# Tableau Reasoners for Probabilistic Ontologies Exploiting Logic Programming Techniques

Riccardo Zese[1], Elena Bellodi[1], Fabrizio Riguzzi[2], and Evelina Lamma[1]

[1] Dipartimento di Ingegneria – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
[2] Dipartimento di Matematica e Informatica – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
{riccardo.zese,elena.bellodi,evelina.lamma,fabrizio.riguzzi}@unife.it

**Abstract.** The adoption of Description Logics for modeling real world domains within the Semantic Web is exponentially increased in the last years, also due to the availability of a large number of reasoning algorithms. Most of them exploit the tableau algorithm which has to manage non-determinism, a feature that is not easy to handle using procedural languages such as Java or C++. Reasoning on real world domains also requires the capability of managing probabilistic and uncertain information. We thus present TRILL, for "Tableau Reasoner for descrIption Logics in proLog" and TRILL$^P$, for "TRILL powered by Pinpointing formulas", which implement the tableau algorithm and return the probability of queries. TRILL$^P$, instead of the set of explanations for a query, computes a Boolean formula representing them, speeding up the computation.

## Introduction

The Semantic Web aims at making information regarding real world domains available in a form that is understandable by machines [7]. The World Wide Web Consortium is working for realizing this vision by supporting the development of the Web Ontology Language (OWL), a family of knowledge representation formalisms for defining ontologies based on Description Logics (DLs). Moreover, uncertain information is intrinsic to real world domains, thus the combination of probability and logic theories becomes of foremost importance.

Efficient DL reasoners, such us Pellet, RacerPro and HermiT, are able to execute inference on the modeled ontologies, but only a few number of reasoners are able to manage probabilistic information. One of the most common approaches for reasoning is the tableau algorithm that exploits some non-deterministic expansion rules. This requires the developers to implement a search strategy in an or-branching search space able to explore all the non-deterministic choices done during the inference in order to find all the possible explanations for the query.

In this paper, we present two systems which implement a tableau reasoner in Prolog: TRILL for "Tableau Reasoner for descrIption Logics in proLog" and TRILL$^P$ for "TRILL powered by Pinpointing formulas", able to reason on

$\mathcal{SHOIQ}$ DL and on $\mathcal{ALC}$ DL respectively. Prolog search strategy is exploited for taking into account the non-determinism of the tableau rules. They use the Thea2 library [18] for translating OWL files into a Prolog representation in which each axiom is mapped into a fact. TRILL and TRILL$^P$ can check the consistency of a concept and the entailment of an axiom from an ontology, and can compute the probability of the entailment following DISPONTE [13], a semantics for probabilistic ontologies based on the Distribution Semantics [15], one of the most widespread approaches in probabilistic logic programming. The availability of a Prolog implementation of a DL reasoner which follows DISPONTE will also facilitate the development of probabilistic reasoners that can integrate probabilistic logic programming with probabilistic DLs.

## Description Logics

DLs are knowledge representation formalisms that are at the basis of the Semantic Web [1, 2] and are used for modeling ontologies. They possess nice computational properties such as decidability and/or low complexity.

Usually, DLs' syntax is based on concepts and roles which correspond respectively to sets of individuals and sets of pairs of individuals of the domain. We now describe $\mathcal{ALC}$ and we refer to [11] for a description of $\mathcal{SHOIQ}$.

Let $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{I}$ be sets of *atomic concepts*, *atomic roles* and *individuals*, respectively. *Concepts* are defined by induction as follows. Each $C \in \mathbf{C}$ is a concept, $\bot$ and $\top$ are concepts. If $C$, $C_1$ and $C_2$ are concepts and $R \in \mathbf{R}$, then $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$ and $\neg C$ are concepts, as well as $\exists R.C$ and $\forall R.C$. A *TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, where $C$ and $D$ are concepts. We use $C \equiv D$ to abbreviate the conjunction of $C \sqsubseteq D$ 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 $C \in \mathbf{C}$, $R \in \mathbf{R}$ and $a, b \in \mathbf{I}$. A *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ and is usually assigned a semantics in terms of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, 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 $R \in \mathbf{R}$.

A query $Q$ over a KB $\mathcal{K}$ is usually an axiom for which we want to test the entailment from the KB, 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. For example, the entailment of the axiom $C \sqsubseteq D$ may be tested by checking the unsatisfiability of the concept $C \sqcap \neg D$ while the entailment of the axiom $a : C$ may be tested by checking the unsatisfiability of $a : \neg C$.

## Querying Description Logics KBs

The problem of finding explanations for a query has been investigated by various authors [16, 8, 6, 9]. It was called *axiom pinpointing* in [16] and considered as

a non-standard reasoning service useful for tracing derivations and debugging ontologies. In particular, *minimal axiom sets* or *MinAs* for short, also called *explanations*, are introduced in [16].

**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 in [16]. ALL-MINAS$(Q, \mathcal{K})$ is the set of all MinAs for query $Q$ in the knowledge base $\mathcal{K}$.

The tableau algorithm is able to returns a single MinA. To solve MIN-A-ENUM, reasoners written in imperative languages, like Pellet [17], have to implement a search strategy in order to explore the entire search space of the possible explanations. A *tableau* is an ABox represented as a graph $G$ where each node corresponds to an individual $a$ and is labeled with the set of concepts to which $a$ belongs. Each edge $\langle a, b \rangle$ in the graph is labeled with the set of roles to which the couple $(a, b)$ belongs. A *tableau algorithm* proves an axiom by refutation, starting from a tableau that contains the negation of the axiom and repeatedly applying 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. If no clashes are found, the tableau represents a model for the negation of the query, which is thus not entailed. The Prolog language allows developers of reasoning algorithms to exploit Prolog's backtracking facilities instead of implementing a search strategy from scratch.

In [3] the authors consider the problem of finding a *pinpointing formula* instead of ALL-MINAS$(Q, \mathcal{K})$. The pinpointing formula is a monotone Boolean formula in which each Boolean variable corresponds to an axiom of the KB. This formula is built using the variables and the conjunction and disjunction connectives. It compactly encodes the set of all MinAs. Let assume that each axiom $E$ of a KB $\mathcal{K}$ is associated with a propositional variable, indicated with $var(E)$. The set of all propositional variables is indicated with $var(\mathcal{K})$. A valuation $\nu$ of a monotone Boolean formula is the set of propositional variables that are true. For a valuation $\nu \subseteq var(\mathcal{K})$, let $\mathcal{K}_\nu := \{t \in \mathcal{K} | var(t) \in \nu\}$.

**Definition 2 (Pinpointing formula).** *Given a query $Q$ and a KB $\mathcal{K}$, a monotone Boolean formula $\phi$ 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 $\nu$ satisfies $\phi$.*

In Lemma 2.4 of [3] the authors proved that the set of all MinAs can be obtained by transforming the pinpoiting formula into DNF and removing disjuncts implying other disjuncts. The example below illustrates axiom pinpointing and the pinpointing formula.

*Example 1 (Pinpointing formula).* The following KB is inspired by the ontology `people+pets` [12] and show also the Boolean variables associated to axioms: $F_1 = \exists hasAnimal.Pet \sqsubseteq NatureLover$, $F_2 = (kevin, fluffy) : hasAnimal$, $F_3 = (kevin, tom) : hasAnimal$, $F_4 = fluffy : Cat$, $F_5 = tom : Cat$, $F_6 = Cat \sqsubseteq Pet$. It states that individuals that own an animal which is a pet are nature lovers

and that *kevin* owns the animals *fluffy* and *tom*, which are cats. Moreover, cats are pets. Let $Q = kevin : NatureLover$ be the query, then ALL-MINAS$(Q, \mathcal{K})$ = $\{\{F_2, F_4, F_6, F_1\}, \{F_3, F_5, F_6, F_1\}\}$, while the pinpointing formula is $((F_2 \wedge F_4) \vee (F_3 \wedge F_5)) \wedge F_6 \wedge F_1$.

## TRILL and TRILL$^P$

Both TRILL and TRILL$^P$ implement a tableau algorithm, the first solves MIN-A-ENUM while the second computes the pinpointing formula representing the set of MinAs. They can answer concept and role membership queries, subsumption queries and can test the unsatifiability of a concept of the KB or the inconsistency of the entire KB. TRILL and TRILL$^P$ are implemented in Prolog, so the management of the non-determinism of the rules is delegated to the language.

We use the Thea2 library [18] for converting OWL DL KBs into Prolog. Thea2 performs a direct translation of the OWL axioms into Prolog facts. For example, a simple subclass axiom between two named classes $Cat \sqsubseteq Pet$ is written using the `subClassOf/2` predicate as `subClassOf('Cat','Pet')`.

Deterministic and non-deterministic tableau expansion rules are treated differently. Non-deterministic rules are implemented by a predicate that, given the current tableau $Tab$, returns the list of tableaux created by the application of the rule to $Tab$. Deterministic rules are implemented by a predicate that, given the current tableau $Tab$, returns the tableau obtained by the application of the rule to $Tab$. The computation of ALL-MINAS$(Q, \mathcal{K})$ is performed by simply calling `findall/3` over the tableau predicate.

The code of TRILL and TRILL$^P$ is available at `https://sites.google.com/a/unife.it/ml/trill`. Experiments presented in [19] show that Prolog is a viable language for implementing DL reasoning algorithms and that their performances are comparable with those of state-of-art reasoners. In order to popularize DISPONTE, we developed a Web application called "TRILL-on-SWISH" and available at `http://trill.lamping.unife.it`. We exploited SWISH [10], a recently proposed Web framework for logic programming that is based on various features and packages of SWI-Prolog. TRILL-on-SWISH is based on SWISH [10] and allows users to write a KB directly in the web page or load it from a URL, and run TRILL to execute queries.

### Computing the Probability

The aim of our work is to implement algorithms which can compute the probability of queries to KBs following DISPONTE [13]. DISPONTE applies the distribution semantics [15] of probabilistic logic programming to DLs. A program following this semantics defines a probability distribution over normal logic programs called *worlds*. Then the distribution is extended to a joint distribution over worlds and queries from which the probability of a query is obtained by marginalization.

In DISPONTE, a *probabilistic knowledge base* $\mathcal{K}$ contains a set of *probabilistic axioms* which take the form $p :: E$ where $p$ is a real number 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 $E$. For example, a probabilistic concept membership axiom $p :: a : C$ means that we have degree of belief $p$ in $C(a)$. The idea of DISPONTE is to associate independent Boolean random variables to the axioms. To obtain a *world* $w$ we decide whether to include each axiom or not in $w$. A world therefore is a non probabilistic KB that can be assigned a semantics in the usual way. A query is entailed by a world if it is true in every model of the world.

To compute the probability of queries to KBs following the DISPONTE semantics, we can first perform MIN-A-ENUM. Then the explanations must be made mutually exclusive, so that the probabilities of individual explanations are computed and summed. This can be done by assigning independent Boolean random variables to the axioms contained in the explanations and define the Disjunctive Normal Form Boolean formula $f_K$ which models the set of explanations $K$. Thus $f_K(\mathbf{X}) = \bigvee_{Ex \in \text{ALL-MINAS(Q,K)}} \bigwedge_{E_i \in Ex} X_i$ where $\mathbf{X} = \{X_i | (E_i) \in \text{ALL-MINAS(Q,K)}\}$ is the set of Boolean random variables. TRILL$^P$, instead, computes directly a pinpointing formula which is a monotone Boolean formula that represents the set of all MinAs.

Irrespective of which representation of the explanations we choose, a DNF or a general pinpointing formula, we can apply knowledge compilation and transform it into a Binary Decision Diagram (BDD), from which we can compute the probability of the query with a dynamic programming algorithm that is linear in the size of the BDD.

A BDD for a function of Boolean variables is a rooted graph that has one level for each Boolean variable. A node $n$ in a BDD has two children: one corresponding to the 1 value of the variable associated with the level of $n$, indicated with $child_1(n)$, and one corresponding to the 0 value of the variable, indicated with $child_0(n)$. The leaves store either 0 or 1. A BDD performs a Shannon expansion of the Boolean formula $f(\mathbf{X})$, so that, if $X$ is the variable associated with the root level of a BDD, the formula $f(\mathbf{X})$ can be represented as $f(\mathbf{X}) = X \wedge f^X(\mathbf{X}) \vee \overline{X} \wedge f^{\overline{X}}(\mathbf{X})$ where $f^X(\mathbf{X})$ ($f^{\overline{X}}(\mathbf{X})$) is the formula obtained by $f(\mathbf{X})$ by setting $X$ to 1 (0). Now the two disjuncts are pairwise exclusive and the probability of $f(\mathbf{X})$ being true can be computed as $P(f(\mathbf{X})) = P(X)P(f^X(\mathbf{X})) + (1 - P(X))P(f^{\overline{X}}(\mathbf{X}))$ by knowing the probabilities of the Boolean variables of being true.

## Conclusions

In this paper we have presented the algorithm TRILL for reasoning on $\mathcal{SHOIQ}$ KBs and the algorithm TRILL$^P$ for reasoning on $\mathcal{ALC}$ KBs.

In the future we plan to apply various optimizations to our systems in order to better manage the expansion of the tableau. In particular, we plan to carefully choose the rule and node application order. We are also studying an extension of

our systems for managing KBs integrating rules and DL axioms. Moreover, we plan to exploit TRILL for implementing algorithms for learning the parameters of probabilistic DISPONTE KBs, along the lines of [4, 5, 14].

## 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. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. J. Log. Comput. 20(1), 5–34 (2010)
4. Bellodi, E., Riguzzi, F.: Learning the structure of probabilistic logic programs. In: Muggleton, S.H., Tamaddoni-Nezhad, A., Lisi, F.A. (eds.) ILP 2011. LNCS, vol. 7207, pp. 61–75. Springer (2012)
5. Bellodi, E., Riguzzi, F.: Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intel. Data Anal. 17(2), 343–363 (2013)
6. Halaschek-Wiener, C., Kalyanpur, A., Parsia, B.: Extending tableau tracing for ABox updates. Tech. rep., University of Maryland (2006)
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRCPress (2009)
8. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. Ph.D. thesis, The Graduate School of the University of Maryland (2006)
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. Lager, T., Wielemaker, J.: Pengines: Web logic programming made easy. TPLP 14(4-5), 539–552 (2014), http://dx.doi.org/10.1017/S1471068414000192
11. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. J. Web Sem. 6(4), 291–308 (2008)
12. Patel-Schneider, P, F., Horrocks, I., Bechhofer, S.: Tutorial on OWL (2003)
13. 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)
14. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R., Cota, G.: Learning probabilistic description logics. In: URSW III. LNCS, vol. 8816, pp. 63–78. Springer (2014)
15. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP 1995. pp. 715–729. MIT Press (1995)
16. 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)
17. 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)
18. Vassiliadis, V., Wielemaker, J., Mungall, C.: Processing OWL2 ontologies using thea: An application of logic programming. In: OWLED 2009. CEUR Workshop Proceedings, vol. 529. CEUR-WS.org (2009)
19. Zese, R., Bellodi, E., Lamma, E., Riguzzi, F., Aguiari, F.: Semantics and inference for probabilistic description logics. In: URSW III. LNCS, vol. 8816, pp. 79–99. Springer (2014)