plingo: A system for probabilistic reasoning in clingo based on lpmln

Susana Hahn[1 [0000−0003−2622−2632]], Tomi Janhunen[2 [0000−0002−2029−7708]], Roland Kaminski[1 [0000−0002−1361−6045]], Javier Romero[1 [0000−0001−5546−9939]], Nicolas Rühling[1 [0000−0001−5157−6788]], and Torsten Schaub[1 [0000−0002−7456−041X]]

[1] University of Potsdam, Germany
[2] University of Tampere, Finland

1 Introduction

Answer Set Programming (ASP; [7]) offers a rich knowledge representation language along with powerful solving technology. In the last years, several probabilisitic extensions of ASP have been proposed, among them LP^{MLN} [5], *ProbLog* [9], and *P-log* [1].

In this work, we present an extension of the ASP system *clingo*, called *plingo*, that features various probabilistic reasoning modes. *Plingo* is centered on LP^{MLN}, a probabilistic extension of ASP based upon a weight scheme from Markov Logic [10]. We rely on translations from *ProbLog* and *P-log* to LP^{MLN} [5, 6], respectively, to capture these approaches as well. In fact, LP^{MLN} has already been implemented in the system *lpmln2asp* [4] by mapping LP^{MLN}-based reasoning into reasoning modes in *clingo* (viz. optimization and enumeration of stable models). As such, *plingo* can be seen as a re-implementation of *lpmln2asp* that is well integrated with *clingo* by using its multi-shot and theory reasoning functionalities. *plingo* offers three alternative frontends, for LP^{MLN}, *P-log*, and *ProbLog*, featuring dedicated language constructs that are in turn translated into the format described above. As regards solving, *plingo* follows the approach of *lpmln2asp* of reducing probabilistic reasoning to *clingo*'s regular optimization and enumeration modes. In addition, *plingo* features an approximation method that calculates probabilities using only the most probable k stable models for an input parameter k. This is accomplished by an improved implementation of answer set enumeration in the order of optimality [8]. We have empirically evaluated *plingo*'s performance by contrasting it to original implementations of LP^{MLN}, *ProbLog* and *P-log*. [3]

2 The language of *plingo*

The main idea of the system is to keep the input language of *clingo*, and re-interpret weak constraints at priority level 0 as soft integrity constraints. These constraints are not considered to determine the optimal stable models, but instead are used to determine the weights of those models, from which their probabilities are calculated. We define a *plingo* program Π as a logic program in the language of *clingo*, and we let $OSM^{pl}(\Pi)$

[3] The full version of this paper is available at `https://arxiv.org/abs/2206.11515`.

denote the optimal models of Π without considering weak constraints at level 0, and $Cost_\Pi(X, 0)$ denote the cost of the interpretation X at priority level 0, according to the definitions of [2]. Then, the *weight* $W_\Pi(X)$ of an interpretation X and its *probability* $P_\Pi(X)$ are defined as:

$$W_\Pi(X) = \begin{cases} TW(\Pi) & \text{if } X \in OSM^{pl}(\Pi) \\ 0 & \text{otherwise} \end{cases} \quad \text{and } P_\Pi(X) = \frac{W_\Pi(X)}{\sum_{Y \in OSM^{pl}(\Pi)} W_\Pi(Y)},$$

where $TW(\Pi) = exp(Cost_\Pi(X, 0))$.

3 The system *plingo*

The implementation of *plingo* is based on *clingo* and its Python API (v5.5, [3]). The system architecture is described in Figure 1. The input is a logic program written in some probabilistic language: *plingo*, LP^{MLN}, *ProbLog* or *P-log*. For *plingo*, the input language (orange element of Figure 1) is the same as the input language of *clingo*, except for the fact that the weights of the weak constraints can be strings representing real numbers. For the other languages, the system uses the corresponding frontends, that translate the input logic programs (yellow elements of Figure 1) to the input language of *plingo* using the translations from [5, 6]. *Plingo* can be used to solve two reasoning tasks: finding the most probable stable model and marginal inference.

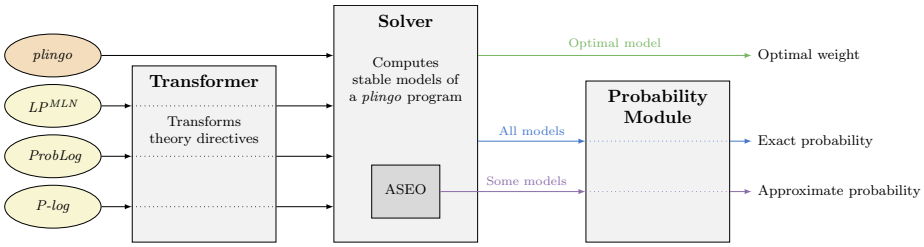

Fig. 1. System architecture of *plingo*. Inputs are yellow for the different frontends provided. Modules of the system are gray boxes. The green flow corresponds to MAP inference, the blue to Exact Marginal Inference, and the purple to Approximate Marginal Inference.

We evaluate our system *plingo* and compare it to native implementations of LP^{MLN}, *ProbLog* and *P-log*.[4] While *lpmln2asp* and *plog* also use *clingo* as backend, *problog* is based on the well-founded semantics and has a algorithm for marginal inference using knowledge compilation. Our results show that for the task of marginal inference the former systems have comparable runtimes, while *problog* clearly outperforms them. However, the approximation algorithm of *plingo* gives us good results with a fast runtime.

[4] Available, respectively, at `https://github.com/azreasoners/lpmln`, `https://github.com/ML-KULeuven/problog`, and `https://github.com/iensen/plog2.0`.

References

1. Baral, C., Gelfond, M., Rushton, J.: Probabilistic reasoning with answer sets. Theory and Practice of Logic Programming **9**(1), 57–144 (2009)
2. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: ASP-Core-2: Input language format. Available at `https://www.mat.unical.it/aspcomp2013/ASPStandardization` (2012)
3. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: Carro, M., King, A. (eds.) Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16). OpenAccess Series in Informatics (OASIcs), vol. 52, pp. 2:1–2:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2016)
4. Lee, J., Talsania, S., Wang, Y.: Computing LPMLN using ASP and MLN solvers. Theory and Practice of Logic Programming **17**(5-6), 942–960 (2017)
5. Lee, J., Wang, Y.: Weighted rules under the stable model semantics. In: C. Baral, J. Delgrande, F.W. (ed.) Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning. pp. 145–154. AAAI/MIT Press (2016)
6. Lee, J., Yang, Z.: LPMLN, weak constraints and P-log. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence. pp. 1170–1177 (2017)
7. Lifschitz, V.: Answer set programming and plan generation. Artificial Intelligence **138**(1-2), 39–54 (2002)
8. Pajunen, J., Janhunen, T.: Solution enumeration by optimality in answer set programming. Theory and Practice of Logic Programming **21**(1), 750–767 (2021)
9. Raedt, L.D., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic prolog and its applications in link discovery. In: Proceedings of the Twenty-second National Conference on Artificial Intelligence (AAAI'07). pp. 2468–2473. AAAI Press (2007)
10. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2), 107–136 (2006)