Combining Logic and Probability: P-log Perspective

Evgenii Balai
Texas Tech University, Lubbock, Texas
evgenii.balai@ttu.edu

Abstract

My research focuses on investigation and improvement of knowledge representation (KR) language P-log which was designed to reason about both logical and probabilistic knowledge. In particular, I aim to extend P-log with new constructs, clarify its semantics, develop a new efficient inference engine for it and establish its relationship with other related formalisms. Successful completion of this work will greatly increase the scope of practical applications of the language.

1 Introduction

The development of KR languages and inference engines for them has been an important direction of AI research during the last decades. Having a powerful language with an efficient inference engine allows a reasoner to reduce certain AI tasks to representing them in a given language and invoking its inference engine to compute the answer. Recently a substantial work was done to develop languages combining logical and probabilistic reasoning (e.g, Problog [De Raedt *et al.*, 2007]). However, most such languages are based on classical first-order logic. In [Baral *et al.*, 2009] the authors define a language named P-log which combines a powerful non-monotonic logical formalism Answer Set Prolog (ASP) [Gelfond and Lifschitz, 1991] with probability theory based on Causal Bayesian Networks [Pearl, 2009]. The nonmonotonocity provides numerous advantages, such as natural representation of defaults, belief updates, abductive reasoning, etc. This language is the focus of my research, where the main directions are as follows:

- improving the usability of P-log by
  - (a) providing means for specifying the sorts (types) of objects dealt with by a program and designing corresponding type-checking algorithms,
  - (b) extending the language with new constructs to increase its expressive power,
  - (c) developing a new efficient inference engine for the language which will increase the scope of its applicability; and
- establishing the relationship between P-log and other related formalisms.

In the next section I describe the improvements of the usability of the language. The last section summarizes the progress and describes the work to be completed.

2 Improving the Usability of P-log

2.1 Sorts and Error-Checking Algorithms

In the original P-log from [Baral *et al.*, 2009] only very simple sorts of objects can be specified concisely. In published work [Balai *et al.*, 2013] we developed a framework which allows for concise representations of sorts hierarchies in a logic program and automatically detects certain semantic errors in it. The framework was implemented and successfully used in AI classes, various research projects (e.g, robotics application in [Zhang *et al.*, 2014]), and in the development of other logical languages at our lab (e.g, [Kahl *et al.*, 2015]). This work, including the implementation, will be expanded to P-log.

2.2 Extending the Language and Clarifying Its Semantics

I extend the language by allowing certain language constructs to occur in the bodies of its logical rules. This simplifies the use P-log for causal reasoning. Clarifications are primarily concerned with P-log treatment of partial functions.

2.3 Algorithms

A straightforward inference in P-log requires the computation of all possible worlds of a program. In [Zhu, 2012] Weijun designs an algorithm which performs a more optimal computation. The algorithm from [Zhu, 2012] however has some restrictions. Firstly, it was proved to be sound only for a comparatively small class of programs. Secondly, I found a certain number of errors in its pseudocode and implementation.

In the thesis I define a class of programs which extend the class used in the algorithm from [Zhu, 2012], prove their coherency (an important property defined in [Baral *et al.*, 2009]) and design a new inference algorithm which works for programs in this class. The algorithm consists of two main steps as shown below:

Algorithm 1 Inference in P-log

Input: A P-log program Π, a query Q to Π
Output: The probability of Q with respect to Π
1: Π_Q := $Simplify(\Pi,\ Q)$
2: **Return** $Compute\_Probability(\Pi_Q, Q)$

$Simplify(\Pi, Q)$ returns a program Π_Q which is normally smaller than Π and preserves the probability of Q. The procedure is similar to the one defined by magic sets [Faber *et al.*, 2007] for ASP programs. $Compute\_Probability(\Pi_Q,Q)$, similarly to DPLL search, constructs a tree, whose nodes are partial interpretations of functional symbols of Π which is sufficient to compute the necessary probability. For example, consider the following program Π:

$a, b, c, d, e, f, g : boolean$	$r5 : pr(d) = 0.6$
$r1 :\ random(a).$	$r6 :\ b \leftarrow d, not\ c.$
$r2 :\ f \leftarrow a.$	$r7 :\ c \leftarrow d, not\ b.$
$r3 :\ e.$	$r8 :\ \leftarrow d, b.$
$r4 :\ random(d) \leftarrow e.$	$r9 :\ random(g) \leftarrow c.$

and query c. $Simplify(\Pi, c)$ drops the first two rules from the program, since c does not depend on functional symbols a and f. $Compute\_Probability(\Pi_c, c)$ constructs the tree:

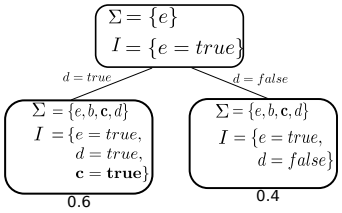

Figure 1: The tree to answer query c in Π_c

Each node of the tree stores a pair (Σ, I), a set of functional symbols of the program and their partial interpretation. The children of a node of the tree are obtained by:

- selecting a **random functional symbol** of the program whose value has not been decided;
- assigning **possible values** to the selected random functional symbol; and
- computing **the consequences** (i.e, the values of other functional symbols which do not depend on random functional symbols not decided so far).

To extend the root of the tree shown on Figure 1, the functional symbol d was selected. Two children corresponding to the two possible values of d ($true$ and $false$) were added. The consequences are values of functional symbols b and c.

Note that there is no need to extend the tree further by selecting g, since every leaf already includes c from the query.

Each leaf node of the tree has an assigned probability. The probability of the query c is equal to the probability of the left leaf, 0.6, obtained from the pr-atom $pr(d) = 0.6$ of Π.

3 Progress and Future Work

So far I have:

- defined extensions of the language (including those by sort definitions from [Balai *et al.*, 2013]);
- designed a pseudo-code for a new inference algorithm;
- defined a new class of P-log programs which extend those from [Zhu, 2012] and [Baral *et al.*, 2009];
- investigated the relationship between P-log and LP$^{\text{MLN}}$ in [Balai and Gelfond, 2016]

Future work related to my thesis will consist of :

- completing the work on the language extensions;
- proving the correctness of the designed algorithm, implementing it and investigating the efficiency;

References

[Balai and Gelfond, 2016] Evgenii Balai and Michael Gelfond. On the relationship between P-log and LP$^{\text{MLN}}$. In *IJCAI 2016*, 2016.

[Balai *et al.*, 2013] Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. Towards answer set programming with sorts. In *Logic Programming and Nonmonotonic Reasoning*, pages 135–147. Springer, 2013.

[Baral *et al.*, 2009] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(01):57–144, 2009.

[De Raedt *et al.*, 2007] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467, 2007.

[Faber *et al.*, 2007] Wolfgang Faber, Gianluigi Greco, and Nicola Leone. Magic sets and their application to data integration. *Journal of Computer and System Sciences*, 73(4):584–609, 2007.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385, 1991.

[Kahl *et al.*, 2015] Patrick Kahl, Richard Watson, Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. The language of epistemic specifications (refined) including a prototype solver. *Journal of Logic and Computation*, page exv065, 2015.

[Pearl, 2009] Judea Pearl. *Causality*. Cambridge university press, 2009.

[Zhang *et al.*, 2014] Shiqi Zhang, Mohan Sridharan, Michael Gelfond, and Jeremy Wyatt. Integrating probabilistic graphical models and declarative programming for knowledge representation and reasoning in robotics. In *Planning and Robotics (PlanRob) Workshop at ICAPS, Portsmouth, USA*, 2014.

[Zhu, 2012] Weijun Zhu. *Plog: Its algorithms and applications*. PhD thesis, Texas Tech University, 2012.