Towards Digesting the Alphabet-Soup of Statistical Relational Learning

Luc De Raedt[1] **Bart Demoen**[1] **Daan Fierens**[1] **Bernd Gutmann**[1] **Gerda Janssens**[1]

Angelika Kimmig[1] **Niels Landwehr**[1] **Theofrastos Mantadelis**[1] **Wannes Meert**[1]

Ricardo Rocha[2] **Vítor Santos Costa**[2] **Ingo Thon**[1] **Joost Vennekens**[1]

[1]Katholieke Universiteit Leuven, [2]University of Porto

Abstract

This paper reports on our work towards the development of a probabilistic logic programming environment intended as a target language in which other probabilistic languages can be compiled, thereby contributing to the digestion of the "alphabet soup".

1 Introduction

The vast interest in statistical relational learning, probabilistic (inductive) logic programming and probabilistic programming languages has resulted in a wide variety of different formalisms, models and probabilistic programming languages. Whereas on the one hand, this provides evidence for the richness and maturity of the field, on the other hand, it also exhibits some important weaknesses (which are sometimes referred to by the term "alphabet-soup"). In particular, the relationships between the different formalisms are often unclear, there is not yet a common core theory or model, and while there exist a few publicly available systems and benchmark data sets, it is often unclear how to compare different systems on a fair basis.

We aim at alleviating this situation by developing a high-speed probabilistic logic programming environment (on top of the well-known YAP-Prolog [13, 8]). Our goal is not to design yet another member of the alphabet soup, but rather a general probabilistic logic programming environment. It would offer a low-level probabilistic language, to which other probabilistic (inductive) logic programming and statistical relational learning formalisms can be "compiled". In addition, it would also offer efficient implementations of a number of inference tasks for this low-level language, which can be used to answer queries or perform learning tasks for the original languages.

2 Reducing Probabilistic Languages

Our compilation approach follows ideas similar to that of [5, 6] and the many reductions known from theoretical computer science. For a source and a target probabilistic (programming) language L_S and L_T, we define reductions $\varphi : L_S \rightarrow L_T$ mapping (at least) models and queries from the source to the target language, and $\psi : L_T \rightarrow L_S$ mapping (at least) answer sets back. The bottom line is that if the reductions can be computed efficiently and an efficient implementation of the target language L_T is available, language L_S can be implemented or emulated by first applying the reduction, then using the inference engine for L_T and finally mapping results back to L_S. That is, φ

maps any model $P \in L_S$ onto a model (or program) $\varphi(P) \in L_T$ and each query q_P on P that can be answered using L_S onto a query $\varphi(q_P)$ on $\varphi(P)$ that can be answered using L_T. Furthermore, ψ maps answers obtained in L_T back to L_S such that $ans_S(q_P, P) = \psi(ans_T(\varphi(q_P), \varphi(P)))$.

To alleviate the above sketched problems surrounding the "alphabet soup" in probabilistic programming using this approach, two problems need to be solved:

1. identify a probabilistic programming language and environment that is sufficiently simple, powerful and efficient to serve as a target language and a common core;

2. develop reductions between a wide range of probabilistic languages and the target language.

In the remainder of this note, we sketch the key ingredients of the probabilistic programming language and technology that we are currently developing, and also provide evidence as to why a significant range of statistical relational models and probabilistic programming languages can be connected to our target language.

3 The Probabilistic Prolog Technology

The semantics of the probabilistic Prolog will be based on Sato's distribution semantics [15], which forms the basis of Sato's PRISM [16], Poole's ICL [10], De Raedt et al.'s ProbLog [3], and probabilistic database technology (e.g., Suciu's work [2]). The idea is to associate probabilities to (possibly non-ground) facts using expressions of the form $p : f(t_1, ..., t_m)$, where each ground instance of such a fact $f(t_1, ..., t_m)$ is independently true with probability p. In addition, constraints on possible combinations of such facts can be specified, such as the disjoint statements of PRISM and ICL. To simplify probabilistic inference, PRISM and ICL, however, assume that abductive explanations are mutually exclusive, which is often too restrictive. While the ProbLog system does not make this assumption, it does not yet allow for specifying constraints. Hence, our proposal is to merge principles of PRISM and ProbLog, thereby avoiding additional assumptions but supporting the use of constraints.

At the same time, it is essential that state-of-the-art programming language technology and probabilistic inference methods can be used. Our recent work on ProbLog has focused on 1) the use of Binary Decision Diagrams and tries for supporting inference, and 2) the incorporation in YAP-Prolog, a state-of-the-art Prolog implementation; cf. [8].

4 Reductions

The literature contains ample evidence and arguments as to why one statistical relational learning model is more expressive than another one. Although these arguments are sometimes informal, they often indicate how one language (or parts thereof) can be reduced to another one. Certainly, for those languages built on the logic programming paradigm (such as BLPs [7], PRISM, ICL, ProbLog, CP-logic [17], CLP($\mathcal{BN}$) [14] and SLPs [9]), the relationships and possible mappings are quite clear. PRISM and ICL are very similar, and have been used as target language to which CP-logic and SLPs can be reduced [18, 1] (and PRISM has actually been used to learn SLPs in this way). A common core of BLPs and SLPs is identified in [11], where mappings from a subclass of one formalism to the other are provided for both directions, along with an extension of SLPs to which arbitrary BLPs can be reduced. At present the situation is less clear for languages not based on logic programming, though also here some interesting results are known. For instance, [6] shows that MLNs [12] can be reduced to RBNs [4].

We believe that a common probabilistic logic programming environment of the kind we are developing will make it significantly easier to address various open issues, including not only the mere existence of reductions as sketched above, but also other questions such as e.g. whether they preserve certain parts of the structure of a theory or are especially suited for specific inference or learning tasks.

References

[1] J. Cussens. Integrating by separating: Combining probability and logic with ICL, PRISM and SLPs. APRIL project report, 2005.

[2] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, p. 864–875, 2004.

[3] L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *IJCAI*, p. 2462–2467, 2007.

[4] M. Jaeger. Relational bayesian networks. In *UAI*, p. 266–273, 1997.

[5] M. Jaeger, K. Kersting, and L. De Raedt. Expressivity Analysis for PL-Languages (position paper). In *Online proceedings of SRL*, 2006.

[6] M. Jaeger. Model-Theoretic Expressivity Analysis. In L. De Raedt et al., editors, *Probabilistic Inductive Logic Programming*, vol.4911 of LNCS, 2008.

[7] K. Kersting, and L. De Raedt. Basic Principles of Learning Bayesian Logic Programs In L. De Raedt et al., editors, *Probabilistic Inductive Logic Programming*, vol.4911 of LNCS, 2008.

[8] A. Kimmig, V. Santos Costa, R. Rocha, B. Demoen, and L. De Raedt. On the efficient execution of ProbLog programs. In *ICLP*, 2008.

[9] S. H Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*. IOS Press, 1996.

[10] D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56, 1997.

[11] A. Puech, S.H. Muggleton. A comparison of Stochastic logic programs and Bayesian logic programs. In *IJCAI03 workshop on learning statistical models from relational data*, 2003.

[12] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[13] V. Santos Costa, L. Damas, R. Reis, R. Azevedo. YAP User's Manual. http://www.ncc.up.pt/~vsc/Yap, 2002.

[14] V. Santos Costa, D. Page, and J. Cussens. CLP($\mathcal{BN}$): Constraint Logic Programming for Probabilistic Knowledge. In L. De Raedt et al., editors, *Probabilistic Inductive Logic Programming*, vol.4911 of LNCS, 2008.

[15] T. Sato. A statistical learning method for logic programs with distribution semantics. In *ICLP*, p. 715–729, 1995.

[16] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[17] J. Vennekens, M. Denecker, and M. Bruynooghe. Representing causal information about a probabilistic process. In *JELIA*, 2006.

[18] J. Vennekens. Algebraic and logical study of constructive processes in knowledge representation. PhD thesis, K.U. Leuven, 2007.