ProbLog

Angelika Kimmig[1], Bernd Gutmann[1],
Theofrastos Mantadelis[1], Guy Van den Broeck[1],
Vítor Santos Costa[2], Gerda Janssens[1], Luc De Raedt[1]

[1]Department of Computer Science, K.U. Leuven, Belgium
[2]CRACS-INESC LA and DCC/FCUP, Portugal

ProbLog [1] is a probabilistic programming language that extends Prolog along the lines of Sato's distribution semantics. Its development focusses especially on machine learning techniques and implementation aspects. The ProbLog implementation is publicly available as part of Yap Prolog at `http://www.dcc.fc.up.pt/~vsc/Yap/` [2]; more information, including all publications, can be found at `http://dtai.cs.kuleuven.be/problog`.

From a modeling perspective, a ProbLog program has two parts: a probabilistic part that defines a probability distribution over truth values of a subset of the program's Herbrand base, and a logical part that derives truth values of remaining atoms just as in Prolog. While the latter part simply contains Prolog clauses, the former is specified by *probabilistic facts* $\mathtt{p} :: \mathtt{fact}$, meaning that `fact` is true with probability `p`. All these are probabilistically independent; in case they contain variables, all ground instances are independent as well. For ease of modeling, ProbLog also allows the use of *annotated disjunctions* $\mathtt{p_1} :: \mathtt{h_1}; \ldots; \mathtt{p_n} :: \mathtt{h_n} \leftarrow \mathtt{body}$ with $\sum_{i=1}^{n} \mathtt{p_i} \leq 1$, meaning that if `body` is true, one of the $\mathtt{h_i}$ will be true according to the specified probabilities $\mathtt{p_i}$. If the probabilities do not sum to one, it is also possible that none of the $\mathtt{h_i}$ is true (with probability $1 - \sum_{i=1}^{n} \mathtt{p_i}$). Annotated disjunctions are internally preprocessed into probabilistic facts.

The following ProbLog program models a tiny social network, where any two persons living in the same town are either friends, family, colleagues, or in no direct relationship at all, and any two persons live in the same town with a probability of 0.3. Based on these direct relationships, the predicate `knows/2` then allows one to infer indirect connections as well.

```
person(ann). person(bob). person(cat). person(dan). person(ed).
0.2 :: link(A,B,friend);0.2 :: link(A,B,family); 0.1 :: link(A,B,work)
                                  ← person(A),person(B),same_town(A,B).
0.3 :: same_town(A,B).
knows(A,B)   :-  path(A,B,T).
path(A,B,T)  :-  link(A,B,T).
path(A,B,T)  :-  link(A,C,T),path(C,B,T).
```

The main inference task addressed in ProbLog is that of calculating the probability that a query, for instance `?- knows(ed,ann)`, succeeds in a randomly sampled program. The ProbLog implementation offers both exact and approximate inference methods. Exact inference is based on the set of all proofs of a query, while approximate inference methods either use a subset of proofs only (with the single most likely proof as an extreme case) or obtain an estimate by sampling large numbers of possible worlds. To calculate probabilities conditioned on evidence, we have recently introduced an alternative approach based on weighted CNFs [3].

ProbLog has been and still is used as a tool to develop various machine learning techniques and to explore different directions for extensions. We have introduced parameter learning methods that estimate fact probabilities for a given program based on either queries or proofs annotated with their success probability [4] or on (partial) interpretations [5]. ProbLog theory compression [6] reduces the size of a ProbLog program while maintaining probabilities of given example queries as well as possible. Probabilistic explanation based learning [7] and probabilistic local pattern mining [8] learn single clauses based on training examples, which can then be used to reason by analogy, that is, to find similar examples with high probability. Hybrid ProbLog [9] introduces a novel type of probabilistic fact where arguments of the fact can be distributed according to a continuous distribution. DTProbLog [10] is a decision-theoretic extension of ProbLog that allows one to reason about alternative actions based on a utility function in a probabilistic context. FOProbLog [11] and aProbLog [12] generalize ProbLog along different lines: FOProbLog replaces Prolog by first order logic formulae guarded by probabilistic facts, while aProbLog replaces probability labels by algebraic labels representing for instance costs, distances, or datastructures.

The publicly available implementation in YAP currently includes parameter learning, both from queries and proofs and from interpretations, Hybrid ProbLog, and DTProbLog; further modules will be included in the future.

References

[1] De Raedt, L., Kimmig, A., Toivonen, H. (2007). ProbLog: A probabilistic Prolog and its application in link discovery. In Proceedings of the 20th International Joint Conference on Artificial Intelligence, pages 2462–2467.

[2] Kimmig, A., Demoen, B., De Raedt, L., Santos Costa, V., Rocha, R. (2011). On the implementation of the probabilistic logic programming language ProbLog. Theory and Practice of Logic Programming (TPLP) **11** 235–262.

[3] Fierens, D., Van den Broeck, G., Thon, I., Gutmann, B., De Raedt, L. (2011). Inference in probabilistic logic programs using weighted CNFs. In Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence. Accepted.

[4] Gutmann, B., Kimmig, A., Kersting, K., De Raedt, L. (2008). Parameter learning in probabilistic databases: A least squares approach. In Proceedings of the European Conference on Machine Learning. Volume 5211 of LNCS, pages 473–488.

[5] Gutmann, B., Thon, I., De Raedt, L. (2011). Learning the parameters of probabilistic logic programs from interpretations. In Proceedings of the European Conference on Machine Learning. Accepted.

[6] De Raedt, L., Kersting, K., Kimmig, A., Revoredo, K., and Toivonen, H. (2008). Compressing probabilistic Prolog programs. Machine Learning, 70(2-3):151–168.

[7] Kimmig, A., De Raedt, L., and Toivonen, H. (2007). Probabilistic explanation based learning. In Proceedings of the 18th European Conference on Machine Learning, volume 4701 of LNCS, pages 176–187.

[8] Kimmig, A. and De Raedt, L. (2009). Local query mining in a probabilistic Prolog. In Proceedings of the 21st International Joint Conference on Artificial Intelligence, pages 1095–1100.

[9] Gutmann, B., Jaeger, M., and De Raedt, L. (2010a). Extending ProbLog with continuous distributions. In Proceedings of the 20th International Conference on Inductive Logic Programming, volume 6489 of LNCS.

[10] Van den Broeck, G., Thon, I., van Otterlo, M., and De Raedt, L. (2010). DTProbLog: A decision-theoretic probabilistic Prolog. In Proceedings of the 24th AAAI Conference on Artificial Intelligence, pages 1217–1222.

[11] Bruynooghe, M., Mantadelis, T., Kimmig, A., Gutmann, B., Vennekens, J., Janssens, G., and De Raedt, L. (2010). ProbLog technology for inference in a probabilistic first order logic. In Proceedings of the 19th European Conference on Artificial Intelligence, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 719–724.

[12] Kimmig, A., Van den Broeck, G., De Raedt, L. (2011). An algebraic Prolog for reasoning about possible worlds. In AAAI Conference on Artificial Intelligence. Accepted.