

# Extending ProbLog with Continuous Distributions

Bernd Gutmann<sup>1</sup>, Manfred Jaeger<sup>2</sup>, and Luc De Raedt<sup>1</sup>

<sup>1</sup> Department of Computer Science, Katholieke Universiteit Leuven, Belgium

<sup>2</sup> Department of Computer Science, Aalborg University, Denmark

**Abstract.** ProbLog is a recently introduced probabilistic extension of Prolog. The key contribution of this paper is that we extend ProbLog with abilities to specify continuous distributions and that we show how ProbLog’s exact inference mechanism can be extended to cope with such distributions. The resulting inference engine combines an interval calculus with a dynamic discretization algorithm into an effective solver.

## 1 Introduction

Continuous distributions are needed in many applications for building a natural model. Probabilistic logic programming languages, such as ProbLog and CP-Logic [1], have, so far, largely focused on modeling discrete distributions and typically perform exact inference. The PRISM [2] system provides primitives for Gaussian distributions but requires the exclusive explanation property which complicates modeling. On the other hand, many of the functional probabilistic programming languages, such as BLOG [3] and Church [4], can cope with continuous distributions but only perform approximate inference by a Markov Chain Monte Carlo approach and also, typical statistical relational learning systems such as Markov Logic and Bayesian Logic Programs, have been extended with continuous distributions. The key contribution of this paper is, that we extend a simple probabilistic extension of Prolog based on the distribution semantics with continuous distributions. This is realized by introducing a novel type of probabilistic fact where arguments of the fact can be distributed according to a continuous distribution. Queries can then be posed about the probability that the resulting arguments fall into specific intervals. We introduce the semantics of using continuous distributions in this way and also show how ProbLog’s inference mechanism, based on Binary Decision Diagrams (BDDs), can be extended to cope with these distributions. The resulting language is called Hybrid ProbLog. We also extended ProbLog’s learning algorithm towards continuous distributions. Due to space restrictions we focus on inference but we will include learning in the longer version of this paper.

Similarly to Hybrid ProbLog, Hybrid Markov Logic Networks (HMLNs) [5] aim at integrating Boolean and numerical random variables in a probabilistic-logic modeling framework. The kind of modeling supported by HMLNs is quite different in nature, however, from the kind of modeling for which Hybrid ProbLog

is designed. In an HMLN, one defines equations that function as soft constraints for relationships among numerical and logical variables. For example, one could express that the temperature on day  $d$  is typically around 20 using the weighted equality  $w \text{ temperature}(d) = 20$ , where larger weights  $w$  lead to a larger penalty for deviations of  $\text{temperature}(d)$  from 20. All weighted formulae containing  $\text{temperature}(d)$  – together – implicitly define a probability distribution for the random variable  $\text{temperature}(d)$  due to HMLN semantics. However, one cannot directly specify this distribution to be Gaussian with mean 20 and standard deviation 5, for example. No exact inference methods have been developed for HMLNs.

The rest of this paper is organized as follows. Section 2 reviews basic concepts from ProbLog. Section 3 introduces the syntax and semantics of Hybrid ProbLog. Section 4 describes our exact inference algorithm. Before concluding, we evaluate the algorithm in Section 5.

## 2 ProbLog

ProbLog [6] is a recent probabilistic extension of Prolog. A ProbLog theory  $T = F \cup \mathcal{BK}$  consists of a set of labeled facts  $F = \{p_1 :: f_1, \dots, p_n :: f_n\}$  and a set of definite clauses  $\mathcal{BK}$  that express the background knowledge. The facts  $p_j :: f_j$  in  $F$  are annotated with a probability  $p_j$  stating that  $f_j\theta$  is true with probability  $p_j$  for all substitutions  $\theta$  grounding  $f_j$ . The resulting facts  $f_j\theta$  are called atomic choices and represent random variables; they are assumed to be mutually independent. Let  $\Theta = \{\theta_{j1}, \dots, \theta_{ji_j} | j = 1, \dots, n\}$  be a finite<sup>3</sup> set of possible substitutions for the variables in the probabilistic facts where  $i_j$  is the number of substitutions for fact  $j$ , then a ProbLog theory describes a probability distribution over Prolog programs  $L \subseteq L_F$  where  $L_F = F\Theta$  and  $F\Theta$  denotes the set of all possible ground instances of facts in  $F$ .

$$P_P(L|F) := \prod_{f_j\theta_{jk} \in L} p_j \prod_{f_j\theta_{jk} \in L_F \setminus L} (1 - p_j) \quad (1)$$

The *success probability* of a query  $q$  is then

$$P_s(q|T) := \sum_{\substack{L \subseteq L_F: \\ L \cup \mathcal{BK} \models q}} P(L|T) . \quad (2)$$

Observe that ProbLog defines a probability distribution  $P_w$  over possible worlds, that is Herbrand interpretations. Each total choice, that is, each  $L \subseteq L_F$ , can be extended to a possible world by computing the least Herbrand model of  $L$ . This possible world is assigned the probability  $P_w = P(L|T)$ . Thus a set of total choices represents an assignment of truth-values to all the atomic choices.

<sup>3</sup> Throughout the paper, we assume that  $F\Theta$  is finite, but see [2] for the infinite case.

### 3 Hybrid ProbLog

A Hybrid ProbLog theory  $T = F \cup F^c \cup \mathcal{BK}$  extends ProbLog with continuous probabilistic facts  $F^c = \{(X_1, \phi_1) :: f_1^c, \dots, (X_m, \phi_m) :: f_m^c\}$  where  $X_i$  is a Prolog variable, appearing in the atom  $f_i^c$  and  $\phi_i$  is a density function. The fact  $(X, \text{gaussian}(2, 8)) :: \text{temp}(\text{D}, X)$  – for example – states that the temperature for day D is normal-distributed with mean 2 and standard deviation 8. The syntax allows one to specify multivariate distributions, i.e.,

$$((X, Y), \text{gaussian}([1, 0], [[1, 0.5], [0.5, 1]])) :: \mathbf{f}(X, Y) .$$

In this paper – however – we restrict ourself to the univariate case. Hybrid ProbLog adds the following predicates to the background knowledge.

- `below(X,c)` succeeds if X is a value from a continuous fact,  $c$  is a number constant, and  $X < c$
- `above(X,c)` succeeds if  $X < c$
- `ininterval(X,c1,c2)` succeeds if  $X \in [c_1, c_2]$

For ease of implementation and – more important – to keep exact inference tractable, values from continuous facts may only be used by these predicates.

*Example 1.* This theory models the weather during winter time. The background knowledge states that a person catches a cold when the temperature is below  $0^\circ$  Celsius or when the temperature is below  $5^\circ$  Celsius while it rains:

$$\begin{aligned} 0.8 :: \text{rain}. \quad \text{catchcold} :- \text{rain}, \text{temp}(\text{T}), \text{below}(\text{T}, 5). \\ (\text{T}, \text{gaussian}(2, 8)) :: \text{temp}(\text{T}). \quad \text{catchcold} :- \text{temp}(\text{T}), \text{below}(\text{T}, 0). \end{aligned}$$

The semantics of Hybrid ProbLog theory  $T = F \cup F^c \cup \mathcal{BK}$  is given by probability distributions over subsets of the facts  $f_i$  (called *subprograms*), and over sample values for the numeric variables in the continuous facts  $f_i^c$  (called *continuous subprograms*)<sup>4</sup>. The subprograms  $L \subseteq L_F$  are distributed as in ProbLog (cf. Equation (1)), and the continuous subprograms are distributed as described in Section 3.1. Combining both, gives one the success probability of queries in a Hybrid ProbLog theory as described in Section 3.2.

#### 3.1 Distribution over Continuous Subprograms

Let  $\Theta^c = \{\theta_{j1}^c, \dots, \theta_{ji'}^c \mid j = 1, \dots, m\}$  be a finite set of possible substitutions for the non-numeric variables in the continuous facts  $(X_j, \phi_j) :: f_j^c$ . Each substitution instance  $f_j^c \theta_{jk}^c$  is associated with a random variable  $X_{jk}$  with probability distribution  $\phi_j$ . The  $X_{jk}$  are assumed to be independent. Let  $\mathbf{X}$  denote the  $|\Theta^c|$ -dimensional vector of the random variables, and  $f(\mathbf{x})$  their joint density function. A sample value  $\mathbf{x}$  for  $\mathbf{X}$  defines the continuous subprogram  $L_{\mathbf{x}} := \{f_j^c \theta_{jk}^c [X_j \leftarrow x_{jk}] \mid j = 1, \dots, m; k = 1, \dots, i'_j\}$ .

<sup>4</sup> We denote facts, values, and substitutions related to the continuous part of  $T$  by the superscript  $^c$ .

*Example 2.* The substitutions  $\theta_{1,1}^c = \emptyset$ ,  $\theta_{2,1}^c = \{Y \leftarrow a\}$ ,  $\theta_{2,2}^c = \{Y \leftarrow b\}$  together with the point  $x_{1,1} = 0.9$ ,  $x_{2,1} = 2.3$ ,  $x_{2,2} = 4.2$  applied on the theory

$$(\mathbf{X}, \text{gaussian}(1, 2)) :: \mathbf{h}(\mathbf{X}). \quad (\mathbf{X}, \text{gaussian}(4, 3)) :: \mathbf{f}(\mathbf{X}, Y).$$

yield the continuous subprogram  $L_{\mathbf{x}} = \{\mathbf{h}(0.9), \mathbf{f}(2.3, \mathbf{a}), \mathbf{f}(4.2, \mathbf{b})\}$ .

The joint distribution of  $\mathbf{X}$  thus defines a distribution over continuous subprograms. Specifically, for a  $|\Theta^c|$ -dimensional interval  $I = [a_{\theta_{1,1}^c}, b_{\theta_{1,1}^c}] \times \dots \times [a_{\theta_{m,1}^c}, b_{\theta_{m,1}^c}]$  (which may also be open or half-open in every dimension), one obtains the probability of the set of continuous subprograms with continuous parameters in  $I$ :

$$P_P(\mathbf{X} \in I | F^c) := \int_{a_{\theta_{1,1}^c}}^{b_{\theta_{1,1}^c}} \dots \int_{a_{\theta_{m,1}^c}}^{b_{\theta_{m,1}^c}} f(\mathbf{x}) d\mathbf{x} \quad (3)$$

*Example 3.* In example 2,  $f(\mathbf{x}) = f(x_{1,1}, x_{2,1}, x_{2,2}) = \varphi_{1,2}(x_{1,1}) \times \varphi_{4,3}(x_{1,2}) \times \varphi_{4,3}(x_{2,2})$  where  $\varphi_{\mu,\sigma}$  is the density function of a normal distribution  $N(\mu, \sigma)$ .

### 3.2 Success Probabilities of Queries

The success probability  $P_s(q)$  of a query  $q$  is the probability that  $q$  is provable from  $L \cup L_{\mathbf{x}} \cup \mathcal{BK}$ , where  $L$  is distributed according to (1), and  $\mathbf{x}$  according to  $f(\mathbf{x})$  respectively. The key to computing success probabilities is the consideration of admissible intervals, as introduced in the following definition.

**Definition 1.** An interval  $I \subseteq \mathbb{R}^{|\Theta^c|}$  is called admissible for a query  $q$  and a theory  $T = F \cup F^c \cup \mathcal{BK}$  iff

$$\forall \mathbf{x}, \mathbf{y} \in I, L \subseteq L_F : (L \cup L_{\mathbf{x}} \cup \mathcal{BK}) \models q \Leftrightarrow (L \cup L_{\mathbf{y}} \cup \mathcal{BK}) \models q \quad (4)$$

If (4) holds, we can also write  $L \cup L_I \cup \mathcal{BK} \models q$ .

A partition  $\mathcal{A} = I_1, I_2, \dots, I_k$  of  $\mathbb{R}^{|\Theta^c|}$  is called admissible for a query  $q$  and a theory  $T$  iff all  $I_i$  are admissible intervals for  $q$  and  $T$ .

In other words, an admissible interval  $I$  is “small enough” such that the values of the continuous variables – as long as they are in  $I$  – do not influence the provability of  $q$ . Within an admissible interval, the query either always fails or always succeeds for any sampled subset  $L \subseteq L_F$  of probabilistic facts.

*Example 4.* The interval  $[0, 10]$  is not admissible for the theory below and the query  $? - \mathbf{h}(\mathbf{X}), \text{ininterval}(\mathbf{X}, 5, 10)$

$$(\mathbf{X}, \text{gaussian}(1, 2)) :: \mathbf{h}(\mathbf{X})$$

since for  $x = 4 \in [0, 10]$  the query fails and for  $x = 6 \in [0, 10]$  it succeeds. The intervals  $[6, 9]$ ,  $[5, 10]$ , or  $(-\infty, 5)$  – for example – are all admissible.

**Theorem 1.** For every theory  $T$ , every query  $q$  that has only finitely many proofs, and all finite sets of possible substitutions  $\Theta, \Theta^c$ , there exists a partition of  $\mathbb{R}^{|\Theta^c|}$  that is admissible for  $T$  and  $q$ .

*Proof.* This follows from the fact that conditions defined by predicates `below/2`, `above/2`, and `ininterval/3` are satisfied by intervals of sample values, and finite combinations of such conditions that may appear in a proof still define intervals.

Given an admissible partition  $\mathcal{A}$  one obtains the success-probability of a query  $q$  as follows:

$$P_{s,\mathcal{A}}(q|T) := \sum_{L \subseteq L_F} \sum_{\substack{I \in \mathcal{A}: \\ L \cup L_I \cup \mathcal{BK} = q}} P_P(L|F) \cdot P_P(\mathbf{X} \in I|F^c) \quad (5)$$

The following theorem shows that the values of  $P_s$  are independent of the partition  $\mathcal{A}$  and we can safely write  $P_s(q|T)$  instead of  $P_{s,\mathcal{A}}(q|T)$ .

**Theorem 2.** Let  $\mathcal{A}$  and  $\mathcal{B}$  be admissible partitions for the query  $q$  and the theory  $T$  then  $P_{s,\mathcal{A}}(q|T) = P_{s,\mathcal{B}}(q|T)$ .

*Proof.* By contradiction when assuming there are two admissible partitions resulting in different distributions.

The restrictions imposed on the usage of continuous variables – namely the limitation to `below/2`, `above/2` and `ininterval/3` – provide a balance between expressivity and tractability. They allow one to discretize the space  $\mathbb{R}^n$  of possible assignments to the continuous facts in multidimensional interval such that the actual values of the continuous facts do not matter. This in turn makes efficient inference algorithms possible.

Comparing two continuous values against each other would couple them. This would require a more complicated discretization of  $\mathbb{R}^n$  in the form of polyhedra which are harder to represent and to integrate over. Allowing arbitrary functions to be applied on continuous values – ultimately – leads to a fragmentation of the space in arbitrary sets which makes exact inference virtually intractable.

## 4 Exact Inference

In this section we present an exact inference algorithm for Hybrid ProbLog. Our approach generalizes De Raedt *et. al*'s BDD algorithm [6] and generates a BDD that is evaluated by the original algorithm. The pseudocode is shown in Algorithm 1. We explain the algorithm by executing it on Example 1 and calculating the success probability of `catchcold`.

1. All proofs for `catchcold` are collected by SLD resolution. Each proof is described by a set of probabilistic facts  $f_i$ , a set of continuous facts  $c_i$ , and an interval for each continuous variable in  $c_i$ . When a continuous fact is used within a proof, it is added to  $c_i$  and the corresponding variable  $X$  is added to

---

**Algorithm 1** The inference algorithm collects all possible proofs and partitions the  $\mathbb{R}^n$  space according to the constraints imposed by each proof.

---

```

1: function SUCCESSPROB(query  $q$ , theory  $T$ )
2:    $\{(f_i, c_i, d_i)\}_{1 \leq i \leq m} \leftarrow \text{FINDALLPROOFS}(T, q)$  ▷ Backtracking in Prolog
3:   for  $c\theta \in \cup_{1 \leq i \leq m} c_i$  do ▷ Iterate over used ground continuous facts
4:      $\mathbf{A}_{c\theta} \leftarrow \text{CREATEPARTITION}(c\theta, \{d_1, \dots, d_m\})$ 
5:      $\{b_{c\theta, I}\}_{I \in A_{c\theta}} \leftarrow \text{CREATEAUXBODIES}(\mathbf{A}_{c\theta})$ 
6:      $u \leftarrow 0$  ▷ # disjoint proofs
7:     for  $i = 1, 2, \dots, m$  do ▷ Go over all proofs
8:       for  $I_1 \in \mathbf{A}_{c_1\theta_1}, \dots, I_k \in \mathbf{A}_{c_k\theta_k}$  do ▷ Go over all possible intervals
9:          $f'_u, c'_u, d'_u \leftarrow \text{DISJOIN}(f_i, c_i, d_i, \{I_1, \dots, I_k\})$  ▷  $k$  is # used cont. facts
10:         $f''_u \leftarrow f'_u \cup \{b_{c\theta, I} | c\theta \in c'_u, I \in d'_u\}$ 
11:         $u \leftarrow u + 1$ 
12:    $BDD \leftarrow \text{GENERATEBDD}(\bigvee_{1 \leq i \leq u} \bigwedge_{f \in f'_i} f)$  ▷ cf. [7]
13:   return PROB(root( $BDD$ )) ▷ cf. [6]

```

---

$d_i$  with  $X \in (-\infty, \infty)$ . When `above(X,y)` is used, the domain  $X \in I$  stored in  $d_i$  is replaced by  $X \in I \cap (\infty, y)$ , similarly for `below/2` and `ininterval/2`.

$$\begin{array}{lll}
f_1 = \{\text{rain}\} & c_1 = \{\text{temp}(\mathbf{T})\} & d_1 = \{\mathbf{T} \in (-\infty, 5)\} \\
f_2 = \emptyset & c_2 = \{\text{temp}(\mathbf{T})\} & d_2 = \{\mathbf{T} \in (-\infty, 0)\}
\end{array}$$

2. We partition  $\mathbb{R}^1$  because one continuous fact is used. The function `CREATEPARTITION(temp(T), {d1, d2})` returns the admissible partition  $[-\infty, 0)$ ,  $[0, 5)$ ,  $[5, \infty)$  which is used to split the proofs with the `DISJOIN` function:

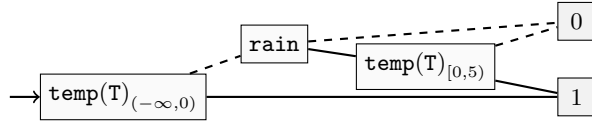
$$\begin{array}{lll}
f'_1 = \{\text{rain}\} & c'_1 = \{\text{temp}(\mathbf{T})\} & d'_1 = \{\mathbf{T} \in (-\infty, 0)\} \\
f'_2 = \{\text{rain}\} & c'_2 = \{\text{temp}(\mathbf{T})\} & d'_2 = \{\mathbf{T} \in [0, 5)\} \\
f'_3 = \emptyset & c'_3 = \{\text{temp}(\mathbf{T})\} & d'_3 = \{\mathbf{T} \in (-\infty, 0)\}
\end{array}$$

3. We create one auxiliary fact per continuous fact and interval. They are dependent – e.g.  $\text{temp}(\mathbf{T})_{[-\infty, 0)}$  and  $\text{temp}(\mathbf{T})_{[0, 5)}$  can not be true at the same time. We make the dependencies explicit by adding (conceptually) the following clauses to the theory and replacing calls to continuous facts and background predicates by the bodies  $b_{c\theta, I}$ :

$$\begin{array}{l}
\text{call\_temp}(\mathbf{T})_{(-\infty, 0)} \text{ :- } \text{temp}(\mathbf{T})_{(-\infty, 0)} \\
\text{call\_temp}(\mathbf{T})_{[0, 5)} \text{ :- } \neg \text{temp}(\mathbf{T})_{(-\infty, 0)}, \text{temp}(\mathbf{T})_{[0, 5)} \\
\text{call\_temp}(\mathbf{T})_{[5, \infty)} \text{ :- } \neg \text{temp}(\mathbf{T})_{(-\infty, 0)}, \neg \text{temp}(\mathbf{T})_{[0, 5)}, \text{temp}(\mathbf{T})_{[5, \infty)}
\end{array}$$

The probability attached to an auxiliary fact  $\text{temp}_{[l, h)}$  is the conditional probability that `temp`'s value is in  $[l, h)$  given it is not in  $(-\infty, l)$

$$\begin{array}{ll}
f''_1 = \{\text{rain}, \text{temp}(\mathbf{T})_{(-\infty, 0)}\} & f''_2 = \{\text{rain}, \neg \text{temp}(\mathbf{T})_{(-\infty, 0)}, \text{temp}_{[0, 5)}\} \\
f''_3 = \{\text{temp}(\mathbf{T})_{(-\infty, 0)}\} &
\end{array}$$



**Fig. 1.** This BDD encodes all proofs of `catchcold` in the theory from Example 1.

- As in ProbLog, this is translated into a DNF and represented as BDD, for instance the one in Figure 1.

$$\left(\text{rain} \wedge \text{temp}(\mathbf{T})_{(-\infty,0)}\right) \vee \left(\text{rain} \wedge \neg \text{temp}(\mathbf{T})_{(-\infty,0)} \wedge \text{temp}(\mathbf{T})_{[0,5)}\right) \vee \left(\text{temp}(\mathbf{T})_{(-\infty,0)}\right)$$

Evaluating the BDD with De Raedt *et al.*'s algorithm yields the success probability of `catchcold` (cf. [6, 7] for the details).

## 5 Experiments

We set up experiments<sup>5</sup> to answer the question: How does the algorithm scale in the size of the partitions and in the number of ground continuous facts?

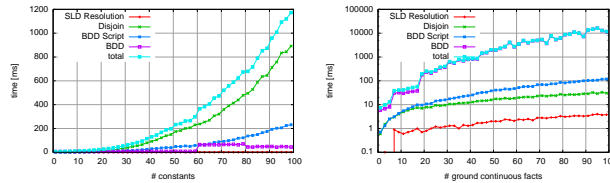
In domains where exact inference is still feasible, the number of continuous facts and comparison operations will be small in contrast to the rest of the theory. Our algorithm is an intermediate step between SLD resolution and BDD generation. Therefore, it is worthwhile to know how much the disjoining operations cost compared to the other inference steps. We used the following theory

```
(Val, gaussian(0, 1)) :: f(Val, ID).
s(Consts, Facts) :- between(1, Facts, ID), between(1, Consts, Top),
                    f(Val, ID), ininterval(Val, 0, Top/Consts).
```

and ran our algorithm to calculate the success probability of `s(Consts, Facts)`. Due to backtracking over `between/3`, the query `s(5, 3)` – for instance – uses three continuous facts (`f(Val1, 1)`, `f(Val2, 1)`, `f(Val3, 3)`) and compares them to the intervals  $[0, 1], [0, \frac{1}{2}], \dots, [0, \frac{1}{5}]$ . In general, this leads to partitions of size  $(\text{Consts} + 1)^{\text{Facts}}$  of the space  $\mathbb{R}^{\text{Facts}}$ .

We ran two series of queries. First, we used one continuous fact and varied the number of constants from 1 to 100. As the graph in Figure 2 (left) shows, the disjoin operation – that is finding all partitions, generating the auxiliary bodies and rewriting the proofs – runs in  $O(\text{Consts}^2)$  when everything else stays constant. Due to compression and pruning operations during the BDD script generation [7] building and evaluating the BDD runs in quasi-linear time

<sup>5</sup> We implemented Hybrid ProbLog in YAP 6.0.0. The experiments were performed on an Intel Core 2 Quad machine with 2.83GHz and 8GB of memory. We used the CUDD package for BDD operations and set the reordering heuristics to CUDD.REORDER.GROUP\_SIFT. Each query has been evaluated ten times and the runtimes were averaged.



**Fig. 2.** Runtimes for calculating the success probability of  $s(\text{Consts}, \text{Facts})$ , (left) for varying the number of constants  $s(1, 1), \dots, s(100, 1)$ , and (right) for varying the number of dimensions  $s(5, 1), \dots, s(5, 100)$

in this case. In the second run, we varied the number of continuous facts As Figure 2 (right) shows, our algorithm (depicted by the Disjoin graph) runs in linear time. The runtime for the BDD operations grows exponentially due to the heuristics used to find an optimal variable ordering.

## 6 Conclusions and Future Work

We extended ProbLog with continuous distributions and introduced an exact inference algorithm. We restricted the expressivity to keep inference tractable. We omitted results on parameter learning but plan to include them in the longer version of this paper. Possible directions for future work include allowing comparisons between continuous facts and applying functions on continuous values.

## Acknowledgments

Bernd Gutmann is supported by the Research Foundation-Flanders (FWO-Vlaanderen). This work is supported by the GOA/08/008 project “Probabilistic Logic Learning”.

## References

1. Vennekens, J., Denecker, M., Bruynooghe, M.: Representing causal information about a probabilistic process. In: JELIA. Volume 4160 of LNCS. (2006) 452–464
2. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP. (1995) 715–729
3. Milch, B., Marthi, B., Russell, S.J., Sontag, D., Ong, D.L., Kolobov, A.: Blog: Probabilistic models with unknown objects. In: IJCAI. (2005) 1352–1359
4. Goodman, N., Mansinghka, V.K., Roy, D.M., Bonawitz, K., Tenenbaum, J.B.: Church: a language for generative models. In: UAI. (2008) 220–229
5. Wang, J., Domingos, P.: Hybrid markov logic networks. In: Proceedings of the Twenty-Third National Conference on Artificial Intelligence. (2008)
6. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: IJCAI. (2007) 2462–2467
7. Kimmig, A., Santos Costa, V., Rocha, R., Demoen, B., De Raedt, L.: On the efficient execution of ProbLog programs. In: ICLP. Volume 5366 of LNCS. (2008) 175–189