frontiers in
ROBOTICS AND AI

PERSPECTIVE ARTICLE
published: 18 September 2014
doi: 10.3389/frobt.2014.00006

A history of Probabilistic Inductive Logic Programming

***Fabrizio Riguzzi*[1]*, *Elena Bellodi*[2] *and *Riccardo Zese*[2]**

[1] *Dipartimento di Matematica e Informatica, Università di Ferrara, Ferrara, Italy*
[2] *Dipartimento di Ingegneria, Università di Ferrara, Ferrara, Italy*

Edited by:
Heni Ben Amor, Georgia Institute of Technology, USA

Reviewed by:
Wannes Meert, KU Leuven, Belgium
Neil Thomas Dantam, Georgia Institute of Technology, USA

***Correspondence:**
Fabrizio Riguzzi, Dipartimento di Matematica e Informatica, Università di Ferrara, Via Saragat 1, Ferrara 44122, Italy
e-mail: fabrizio.riguzzi@unife.it

The field of Probabilistic Logic Programming (PLP) has seen significant advances in the last 20 years, with many proposals for languages that combine probability with logic programming. Since the start, the problem of learning probabilistic logic programs has been the focus of much attention. Learning these programs represents a whole subfield of Inductive Logic Programming (ILP). In Probabilistic ILP (PILP), two problems are considered: learning the parameters of a program given the structure (the rules) and learning both the structure and the parameters. Usually, structure learning systems use parameter learning as a subroutine. In this article, we present an overview of PILP and discuss the main results.

Keywords: logic programming, probabilistic programming, inductive logic programming, probabilistic logic programming, statistical relational learning

1. INTRODUCTION

Probabilistic Logic Programming (PLP) started in the early 90s with seminal works such as those of Dantsin (1991), Ng and Subrahmanian (1992), Poole (1993), and Sato (1995).

Since then, the field has steadily developed and many proposals for the integration of logic programming and probability have appeared, allowing the representation of both complex relations among entities and uncertainty over them. These proposals can be grouped into two classes: those that use a variant of the distribution semantics (Sato, 1995) and those that follow a Knowledge Base Model Construction (KBMC) approach (Wellman et al., 1992; Bacchus, 1993).

The distribution semantics underlines many languages such as Probabilistic Logic Programs (Dantsin, 1991), Probabilistic Horn Abduction (Poole, 1993), PRISM (Sato, 1995), Independent Choice Logic (ICL) (Poole, 1997), pD (Fuhr, 2000), Logic Programs with Annotated Disjunctions (LPADs) (Vennekens et al., 2004), ProbLog (De Raedt et al., 2007), P-log (Baral et al., 2009), and CP-logic (Vennekens et al., 2009). While the number of languages is large, all share a common approach so that there are transformations with linear complexity that can translate one language into another. Under the distribution semantics, a probabilistic logic program defines a probability distribution over normal logic programs (termed *worlds*). The probability of a ground query Q is then obtained from the joint distribution of the query and the worlds: it is the sum of the probability of the worlds where the query is true.

The languages following a KBMC approach include Relational Bayesian Network (Jaeger, 1998), CLP(BN) (Santos Costa et al., 2003), Bayesian Logic Programs (Kersting and De Raedt, 2001), and the Prolog Factor Language (Gomes and Santos Costa, 2012). In these languages, a program is a template for generating a ground graphical model, be it a Bayesian network or a Markov network.

Learning probabilistic logic programs has been considered from the start: Sato (1995) already presented an algorithm for learning the parameters of programs under the distribution semantics. This is the first problem that was considered in the domain of learning and was the only one until recently, when works regarding the induction of the structure (the rules) and the parameters at the same time began to appear. The whole field was called Probabilistic Inductive Logic Programming (PILP) in (De Raedt and Kersting, 2004) and an overview of the field was provided in De Raedt et al. (2008a).

PILP uses declarative probabilistic languages that allow learned models to be easily understood by humans. Moreover, languages based on the distribution semantics are Turing complete, thus representing very expressive target formalisms. Recently, effective PILP systems have been proposed that achieve good results on a variety of domains, including biology, chemistry, medicine, entity resolution, link prediction, and web page classification.

In the following, we present an updated overview of PILP by concentrating on languages under the distribution semantics.

2. LANGUAGES UNDER THE DISTRIBUTION SEMANTICS

We illustrate the distribution semantics through ProbLog (De Raedt et al., 2007), the language with the simplest syntax. A ProbLog program consists of a set of (certain) rules and a set of *probabilistic facts* of the form:

$$p_i :: A_i.$$

where $p_i \in [0,1]$ and A_i is an atom, meaning that each ground instantiation $A_i\theta$ of A_i is true with probability p_i and false with probability $1 - p_i$. From a ProbLog program, we obtain normal programs called worlds by including the set C of certain rules and a subset L of the (ground) probabilistic facts. Each world is obtained by selecting or rejecting each grounding of each probabilistic fact. The probability of a world is given by the product of a factor p_i for each grounding of a probabilistic fact $p_i :: A_i$ included in the world and of a factor $1 - p_i$ for each grounding of a probabilistic fact not included in the world. The probability of a ground

atom (query Q) is then the sum of the probabilities of the worlds where the query is true.

Example 1: The following program encodes the fact that a person sneezes if he has the flu and this is the active cause of sneezing, or if he has hay fever and hay fever is the active cause of sneezing:

$$sneezing(X) \text{ :- } flu(X), fluSneezing(X).$$
$$sneezing(X) \text{ :- } hayFever(X), hayFeverSneezing(X).$$
$$flu(bob).$$
$$hayFever(bob).$$
$$0.7 :: fluSneezing(X).$$
$$0.8 :: hayFeverSneezing(X).$$

This program has 4 worlds, $Q = sneezing(bob)$ is true in 3 of them and its probability $P(Q)$ is $0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$.

Note that, in the case a ground atom can be derived from more than one ground rule, the contributions in terms of probability of the ground rules are combined with a noisy-OR gate. Hommerson and Lucas (2011) generalize the distribution semantics by allowing the replacement of the noisy OR combining rules with user-defined rules.

3. INFERENCE

The problem of computing the probability of queries is called inference. Solving it by computing all the worlds and then identifying those that entail the query is impractical as the number of possible worlds is exponential in the number of ground probabilistic facts. Usually, inference is performed by resorting to knowledge compilation (Darwiche and Marquis, 2002): according to this, a propositional theory and a query are compiled into a "target language", which is then used to answer queries in polytime. The compilation becomes the main computational bottleneck, but considerable effort has been devoted to the development of efficient compilers. The compilation methods differ for the compactness of the target language and the class of queries and transformations that it supports in polynomial time. We describe in the following Section two major compilation approaches.

3.1. EXACT INFERENCE

An early method for exact inference in Relational Bayesian Networks (RBNs) was proposed in Chavira et al. (2006) where RBNs were compiled into arithmetic circuits.

The first knowledge compilation approach for performing inference on languages based on the distribution semantics (De Raedt et al., 2007) required to find a covering set of explanations for the query. An explanation is a minimal set of probabilistic facts that is sufficient for entailing the query and a covering set of explanations is a set that contains all possible explanations for the query. From the set of explanations, a Boolean formula in Disjunctive Normal Form (DNF) can be built, where each probabilistic fact is associated with a Boolean variable, an explanation is the conjunction of the facts that it contains and the whole formula is the disjunction of the formulas for the different explanations. In Example 1, if we associate the Boolean variable X_1 with $fluSneezing(bob)$ and X_2 with $hayFeverSneezing(bob)$, the Boolean formula that encodes the set of explanations for Q will be $X_1 \vee X_2$.

Computing the probability of a Boolean DNF formula is an intractable problem (Rauzy et al., 2003) but, by exploiting the advances made in knowledge compilation, we can compile the formula into a Binary Decision Diagram (BDD). BDDs allow to compute the probability of a query with a dynamic programming algorithm that is linear in the size of the diagram (De Raedt et al., 2007). BDDs are rooted graphs with one level for each Boolean variable; a node n in a BDD has two children: one corresponding to the 1 value of the variable associated with the level of n and one corresponding to the 0 value of the variable. The leaves store either 0 or 1. A BDD for the function $X_1 \vee X_2$ is shown in **Figure 1**.

Other reasoning systems based on the BDD language are PICL (Riguzzi, 2009), that was developed for ICL and computes the explanations of queries using a modification of SLDNF-resolution, and PITA (Riguzzi and Swift, 2011), which translates a general PLP program into a normal program evaluated by a Prolog engine with tabling. Library functions that perform BDD operations are added to the normal program and BDDs are built representing the set of explanations of the goals encountered during inference.

An alternative approach to exact inference can be realized using compilation to d-DNNFs (deterministic Decomposable Negation Normal Form) rather than BDDs (Fierens et al., 2011). In the first step, the theory and the evidence are converted into Boolean formulas in Conjunctive Normal Form (CNF), i.e., conjunctions of disjunctions of literals. The Boolean formulas are then compiled to d-DNNFs and the probability is computed by weighted model counting (WMC). In WMC, the literals of the CNFs are assigned weight p to A and weight $1 - p$ to $\neg A$, if the program contains a probabilistic fact $p :: A$, and weight 1 otherwise. A d-DNNF is a rooted directed cyclic graph in which each leaf node is labeled with a literal and each internal node is labeled with a conjunction or disjunction. The graph must moreover satisfy a number of restrictions. BDDs form a subclass of d-DNNFs. WMC on d-DNNFs is linear in the size of the graph.

3.2. APPROXIMATE INFERENCE

Since the cost of inference may be very high, approximate algorithms have been developed. They either compute subsets of possibly incomplete explanations or use random sampling. In the first approach, a subset of the explanations provides a lower bound and the set of partially expanded explanations provides an upper bound (Kimmig et al., 2011; Renkens et al., 2014). In the second approach, the truth of the query is repeatedly checked in a normal program sampled from the probabilistic program. The probability

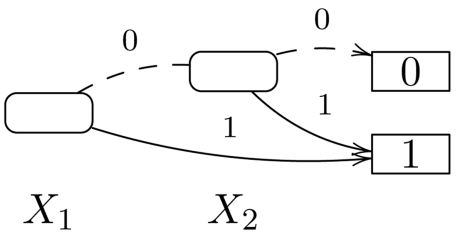

FIGURE 1 | Binary Decision Diagram for Example 1.

of the query is then given by the fraction of the successes (Kimmig et al., 2011).

In Choi and Darwiche (2011), the system called Relax, Compensate, and then Recover performs inference by first approximating the exact model by relaxing equality constraints contained in the model. Then, this approximate model is improved by compensation that enforces weaker notions of equality. Finally, the system recovers some equivalence constraints in the relaxed and compensated model, which can improve the quality of the approximation.

In Riguzzi (2013), the disjunctive clauses of an LPAD are transformed into normal clauses with auxiliary atoms in the body in order to take the samples; tabling is exploited to avoid sampling twice the same atom.

3.3. LIFTED INFERENCE

Recently, lifted inference approaches have appeared that perform inference without first grounding the model. In this way, groups of indistinguishable individuals are treated as a whole and not individually. The exploitation of the symmetries in the model can significantly speed up inference. For example, consider the following program:

$$p :: famous(Y).$$
$$popular(X) \text{ :- } friends(X, Y), famous(Y).$$

In this case $P(\text{popular(john)}) = 1 - (1 - p)^m$ where m is the number of friends of *john*. This is because an atom in the head of a clause with variables in the body only represents the noisy-OR of the atoms in the body. In this case, we do not need to know the identities of these friends, we just need to know how many are there. Hence, we need not ground the clauses.

Bellodi et al. (2014) proposed to use Lifted Variable Elimination (Poole, 2003) for performing lifted inference in PLP. This algorithm eliminates random variables from the factorization of the probability distribution. It repeatedly applies operators that either eliminate a group of variables or prepare the factorization for elimination. In PLP, random variables corresponding to the head of clauses represent the noisy-OR of variables in the body. In order to deal with noisy-OR structures Bellodi et al. (2014) introduced two new operators.

Van den Broeck et al. (2014) presented an algorithm that can perform lifted inference in PLP by weighted first-order model counting (WFOMC). The program is transformed by knowledge compilation into a First-Order d-DNNF circuit from which the weighted model count is computed. The paper presents a Skolemization procedure that maps a logical input theory to an output theory devoid of existential quantifiers and functions but with identical WFOMC; the procedure takes into account the cases in which program clauses may have variables in the body that do not appear in the head too.

4. LEARNING

The problem that PILP aims at solving can be expressed as:

- Given
  - background knowledge as a probabilistic logic program B
  - a set of positive and negative examples E^+ and E^-
  - a language bias $\mathcal{L}$
- Find
  - a probabilistic logic program P such that the probability of positive examples according to $P \cup B$ is maximized and the probability of negative examples is minimized.

This problem has two variants: parameter learning and structure learning. In the first, we are given the structure (the rules) of P and we just want to infer the parameters of P, while in the second we want to infer both the structure and the parameters of P. Moreover, the examples can be given in the form of (partial) interpretations, ground atoms, or (partial) proofs.

4.1. PARAMETER LEARNING

Parameter learning for languages following the distribution semantics has been performed by using the Expectation Maximization (EM) algorithm or by gradient descent.

The EM algorithm is used to estimate the probability of models containing random variables that are not observed in the data. This is the case of PLP under the distribution semantics because of the use of combining rules: these imply the presence of unobserved variables. The EM algorithm consists of a cycle in which the steps of Expectation and Maximization are repeatedly performed. In the Expectation step, the distribution of the hidden variables is computed according to the current values of the parameters, while in the Maximization step, the new values of the parameters are computed. Examples of approaches that use EM are PRISM (Sato and Kameya, 2001), LFI-ProbLog (Fierens et al., 2013), and EMBLEM (Bellodi and Riguzzi, 2013). The latter two use knowledge compilation for computing the distribution of the hidden variables. RIB (Riguzzi and Di Mauro, 2012) is a system for parameter learning that uses a special EM algorithm called information bottleneck that was shown to avoid some local maxima of EM.

Gradient descent methods compute the gradient of the target function and iteratively modify the parameters moving in the direction of the gradient. An example of these methods is LeProbLog (Gutmann et al., 2008) that uses a dynamic programming algorithm for computing the gradient exploiting BDDs.

4.2. STRUCTURE LEARNING

One of the first structure learning works is (Koller and Pfeffer, 1997) where the authors learn the structure of first-order rules with associated probabilistic uncertainty parameters. Their approach involves generating the underlying graphical model using a Knowledge Based Model Construction approach. EM is then applied on the graphical model.

De Raedt et al. (2008b) presented an algorithm for performing theory compression on ProbLog programs. Theory compression means removing as many clauses as possible from the theory in order to maximize the probability. No new clause can be added to the theory.

SEM-CP-logic (Meert et al., 2008) learns parameters and structure of ground CP-logic programs. It performs learning by considering the Bayesian networks equivalent to CP-logic programs and by applying techniques for learning Bayesian networks. In

particular, it applies the Structural Expectation Maximization (SEM) algorithm (Friedman, 1998): it iteratively generates refinements of the equivalent Bayesian network and it greedily chooses the one that maximizes the BIC score (Schwarz, 1978).

ProbFOIL (De Raedt and Thon, 2010) combines the rule learner FOIL (Quinlan and Cameron-Jones, 1993) with ProbLog. Logical rules are learned from probabilistic data in the sense that both the examples themselves and their classifications can be probabilistic. The set of rules has to allow to predict the probability of the examples from their description. In this setting, the parameters (the probability values) are fixed and the structure has to be learned.

SLIPCASE (Bellodi and Riguzzi, 2011) performs a beam search in the space of LPADs by iteratively refining probabilistic theories and optimizing the parameters of each theory with EMBLEM. This is possible as parameter learning is usually fast. SLIPCOVER (Bellodi and Riguzzi, 2014) is an evolution of SLIPCASE that uses bottom clauses generated as in Progol (Muggleton, 1995) to guide the refinement process, thus reducing the number of revisions and exploring more effectively the search space. Moreover, SLIPCOVER separates the search for promising clauses from that of the theory: the space of clauses is explored with a beam search, while the space of theories is searched greedily. Both of them use the log likelihood of the data as the guiding heuristics in the search phases, evaluated by EMBLEM.

5. DISCUSSION AND DIRECTIONS FOR FUTURE WORK

PLP can be framed into the broader area of Probabilistic Programming (PP), which is receiving an increasing attention especially in the field of Machine Learning, as is testified by the ongoing DARPA project "Probabilistic Programming for Advancing Machine Learning." PLP differs for the use of Logic Programming, which provides a declarative reading of the programs. The array of algorithms for performing inference with PLP is constantly expanding, quickly approaching the variety of algorithms available for other PP languages.

In the field of Statistical Relational Learning (SRL) (Getoor and Taskar, 2007), relational probabilistic languages are used as the representation of the data and of the theory to be learned. These languages provide a compact way of specifying complex graphical models but are not usually programming languages, i.e., they are not Turing complete. SRL systems have achieved impressive results on a plethora of datasets, especially the systems using Markov Logic Networks (MLNs) (Richardson and Domingos, 2006). For example, MLNs showed very good performances in link prediction, entity resolution, and information extraction. Recently, the performance of PILP systems has been compared with those using MLNs with promising results. EMBLEM has been compared with Alchemy (Richardson and Domingos, 2006) on the problem of parameter learning and has achieved better results (Bellodi and Riguzzi, 2012). A comparison (Bellodi and Riguzzi, 2014) of SLIPCOVER with the LSM algorithm (Kok and Domingos, 2005) for learning the structure of MLNs has shown that SLIPCOVER has better performance. These results indicate that PLP is a viable alternative to mainstream SRL languages and that PILP is a promising research area, given the better readability and Turing completeness of PLP programs. The aim of PILP is to develop systems that are fast, easy to configure and use, and that return accurate models on a wide variety of domains.

There are many avenues for future research. Improving the efficiency of inference is very important, since it is a basic component of learning systems. The use of new languages for knowledge compilation, such as Sentential Decision Diagrams (Darwiche, 2011) is one possibility, as is the development of lifted inference systems. In particular, the latter can be combined with techniques for identifying the portion of the program that is relevant to the query, such as First-Order Bayes Ball (Meert et al., 2010), again without grounding the model first.

Regarding learning systems, parameter learning should be combined with lifted inference to speed up the process. Other forms of parameter optimizations can be applied, drawing inspiration from the algorithms developed for related formalisms such as Markov Logic. For structure learning, other search approaches can be investigated, such as local and randomized search, and methods that learn the parameters and the structure at the same time can be considered as Natarajan et al. (2012) do for Relational Dependency Networks. Moreover, the configuration of PILP systems should be simplified so that they can be used as much as possible out of the box: at the moment, the user has to set many parameters; the aim is to fully understand the effects of each parameter in order to provide the user with strong guidelines.

AUTHOR NOTE

An earlier version of this paper appeared in the ALP Newsletter (http://www.cs.nmsu.edu/ALP/2014/03/probabilistic-inductive-logic-programming/).

REFERENCES

Bacchus, F. (1993). "Using first-order probability logic for the construction of Bayesian networks," in *Conference on Uncertainty in Artificial Intelligence*, eds D. Heckerman and E. H. Mamdani (San Francisco: Morgan Kaufmann), 219–226.

Baral, C., Gelfond, M., and Rushton, N. (2009). Probabilistic reasoning with answer sets. *Theory Pract. Logic Program.* 9, 57–144. doi:10.1017/S1471068408003645

Bellodi, E., Lamma, E., Riguzzi, F., Costa, V. S., and Zese, R. (2014). "Lifted variable elimination for probabilistic logic programming," in *Theory and Practice of Logic Programming, Special Issue on the International Conference on Logic Programming(CoRR abs/1405.3218)*.

Bellodi, E., and Riguzzi, F. (2011). "Learning the structure of probabilistic logic programs," in *International Conference on Inductive Logic Programming, Volume 7207 of LNCS* (Berlin: Springer), 61–75.

Bellodi, E., and Riguzzi, F. (2012). Experimentation of an expectation maximization algorithm for probabilistic logic programs. *Intell. Artif.* 8, 3–18. doi:10.3233/IA-2012-0027

Bellodi, E., and Riguzzi, F. (2013). Expectation maximization over binary decision diagrams for probabilistic logic programs. *Intell. Data Anal.* 17, 343–363. doi:10.3233/IDA-130582

Bellodi, E., and Riguzzi, F. (2014). Structure learning of probabilistic logic programs by searching the clause space. *Theory and Practice of Logic Programming.*

Chavira, M., Darwiche, A., and Jaeger, M. (2006). Compiling relational Bayesian networks for exact inference. *Int. J. Approx. Reason.* 42, 4–20. doi:10.1016/j.ijar.2005.10.001

Choi, A., and Darwiche, A. (2011). "Relax, compensate and then recover," in *New Frontiers in Artificial Intelligence*, eds T. Onada, D. Bekki, and E. McCready (Berlin: Springer), 167–180.

Dantsin, E. (1991). "Probabilistic logic programs and their semantics," in *Russian Conference on Logic Programming, Volume 592 of LNCS* (Berlin: Springer), 152–164.

Darwiche, A. (2011). "SDD: a new canonical representation of propositional knowledge bases," in *International Joint Conference on Artificial Intelligence*, Palo Alto, CA, 819–826.

Darwiche, A., and Marquis, P. (2002). A knowledge compilation map. *J. Artif. Intell. Res.* 17, 229–264. doi:10.1613/jair.989

De Raedt, L., Frasconi, P., Kersting, K., and Muggleton, S. (eds) (2008a). *Probabilistic Inductive Logic Programming – Theory and Applications, Volume 4911 of LNCS* (Berlin: Springer).

De Raedt, L., Kersting, K., Kimmig, A., Revoredo, K., and Toivonen, H. (2008b). Compressing probabilistic Prolog programs. *Mach. Learn.* 70, 151–168. doi:10.1007/s10994-007-5030-x

De Raedt, L., and Kersting, K. (2004). "Probabilistic inductive logic programming," in *International Conference on Algorithmic Learning Theory, Volume 3244 of LNCS* (Berlin: Springer), 19–36.

De Raedt, L., Kimmig, A., and Toivonen, H. (2007). "ProbLog: a probabilistic prolog and its application in link discovery," in *International Joint Conference on Artificial Intelligence* (San Francisco: Morgan Kaufmann), 2462–2467.

De Raedt, L., and Thon, I. (2010). "Probabilistic rule learning," in *International Conference on Inductive Logic Programming, Volume 6489 of LNCS* (Berlin: Springer), 47–58.

Fierens, D., den Broeck, G. V., Renkens, J., Shterionov, D. S., Gutmann, B., Thon, I., et al. (2013). Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming.*

Fierens, D., Van den Broeck, G., Thon, I., Gutmann, B., and De Raedt, L. (2011). "Inference in probabilistic logic programs using weighted CNF's," in *Conference on Uncertainty in Artificial Intelligence* (San Francisco: Morgan Kaufmann), 211–220.

Friedman, N. (1998). "The Bayesian structural EM algorithm," in *Conference on Uncertainty in Artificial Intelligence* (San Francisco: Morgan Kaufmann), 129–138.

Fuhr, N. (2000). Probabilistic datalog: implementing logical information retrieval for advanced applications. *J. Am. Soc. Inform. Sci.* 51, 95–110. doi:10.1002/(SICI)1097-4571(2000)51:2<95::AID-ASI2>3.0.CO;2-H

Getoor, L., and Taskar, B. (eds) (2007). *Introduction to Statistical Relational Learning.* Cambridge: MIT Press.

Gomes, T., and Santos Costa, V. (2012). "Evaluating inference algorithms for the prolog factor language," in *International Conference on Inductive Logic Progamming, Volume 7842 of LNCS* (Berlin: Springer), 74–85.

Gutmann, B., Kimmig, A., Kersting, K., and De Raedt, L. (2008). "Parameter learning in probabilistic databases: a least squares approach," in *European Conference on Machine Learning and Knowledge Discovery in Databases, Volume 5211 of LNCS* (Berlin: Springer), 473–488.

Hommerson, A., and Lucas, P. J. F. (2011). "Generalising the interaction rules in probabilistic logic," in *International Joint Conference on Artificial Intelligence* (Palo Alto: IJCAI/AAAI), 912–917.

Jaeger, M. (1998). "Reasoning about infinite random structures with relational Bayesian networks," in *International Conference on Principles of Knowledge Representation and Reasoning* (San Francisco: Morgan Kaufmann), 570–581.

Kersting, K., and De Raedt, L. (2001). "Towards combining inductive logic programming with Bayesian networks," in *International Conference on Inductive Logic Programming, Volume 2157 of LNCS* (Berlin: Springer), 118–131.

Kimmig, A., Demoen, B., De Raedt, L., Costa, V. S., and Rocha, R. (2011). On the implementation of the probabilistic logic programming language ProbLog. *Theory Pract. Logic Program.* 11, 235–262. doi:10.1093/bioinformatics/bts166

Kok, S., and Domingos, P. (2005). "Learning the structure of Markov logic networks," in *International Conference on Machine Learning* (New York: ACM), 441–448. doi:10.1145/1102351.1102407

Koller, D., and Pfeffer, A. (1997). "Learning probabilities for noisy first-order rules," in *International Joint Conference on Artifical Intelligence*, Vol. 2 (San Francisco: Morgan Kaufmann), 1316–1321.

Meert, W., Struyf, J., and Blockeel, H. (2008). Learning ground CP-Logic theories by leveraging Bayesian network learning techniques. *Fundam. Inform.* 89, 131–160.

Meert, W., Taghipour, N., and Blockeel, H. (2010). "First-order Bayes-ball," in *European Conference on Machine Learning and Knowledge Discovery in Databases, Volume 6322 of LNCS* (Berlin: Springer), 369–384.

Muggleton, S. (1995). Inverse entailment and Progol. *New Gen. Comput.* 13, 245–286. doi:10.1007/BF03037227

Natarajan, S., Khot, T., Kersting, K., Gutmann, B., and Shavlik, J. W. (2012). Gradient-based boosting for statistical relational learning: the relational dependency network case. *Mach. Learn.* 86, 25–56. doi:10.1007/s10994-011-5244-9

Ng, R., and Subrahmanian, V. S. (1992). Probabilistic logic programming. *Inform. Comput.* 101, 150–201. doi:10.1016/0890-5401(92)90061-J

Poole, D. (1993). Logic programming, abduction and probability – a top-down anytime algorithm for estimating prior and posterior probabilities. *New Gen. Comput.* 11, 377–400. doi:10.1007/BF03037184

Poole, D. (1997). The Independent Choice Logic for modelling multiple agents under uncertainty. *Artif. Intell.* 94, 7–56. doi:10.1016/S0004-3702(97)00027-1

Poole, D. (2003). "First-order probabilistic inference," in *International Joint Conference on Artificial Intelligence* (Berlin: Morgan Kaufmann), 985–991.

Quinlan, J. R., and Cameron-Jones, R. M. (1993). "FOIL: a midterm report," in *European Conference on Machine Learning, Volume 667 of LNCS* (Berlin: Springer), 3–20.

Rauzy, A., Châtelet, E., Dutuit, Y., and Bérenguer, C. (2003). A practical comparison of methods to assess sum-of-products. *Reliabil. Eng. Syst. Safety* 79, 33–42. doi:10.1016/S0951-8320(02)00165-5

Renkens, J., Kimmig, A., den Broeck, G. V., and Raedt, L. D. (2014). "Explanation-based approximate weighted model counting for probabilistic logics," in *AAAI Conference on Artificial Intelligence* (Palo Alto: AAAI Press), 2490–2496.

Richardson, M., and Domingos, P. (2006). Markov logic networks. *Mach. Learn.* 62, 107–136. doi:10.1007/s10994-006-5833-1

Riguzzi, F. (2009). Extended semantics and inference for the Independent Choice Logic. *Log. J. IGPL* 17, 589–629. doi:10.1093/jigpal/jzp025

Riguzzi, F. (2013). MCINTYRE: a Monte Carlo system for probabilistic logic programming. *Fundam. Inform.* 124, 521–541. doi:10.3233/FI-2013-847

Riguzzi, F., and Di Mauro, N. (2012). Applying the information bottleneck to statistical relational learning. *Mach. Learn.* 86, 89–114. doi:10.1007/s10994-011-5247-6

Riguzzi, F., and Swift, T. (2011). The PITA system: tabling and answer subsumption for reasoning under uncertainty. *Theory Pract. Logic Program.* 11, 433–449. doi:10.1017/S147106841100010X

Santos Costa, V., Page, D., Qazi, M., and Cussens, J. (2003). "CLP(BN): constraint logic programming for probabilistic knowledge," in *Conference on Uncertainty in Artificial Intelligence* (San Francisco: Morgan Kaufmann), 517–524.

Sato, T. (1995). "A statistical learning method for logic programs with distribution semantics," in *International Conference on Logic Programming* (Cambridge: MIT Press), 715–729.

Sato, T., and Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. *J. Artif. Intell. Res.* 15, 391–454. doi:10.1613/jair.912

Schwarz, G. (1978). Estimating the dimension of a model. *Ann. Stat.* 6, 461–464. doi:10.1214/aos/1176344136

Van den Broeck, G., Meert, W., and Darwiche, A. (2014). "Skolemization for weighted first-order model counting," in *International Conference on Principles of Knowledge Representation and Reasoning* (Palo Alto: AAAI Press).

Vennekens, J., Denecker, M., and Bruynooghe, M. (2009). CP-logic: a language of causal probabilistic events and its relation to logic programming. *Theory Pract. Log. Program.* 9, 245–308. doi:10.1017/S1471068409003767

Vennekens, J., Verbaeten, S., and Bruynooghe, M. (2004). "Logic programs with annotated disjunctions," in *International Conference on Logic Programming, Volume 3131 of LNCS* (Berlin: Springer), 195–209.

Wellman, M. P., Breese, J. S., and Goldman, R. P. (1992). From knowledge bases to decision models. *Knowl. Eng. Rev.* 7, 35–53. doi:10.1017/S0269888900006147

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 09 July 2014; accepted: 02 September 2014; published online: 18 September 2014.

Citation: Riguzzi F, Bellodi E and Zese R (2014) A history of Probabilistic Inductive Logic Programming. Front. Robot. AI ***1****:6. doi: 10.3389/frobt.2014.00006*

This article was submitted to Computational Intelligence, a section of the journal Frontiers in Robotics and AI.