that ϕ is an explanation for Q if Q is entailed by every world of ω_ϕ.*

Definition 9 (DISPONTE Justification). *We say that an explanation γ is a justification if, for all $\gamma' \subset \gamma$, γ' is not an explanation for Q.*

The set of worlds $\omega_{\boldsymbol{\Phi}}$ identified by the set of explanations $\boldsymbol{\Phi}$ is *covering* Q if every world $w_\sigma \in \mathcal{W}$ in which Q is entailed is such that $w_\sigma \in \omega_{\boldsymbol{\Phi}}$. In other words, a covering set $\boldsymbol{\Phi}$ identifies all the worlds in which Q succeeds.

Two composite choices κ_1 and κ_2 are *incompatible* if their union is inconsistent. For example, $\kappa_1 = \{(\beta_i, 1)\}$ and $\kappa_2 = \{(\beta_i, 0)\}$ are incompatible. A set $\boldsymbol{K}$ of composite choices is *pairwise incompatible* if for all $\kappa_1 \in \boldsymbol{K}$, $\kappa_2 \in \boldsymbol{K}$, $\kappa_1 \neq \kappa_2$ implies that κ_1 and κ_2 are incompatible. The *probability of a pairwise incompatible set of composite choices* K is $P(\boldsymbol{K}) = \sum_{\kappa \in \boldsymbol{K}} P(\kappa)$.

Given a query Q and a covering set of pairwise incompatible explanations $\boldsymbol{\Phi}$, the probability of Q is [27]:

$$P(Q) = \sum_{w_\sigma \in \omega_{\boldsymbol{\Phi}}} P(w_\sigma) = P(\omega_{\boldsymbol{\Phi}}) = P(\boldsymbol{\Phi}) = \sum_{\phi \in \boldsymbol{\Phi}} P(\phi) \tag{1}$$

Unfortunately, in general, explanations (and hence justifications) are not pairwise incompatible, therefore the covering set of justifications cannot be directly used to compute the probability. Even more so, the pinpointing formula, which compactly encodes the covering set of justifications, cannot be directly used for probability computation. The problem of calculating the probability of a query is therefore reduced to that of finding a covering set of justifications or the pinpointing formula and then transforming it into a covering set of pairwise incompatible explanations.

We can think of using non-probabilistic reasoners for justification finding or pinpointing formula extraction, then consider only the probabilistic axioms and transform the covering set of justifications or the pinpointing formula into a pairwise incompatible covering set of explanations from which it is easy to compute the probability.

Example 5. Consider the KB and the query $Q = \mathsf{raskolnikov} : \mathsf{GreatMan}$ of Example 4. If we use justification finding algorithms by ignoring the probabilistic annotations, we find the following non-probabilistic justifications: $\boldsymbol{\mathcal{J}} = \{\ \{\beta_1, \alpha_1, \beta_2\},\ \{\beta_1, \alpha_1, \beta_3\}\ \}$. Then we can translate them into DISPONTE justifications: $\boldsymbol{\Gamma} = \{\ \{(\beta_1, 1), (\beta_2, 1)\},\ \{(\beta_1, 1), (\beta_3, 1)\}\ \}$. Note that $\boldsymbol{\Gamma}$ is not pairwise incompatible, therefore we cannot directly use Equation (1). The solution to this problem will be shown in the following section.

6 BUNDLE

The reasoner BUNDLE [27,28] computes the probability of a query w.r.t. DISPONTE KBs by first computing all the justifications for the query, then converting them into a pairwise incompatible covering set of explanations by building a Binary Decision Diagram (BDD). Finally, it computes the probability by traversing the BDD using function PROBABILITY described in [19]. A BDD for

a function of Boolean variables is a rooted graph that has one level for each Boolean variable. A node n has two children corresponding respectively to the 1 value and the 0 value of the variable associated with the level of n. When drawing BDDs, the 0-branch is distinguished from the 1-branch by drawing it with a dashed line. The leaves store either 0 or 1.

Example 6 (Example 4 cont.). Let us consider the KB and the query of Example 4. If we associate random variables X_1 with axiom β_1, X_2 with β_2 and X_3 with β_3, the BDD representing the set of explanations is shown in Figure 1. By applying function PROBABILITY [19] to this BDD we get

$$\text{PROBABILITY}(n_3) = 0.7 \cdot 1 + 0.3 \cdot 0 = 0.7$$
$$\text{PROBABILITY}(n_2) = 0.6 \cdot 1 + 0.4 \cdot 0.7 = 0.88$$
$$\text{PROBABILITY}(n_1) = 0.2 \cdot 0.88 + 0.8 \cdot 0 = 0.176$$

and therefore $P(Q) = \text{PROBABILITY}(n_1) = 0.176$, which corresponds to the probability given by DISPONTE.

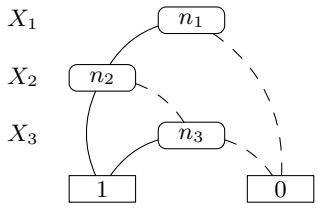

Fig. 1. BDD representing the set of explanations for the query of Example 4.

BUNDLE uses implementations of the HST algorithm to incrementally obtain all the justifications from non-probabilistic reasoner. Moreover, in the latest version, BUNDLE can exploit the reasoners provided by the TRILL framework, thus providing a simple and uniform way to execute probabilistic queries by using the preferred method and reasoner.

Figure 2 shows the new architecture of BUNDLE. The main novelties are the interfaces for the probabilistic reasoners of the TRILL system. These reasoners directly return the probability of the query, therefore, when using one of the TRILL reasoner interfaces, the Probability Computation module, which converts a covering set of justifications into a BDD, is not used.

BUNDLE now supports: (1) four different OWL reasoners: Pellet 2.5.0, Hermit 1.3.8.413 [32], Fact++ 1.6.5 [34], and JFact 4.0.4[4]; (2) three probabilistic reasoners TRILL, TRILL$^{\text{P}}$ and TORNADO, which exploit Prolog's backtracking feature to obtain the justifications or the pinpointing formula; and (3) three different strategies for finding a justification using a non probabilistic reasoner, which are:

[4] `http://jfact.sourceforge.net/`

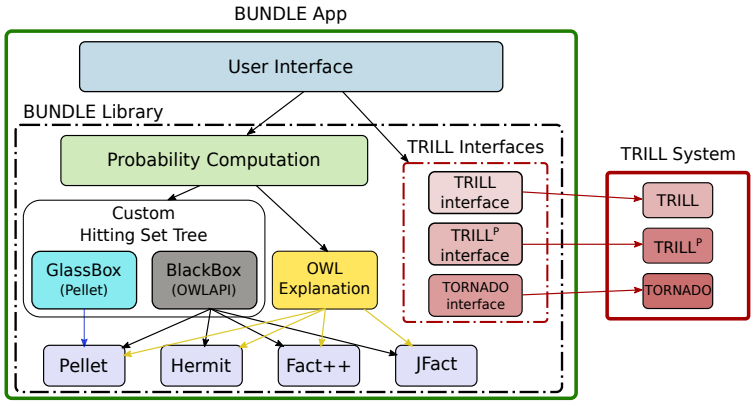

Fig. 2. Software architecture of BUNDLE.

GlassBox A glass-box approach which depends on Pellet. It is a modified version of the `GlassBoxExplanation` class contained in the Pellet Explanation library.

BlackBox A black-box approach offered by the OWL API[5] [12]. The OWL API is a Java API for the creation and manipulation of OWL 2 ontologies.

OWL Explanation A library that is part of the OWL Explanation Workbench [13]. The latter also contains a Protégé plugin, underpinned by the library, that allows Protégé users to find justifications for entailments in their OWL 2 ontologies.

All the supported non-probabilistic reasoners can be paired with the BlackBox or the OWL Explanation methods, while only Pellet can exploit the GlassBox method.

To find all justifications using the GlassBox and BlackBox approaches, we extended the `HSTExplanationGenerator` class of the OWL API, which provides an implementation of the HST algorithm, in order to support annotated axioms (DISPONTE axioms are OWL axioms annotated with a probability). OWL Explanation, instead, already contains an HST implementation and a black-box approach that supports annotated axioms.

Table 1 provides an overview of the reasoners and methods supported by the BUNDLE framework.

BUNDLE can be easily extended in three main ways:

- By adding a non-probabilistic reasoner that implements the `OWLReasoner` interface of the OWL API library.
- By adding a probabilistic reasoner that implements the interface `ProbabilisticReasoner` defined in BUNDLE. Indeed, the interfaces for the TRILL reasoners implement this Java interface.

[5] `http://owlcs.github.io/owlapi/`

Table 1. Reasoners supported by the BUNDLE framework. The symbol ✓means that the reasoner is compatible with the method used for computing the probability.

		Justification Finding				Pinpointing Formula
		Hitting Set Tree			Prolog backtracking	
Reasoner	**DL**	**GlassBox**	**BlackBox**	**OWL Expl.**	**built-in**	**built-in**
Pellet	$\mathcal{SROIQ}(\mathbf{D})$	✓	✓	✓		
Hermit	$\mathcal{SROIQ}(\mathbf{D})$		✓	✓		
Fact++	$\mathcal{SROIQ}(\mathbf{D})$		✓	✓		
JFact	$\mathcal{SROIQ}(\mathbf{D})$		✓	✓		
TRILL	$\mathcal{SHIQ}$				✓	
TRILL$^{\mathrm{P}}$	$\mathcal{SHI}$					✓
TORNADO	$\mathcal{SHI}$					✓

– By adding a non-probabilistic reasoner that is able to perform the reasoning task of justification finding. This reasoner should implement the `ExplanationReasoner` interface defined in BUNDLE.

BUNDLE can be used as standalone desktop application or as a library. Moreover, we developed a web application available at `http://bundle.ml.unife.it/` in a similar way to what has been done for TRILL [6] and `cplint` [1]. The web application allows the user to test BUNDLE without installing any software on the local machine.

6.1 Using BUNDLE as an Application

BUNDLE is an open-source software and is available on Bitbucket, together with its manual, at `https://bitbucket.org/machinelearningunife/bundle`.

A BUNDLE image was deployed in Docker Hub. Users can start using BUNDLE by executing the following commands:

```
sudo docker pull giuseta/bundle:4.0.0
sudo docker run -it giuseta/bundle:4.0.0 bash
```

A bash shell of the container then starts and users can execute probabilistic queries by running the command `bundle`.

6.2 Using BUNDLE as a Library

BUNDLE can also be used as a library. Once the developer has added BUNDLE dependency in the project's POM file, the probability of the query can be obtained in just few lines:

```
1  BundleConfigurationBuilder configBuilder = new
       BundleConfigurationBuilder(ontology);
2  BundleConfiguration config = configBuilder
3    .hstMethod(hstMethod).reasoner(reasonerName)
4    .buildConfiguration();
5  Bundle reasoner = new Bundle(config);
6  reasoner.init();
7  QueryResult result = reasoner.computeQuery(query);
```

where `ontology` and `query` are objects of the classes `OWLOntology` and `OWLAxiom` of the OWL API library respectively.

Lines 1-4 show that the developer can inject the preferred HST method for justification finding (none for the TRILL reasoners) and the favorite reasoner by using a configuration builder. After initialization (lines 5-6), probabilistic inference is performed (line 7).

7 Experiments

We performed four different tests to compare the possible configurations of BUNDLE, which depend on the reasoner and the justification search strategy chosen, for a total of 12 combinations. For each query we set a timeout of 10 minutes. In the first test we compared all configurations on four different datasets, in order to highlight which combination reasoner/strategy shows the best behavior in terms of inference time. To investigate the scalability of the different configurations, in the last three experiments, we considered KBs of increasing size in terms of the number of probabilistic axioms.

All tests were performed on the HPC System Galileo[6] equipped with Intel Xeon E5-2697 v4 (Broadwell) @ 2.30 GHz, using 1 core for each test.

Test 1 The first test considers 4 real world KBs of various complexity as in [38]: (1) **BRCA** [20], which models the risk factors of breast cancer; (2) an extract of **DBPedia** [22], containing structured information from Wikipedia, usually those contained in the information box on the righthand side of pages; (3) **Biopax level 3** [9], which models metabolic pathways; (4) **Vicodi** [25], which contains information on European history and models historical events and important personalities.

We used a version of the DBPedia, Biopax and BRCA KBs without the ABox and a version of Vicodi with an ABox of 19 individuals. For each KB we added a probability annotation to each axiom. The probabilistic values of a KB were randomly assigned using a uniform distribution $\mathcal{U}(0, 1)$ and are fixed for all the queries performed on that KB. We randomly created 50 different subclass-of queries for BRCA, DBPedia and BioPax, and 50 different instance-of queries for Vicodi, following the concepts hierarchy of the KBs, ensuring each query had at least one explanation.

Table 2 shows the average time in seconds to answer queries with different BUNDLE configurations. Bold values highlight the fastest configuration for each KB. Cells with "–" indicate that the timeout was reached in at least one query. Cells with "crash", instead, indicate that the reasoner experienced an internal error and was not able to return a result.

Overall, the best results are obtained by Pellet with the GlassBox approach. However, the use of OWL Explanation library shows competitive results. For BioPax and Vicodi KBs, the Fact++/BlackBox configuration was not able to

[6] `http://www.hpc.cineca.it/hardware/galileo`

return a result for any query. TRILL and TRILL^{P} reached the timeout at least in one query for BioPax and BRCA, while using TORNADO a timeout occurred when querying BioPax.

Table 2. Average time (in seconds) for probabilistic inference with all possible configurations of BUNDLE over different datasets (Test 1). "–" means that the execution timed out (600 s). crash means that the reasoner experienced an internal error.

		Dataset				
Reasoner	**Method**	**BioPax**	**BRCA**	**DBPedia**	**Vicodi**	**Average**
Pellet	GlassBox	1.316	0.886	0.623	0.956	**0.945**
Pellet	BlackBox	1.619	1.653	0.570	1.614	1.364
Pellet	OWLExp	1.070	1.034	0.914	1.420	1.110
Hermit	BlackBox	4.199	6.694	1.299	4.832	4.256
Hermit	OWLExp	1.198	2.071	0.835	2.024	1.532
JFact	BlackBox	1.648	1.852	0.541	1.341	1.346
JFact	OWLExp	**0.887**	1.012	0.878	1.179	0.989
Fact++	BlackBox	crash	0.649	0.363	crash	n.a.
Fact++	OWLExp	0.903	**0.554**	0.588	3.285	1.333
TRILL		–	–	0.442	0.470	n.a.
TRILL^{P}		–	–	0.335	0.424	n.a.
TORNADO		–	2.439	**0.287**	**0.373**	n.a.

Test 2 The second test was performed following the approach presented in [20] on the BRCA KB ($\mathcal{ALCHF}(D)$, 490 axioms). To test BUNDLE, we randomly generated and added an increasing number of subclass-of probabilistic axioms. The number of these axioms was varied from 9 to 16, and, for each number, 30 different consistent KBs were created. Every time a KB is generated, the probability values of the axioms are generated anew. The number of additional axioms may cause an exponential increase of the inference complexity (please see [20] for a detailed explanation). In these tests we consider those possible cases where most of our knowledge is certain but there are some uncertainties.

Finally, an individual was added to every KB, randomly assigned to each simple class that appeared in the probabilistic axioms, and a random probability was attached to it. We ran 60 probabilistic queries of the form $a : C$ where a is the added individual and C is a class randomly selected among those that represent women under increased and lifetime risk such as *WomanUnderLifetimeBRCRisk* and *WomanUnderStronglyIncreasedBRCRisk*, which are at the top of the concept hierarchy.

Table 3 shows the execution time averaged over the 60 queries as a function of the number of probabilistic axioms. For each size, bold values indicate the best configuration. The results show that TORNADO is the only reasoner that can provide answers by respecting the time limits and all its execution timings where competitive with the best configurations.

Table 3. Average execution time (in seconds) for probabilistic inference with different configurations of BUNDLE on versions of the BRCA KB of increasing size (Test 2). "–" means that the execution timed out (600 s).

Reasoner	Method	9	10	11	12	13	14	15	16
Pellet	GlassBox	**2.437**	**1.812**	14.914	–	**4.101**	–	**3.743**	7.102
Pellet	BlackBox	8.082	5.152	29.151	–	14.470	–	12.453	18.727
Pellet	OWLExp	3.034	2.339	10.507	–	4.759	–	4.025	5.405
Hermit	BlackBox	38.431	22.625	–	–	73.854	–	–	–
Hermit	OWLExp	11.249	8.497	29.536	–	18.850	–	–	19.692
JFact	BlackBox	8.937	5.759	35.504	–	16.451	–	15.096	23.108
JFact	OWLExp	3.068	2.384	8.494	–	4.493	–	3.978	5.103
Fact++	BlackBox	–	–	17.015	–	–	–	–	–
Fact++	OWLExp	–	–	–	–	–	–	–	**4.510**
TRLL		–	–	–	–	–	–	–	–
TRILL^{P}		–	–	–	–	–	–	–	–
TORNADO		3.425	2.659	**4.705**	**4.252**	4.604	**4.277**	4.022	4.745

Test 3 In the third test we artificially created a set of KBs of increasing size of the following form:

$$(\beta_{1,i})\ 0.6 :: B_{i-1} \sqsubseteq P_i \sqcap Q_i \qquad (\beta_{2,i})\ 0.6 :: P_i \sqsubseteq B_i \qquad (\beta_{3,i})\ 0.6 :: Q_i \sqsubseteq B_i$$

where $n \geq 1$ and $1 \leq i \leq n$. The query $Q = B_0 \sqsubseteq B_n$ has 2^n justifications, even if the KB has a size that is linear in n. We increased n from 2 to 10 in steps of 2 and we collected the running time, averaged over 50 executions. Table 4 shows, for each n, the average time in seconds that the systems took for computing the probability of the query Q (in bold the best time for each size). Cells with "–" indicate that the timeout occurred at least in one query.

The experimental results show that using TORNADO outperforms the other settings. If the size of the synthetic dataset is greater or equal to 10, all the runs reach a timeout.

Table 4. Average execution time (in seconds) for probabilistic inference with different configurations of BUNDLE on synthetic datasets (Test 3). "–" means that the execution timed out (600 s).

Reasoner	Method	2	4	6	8	10
Pellet	GlassBox	0.890	1.573	7.720	–	–
Pellet	BlackBox	1.060	2.446	12.738	–	–
Pellet	OWLExp	1.843	4.055	10.443	31.118	–
Hermit	BlackBox	5.968	29.316	168.286	–	–
Hermit	OWLExp	4.410	16.637	63.062	239.256	–
JFact	BlackBox	1.039	2.205	10.978	–	–
JFact	OWLExp	1.749	3.869	9.833	28.170	–
Fact++	BlackBox	–	2.229	–	–	–
Fact++	OWLExp	1.728	–	10.159	30.923	–
TRILL		0.549	1.244	3.708	34.105	–
TRILL^{P}		0.267	0.443	23.767	–	–
TORNADO		**0.194**	**0.203**	**0.240**	**0.343**	–

Test 4 To further test the various settings of the BUNDLE framework on real world KBs, we have conducted a test using the *Foundational Model of Anatomy Ontology* (FMA for short)[7]. FMA is a KB for biomedical informatics that models the phenotypic structure of the human body anatomy. It contains 4,706 axioms in the TBox and RBox, with 2,626 different classes. To perform this test, we created 7 versions of the KB containing an increasing number of individuals. The first 5 versions contains a number of individuals varying from 20 to 100 with steps of 20, while the last 2 versions contain 200 and 300 individuals. Each individual can be the subject of 1 to 11 assertions. Then we added a probability annotation to each axiom with random values sampled from a uniform distribution $\mathcal{U}(0, 1)$. For each KB of a given size, we ran 10 times the query $i_0 : \mathsf{Organ\_zone}$, where i_0 is an individual which is present in all the KBs. The averaged running time is reported in Table 5.

The results show that Pellet with GlassBox obtains the best performances and that with OWL Explanation paired with any reasoner we obtain competitive results. However, it seems that the Prolog-based approaches have some issues handling high numbers of individuals.

Table 5. Average execution time (in seconds) for probabilistic inference with different configurations of BUNDLE on versions of the FMA KB of increasing size (Test 4). "–" means that the execution timed out (600 s).

Reasoner	**Method**	**20**	**40**	**60**	**80**	**100**	**200**	**300**
Pellet	GlassBox	**1.392**	**1.491**	**1.611**	**1.594**	**2.457**	1.890	1.931
Pellet	BlackBox	9.253	9.458	11.772	15.509	43.435	20.649	22.118
Pellet	OWLExp	2.000	2.004	2.091	2.092	5.757	2.371	2.470
Hermit	BlackBox	23.660	24.306	26.570	32.344	97.917	40.773	40.969
Hermit	OWLExp	3.913	3.907	3.950	4.201	10.349	4.169	4.184
JFact	BlackBox	7.715	8.206	9.612	12.129	31.382	17.708	17.547
JFact	OWLExp	1.900	1.985	1.956	2.003	4.995	2.290	2.294
Fact++	BlackBox	8.219	–	9.526	11.676	34.216	16.953	–
Fact++	OWLExp	1.644	1.687	1.689	1.774	4.648	**1.853**	**1.845**
TRILL		17.286	23.912	35.932	59.228	93.456	–	–
TRILL$^{\mathrm{P}}$		17.884	31.431	50.811	89.426	142.585	–	–
TORNADO		17.035	24.192	37.112	62.409	97.258	–	–

8 Conclusions

In this chapter, we illustrated the state of the art of BUNDLE, a framework for reasoning on Probabilistic Description Logics KBs that follow DISPONTE. The framework can be used both as a standalone application and as a library and it allows to pair 4 different OWL reasoners with 3 different approaches to find query justifications. Moreover, the latest version add the possibility of using the Prolog-based probabilistic reasoners TRILL, TRILL$^{\mathrm{P}}$ and TORNADO. We also

[7] `http://si.washington.edu/projects/fma`

developed a web application that allows to test BUNDLE without installing any software on the local machine.

We provided a comparison between the various configurations reasoner/approach over different datasets, showing that in 2 out of 4 experiments (Test 1 and Test 4) Pellet paired with GlassBox or any reasoner paired with the OWLExplanation library achieve the best results in terms of inference time on a probabilistic ontology. However, for BRCA datasets of increasing size (Test 2) and for synthetic datasets with queries that have an exponential number of justifications (Test 3), TORNADO showed the best performance.

In the future, we plan to study the effects of glass-box or grey-box methods for collecting explanations. Moreover, we plan to integrate in BUNDLE a reasoner based on Abductive Logic Programming [11].

Acknowledgement This work was supported by the "GNCS-INdAM".

References

1. Alberti, M., Cota, G., Riguzzi, F., Zese, R.: Probabilistic logical inference on the web. In: Adorni, G., Cagnoni, S., Gori, M., Maratea, M. (eds.) AI*IA 2016. LNCS, vol. 10037, pp. 351–363. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-49130-1_26
2. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. J. Autom. Reasoning **45**(2), 91–129 (2010)
3. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. J. Logic Comput. **20**(1), 5–34 (2010)
4. Baader, F., Horrocks, I., Sattler, U.: Description Logics, chap. 3, pp. 135–179. Elsevier, Amsterdam (2008)
5. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic $\mathcal{EL}^+$. Appl. Artif. Intell. p. 52 (2007)
6. Bellodi, E., Lamma, E., Riguzzi, F., Zese, R., Cota, G.: A web system for reasoning with probabilistic OWL. Softw.-Pract. Exper. **47**(1), 125–142 (2017)
7. Cota, G., Riguzzi, F., Zese, R., Bellodi, E., Lamma, E.: A modular inference system for probabilistic description logics. In: Ciucci, D., Pasi, G., Vantaggi, B. (eds.) SUM 2018. LNCS, vol. 11142, pp. 78–92. Springer, Heidelberg, Germany (2018). https://doi.org/10.1007/978-3-030-00461-3_6, `http://mcs.unife.it/~friguzzi/Papers/CotRigZes-SUM18.pdf`
8. Cota, G., Zese, R., Bellodi, E., Lamma, E., Riguzzi, F.: Structure learning with distributed parameter learning for probabilistic ontologies. In: Hollmen, J., Papapetrou, P. (eds.) Doctoral Consortium of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2015). pp. 75–84 (2015), `http://urn.fi/URN:ISBN:978-952-60-6443-7`
9. Demir, E., Cary, M.P., Paley, S., Fukuda, K., Lemer, C., Vastrik, I., Wu, G., D'Eustachio, P., Schaefer, C., Luciano, J., et al.: The biopax community standard for pathway data sharing. Nature biotechnology **28**(9), 935–942 (2010)
10. Ding, Z., Peng, Y.: A probabilistic extension to ontology language OWL. In: Sprague, Jr., R.H. (ed.) 37th Hawaii International Conference on System Sciences (HICSS-37 2004), CD-ROM / Abstracts Proceedings, 5-8 January 2004, Big Island, HI, USA. pp. 1–10. IEEE Computer Society (2004)

11. Gavanelli, M., Lamma, E., Riguzzi, F., Bellodi, E., Zese, R., Cota, G.: An abductive framework for datalog± ontologies. In: Vos, M.D., Eiter, T., Lierler, Y., Toni, F. (eds.) Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015). CEUR-WS, vol. 1433. CEUR-WS.org (2015)
12. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semant. Web **2**(1), 11–21 (2011)
13. Horridge, M., Parsia, B., Sattler, U.: The OWL explanation workbench: A toolkit for working with justifications for entailments in OWL ontologies (2009)
14. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006. pp. 57–67. AAAI Press (2006), http://dl.acm.org/citation.cfm?id=3029947.3029959
15. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. J. Logic Comput. **9**(3), 385–410 (1999)
16. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Logic for Programming and Automated Reasoning. vol. 99, pp. 161–180. Springer (1999)
17. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Doyle, J., Sandewall, E., Torasso, P. (eds.) 4th International Conference on Principles of Knowledge Representation and Reasoning. pp. 305–316. Morgan Kaufmann (1994)
18. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. Ph.D. thesis, The Graduate School of the University of Maryland (2006)
19. Kimmig, A., Demoen, B., De Raedt, L., Costa, V.S., Rocha, R.: On the implementation of the probabilistic logic programming language ProbLog. Theor. Pract. Log. Prog. **11**(2-3), 235–262 (2011)
20. Klinov, P., Parsia, B.: Optimization and evaluation of reasoning in probabilistic description logic: Towards a systematic approach. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5318, pp. 213–228. Springer (2008)
21. Koller, D., Levy, A.Y., Pfeffer, A.: P-CLASSIC: A tractable probabilistic description logic. In: Kuipers, B., Webber, B.L. (eds.) Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island. pp. 390–397. AAAI Press / The MIT Press (1997)
22. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web **6**(2), 167–195 (2015)
23. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. **172**(6-7), 852–883 (2008)
24. Lutz, C., Schröder, L.: Probabilistic Description Logics for subjective uncertainty. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) 12th International Conference on Principles of Knowledge Representation and Reasoning (KR 2010). pp. 393–403. AAAI Press, Menlo Park, CA, USA (2010)
25. Nagypál, G., Deswarte, R., Oosthoek, J.: Applying the semantic web: The VICODI experience in creating visual contextualization for history. Lit. Linguist. Comput. **20**(3), 327–349 (2005). https://doi.org/10.1093/llc/fqi037, https://doi.org/10.1093/llc/fqi037

26. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
27. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Probabilistic description logics under the distribution semantics. Semant. Web **6**(5), 447–501 (2015). https://doi.org/10.3233/SW-140154
28. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Reasoning with probabilistic ontologies. In: Yang, Q., Wooldridge, M. (eds.) 24th International Joint Conference on Artificial Intelligence (IJCAI 2015). pp. 4310–4316. AAAI Press/International Joint Conferences on Artificial Intelligence, Palo Alto, California USA (2015)
29. Riguzzi, F., Bellodi, E., Zese, R., Cota, G., Lamma, E.: Scaling structure learning of probabilistic logic programs by MapReduce. In: Fox, M., Kaminka, G. (eds.) ECAI 2016. vol. 285, pp. 1602–1603. IOS Press (2016). https://doi.org/10.3233/978-1-61499-672-9-1602
30. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Sterling, L. (ed.) ICLP 1995. pp. 715–729. MIT Press (1995)
31. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003. pp. 355–362. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)
32. Shearer, R., Motik, B., Horrocks, I.: HermiT: A highly-efficient OWL reasoner. In: Dolbear, C., Ruttenberg, A., Sattler, U. (eds.) Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008. CEUR-WS, vol. 432. CEUR-WS.org (2008)
33. Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Semant. **5**(2), 51–53 (2007)
34. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Furbach, U., Shankar, N. (eds.) Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings. LNCS, vol. 4130, pp. 292–297. Springer (2006). https://doi.org/10.1007/11814771_26, `https://doi.org/10.1007/11814771_26`
35. W3C: OWL 2 Web Ontology Language (12 2012), `http://www.w3.org/TR/2012/REC-owl2-overview-20121211/`
36. Zese, R.: Probabilistic Semantic Web: Reasoning and Learning, Studies on the Semantic Web, vol. 28. IOS Press, Amsterdam (2017). https://doi.org/10.3233/978-1-61499-734-4-i, `http://ebooks.iospress.nl/volume/probabilistic-semantic-web-reasoning-and-learning`
37. Zese, R., Bellodi, E., Cota, G., Riguzzi, F., Lamma, E.: Probabilistic DL reasoning with pinpointing formulas: A Prolog-based approach. Theor. Pract. Log. Prog. pp. 1–28 (2018). https://doi.org/10.1017/S1471068418000480
38. Zese, R., Bellodi, E., Riguzzi, F., Cota, G., Lamma, E.: Tableau reasoning for description logics and its extension to probabilities. Ann. Math. Artif. Intell. **82**(1-3), 101–130 (2018). https://doi.org/10.1007/s10472-016-9529-3, `http://dx.doi.org/10.1007/s10472-016-9529-3f`