Pronto: A Practical Probabilistic Description Logic Reasoner

Pavel Klinov and Bijan Parsia

School of Computer Science
University of Manchester, United Kingdom
{pklinov,bparsia}@cs.man.ac.uk

Abstract. This paper presents a system description of Pronto — the first probabilistic Description Logic reasoner capable of processing knowledge bases containing about a thousand of probabilistic axioms. We describe the design and architecture of the reasoner with an emphasis on the components that implement algorithms which are crucial for achieving such level of scalability. Finally, we present the results of the experimental evaluation of Pronto's performance on series of propositional and non-propositional probabilistic knowledge bases.

1 Introduction

There are many proposed formalisms for combining Description Logics (DLs) with various sorts of uncertainty, although, to our knowledge, none have been used for a production ontology. We believe that this is due to two reasons: 1) there is comparatively little knowledge about how to use these formalisms effectively (or even, which are best suited for what purposes) and 2) there is a severe lack of tooling, in particular, there have been no sufficiently effective reasoners.

This paper describes our work on the second problem. We present Pronto — the reasoner for the probabilistic extension of DL $\mathcal{SHIQ}$ (named P-$\mathcal{SHIQ}$) [1]. This logic can be viewed either as a generalization of the Nilsson's propositional probabilistic logic [2] or as a fragment of first-order probabilistic logic of Halpern and Bacchus [3] [4] (with certain non-monotonic extensions). One attractive feature of these probabilistic logics is that they allow modelers to declaratively describe their uncertain knowledge without fully specifying any probability distribution (in contrast to, for example, Bayesian networks). They are also proper generalizations of their classical counterparts which, in the case of P-$\mathcal{SHIQ}$, means that modelers can take an existing $\mathcal{SHIQ}$ ontology and add probabilistic axioms to capture uncertain, such as statistical, relationships.

In spite of their attractive features Nilsson-style logics have been criticized, partly for the intractability of probabilistic inference. Reasoning procedures are typically implemented via reduction to linear programming but it is well known that corresponding linear programs are exponentially large so the scalability is very limited. Over the last two decades there have been several attempts to overcome that issue in the propositional case which led to some promising results, such as solving the probabilistic satisfiability problem (PSAT) for 800-1000 formulas [5]. It has been unclear whether the methods used to solve large propositional PSATs can be directly applied to PSATs in probabilistic DLs.

To the best of our knowledge, Pronto is the first reasoner for a Nilsson-style probabilistic DL which scalability is comparable (and often better) than scalability of propositional solvers. In particular, it can solve propositional PSATs of the same size as them but can also effectively deal with KBs with non-propositional classical knowledge, such as large $\mathcal{SHIQ}$ terminologies. We present experimental results which show that the level of scalability is comparable in the propositional and non-propositional cases. In addition, Pronto implements all the standard reasoning services for P-$\mathcal{SHIQ}$ as well as useful extra services, in particular, finding *all* minimal unsatisfiable fragments of a KB which is crucial for analyzing large bodies of conflicting probabilistic knowledge.

2 Preliminaries

P-$\mathcal{SHIQ}$ [1] is a probabilistic generalization of the DL $\mathcal{SHIQ}$ [6]. It supports probabilistic subsumptions between arbitrary $\mathcal{SHIQ}$ concepts and a certain class of probabilistic concept assertions (but no form of probabilistic role assertions). Any $\mathcal{SHIQ}$ ontology can be used as a basis for a P-$\mathcal{SHIQ}$ ontology which facilitates transition from classical to probabilistic ontological models. Finally, it combines probabilistic and default reasoning. This allows for a consistent treatment of exceptional individuals and subconcepts.

P-$\mathcal{SHIQ}$ is extends the syntax of $\mathcal{SHIQ}$ with *conditional constraints*, that is, expressions of the form $(D|C)[l, u]$ where C and D are arbitrary $\mathcal{SHIQ}$ concept expressions. Conditional constraints can be used for representing uncertainty in both terminological (TBox) and assertional (ABox) knowledge. A probabilistic TBox (PTBox) is a 2-tuple $PT = (\mathcal{T}, \mathcal{P})$ where $\mathcal{T}$ is a $\mathcal{SHIQ}$ TBox (sometimes called *the classical part*) and $\mathcal{P}$ is a finite set of default conditional constraints (or *probabilistic part*). Informally, a PTBox axiom $(D|C)[l, u]$ means that "generally, if a *randomly* chosen individual belongs to C, its probability of belonging to D is in $[l, u]$". A probabilistic ABox (PABox) is a finite set of strict conditional constraints pertaining to a *concrete* probabilistic individual o. A knowledge base (ontology) in P-$\mathcal{SHIQ}$ is a combination of a PTBox and a collection of PABoxes (one for each probabilistic individual).

P-$\mathcal{SHIQ}$ semantics is based on probability distributions over *possible worlds*, where each possible world is a subset of probabilistically relevant concepts Φ (i.e. concepts used to define conditional constraints). Informally, each world can be thought of as a concept type for a randomly chosen individual. A world is *possible* if there exists an individual that is an instance of all concepts in the world (i.e. the concept type is *realizable*). A KB is satisfiable if there exists a probability distribution that satisfies all conditional constraints.

Standard reasoning tasks in P-$\mathcal{SHIQ}$ include PSAT, tight logical entailment (TLogEnt), and tight lexicographic entailment (TLexEnt). The first two tasks are probabilistic counterparts of classical satisfiability and entailment problems in DL. In contrast, TLexEnt is a *non-monotonic* reasoning task which is reducible to the logical entailment from the largest, conflict-free fragments of the KB.

See [1] for a formal presentation of P-$\mathcal{SHIQ}$ semantics, reasoning procedures and complexity results.

3 Probabilistic Satisfiability Algorithm

In this section we briefly sketch the novel PSAT algorithm implemented in Pronto (see Section 6 for its differences from the previously developed methods). For clarity we will consider a special case of PSAT where the PTBox is of the form $PT = (\mathcal{T}, \{(C_i|\top)[p_i, p_i]\})$ (i.e. all probabilistic statements are unconditional constraints with point-valued probabilities). It is straightforward, but technically awkward, to generalize the procedure to handle conditional interval statements.

A PTBox $PT = (\mathcal{T}, \{(C_i|\top)[p_i, p_i]\})$ is satisfiable iff the following linear program admits a solution:

$$\begin{aligned} &max \sum_{I \in I_{\Phi}} x_I \\ &s.t. \sum_{C_i \in I} x_I = p_i, \text{ for each } (C_i|\top)[p_i, p_i] \in \mathcal{P} \\ &\sum_{I \in I_{\Phi}} x_I = 1 \text{ and all } x_I \geq 0 \end{aligned} \tag{1}$$

where I_{Φ} is the set of all possible worlds for the set of concepts Φ in $\mathcal{T}$.

Let A denote the matrix of linear coefficients in (1). At every step of the simplex algorithm, A is represented as a combination (B, N) where B and N are the submatrices of the *basic* and *non-basic* variables, respectively. Values of non-basic variables are fixed to zero, and the solver proceeds by replacing one basic variable by a non-basic one until the optimal solution is found. The index of the non-basic column is determined according to the following expression [5]:

$$j \in \{1, \dots, |N|\} \text{ s.t. } c_j - u^T A^j \text{ is minimal} \tag{2}$$

where c_j is the objective coefficient for the new variable (it is always equal to 1 in (1)) and u^T is the current dual solution of (1).

As the size of N is exponential in $|\Phi|$, one should compute (2) without examining all columns in N. This is done using the column generation technique in which (2) is treated as an optimization problem with the following objective function:

$$min \; (1 - \sum_{i=1}^{m+1} u_i a_i^j), \; A^j = (a_i^j) \in \{0,1\}^{m+1} \tag{3}$$

Since columns in (1) correspond to possible worlds, $a_i^j = 1$ means that $C_i \in I_j$ while $a_i^j = 0$ means that $\neg C_i \in I_j$, where I_j is the possible world corresponding to the column A^j. Thus it is possible to represent I_j as a conjunctive $\mathcal{SHIQ}$ concept expression as follows (we call the correspondence function η):

$$I_j = \eta(A^j) = \bigsqcap X_i, \text{ where } X_i = \begin{cases} C_i, & a_i^j = 1 \\ \neg C_i, & a_i^j = 0 \end{cases} \tag{4}$$

The critical step is to formulate linear constraints for (3) such that every solution corresponds to a concept expression that is satisfiable w.r.t. $\mathcal{T}$, i.e. a possible world. In the propositional case, where each C_i is a clause, this can be done by employing a well known formulation of SAT as a mixed-integer linear program [7]. For example, if $C_i = x_{i1} \vee \neg x_{i2} \vee x_{i3}$ then (3) will have the constraint $a_i = x_{i1} + (1 - x_{i2}) + x_{i3}$ where all variables are binary.

In the case of an expressive language, such as $\mathcal{SHIQ}$, there appears to be no easy way of determining a set of constraints H for (3) such that its set of solutions in one-to-one correspondence with $\mathcal{I}_\Phi$ (in particular, it is important to ensure that, if A^j is a solution then $\mathcal{T} \not\models \eta(A^j) \sqsubseteq \bot$, i.e. H faithfully captures the TBox $\mathcal{T}$). Instead, Pronto implements a novel *hybrid*, *iterative* procedure to compute H which can be summarized as follows:

```
   Input: PTBox $PT = (\mathcal{T}, \mathcal{P})$, current dual solution $u^T$ of (1)
   Output: New column $A^j$ or null
1  Initialize (3) using $u^T$, $H \leftarrow \emptyset$
2  while $A^j \neq null$ do
3      $A^j \leftarrow$ current optimal solution of (3)
4      if $A^j \neq null$ then
5          if satisfiable($\eta(A^j), \mathcal{T}$) then
6              return $A^j$
7          else
8              add constraints to H that prohibit $A^j$
9          end
10     end
11 end
12 return null;
```

Algorithm 1: Hybrid iterative column generation algorithm

The key steps are 5 and 8. On step 5 the algorithm invokes a $\mathcal{SHIQ}$ SAT solver (in our case, Pellet) to determine if the computed column corresponds to a *possible* world. If yes, the column is returned. If no, the current set of constraints H is augmented on step 8 to exclude A^j from the set of solutions to (3). The algorithm is called “hybrid” because it combines invocations of simplex and $\mathcal{SHIQ}$ solvers and “iterative” because it iteratively tightens the set of solutions to (3) until either a valid column is found or provably no such column exists.

The actual implementation is considerably more involved, in particular, because it is important to minimize the number of calls to the $\mathcal{SHIQ}$ solver.

Therefore the algorithm tries to learn the set H as quickly as possible. Such details, as well as the description of step 8, are beyond the scope of this paper.

4 Architecture

Pronto has layered architecture presented on Figure 1. Each layer has one or more components which invoke other components at the same or lower levels (but not the other way around).

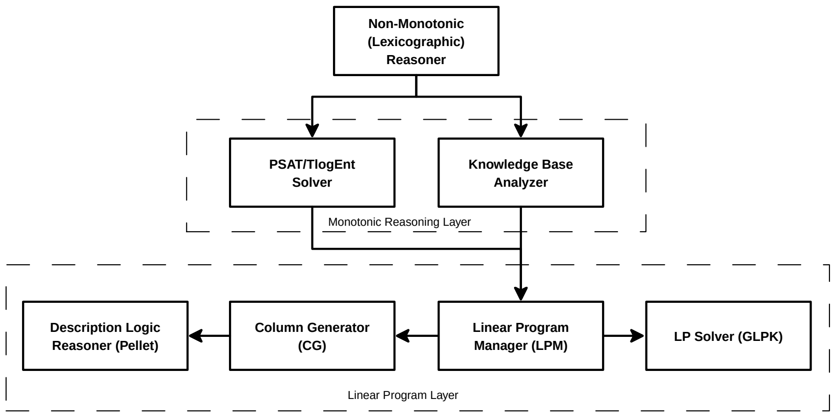
Non-Monotonic (Lexicographic) Reasoner
PSAT/TlogEnt Solver
Knowledge Base Analyzer
Monotonic Reasoning Layer
Description Logic Reasoner (Pellet)
Column Generator (CG)
Linear Program Manager (LPM)
LP Solver (GLPK)
Linear Program Layer

Fig. 1: Architecture of Pronto

Linear Program Layer The components at the lowermost level are responsible for managing linear programs which are optimized in order to solve PSAT or TLogEnt problems. As mentioned earlier, these linear programs usually have exponentially many variables so it is futile to try to represent them explicitly. Therefore, the main function of the components, namely, the linear program manager (LPM) and the column generator (CG), is to generate partial linear systems (1) which have the same optimal objective values as their complete versions.

The LPM is responsible for producing the initial version of the linear program (1), incorporating each new column into it, and checking the optimality (i.e. stopping) criteria. It interacts with the simplex solver, such as GLPK, which solves the current program (1) and returns its primal and dual solutions. The latter is supplied to the CG component.

The CG component implements Algorithm 1. It maintains the binary linear program 3, accepts the dual values u^T, and interacts with Pellet in order to produce improving columns which are then returned back to the LPM. This component implements a number of optimizations such as tuning the binary program (3), learning and re-using constraints H that reflect the structure of the TBox $\mathcal{T}$, and others.

Monotonic Probabilistic Reasoning Layer The components on the next layer use the underlying linear programs to perform monotonic reasoning, i.e. solve PSAT and TLogEnt, and analyze unsatisfiable probabilistic KBs. The first two tasks are straightforward. They amount to checking if a linear system generated by the components of the lower layer admits a solution (PSAT) or solving it to optimality (TLogEnt).

The analysis of an unsatisfiable probabilistic KB is a problem of finding all minimal unsatisfiable subsets of the KB where minimality is defined with respect to the set inclusion. This is essential for 1) computing all maximal satisfiable fragments of the KB during non-monotonic (lexicographic) reasoning, and 2) computing explanations for the results of probabilistic reasoning.

On the linear system level this analysis is equivalent to discovering all irreducible infeasible subsystems (IIS) which can be exponentially many [8]. The task of finding all IISes is somewhat complicated by the fact that the linear systems are never complete, so their set of IISes may not be the same as for the complete system (despite that their objective values are optimal). Therefore the analyzer has to repeatedly invoke the LP layer components to enrich the system with new columns after finding each new IIS. Apart of that the analysis follows the classical model-based diagnosis methodology based on hitting set trees [9].

Non-Monotonic Probabilistic Reasoning Layer The uppermost layer consists of a single component: the lexicographic reasoner. It implements the TLexEnt algorithm which relies on the KB analyzer and the TLogEnt reasoner. TLexEnt is equivalent to solving TLogEnt for all lexicographically minimal subsets of the KB [1]. The latter are computed in three phases: 1) KB analyzer computes the structure called *conflict graph* which represents conflicts between pieces of probabilistic knowledge, 2) the graph is used to rank conflicting statements by specificity, and 3) conflicts are resolved by preferring more specific statements to less specific (the resulting conflict-free fragments of the KB are lexicographically minimal). The last phase may fail if equally specific statements happen to be in conflict. In that case the reasoner reports probabilistic inconsistency (which is different from probabilistic unsatisfiability, see [1] for details).

5 Experimental Evaluation

We have conducted two experiments to demonstrate that Pronto is practical to use on ontologies of realistic size. Both experiments evaluate the performance of solving PSAT (since all other reasoning tasks are reducible to it). KBs in the first experiments are randomly generated sets of propositional conditional constraints with no classical part. KBs in the second experiment are randomly generated conditional constraints with TBoxes from real-life ontologies represented in expressive DLs: the GeoSkills ontology and the SWEET Process ontology, both taken from the TONES repository[1]. There does not seem to be an easy way to "propositionalize" these ontologies in order to use propositional PSAT solvers.

The results, which are presented in Table 1, were averaged over 10 PSAT instances solved for each size. In the table n stands for the number of concepts in the classical part of the KB and m — for the number of conditional constraints. We used a conventional PC with 2GHz CPU and 2GB RAM.

Table 1: Performance on random propositional and non-propositional probabilistic KBs

Propositional KBs			GeoSkills ($\mathcal{ALCHOIN}$)			Process ($\mathcal{ALCHOF}$)		
n	m	Time (s)	n	m	Time (s)	n	m	Time (s)
50	100	3	603	100	20	1537	100	5
100	200	12	603	200	38	1537	200	30
250	500	30	603	500	151	1537	500	92
500	1000	291	603	1000	332	1537	1000	176

The results show that Pronto performs comparably to the state-of-the-art propositional PSAT solvers on propositional KBs [5][2]. However, it can also handle probabilistic KBs of the same size defined over highly expressive DL ontologies without significant loss of performance. In fact, expressive TBoxes may improve the performance because they tend to shrink the space of all potential columns (possible worlds) by constraining models. This can explain why the probabilistic extension of the Process ontology is sometimes easier than random propositional KBs of the same size. We are now in the process of investigating this and other complexity factors through a more comprehensive and systematic evaluation. It is anticipated that understanding of such factors will help to develop corresponding optimizations.

6 Related Work

To the best of our knowledge there are no other reasoners for Nilsson-style probabilistic DLs. One exception is [10] but that system does not implement any technique to reduce the size of the linear programs and is, therefore, limited to 10-15 probabilistic statements.

Among other tools the most closely related are propositional PSAT solvers which also use the column generation method to cope with the size of the linear systems [5] [11]. However, the main difference between Pronto and those tools lies in the optimization problem (3) used to produce columns. They can encode the

[1] http://owl.cs.manchester.ac.uk/repository/

[2] We must mention that our problem generation methodology is slightly different. Hansen and Perron experimented with unconditional constraints and point-valued probabilities while we generated mixtures of conditional and unconditional statements with intervals which, as we believe, are more useful for practical modeling.

entire structure of the *propositional* KB in the set of constraints H for (3) while Pronto employs a $\mathcal{SHIQ}$ reasoner to compute such set iteratively. One important consequence is that Pronto can be used for a Nilsson-style probabilistic extension to *any* logic for which a SAT solver is available.

7 Summary

The primary conclusion of this work is that Pronto is *practical* to use for probabilistic ontologies of a realistic size. PSAT and the various entailment problems for P-$\mathcal{SHIQ}$ are EXPTIME-complete, so, as with any logic in the $\mathcal{SH}$ family, practicality, efficiency, and scalability claims must be carefully qualified. However, Pronto handles P-$\mathcal{SHIQ}$ ontologies which are comparable in size to, for example, various handcrafted Bayesian networks[3], which gives good reasons to believe that it will prove practical for future probabilistic ontologies.

More generally, our work also shows that P-$\mathcal{SHIQ}$, as well as other Nilsson-style extensions to DLs, can be *practical* (in terms of reasoning complexity) probabilistic ontology languages. In a certain sense they are no less practical than $\mathcal{SHIQ}$. Although there will certainly be P-$\mathcal{SHIQ}$ ontologies that will defeat the current version of Pronto, this is also the case with any NP-hard logic. However, now it makes sense for modelers to experiment with P-$\mathcal{SHIQ}$ and report troublesome ontologies, so that tool developers can adapt to new challenges.

References

1. Lukasiewicz, T.: Expressive probabilistic description logics. Artificial Intelligence **172(6-7)** (2008) 852–883
2. Nilsson, N.J.: Probabilistic logic. Artificial Intelligence **28**(1) (1986) 71–87
3. Halpern, J.Y.: An analysis of first-order logics of probability. Artificial Intelligence **46** (1990) 311–350
4. Bacchus, F.: Representing and reasoning with probabilistic knowledge. MIT Press (1990)
5. Hansen, P., Perron, S.: Merging the local and global approaches to probabilistic satisfiability. Int. Journal of Approximate Reasoning **47**(2) (2008) 125–140
6. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Journal of the IGPL **8**(3) (2000)
7. Hooker, J.N.: Quantitative approach to logical reasoning. Decision Support Systems **4** (1988) 45–69
8. Gleeson, J., Ryan, J.: Identifying minimally infeasible subsystems of inequalities. INFORMS Journal on Computing **2**(1) (1990) 61–63
9. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **32** (1987) 57–95
10. Näth, T.H., Möller, R.: ContraBovemRufum: A system for probabilistic lexicographic entailment. In: Description Logics. (2008)
11. de Souza Andrade, P.S., da Rocha, J.C.F., Couto, D.P., da Costa Teves, A., Cozman, F.G.: A toolset for propositional probabilistic logic. In: Encontro Nacional de Inteligencia Artificial. (2007) 1371–1380

[3] See, in particular, the repository at http://genie.sis.pitt.edu/networks.html