Handling Uncertainty in Answer Set Programming

Yi Wang and **Joohyung Lee**
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, USA
{ywang485, joolee}@asu.edu

Abstract

We present a probabilistic extension of logic programs under the stable model semantics, inspired by the concept of Markov Logic Networks. The proposed language takes advantage of both formalisms in a single framework, allowing us to represent commonsense reasoning problems that require both logical and probabilistic reasoning in an intuitive and elaboration tolerant way.

Introduction

Answer Set Programming (ASP) is a successful logic programming paradigm that takes advantage of nonmonotonicity of the underlying semantics, the stable model semantics. Many useful knowledge representation constructs and efficient solvers allow ASP to handle various commonsense reasoning problems elegantly and efficiently. However, difficulty still remains when it comes to domains with uncertainty. Imprecise and vague information cannot be handled well since the stable model semantics is mainly restricted to Boolean values. Further, probabilistic information cannot be handled since the stable model semantics does not distinguish which stable models are more likely to be true.

To address the first aspect of the issue, several approaches to incorporating fuzzy logic into ASP have been proposed, such as (Lukasiewicz 2006). However, most of the work is limited to simple rules. In our previous work (Lee and Wang 2014), we proposed a stable model semantics for fuzzy propositional formulas, which properly generalizes both fuzzy logic and the Boolean stable model semantics, as well as many existing approaches to combining them. The resulting language combines the many-valued nature of fuzzy logic and the nonmonotonicity of stable model semantics, and consequently shows competence in commonsense reasoning involving fuzzy values.

In this work, we focus on the other aspect of the issue, namely handling probabilistic information in commonsense reasoning. We adapt the idea of Markov Logic Networks (MLN) (Richardson and Domingos 2006), a well-known approach to combining first-order logic and probabilistic graphical models in a single framework, to the context of logic programming. The resulting language LP$^{\mathrm{MLN}}$ combines the attractive features of each of the stable model semantics and MLN.

Language LP$^{\mathrm{MLN}}$

Since a Markov Logic Network is based on the standard first-order logic semantics, it "has the drawback that it cannot express (non-ground) inductive definitions" (Fierens et al. 2013). Consider the following domain description ("α" denotes "hard weight").

$$\begin{array}{ll} w_1: & \mathit{Edge}(1,2) \\ w_2: & \mathit{Edge}(2,3) \\ \ldots & \\ \alpha: & \mathit{Path}(x,y) \leftarrow \mathit{Edge}(x,y) \\ \alpha: & \mathit{Path}(x,y) \leftarrow \mathit{Path}(x,z),\ \mathit{Path}(z,y). \end{array}$$

The above weighted rules are intended to describe probabilistic reachability in a graph, which is induced by the probabilities of the involved edges. The relation *Path*, describing the reachability between two vertices, is supposed to be the transitive closure of the relation *Edge*. However, under the MLN semantics, this is not the case.

We propose the language LP$^{\mathrm{MLN}}$, which overcomes such a limitation. The syntax of an LP$^{\mathrm{MLN}}$ program is essentially the same as that of an MLN instance, except that weighted formulas are replaced by weighted rules.

Similar to MLN, the weight of each rule can be either a real number, or the symbol α that marks a rule a "hard rule." The weight of each interpretation is obtained from the weights of the rules that form the maximal subset of the program for which the interpretation is a stable model.

More precisely, we assume a finite first-order signature σ that contains no function constants of positive arity, so that any non-ground LP$^{\mathrm{MLN}}$ program can be identified with its ground instance. Without loss of generality, we consider ground (i.e., variable-free) LP$^{\mathrm{MLN}}$ programs. For any ground LP$^{\mathrm{MLN}}$ program $\mathbb{P}$ of signature σ, consisting of a set of weighted rules $w : R$, and any Herbrand interpretation I of σ, we define $\mathbb{P}_I$ to be the set of rules in $\mathbb{P}$ which are satisfied by I. The weight of I, denoted by $W_{\mathbb{P}}(I)$, is defined as

$$W_{\mathbb{P}}(I) = exp\left(\sum_{\substack{w:R \in \mathbb{P} \\ R \in \mathbb{P}_I}} w \right)$$

if I is a stable model of $\mathbb{P}_I$; otherwise $W_{\mathbb{P}}(I) = 0$. The probability of I under $\mathbb{P}$, denoted $Pr_{\mathbb{P}}[I]$, is defined as

$$Pr_{\mathbb{P}}[I] = \lim_{\alpha \to \infty} \frac{W_{\mathbb{P}}(I)}{\sum_{J \in PW} W_{\mathbb{P}}(J)}$$

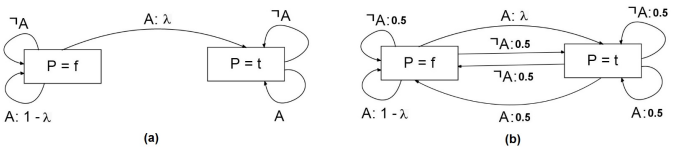

Figure 1: The transition system with probabilistic effects, defined by LP^{MLN} semantics (a) and MLN semantics (b)

where PW is the set of all Herbrand interpretations of σ.

Inductive definitions are correctly handled in LP^{MLN} since it adopts the stable model semantics in place of the standard first-order logic semantics in MLN. For instance, the weighted rules given at the beginning of this section yields the expected result under the LP^{MLN} semantics.

Any logic program under the stable model semantics can be embedded in LP^{MLN} by assigning the hard weight to each rule. MLN can also be embedded in LP^{MLN} via choice formulas which exempt atoms from minimization. LP^{MLN} programs can be turned into MLN instances via the concept of loop formulas (Ferraris, Lee, and Lifschitz 2006). This result allows us to use an existing implementation of MLN, such as Alchemy, to effectively compute "tight" LP^{MLN} programs.

Probabilistic Transitions by LP^{MLN}

LP^{MLN} can be used to model transition systems where actions may have probabilistic effects. For example, the transition system shown in Figure 1(a) can be encoded as the following LP^{MLN} program.

$$
\begin{array}{rlrl}
ln(\lambda): & Aux_i & \alpha: & \{P_{i+1}\} \leftarrow P_i \\
ln(1-\lambda): & \leftarrow Aux_i & \alpha: & \{NP_{i+1}\} \leftarrow NP_i \\
\alpha: & \leftarrow P_i, NP_i & \alpha: & \{A_i\} \\
\alpha: & \leftarrow not\ P_i, not\ NP_i & \alpha: & \{P_0\} \\
\alpha: & P_{i+1} \leftarrow A_i, Aux_i & \alpha: & \{NP_0\}
\end{array}
$$

Here i is a schematic variable ranging over $\{0, \ldots, maxstep - 1\}$. The atom Aux_i represents the probabilistic choice for the success of action A_i. The 3rd and 4th rules say that P_i and NP_i are complementary. The 5th rule defines the probabilistic effect of A_i. The 6th and 7th rules represent the commonsense law of inertia, where we use $\{H\} \leftarrow Body$ to denote the rule $H \leftarrow Body, not\ not\ H$. The last three rules specify that the execution of A at each step and the initial value of P are arbitrary.

Under the MLN semantics, the same program specifies a different probabilistic transition system (Figure 1(b)), which is not aligned with the commonsense understanding of the description. In this transition system, fluents can change arbitrarily even when no relevant actions are executed.

As an application of the idea of using LP^{MLN} to define probabilistic transition systems, a probabilistic variant of the Wolf, Sheep and Cabbage puzzle can be encoded in LP^{MLN}. We consider an elaboration in which with some probability p, eating does not happen even when the farmer is not present. While the minimal length plan for the original puzzle involves 17 actions of loading, moving and unloading, the elaboration has a new minimal plan containing 11 actions only, including two steps in which the wolf does not eat sheep when the farmer is not around. We formalized this domain in LP^{MLN}, and, based on the reduction of LP^{MLN} to MLN, used Alchemy to check that the probability of the success of this plan is $p \times p$ and that of the original 17 step plan is 1.

Discussion

LP^{MLN} is related to many earlier work. Only a few of them are mentioned here due to lack of space. It is not difficult to embed ProbLog (Raedt, Kimmig, and Toivonen 2007) in LP^{MLN}. The language is also closely related to P-log (Baral, Gelfond, and Rushton 2009). The LP^{MLN} formalization of probabilistic transition systems is related to P$\mathcal{C}$+ (Eiter and Lukasiewicz 2003), which extends action language $\mathcal{C}$+ for probabilistic reasoning about actions.

The work presented here calls for more future work. One may design a native computation algorithm for LP^{MLN} which would be feasible to handle certain "non-tight" programs. We expect many results established in ASP may carry over to MLN, and vice versa, which may provide a new opportunity for probabilistic answer set programming.

Acknowledgements We are grateful to Chitta Baral, Michael Bartholomew, Amelia Harrison, Vladimir Lifschitz, Yunsong Meng, and the anonymous referees for their useful comments. This work was partially supported by the National Science Foundation under Grant IIS-1319794 and by the South Korea IT R&D program MKE/KIAT 2010-TD-300404-001, and the Brain Pool Korea program.

References

Baral, C.; Gelfond, M.; and Rushton, J. N. 2009. Probabilistic reasoning with answer sets. *TPLP* 9(1):57–144.

Eiter, T., and Lukasiewicz, T. 2003. Probabilistic reasoning about actions in nonmonotonic causal theories. In *Proceedings Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-2003)*, 192–199. Morgan Kaufmann Publishers.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2006. A generalization of the Lin-Zhao theorem. *Annals of Mathematics and Artificial Intelligence* 47:79–101.

Fierens, D.; Van den Broeck, G.; Renkens, J.; Shterionov, D.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2014. Inference and learning in probabilistic logic programs using weighted boolean formulas. *TPLP*, 44 pages.

Lee, J., and Wang, Y. 2014. Stable models of fuzzy propositional formulas. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*, 326–339.

Lukasiewicz, T. 2006. Fuzzy description logic programs under the answer set semantics for the semantic web. In Eiter, T.; Franconi, E.; Hodgson, R.; and Stephens, S., eds., *RuleML*, 89–96. IEEE Computer Society.

Raedt, L. D.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, 2462–2467.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.