Computing Instantiated Explanations in OWL DL

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. Finding explanations for queries to Description Logics (DL) theories is a non-standard reasoning service originally defined for debugging purposes but recently found useful for answering queries to probabilistic theories. In the latter case, besides the axioms that are used to entail the query, it is necessary to record also the individuals to which the axioms are applied. We refer, in this case, to instantiated explanations. The system BUNDLE computes the probability of queries to probabilistic $\mathcal{ALC}$ knowledge bases by first finding instantiated explanations for the query and then applying a dynamic programming algorithm. In order to apply BUNDLE to more expressive DLs, such as $\mathcal{SHOIN}(\mathbf{D})$ that is at the basis of OWL DL, instantiated explanations must be found. In this paper, we discuss how we extended BUNDLE in order to compute instantiated explanations for $\mathcal{SHOIN}(\mathbf{D})$.

1 Introduction

Description Logics (DLs) are at the basis of the Semantic Web and the study of reasoning algorithms for DLs is a very active area of research. The problem of finding explanations for queries has been the subject of various works [22,7,9,6]. An explanation for a query is a set of axioms that is sufficient for entailing the query and that is minimal. Finding all explanations for a query is a non-standard reasoning service that was originally devised for ontology debugging, i.e., for helping the knowledge engineer in spotting modeling errors. However, this reasoning service turned out to be useful also for computing the probability of queries from probabilistic knowledge bases. In [3] we presented the algorithm BUNDLE for "Binary decision diagrams for Uncertain reasoNing on Description Logic thEories", that performs inference from DL knowledge bases under the probabilistic DISPONTE semantics. DISPONTE is based on the distribution semantics for probabilistic logic programming [20] and minimally extends the underlying DL language by annotating axioms with a probability and assuming that each axiom is independent of the others. In [3] BUNDLE computes the probability of queries by first finding all the explanations for the queries using

an underlying reasoner (Pellet [23] in particular) and then building Binary Decision Diagrams from which the probability is computed. In [17] we extended DISPONTE by including a second type of probabilistic annotation that represents statistical information, while the previous type only represents epistemic information. With these two types of annotations we are able to seamlessly represent assertional and terminological probabilistic knowledge, including degrees of overlap between concepts. Statistical annotations require to consider the individuals to which the axioms are applied. Thus, in order to perform inference, finding explanations is not enough: we need to know also the individuals involved in the application of each axiom.

In [18] we presented a version of BUNDLE that is able to compute the probabilities of queries from DISPONTE $\mathcal{ALC}$ knowledge bases. This required a modification of the underlying DL resoner, Pellet, in order to find *instantiated explanations* for the queries: sets of couples (axiom, substitution) where the substitution replaces (some) of the universally quantified logical variables of the axiom with individuals. This can be seen as a new non-standard reasoning service that, to the best of our knowledge, has not been tackled before. This result was achieved by modifying the rules that Pellet uses for updating the tracing function during the expansion of its tableau.

In this paper, we present an extension of BUNDLE for computing instantiated explanations for the expressive $\mathcal{SHOIN}(\mathbf{D})$ DL, that is at the basis of the OWL DL language. This paves the way for answering queries to DISPONTE OWL DL, thus allowing the introduction of uncertainty in the Semantic Web, an important goal according to many authors [4,24].

To compute instantiated explanations from $\mathcal{SHOIN}(\mathbf{D})$, we will illustrate how we modified the rules for updating the tracing function during the expansion of the tableau. This allows to find a single instantiated explanation. In order to find all of them, the hitting set algorithm of Pellet has been suitably adapted.

The paper is organized as follows. Section 2 briefly introduces the syntax of $\mathcal{SHOIN}(\mathbf{D})$ and its translation into predicate logic. Section 3 defines the problem of finding instantiated explanations while Section 4 illustrates the motivations that led us to consider this problem. Section 5 describes how BUNDLE finds instantiated explanations and, finally, Section 6 concludes the paper.

2 Description Logics

DLs are knowledge representation formalisms particularly useful for representing ontologies and have been adopted as the basis of the Semantic Web [1,2]. In this section, we recall the expressive description logic $\mathcal{SHOIN}(\mathbf{D})$ [12], which is the basis of the web ontology language OWL DL.

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

A *role* is either an atomic role $R \in \mathbf{R}$ or the inverse R^- of an atomic role $R \in \mathbf{R}$. We use $\mathbf{R}^-$ to denote the set of all inverses of roles in $\mathbf{R}$. An *RBox* $\mathcal{R}$ consists of a finite set of *transitivity axioms* $Trans(R)$, where $R \in \mathbf{R}$, and *role inclusion axioms* $R \sqsubseteq S$, where $R, S \in \mathbf{R} \cup \mathbf{R}^-$.

Concepts are defined by induction as follows. Each $C \in \mathbf{A}$ is a concept, $\bot$ and $\top$ are concepts, and if $a \in \mathbf{I}$, then $\{a\}$ is a concept. 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 $\exists R.C$, $\forall R.C$, $\geq nR$ and $\leq nR$ for an integer $n \geq 0$. 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* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ consists of a TBox $\mathcal{T}$, an RBox $\mathcal{R}$ 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 converting a KB into a predicate logic theory and then using the model-theoretic semantics of the resulting theory. A translation of $\mathcal{SHOIN}$ into First-Order Logic with Counting Quantifiers is given in the following as an extension of the one given in [21]. We assume basic knowledge of logic. The translation uses two functions π_x and π_y that map concept expressions to logical formulas, where π_x is given by

$$
\begin{array}{ll}
\pi_x(A) = A(x) & \pi_x(\neg C) = \neg \pi_x(C) \\
\pi_x(\{a\}) = (x = a) & \pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D) \\
\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D) & \pi_x(\exists R.C) = \exists y.R(x, y) \wedge \pi_y(C) \\
\pi_x(\exists R^-.C) = \exists y.R(y, x) \wedge \pi_y(C) & \pi_x(\forall R.C) = \forall y.R(x, y) \rightarrow \pi_y(C) \\
\pi_x(\forall R^-.C) = \forall y.R(y, x) \rightarrow \pi_y(C) & \pi_x(\geq nR) = \exists^{\geq n} y.R(x, y) \\
\pi_x(\geq nR^-) = \exists^{\geq n} y.R(y, x) & \pi_x(\leq nR) = \exists^{\leq n} y.R(x, y) \\
\pi_x(\leq nR^-) = \exists^{\leq n} y.R(y, x) &
\end{array}
$$

and π_y is obtained from π_x by replacing x with y and vice-versa. Table 1 shows the translation of each axiom of $\mathcal{SHOIN}$ knowledge bases into predicate logic.

$\mathcal{SHOIN}(\mathbf{D})$ adds to $\mathcal{SHOIN}$ datatype roles, i.e., roles that map an individual to an element of a datatype such as integers, floats, etc. Then new concept definitions involving datatype roles are added that mirror those involving roles introduced above. We also assume that we have predicates over the datatypes.

A query Q over a KB $\mathcal{K}$ is usually an axiom for which we want to test the entailment from the knowledge base, written $\mathcal{K} \models Q$. The entailment test may be reduced to checking the unsatisfiability of a concept in the knowledge base, i.e., the emptiness of the concept. $\mathcal{SHOIN}(\mathbf{D})$ is decidable iff there are no number restrictions on non-simple roles. A role is non-simple iff it is transitive or it has transitive subroles.

Given a predicate logic formula F, a *substitution* θ is a set of pairs x/a, where x is a variable universally quantified in the outermost quantifier in F and

Axiom	Translation
$C \sqsubseteq D$	$\forall x.\pi_x(C) \rightarrow \pi_x(D)$
$R \sqsubseteq S$	$\forall x, y.R(x, y) \rightarrow S(x, y)$
$Trans(R)$	$\forall x, y, z.R(x, y) \wedge R(y, z) \rightarrow S(x, z)$
$a : C$	$\pi_a(C)$
$(a, b) : R$	$R(a, b)$
$a = b$	$a = b$
$a \neq b$	$a \neq b$

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

$a \in \mathbf{I}$. The application of θ 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 θ. Formulas not containing variables are called *ground*. A substitution θ is *grounding* for a formula F if $F\theta$ is ground.

Example 1. The following KB is inspired by the ontology `people+pets` [13]:
$\exists hasAnimal.Pet \sqsubseteq NatureLover$
$(kevin, fluffy) : hasAnimal \quad (kevin, tom) : hasAnimal$
$fluffy : Cat \quad tom : Cat \quad Cat \sqsubseteq Pet$
It states that individuals that own an animal which is a pet are nature lovers and that *kevin* owns the animals *fluffy* and *tom*. Moreover, *fluffy* and *tom* are cats and cats are pets. The predicate logic formulas equivalent to the axioms are $F_1 = \forall x.\exists y.hasAnimal(x, y) \wedge Pet(y) \rightarrow NatureLover(x)$, $F_2 = hasAnimal(kevin, fluffy)$, $F_3 = hasAnimal(kevin, tom)$, $F_4 = Cat(fluffy)$, $F_5 = Cat(tom)$ and $F_6 = \forall x.Cat(x) \rightarrow Pet(x)$. The query $Q = kevin : NatureLover$ is entailed by the KB.

The *Unique Name Assumption* (UNA) [1] states that distinct individual names denote distinct objects, i.e., that $a \neq b$ with $a, b \in \mathbf{I}$ implies $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

3 Finding Instantiated Explanations

The problem of finding explanations for a query has been investigated by various authors [22,7,9,6]. It was called *axiom pinpointing* in [22] and considered as a non-standard reasoning service useful for debugging ontologies. In particular, in [22] the authors define *minimal axiom sets* or *MinAs* for short.

Definition 1 (MinA). *Let $\mathcal{K}$ be a knowledge base and Q an axiom that follows from it, i.e., $\mathcal{K} \models Q$. We call a set $M \subseteq \mathcal{K}$ a* minimal axiom set *or* MinA *for Q in $\mathcal{K}$ if $M \models Q$ and it is minimal w.r.t. set inclusion.*

The problem of enumerating all MinAs is called MIN-A-ENUM. ALL-MINAS$(Q, \mathcal{K})$ is the set of all MinAs for Q in $\mathcal{K}$.

Axiom pinpointing has been thoroughly discussed in [8,10,5,9] for the purpose of tracing derivations and debugging ontologies. The techniques proposed in

these papers have been integrated into the Pellet reasoner [23]. Pellet solves MIN-A-ENUM by finding a single MinA using a tableau algorithm and then applying the *hitting set* algorithm to find all the other MinAs.

BUNDLE is based on Pellet and uses it for solving the MIN-A-ENUM problem. However, BUNDLE needs, besides ALL-MINAS$(Q, \mathcal{K})$, also the individuals to which the axiom was applied for each probabilistic axiom appearing in ALL-MINAS$(Q, \mathcal{K})$. We call this problem *instantiated axiom pinpointing*.

In instantiated axiom pinpointing we are interested in instantiated minimal sets of axioms that entail an axiom. We call this type of explanations *InstMinA*. An *instantiated axiom set* 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. Given two instantiated axiom sets $\mathcal{F} = \{(F_1, \theta_1), \ldots, (F_n, \theta_n)\}$ and $\mathcal{E} = \{(E_1, \delta_1), \ldots, (E_m, \delta_m)\}$, we say that $\mathcal{F}$ *precedes* $\mathcal{E}$, written $\mathcal{F} \preceq \mathcal{E}$, iff, for each $(F_i, \theta_i) \in \mathcal{F}$, there exists an $(E_j, \delta_j) \in \mathcal{E}$ and a substitution η such that $F_j\theta_j = E_i\delta_i\eta$.

Definition 2 (InstMinA). *Let $\mathcal{K}$ be a knowledge base and Q an axiom that follows from it, i.e., $\mathcal{K} \models Q$. We call $\mathcal{F} = \{(F_1, \theta_1), \ldots, (F_n, \theta_n)\}$ an* instantiated minimal axiom set *or* InstMinA *for Q in $\mathcal{K}$ if $\{F_1\theta_1, \ldots, F_n\theta_n\} \models Q$ and $\mathcal{F}$ is minimal w.r.t. precedence.*

Minimality w.r.t. precedence means that axioms in a InstMinA are as instantiated as possible. We call INST-MIN-A-ENUM the problem of enumerating all InstMinAs. ALL-INSTMINAS$(Q, \mathcal{K})$ is the set of all InstMinAs for Q in $\mathcal{K}$.

Example 2. The query $Q = kevin : NatureLover$ of Example 1 has two MinAs, the first of which is: $\{ hasAnimal(kevin, fluffy),\ Cat(fluffy), \forall x.Cat(x) \rightarrow Pet(x), \forall x.\ \exists y.hasAnimal(x, y) \wedge Pet(y) \rightarrow NatureLover(x) \}$ while the other is: $\{ hasAnimal(kevin, tom),\ Cat(tom),\ \forall x.Cat(x) \rightarrow Pet(x), \forall x.\exists y.hasAnimal(x, y) \wedge Pet(y) \rightarrow NatureLover(x) \}$, where axioms are represented as first order logic formulas.

The corresponding InstMinAs are $\{ hasAnimal(kevin, fluffy),\ Cat(fluffy), Cat(fluffy) \rightarrow Pet(fluffy),\ hasAnimal(kevin,\ fluffy) \wedge Pet(fluffy) \rightarrow NatureLover(kevin) \}$ and $\{ hasAnimal(kevin, tom),\ Cat(tom),\ Cat(tom) \rightarrow Pet(tom),\ hasAnimal(kevin, tom) \wedge Pet(tom) \rightarrow NatureLover(kevin) \}$ respectively, where instantiated axioms are represented directly in their first order version.

4 Motivation

INST-MIN-A-ENUM is required to answer queries to knowledge bases following the DISPONTE probabilistic semantics. DISPONTE applies the distribution semantics [20] to probabilistic DL theories. In DISPONTE, a *probabilistic knowledge base* $\mathcal{K}$ is a set of certain axioms or probabilistic axioms. *Certain axioms* take the form of regular DL axioms. *Probabilistic axioms* take the form $p ::_{Var} E$ where p is a real number in $[0, 1]$, *Var* is a set of variables from $\{x, y, z\}$ and E is a DL axiom. *Var* is usually written as a string, so xy indicates the subset $\{x, y\}$.

If *Var* is empty, then the :: symbol has no subscript. The variables in *Var* must appear in the predicate logic version of E shown in Table 1.

In order to give a semantics to such probabilistic knowledge bases, we consider their translation into predicate logic and we make the UNA. The idea of DISPONTE [17,16] 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 define a *world*, the set of predicate logic formulas whose random variable takes value 1. The purpose of the UNA is to ensure that the instantiations of formulas are really distinct.

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. Here we assume that $\mathbf{I}$ is countably infinite and, together with the UNA, this entails that the elements of $\mathbf{I}$ are in bijection with the elements of the domain. These are called *standard names* according to [11].

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 the set of all allowed variables, p can be interpreted as a *statistical probability*, i.e., as information regarding random individuals from the domain.

For example, the statement that birds fly with 90% probability can be expressed by the epistemic axiom $0.9 :: \mathit{Bird} \sqsubseteq \mathit{Flies}$. Instead the statistical axiom $0.9 ::_x \mathit{Bird} \sqsubseteq \mathit{Flies}$ means that a random bird has 90% probability of flying. Thus, 90% of birds fly. If we query the probability of a bird flying, both axioms give the same result, 0.9. 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 the first one.

In order to compute the probability of queries, rather than generating all possible worlds, we look for a covering set of explanations, i.e., the set of all InstMinAs for the query. The query is true if all its instantiated axioms in an InstMinA are chosen, according to the values of the random variables. To compute the probability, the explanations must be made mutually exclusive, so that the probability of each individual explanation is computed and summed with the others. This is done by means of Binary Decision Diagrams [18].

Instantiated axiom pinpointing is also useful for a more fine-grained debugging of the ontology: by highlighting the individuals to which the axiom is being applied, it may point to parts of the ABox to be modified for repairing the KB.

5 An Algorithm for Computing Instantiated Explanations

Pellet, on which BUNDLE is based, finds explanations by using a tableau algorithm [7]. A *tableau* is a graph where each node represents an individual a and

is labeled with the set of concepts $\mathcal{L}(a)$ it belongs to. Each edge $\langle a, b\rangle$ in the graph is labeled with the set of roles $\mathcal{L}(\langle a, b\rangle)$ 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 concept C and a node a where C and $\neg C$ are present in the label of a, i.e. $\{C, \neg C\} \subseteq \mathcal{L}(a)$. Each expansion rule updates as well a *tracing function* τ, which associates sets of axioms with labels of nodes and edges. It maps couples (concept, individual) or (role, couple of individuals) to a fragment of the knowledge base $\mathcal{K}$. τ is initialized as the empty set for all the 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. The output of the tableau algorithm is a set S of axioms that is a fragment of $\mathcal{K}$ from which the query is entailed.

In BUNDLE we are interested in solving the INST-MIN-A-ENUM problem. In order to find all the instantiated explanations, we modified the tableau expansion rules of Pellet, reported in [7], to return a set of pairs (axiom, substitution) instead of a set of axioms. The tracing function τ 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. Fig. 1 shows the tableau expansion rules of BUNDLE for OWL DL, where $(A \sqsubseteq D, a)$ is the abbreviation of $(A \sqsubseteq D, \{x/a\})$, $(R \sqsubseteq S, a)$ of $(R \sqsubseteq S, \{x/a\})$, $(R \sqsubseteq S, a, b)$ of $(R \sqsubseteq S, \{x/a, y/b\})$, $(\mathit{Trans}(R), a, b)$ of $(\mathit{Trans}(R), \{x/a, y/b\})$ and $(\mathit{Trans}(R), a, b, c)$ of $(\mathit{Trans}(R), \{x/a, y/b, z/c\})$, with a, b, c individuals and x, y, z variables (see Table 1). In [18] we presented BUNDLE for the $\mathcal{ALC}$ DL and only the rules $\rightarrow$ *unfold* , $\rightarrow$ *CE* and $\rightarrow\ \forall$ were modified with respect to Pellet's ones.

Example 3. Let us consider the knowledge base presented in Example 1 and the query $Q = kevin : NatureLover$.

After the initialization of the tableau, BUNDLE can apply the $\rightarrow$ *unfold* rule to the individuals *tom* or *fluffy*. Suppose it selects *tom*. The tracing function τ becomes (in predicate logic)

$\tau(Pet, tom) = \{\ Cat(tom),\ Cat(tom) \rightarrow Pet(tom)\ \}$

At this point BUNDLE applies the $\rightarrow\ CE$ rule to *kevin*, adding $\neg(\exists hasAnimal.Pet) \sqcup NatureLover = \forall hasAnimal.(\neg Pet) \sqcup NatureLover$ to $\mathcal{L}(kevin)$ with the following tracing function:

$\tau(\forall hasAnimal.(\neg Pet) \sqcup NatureLover, kevin) = \{$
$\quad \exists y.hasAnimal(kevin, y) \wedge Pet(y) \rightarrow NatureLover(kevin)\ \}$

Then it applies the $\rightarrow \sqcup$ rule on *kevin* generating two tableaux. In the first one it adds $\forall hasAnimal.(\neg Pet)$ to the label of *kevin* with the tracing function

$\tau(\forall hasAnimal.(\neg Pet), kevin) = \{$
$\quad \exists y.hasAnimal(kevin, y) \wedge Pet(y) \rightarrow NatureLover(kevin)\}$

Now it can apply the $\rightarrow \forall$ rule to *kevin*. In this step it can use either *tom* or *fluffy*, supposing it selects *tom* the tracing function will be:

$\tau(\neg(Pet), tom) = \{\ hasAnimal(kevin, tom),$
$\quad hasAnimal(kevin, tom) \wedge Pet(tom) \rightarrow NatureLover(kevin)\ \}$

At this point this first tableau contains a clash for the individual *tom*, thus

$\rightarrow$ *unfold*: **if** $A \in \mathcal{L}(a)$, A atomic and $(A \sqsubseteq D) \in K$, **then**
if D $\notin \mathcal{L}(a)$, **then**
$\mathcal{L}(a) = \mathcal{L}(a) \cup \{D\}$
$\tau(D, a) := \tau(A, a) \cup \{(A \sqsubseteq D, a)\}$
$\rightarrow$ *CE*: **if** $(C \sqsubseteq D) \in K$, **then**
if $(\neg C \sqcup D) \notin \mathcal{L}(a)$, **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**
$\mathcal{L}(a) = \mathcal{L}(a) \cup \{C_1, C_2\}$
$\tau(C_i, a) := \tau((C_1 \sqcap C_2), a)$
$\rightarrow \sqcup$: **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** $\forall(S_1, C) \in \mathcal{L}(a_1)$, a_1 is not indirectly blocked and there is an S_1-neighbor b of a_1, **then**
if $C \notin \mathcal{L}(b)$, **then** $\mathcal{L}(b) = \mathcal{L}(a) \cup \{C\}$
if there is a chain of individuals $a_2, \ldots, a_n$ and roles $S_2, \ldots, S_n$ such that
$\bigcup_{i=2}^{n} \{(\mathit{Trans}(S_{i-1}), a_i, a_{i-1}), (S_{i-1} \sqsubseteq S_i, a_i)\} \subseteq \tau(\forall S_1.C, a_1)$
and $\neg\exists a_{n+1} : \{(\mathit{Trans}(S_n), a_{n+1}, a_n), (S_n \sqsubseteq S_{n+1}, a_{n+1})\} \subseteq \tau(\forall S_1.C, a_1)$, **then**
$\tau(C, b) := \tau(\forall S_1.C, a_1) \setminus \bigcup_{i=2}^{n} \{(\mathit{Trans}(S_{i-1}), a_i, a_{i-1}), (S_{i-1} \sqsubseteq S_i, a_i)\} \cup$
$\bigcup_{i=2}^{n} \{(\mathit{Trans}(S_{i-1}), a_i, a_{i-1}, b), (S_{i-1} \sqsubseteq S_i, a_i, b)\} \cup \tau(S_1, \langle a_1, b\rangle)$
else
$\tau(C, b) := \tau(\forall S_1.C, a_1) \cup \tau(S_1, \langle a_1, b\rangle)$
$\rightarrow \forall^+$: **if** $\forall(S.C) \in \mathcal{L}(a)$, a is not indirectly blocked
and there is an R-neighbor b of a, $\mathit{Trans}(R)$ and $R \sqsubseteq S$, **then**
if $\forall R.C \notin \mathcal{L}(b)$, **then** $\mathcal{L}(b) = \mathcal{L}(b) \cup \{\forall R.C\}$
$\tau(\forall R.C, b) := \tau(\forall S.C, a) \cup \tau(R, \langle a, b\rangle) \cup \{(\mathit{Trans}(R), a, b), (R \sqsubseteq S, a)\}$
$\rightarrow \geq$: **if** $(\geq nS) \in \mathcal{L}(a)$, a is not blocked, **then**
if there are no n safe S-neighbors $b_1, \ldots, b_n$ of a with $b_i \neq b_j$, **then**
create n new nodes $b_1, \ldots, b_n$; $\mathcal{L}(\langle a, b_i\rangle) = \mathcal{L}(\langle a, b_i\rangle) \cup \{S\}$; $\dot{\neq}(b_i, b_j)$
$\tau(S, \langle a, b_i\rangle) := \tau((\geq nS), a)$
$\tau(\dot{\neq}(b_i, b_j)) := \tau((\geq nS), a)$
$\rightarrow \leq$: **if** $(\leq nS) \in \mathcal{L}(a)$, a is not indirectly blocked
and there are m S-neighbors $b_1, \ldots, b_m$ of a with $m > n$, **then**
For each possible pair b_i, b_j, $1 \leq i, j \leq m; i \neq j$ **then**
Generate a graph G'
$\tau(\mathit{Merge}(b_i, b_j)) := (\tau((\leq nS), a) \cup \tau(S, \langle a, b_1\rangle) \ldots \cup \tau(S, \langle a, b_m\rangle))$
if b_j is a nominal node, **then** $\mathit{Merge}(b_i, b_j)$ in G',
else if b_i is a nominal node or ancestor of b_j, **then** $\mathit{Merge}(b_j, b_i)$
else $\mathit{Merge}(b_i, b_j)$ in G'
if b_i is merged into b_j, **then** for each concept C_i in $\mathcal{L}(b_i)$,
$\tau(C_i, b_j) := \tau(C_i, b_i) \cup \tau(\mathit{Merge}(b_i, b_j))$
(similarly for roles merged, and correspondingly for concepts in b_j if merged into b_i)
$\rightarrow O$: **if**, $\{o\} \in \mathcal{L}(a) \cap \mathcal{L}(b)$ and not $a \dot{\neq} b$, **then**
$\mathit{Merge}(a, b)$
$\tau(\mathit{Merge}(a, b)) := \tau(\{o\}, a) \cup \tau(\{o\}, b)$
For each concept C_i in $\mathcal{L}(a)$, $\tau(C_i, b) := \tau(C_i, a) \cup \tau(\mathit{Merge}(a, b))$
(similarly for roles merged, and correspondingly for concepts in $\mathcal{L}(b)$)
$\rightarrow NN$: **if** $(\leq nS) \in \mathcal{L}(a)$, a nominal node, b blockable S-predecessor of a
and there is no m s.t. $1 \leq m \leq n$, $(\leq mS) \in \mathcal{L}(a)$
and there exist m nominal S-neighbors $c_1, \ldots, c_m$ of a s.t. $c_i \dot{\neq} c_j$, $1 \leq j \leq m$, **then**
generate new G_m for each m, $1 \leq m \leq n$
and do the following in each G_m:
$\mathcal{L}(a) = \mathcal{L}(a) \cup \{\leq mS\}$, $\tau((\leq mS), a) := \tau((\leq nS), a) \cup (\tau(S, \langle b, a\rangle)$
create $b_1, \ldots, b_m$; add $b_i \dot{\neq} b_j$ for $1 \leq i \leq j \leq m$. $\tau(\dot{\neq}(b_i, b_j) := \tau((\leq nS), a) \cup \tau(S, \langle b, a\rangle)$
$\mathcal{L}(\langle a, b_i\rangle) = \mathcal{L}(\langle a, b_i\rangle) \cup \{S\}$; $\mathcal{L}(b_i) = \mathcal{L}(b_i) \cup \{\{o_i\}\}$;
$\tau(S, \langle a, b_i\rangle) := \tau((\leq nS), a) \cup \tau(S, \langle b, a\rangle)$; $\tau(\{o_i\}, b_i) := \tau((\leq nS), a) \cup \tau(S, \langle b, a\rangle)$

Fig. 1. BUNDLE tableau expansion rules for OWL DL.

BUNDLE moves to the second tableau and tries to expand it. The second tableau was found by applying the $\rightarrow CE$ rule that added $NatureLover$ to the label of $kevin$, so also the second tableau contains a clash. At this point, BUNDLE joins the tracing functions of the two clashes to find the following InstMinA:

$\{hasAnimal(kevin, tom) \wedge Pet(tom) \rightarrow NatureLover(kevin),$
$hasAnimal(kevin, tom),\ Cat(tom),\ Cat(tom) \rightarrow Pet(tom)\ \}.$

For applying BUNDLE to $\mathcal{SHOIN}(\mathbf{D})$, we further modified the rules $\rightarrow \forall^+$ and $\rightarrow \forall$. For the rule $\rightarrow \forall^+$, we record in the explanation a transitivity axiom for the role R in which only two individuals, those connected by the super role S, are involved. For the rule $\rightarrow \forall$ we make a distinction between the case in which $\forall S_1.C$ was added to $\mathcal{L}(a_1)$ by a chain of applications of $\rightarrow \forall^+$ or not. In the first case, we fully instantiate the transitivity and subrole axioms. In the latter case, we simply obtain $\tau(C, b)$ by combining the explanation of $\forall S_1.C(a_1)$ with that of $(a_1, b) : S_1$.

Example 4. Let us consider the query $Q = eva : Person$ with the following knowledge base:

$Trans(friend) \quad kevin : \forall friend.Person \quad (kevin, lara) : friend \quad (lara, eva) : friend$

BUNDLE first applies the $\rightarrow \forall^+$ rule to $kevin$, adding $\forall friend.Person$ to the label of $lara$. In this case *friend* is considered as a subrole of itself. The tracing function τ is updated as (in predicate logic):

$\tau(\forall friend.Person, lara) = \{\ \forall y.friend(kevin, y) \rightarrow Person(y),$
$friend(kevin, lara),\ \forall z.friend(kevin, lara) \wedge friend(lara, z) \rightarrow friend(kevin, z)\}$

Note that the transitivity axiom is not fully instantiated, the variable z is still present. Then BUNDLE applies the $\rightarrow \forall$ rule to $lara$ adding $Person$ to eva. The tracing function τ is modified as (in predicate logic):

$\tau(Person, eva) = \{\ \forall y.friend(kevin, y) \rightarrow Person(y),\ friend(kevin, lara),$
$friend(lara, eva),\ friend(kevin, lara) \wedge friend(lara, eva) \rightarrow friend(kevin, eva)\}$

Here the transitivity axiom has become ground: all variables have been instantiated. At this point the tableau contains a clash so the algorithm stops and returns the explanation given by $\tau(Person, eva)$.

Example 5. Let us consider the knowledge base

$kevin : \forall kin.Person \quad (kevin, lara) : relative \quad (lara, eva) : ancestor$
$(eva, ann) : ancestor \quad Trans(ancestor) \quad Trans(relative)$
$relative \sqsubseteq kin \quad ancestor \sqsubseteq relative$

The query $Q = ann : Person$ has the explanation (in predicate logic):

$\tau(Person, ann) = \{\ \forall y.kin(kevin, y) \rightarrow Person(y),$
$relative(kevin, lara),\ ancestor(lara, eva),\ ancestor(eva, ann),$
$relative(kevin, lara) \wedge relative(lara, ann) \rightarrow relative(kevin, ann),$
$ancestor(lara, eva) \wedge ancestor(eva, ann) \rightarrow ancestor(lara, ann),$
$relative(kevin, ann) \rightarrow kin(kevin, ann),$
$ancestor(lara, ann) \rightarrow relative(lara, ann)\ \}$

It is easy to see that the explanation entails the axiom represented by the arguments of τ. In general, the following theorem holds.

Theorem 1. *Let Q be an axiom entailed by $\mathcal{K}$ and let S be the output of BUNDLE with the tableau expansion rules of Figure 1 with input Q and $\mathcal{K}$. Then $S \in$ All-InstMinAs$(Q, \mathcal{K})$.*

Proof. The proof, whose full details are given in Theorem 5 of [15], proceeds by induction on the number of rule applications following the proof of Theorem 2 of [7].

The complexity of BUNDLE is similar to the one of Pellet due to the fact that only the $\rightarrow \forall$ rule requires a non-constant number of additional operations. BUNDLE has to find chains of individuals in the current explanation, thus the complexity increment depends on the size of the current explanation.

The tableau algorithm returns a single InstMinA. To solve the problem All-MinAs$(Q, \mathcal{K})$, Pellet uses the *hitting set algorithm* [14]. It starts from a MinA S and initializes a labeled tree called *Hitting Set Tree* (HST) with S as the label of its root v [7]. Then it selects an arbitrary axiom F in S, it removes it from $\mathcal{K}$, generating a new knowledge base $\mathcal{K}' = \mathcal{K} - \{F\}$, and tests the entailment of Q w.r.t. $\mathcal{K}'$. If Q is still entailed, we obtain a new explanation for Q. The algorithm adds a new node w in the tree and a new edge $\langle v, w \rangle$, then it assigns this new explanation to the label of w and the axiom F to the label of the edge. The algorithm repeats this process until the entailment test returns negative: in that case the current node becomes a leaf, the algorithm backtracks to a previous node and repeats these operations until the HST is fully built. The distinct non-leaf nodes of the tree collectively represent the set of all MinAs for the query E.

As in Pellet, to compute All-InstMinAs$(E, \mathcal{K})$ we use the hitting set algorithm that calls BUNDLE's tableau algorithm for computing a single InstMinA. However in BUNDLE we need to eliminate instantiated axioms from the KB. This cannot be done by modifying the KB, since it contains axioms in their general form. Therefore we modified BUNDLE's tableau algorithm to take as input a set of banned instantiated axioms, $BannedInstAxioms$, as well. BUNDLE, before applying a tableau rule, checks whether one of the instantiated axioms to be added to τ is in $BannedInstAxioms$. If so, it does not apply the rule. In this way we can simulate the removal of an instantiated axiom from the theory and apply Pellet's hitting set algorithm to find All-InstMinAs$(E, \mathcal{K})$.

Example 6. Let us consider Example 3. Once an InstMinA is found, BUNDLE applies the hitting set algorithm to it. Thus BUNDLE selects an axiom from the InstMinA and removes it from the knowledge base. Suppose it selects the axiom $Cat(tom)$. At this point, a new run of the tableau algorithm w.r.t. the KB without the axiom $Cat(tom)$ can be executed. BUNDLE can apply the $\rightarrow$ *unfold* rule to the individual *fluffy* and following the same steps used in Example 3 it finds a new InstMinA:

$\{hasAnimal(kevin, fluffy) \wedge Pet(fluffy) \rightarrow NatureLover(kevin),$
$hasAnimal(kevin, fluffy),\ Cat(fluffy),\ Cat(fluffy) \rightarrow Pet(fluffy)\ \}$

Now all the tableaux contain a clash so the hitting set algorithm adds a new node to the HST and selects a new axiom from this second InstMinA to be removed

from the knowledge base. At this point, whatever axiom is removed from the KB, the query $Q = kevin : NatureLover$ will be no longer entailed w.r.t. the KB. Once the HST is fully built, BUNDLE returns the set of all instantiated explanations which is:

$$\begin{aligned}&\textsc{All-InstMinAs}(kevin : NatureLover, \mathcal{K}) = \{\\ &\quad \{\ hasAnimal(kevin, tom) \wedge Pet(tom) \rightarrow NatureLover(kevin),\\ &\quad\ \ hasAnimal(kevin, tom),\ Cat(tom),\ Cat(tom) \rightarrow Pet(tom)\ \},\\ &\quad \{\ hasAnimal(kevin, \mathit{fluffy}) \wedge Pet(\mathit{fluffy}) \rightarrow NatureLover(kevin),\\ &\quad\ \ hasAnimal(kevin, \mathit{fluffy}),\ Cat(\mathit{fluffy}),\ Cat(\mathit{fluffy}) \rightarrow Pet(\mathit{fluffy})\ \}\ \}\end{aligned}$$

Theorem 2. *Let Q be an axiom entailed by $\mathcal{K}$ and let* $\textsc{InstExpHST}(Q, \mathcal{K})$ *be the set of instantiated explanations returned by BUNDLE's hitting set algorithm. Then* $\textsc{InstExpHST}(Q, \mathcal{K}) = \textsc{All-InstMinAs}(Q, \mathcal{K})$.

Proof. See Theorem 6 in [15].

6 Conclusions

We have presented an approach for finding instantiated explanations for expressive DLs such as $\mathcal{SHOIN}(\mathbf{D})$. The approach is implemented in the system BUNDLE for performing inference on probabilistic knowledge bases following the DISPONTE semantics. BUNDLE is available for download from `http://sites.unife.it/ml/bundle`. BUNDLE is thus now able to compute the probability of queries from DISPONTE OWL DL theories. We plan to perform an extensive test of BUNDLE performance on ontologies of various sizes. Moreover, we plan to perform parameter learning of DISPONTE OWL DL KBs by extending EDGE [19], an algorithm that learns DISPONTE $\mathcal{ALC}$ KBs' parameters.

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., Lamma, E., Riguzzi, F., Albani, S.: A distribution semantics for probabilistic ontologies. In: Bobillo, F., et al. (eds.) URSW 2011. CEUR Workshop Proceedings, vol. 778. Sun SITE Central Europe (2011)
4. Carvalho, R.N., Laskey, K.B., Costa, P.C.G.: PR-OWL 2.0 - bridging the gap to OWL semantics. In: Bobillo, F., et al. (eds.) URSW 2010. CEUR Workshop Proceedings, vol. 654. Sun SITE Central Europe (2010)
5. Halaschek-Wiener, C., Kalyanpur, A., Parsia, B.: Extending tableau tracing for ABox updates. Tech. rep., University of Maryland (2006)
6. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in OWL ontologies. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 124–137. Springer (2009)

7. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. Ph.D. thesis, The Graduate School of the University of Maryland (2006)
8. Kalyanpur, A., Parsia, B., Cuenca-Grau, B., Sirin, E.: Tableaux tracing in SHOIN. Tech. Rep. 2005-66, University of Maryland (2005)
9. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 267–280. Springer (2007)
10. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.A.: Debugging unsatisfiable classes in OWL ontologies. J. Web Sem. 3(4), 268–293 (2005)
11. Levesque, H., Lakemeyer, G.: The Logic of Knowledge Bases. MIT Press (2000)
12. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. J. Web Sem. 6(4), 291–308 (2008)
13. Patel-Schneider, P, F., Horrocks, I., Bechhofer, S.: Tutorial on OWL (2003)
14. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. 32(1), 57–95 (1987)
15. Riguzzi, F., Bellodi, E., Lamma, E., , Zese, R.: Probabilistic description logics under the distribution semantics. Tech. Rep. ML-01, University of Ferrara (2013), `http://sites.unife.it/ml/bundle`
16. Riguzzi, F., Bellodi, E., Lamma, E.: Probabilistic Datalog+/- under the distribution semantics. In: Kazakov, Y., Lembo, D., Wolter, F. (eds.) DL 2012. CEUR Workshop Proceedings, vol. 846. Sun SITE Central Europe (2012)
17. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Epistemic and statistical probabilistic ontologies. In: Bobillo, F., et al. (eds.) URSW 2012. CEUR Workshop Proceedings, vol. 900, pp. 3–14. Sun SITE Central Europe (2012)
18. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: BUNDLE: A reasoner for probabilistic ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 183–197. Springer (2013)
19. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Parameter learning for probabilistic ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 265–270. Springer (2013)
20. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP 1995. pp. 715–729. MIT Press (1995)
21. Sattler, U., Calvanese, D., Molitor, R.: Relationships with other formalisms. In: Description Logic Handbook, chap. 4, pp. 137–177. Cambridge University Press (2003)
22. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) IJCAI 2003. pp. 355–362. Morgan Kaufmann (2003)
23. 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)
24. URW3-XG: Uncertainty reasoning for the World Wide Web, final report (2005)