A Comparison of Stochastic Logic Programs and Bayesian Logic Programs

Aymeric Puech and Stephen Muggleton
Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2BZ, UK
{atp02,shm}@doc.ic.ac.uk

Abstract

First-order probabilistic models are recognized as efficient frameworks to represent several real-world problems: they combine the expressive power of first-order logic, which serves as a knowledge representation language, and the capability to model uncertainty with probabilities. Among existing models, it is usual to distinguish the *domain-frequency* approach from the *possible-worlds* approach.

Bayesian logic programs (BLPs, which conveniently encode *possible-worlds* semantics) and stochastic logic programs (SLPs, often referred to as a *domain-frequency* approach) are promising probabilistic logic models in their categories.

This paper is aimed at comparing the respective expressive power of these frameworks. We demonstrate relations between SLPs' and BLPs' semantics, and argue that SLPs can encode the same knowledge as a subclass of BLPs. We introduce *extended* SLPs which lift the latter result to any BLP. Converse properties are reviewed, and we show how BLPs can define the same semantics as complete, range-restricted, non-recursive SLPs. Algorithms that translate BLPs into SLPs (and vice versa) are provided, as well as worked examples of the inter-translations of SLPs and BLPs.

1 Introduction

The recent study APrIL [Muggleton and de Raedt, 2001] aimed at assessing how probabilistic reasoning could be integrated with first-order logic representations and machine learning. The project ended in June 2002 and raised the conclusion that, *due to the requirements for the use in functional genomics, stochastic logic programs (SLPs) and Bayesian logic programs (BLPs) [were] the most promising formalisms. They represent the most expressive frameworks as they allow not only for constant and predicate symbols but also for functors. Moreover, both formalisms are alternative and complementary formalisms.* Indeed, in BLPs the possible-worlds perspective is dominant, while SLPs use a domain-frequency approach.

1.1 Motivations

Comparing these two major approaches looks like an interesting issue. The study of the relations between the *possible-worlds* and *domain-frequency* approaches, which is not new, crops up in many articles. Halpern dedicated a paper to that study [Halpern, 1989]. Furthermore, the idea of investigating the relations between BLPs and SLPs in particular has been put forward in [Kersting and Raedt, 2000] by Kersting and de Raedt, who reckon at the very end of their article: *Exploring the relations between Bayesian logic programs and stochastic logic programs is interesting because: (1) both are using SLD trees and (2) transformations between probabilities on the domain and probabilities on possible worlds exist as Halpern noted.*

Essentially two questions at two different levels can be asked:

• Given an expert domain, can we encode the same knowledge in a BLP and in a SLP?

• More practically: can we translate a BLP into an SLP, and an SLP into a BLP? What are the respective interests of these formulations? In particular, does the translation have the same features (computation of the probabilities, inference with or without evidence)?

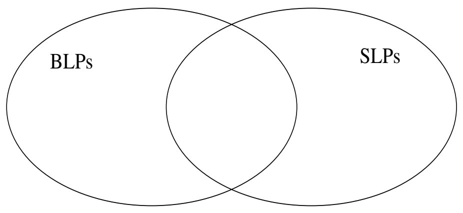


Figure 1: BLPs, SLPs... What is at the intersection?

We argue that it is possible to encode a complete, range-restricted and non-recursive SLP with a BLP, and that the converse holds for a certain subclass of BLPs (the *restricted* BLPs, whose predicate definitions contain a single clause). This latter result is lifted to all BLPs by introducing *extended SLPs* (e-SLPs), which are SLPs augmented with combining functions, and whose evaluation is made in stochastic SLD *and-or* trees instead of stochastic SLD-trees for SLPs.

1.2 Outline

The section 2 of this article is dedicated to some prerequisites (we recall the usual syntax and semantics of SLPs and BLPs). We also demonstrate relations between SLPs' and BLPs' semantics, hence we eventually reduce the issue of inter-translations of SLPs and BLPs to the practical question: given a BLP (resp. an SLP), how can we find an SLP (resp. a BLP) that encodes equivalent distributions of probabilities with respect to the semantics?

The section 3 presents some ways to construct and evaluate an SLP which computes equivalent probabilities to a given BLP.

- Firstly, a *standard translation* is proposed, which deals with the subclass of restricted BLPs, whose *and-or trees* (as defined in [Kersting and Raedt, 2000]) contain only *and*-nodes (these BLPs don't make use of combining rules). *Extended* SLPs are introduced, and the latter result is lifted to all BLPs.
- However, standard translations can only be queried without evidence. Thus a *BN*[1] *translation* is also proposed, which can be queried with evidence but works only for BLPs with finite Herbrandt model (which obviously prevent the use of functors). Finally, we show how this result can be lifted to all BLPs, provided that we insert a KBMC[2]-inspired stage in the query-answering procedure.

In section 4, we provide a converse translation (from SLP to BLP) and prove that it computes equivalent distributions of probabilities.

We conclude that extended SLPs and BLPs can encode the same knowledge (although their formalism is more or less intuitive, depending on the kind of knowledge we want to model).

2 Background

2.1 Stochastic Logic Programs (SLP)

Stochastic logic programs were first introduced by Stephen Muggleton in 1996, as a generalization of stochastic grammars.

Syntax:

As defined in [Muggleton, 2001], a SLP consists of a set of labelled clauses $p : C$, where p is from the interval $[0, 1]$, and C is a range-restricted[3] definite clause. Later in this report, the labelled clauses $p : C$ will be named *parameterized* clauses or *stochastic* clauses. The simplest example of SLP is the coin example which mimics the action of a fair coin. The probability of the coin coming up either head-side up (0) or tail-side up (1) is 0.5:

$0.5 \quad : \quad coin(0) \leftarrow$
$0.5 \quad : \quad coin(1) \leftarrow$

In [Muggleton, 2001], SLP definition requires that for each predicate symbol q, the probability labels for all clauses with q in the head sum to 1. However, this can be a restrictive definition of SLPs. In other articles ([Cussens, 2000] for instance), SLPs having this property are called ***complete*** SLPs, while in ***uncomplete*** SLPs, the probability labels for all clauses with a same predicate symbol in the head sum to less than 1. In [Cussens, 2000] James Cussens introduces ***pure*** SLPs, whose clauses are all parameterized (whereas ***impure*** SLPs can have non-parameterized clauses, that is, definite logical clauses). Furthermore, ***normalized*** SLPs are like complete SLPs, but in ***unnormalised*** SLPs, the probability labels for all clauses with a same predicate symbol in the head can sum to any positive value other than 1.

Semantics:

A stochastic logic program P has a distributional semantics, that is one which assigns a probability distribution to the atoms of each predicate in the Herbrand base of the clauses in P. These probabilities are assigned to atoms according to an SLD-resolution strategy which employs a **stochastic selection rule**[4].

In [Cussens, 1999b], three different related distributions are defined, over derivations, refutations and atoms. Given an SLP P with n parameterized clauses and a goal G, it is easy to define a ***log-linear probability distribution over the set of derivations***, by considering the function:

$$\psi_\lambda(x) = e^{\lambda.\nu(x)} = \prod_{i=1}^{n} {l_i}^{\nu_i(x)}$$

where

- x is a derivation in the set of derivations from the goal G.
- $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n) \in \Re^n$ is a vector of log-parameters where $\lambda_i = log(l_i)$, l_i being the label of the clause i.
- $\nu = (\nu_1, \nu_2, ..., \nu_n) \in N^n$ is a vector of clause counts s.t. $\nu_i(x)$ is the number of times the ith parameterized clause is used in the derivation x.

The proof that $\psi_\lambda(x)$ is a probability distribution (provided that P is pure and normalized) can be found in [Cussens, 2000]. So far we have defined a probability distribution over all possible derivations, but we are mainly interested in the refutations of the goal G. Now, if we assign the probability 0 to all derivations that are not

[1]BN stands for Bayesian network.

[2]Knowledge-Based Model Construction, as defined in [Kersting and Raedt, 2000].

[3]C is said to be range-restricted iff every variable in the head of C is found in the body of C.

[4]The selection rule is not deterministic but stochastic; the probability that a clause is selected depends on the values of the labels (details can be found in [Muggleton, 2001]).

refutations of the goal G, and if we normalize the remaining probabilities with a normalization factor Z, we obtain the ***probability distribution*** $f_\lambda(r)$ ***over the set*** R ***of the refutations of G***:

$$f_\lambda(r) \;=\; Z_{\lambda,G}^{-1}\; e^{\lambda.\nu(r)}$$

Given an SLP S and a ground query G, the probability of G as defined in the distributional semantics [Muggleton, 2000] is equal to $\sum_{r\in R}\psi_\lambda(r)$ and is noted $P(G\;/\;S)$.

Each refutation r involves some bindings along the SLD-tree, which permits finding the computed answer for G using r. The computed answer is the most general instance of the goal G that is refuted by r; it is also named the ***yield atom***. Let us note $X(y)$ the set of refutations which lead to the yield atom y. We can finally define a ***distribution of probabilities over the set of yield atoms***, with the function:

$$p_{\lambda,G}(y) \;=\; \sum_{r\in X(y)} f_\lambda(r) \;=\; Z_{\lambda,G}^{-1} \sum_{r\in X(y)} \left(\prod_{i=1}^{n} l_i^{\nu_i(r)}\right)$$

Thus it is fairly simple to define probability-distributions over the yield atoms in SLPs: we can use the **stochastic SLD-tree**, which is the SLD-tree in which each vertex (which corresponds to a resolution step) is labelled with the parameter of the clause that is used in the resolution step. Since every refutation of the goal G corresponds in the stochastic SLD-tree to a branch from the root to a leaf, we only have to:
- draw the stochastic SLD tree,
- compute the probability of each refutation by multiplying the labels along the corresponding branch,
- normalize the probabilities, so that they sum to 1,
- associate each refutation to a yield atom, and sum the probabilities of the refutations that lead to the same yield atom.

We can illustrate this method with Cussens' sample SLP:

0.4	:	$s(X) \leftarrow\; p(X), p(X).$
0.6	:	$s(X) \leftarrow\; q(X).$
0.3	:	$p(a).$
0.7	:	$p(b).$
0.2	:	$q(a).$
0.8	:	$q(b).$

We take the goal $G \;=\; s(X)$; the stochastic SLD tree which derives from the query $:-\; s(X)$ is:

There are 4 refutations of the goal (r_1 to r_4). The yield atom is $s(a)$ for r_1 and r_3, $s(b)$ for r_2 and r_4. Thus the probability distribution over $\{s(a), s(b)\}$ is $\{Z^{-1}\times(0.4\times 0.3\times 0.3+0.6\times 0.2),\; Z^{-1}\times(0.4\times 0.7\times 0.7+0.6\times 0.8)\}$ where Z is a normalization constant.

2.2 BLPs

Bayesian logic programs were first introduced by Luc de Raedt and Kristian Kersting in 2000, as a generalization of Bayesian nets (BNs) and Logic Programs.

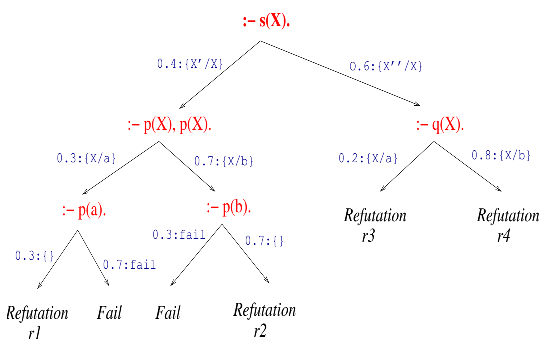


Figure 2: Stochastic SLD tree for the query *?- s(X).*

Syntax:

A Bayesian logic program has 2 components: a *logical* one (which is a set of *Bayesian clauses*) and a *quantitative* one (a set of conditional probability distributions and *combining rules* corresponding to that logical structure).

A *Bayesian clause* is an expression of the form:

$$A \mid A_1, ..., A_n$$

where $n \geq 0$ and the A_i are *Bayesian atoms* which are (implicitly) universally quantified. The difference between a logical definite clause and a Bayesian clause is that:
- the sign $\mid$ is employed instead of $:-$,
- Bayesian atoms are assigned a (finite) *domain*, whereas first order logic atoms have binary values. Following the definitions in [Kersting and Raedt, 2000], we assume that atom domains in BLPs are discrete.

In order to represent a probabilistic model, we associate with each Bayesian clause c a conditional probability distribution $cpd(c)$ which encodes the probability that $head(c)$ takes some value, given the values of the Bayesian atoms in $body(c)$:

$$\mathbf{P}(head(c)|body(c))$$

This conditional probability distribution is represented with a matrix called *conditional probability table* (CPT).

As there can be many clauses with the same head (or non-ground heads that can be unified), we use *combining rules* to obtain the distribution required, i.e. functions which map finite sets of conditional probability distributions onto one *combined* conditional probability distribution. Common combining rules include the *noisy-or* rule, when domains are boolean, and the *max* rule, which is defined on finite domains.

Semantics:

The link to Bayesian networks is now straightforward: each ground Bayesian atom can be associated to a chance node (a standard random variable), whose set of states is the domain of the Bayesian atom. The links (influence relations) between chance nodes are given by the Bayesian clauses, and the link matrices by the conditional probability distributions associated to these Bayesian clauses.

The set of ground Bayesian atoms in the least Herbrand model[5] together with the structure defined by the set of ground instances of the Bayesian clauses and the conditional probability tables, define a global (possibly infinite) Bayesian network that can be queried like any other Bayesian net[6].

Thus the query-answering procedure actually consists of two parts: first, given a ground query and some evidence, the Bayesian network containing all *relevant* atoms is computed, using KBMC (*Knowledge Based Model Construction*). Then the resulting Bayesian network can be queried using any available inference algorithm, the results we were looking for being the probability of the initial ground query over its domain. Further details about the query-answering procedure can be found in [Kersting and Raedt, 2000].

Remark: The distribution is not defined if there is a cycle, as stated in [Kersting and Raedt, 2000]. In the remainder of the paper, in order to use the conditional independency assumption between the parents of a node in a Bayesian net, we assume that no merging of nodes takes place in the and-or tree (when constructing the Bayesian net from the and-or tree). This leads to a Bayesian net which consists in a singly connected network (with no loop). Ground queries satisfying this property are said to be *safe* with regards to the considered BLP (cf. next definition).

2.3 Formulation of the problem

Halpern's paper [Halpern, 1989] as well as [Cussens, 1999a] are good clarifications about what respective kinds of knowledge can be captured with probabilities on the domain (such as those defined by SLPs) and probabilities on possible worlds (BLPs). Links between these probabilities are also provided.

Let B be a BLP and G_a a ground query. The Bayesian network constructed with KBMC (as defined in [Kersting and Raedt, 2000]) is denoted by BN_{B,G_a}. The probability of a chance node Q taking the value v in BN_{B,G_a} (i.e. the probability of the set of possible worlds of BN_{B,G_a} in which Q has the value v) will be denoted $P_{B,G_a}(Q = v)$. Given a SLP S, the probability of a ground query G (as defined in the distributional semantics [Muggleton, 2000] and in section 3) is noted $P(G \ / \ S)$.

The fact that a k-ary Bayesian atom G_a takes the value v can be represented with a $(k + 1)$-ary logical atom G having the same predicate and k first arguments as G_a, and the value v as last argument. Conversely, we can identify any logical atom to a Bayesian atom having the domain $\{true, false\}$ and taking the value *true* whenever the logical atom holds.

Hence we will claim that a BLP and an SLP define *equivalent semantics* if the probability that any ground Bayesian atom G_a in the Herbrand model of the BLP takes some value v is identical to the probability of the associated logical atom G in S:

$$P(G \ / \ S) \ = \ P_{B,G_a}(G_a = v)$$

Given these relations between SLPs' and BLPs' semantics, we eventually reduce the issue of inter-translations of SLPs and BLPs to the practical question: given a BLP (resp. an SLP), how can we find an SLP (resp. a BLP) that encodes equivalent distributions of probabilities with respect to the semantics?

3 From BLPs to SLPs

We provide two translations from BLPs to SLPs. The *standard translation* (3.1) exists in 2 versions: one uses SLPs and works for a subclass of BLPs (the *restricted* BLPs). It is generalized to all BLPs in the second version , which makes use of e-SLPs (SLPs augmented with combining functions). However, these standard translations eventually work for all BLPs but do not handle *evidence* (that is: some prior knowledge about the domain, which corresponds to the instantiation of a chance node in a BN). The reason is that SLPs and e-SLPs define semantics on tree structures, whereas KBMC (in BLPs) permits the union of several trees, hence the computation of probabilities in singly connected networks. Thus we also provide a more global approach with the *BN translation* (3.2), which is based on Cussens' translation of Bayesian networks into SLPs.

3.1 Standard Translations

Standard Translation for Restricted BLPs:

Definition: If S is an SLP, the subset S_h of clauses in S with predicate symbol h in the head is called the definition of h. A restricted BLP is a BLP whose predicate definitions contain one single stochastic clause each. A ground query G_a is said to be safe with regards to a BLP B if the and-or tree rooted at G_a does not contain 2 identical nodes (no merging of nodes takes place during KBMC). $\mathcal{N}_n$ is the set of natural numbers from 1 to n.

Definition (standard translation of a restricted BLP):

Let B denote a restricted BLP.

- Identify each k-ary Bayesian atom b, which appears in B and has the value domain V, to the $(k+1)$-ary (logical) atom $b(v_b)$ having the same k first arguments and a value v_b of V as last argument.
- For each Bayesian clause $head|b_1, ..., b_n$ in B, for each value in the associated CPT, which indicates the probability $p_{v_h,v_{b1},...,v_{bn}}$ that the Bayesian atom *head* takes the value v_h given that the $\{b_i \ : \ i \in \mathcal{N}_n\}$ take the values $(v_{b1}, ..., v_{bn})$, construct the stochastic

[5]We define the least Herbrand model of a BLP in the same way as in logic programs.

[6]Bayesian networks are formally defined only for finite sets of chance nodes; this point of view is put forward because it provides a better idea of the relations between BLPs and Bayesian nets.

clause consisting of the parameter $p_{v_h, v_{b1}, \ldots, v_{bn}}$, and the definite clause:

$$head(v_h) \;\leftarrow\; b_1(v_{b1}),\; \ldots,\; b_n(v_{bn})$$

- The standard translation of B consists of the n stochastic clauses constructible in that way, n being the sum of the numbers of coefficients in the CPTs. This SLP is pure and unnormalised (the parameters of the clauses in $S_h \subseteq S$ sum to the product of the domain sizes of the Bayesian atoms in the body of the Bayesian clause with head h).

Theorem: Given a restricted BLP B, its standard translation S obtained as defined above, and a ground Bayesian query G_a which is safe with regards to B. Let us associate to G_a the logical query $G(v)$, $v \in dom(G_a)$. Then: $P(G(v)/S) = P_{B,G_a}(G_a = v)$.

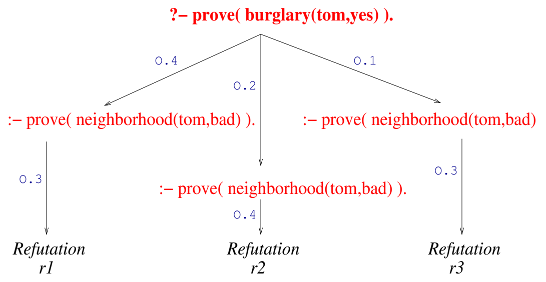
We illustrate the standard translation mechanism through this simple example:

Example:

Let us take the following standard translation of a BLP (the original BLP does not need to be mentioned):

0.99	:	$alarm(A, yes) \leftarrow burglary(A, yes), tornado(A, yes).$
0.80	:	$alarm(A, yes) \leftarrow burglary(A, yes), tornado(A, no).$
0.90	:	$alarm(A, yes) \leftarrow burglary(A, no), tornado(A, yes).$
0.05	:	$alarm(A, yes) \leftarrow burglary(A, no), tornado(A, no).$
0.01	:	$alarm(A, no) \leftarrow burglary(A, yes), tornado(A, yes).$
0.20	:	$alarm(A, no) \leftarrow burglary(A, yes), tornado(A, no).$
0.10	:	$alarm(A, no) \leftarrow burglary(A, no), tornado(A, yes).$
0.95	:	$alarm(A, no) \leftarrow burglary(A, no), tornado(A, no).$
0.4	:	$burglary(A, yes) \leftarrow neighborhood(A, bad).$
0.2	:	$burglary(A, yes) \leftarrow neighborhood(A, avg).$
0.1	:	$burglary(A, yes) \leftarrow neighborhood(A, good).$
0.6	:	$burglary(A, no) \leftarrow neighborhood(A, bad).$
0.8	:	$burglary(A, no) \leftarrow neighborhood(A, avg).$
0.9	:	$burglary(A, no) \leftarrow neighborhood(A, good).$
0.3	:	$neighborhood(tom, bad).$
0.4	:	$neighborhood(tom, avg).$
0.3	:	$neighborhood(tom, good).$
0.01	:	$tornado(tom, yes).$
0.99	:	$tornado(tom, no).$

We want to compute the probability $P(burglary(tom, yes) \;/\; S)$. Each refutation in the stochastic SLD-tree rooted at $burglary(tom, yes)$ permits the computation of the probability of a particular set of possible worlds. The nodes along the refutation correspond to instantiations of some Bayesian atoms. The set of possible worlds we are talking about is the set of worlds where the instantiations defined along the refutation hold.



?– prove(burglary(tom,yes)).
0.4
0.2
0.1
:– prove(neighborhood(tom,bad)).
:– prove(neighborhood(tom,bad)
:– prove(neighborhood(tom,bad)).
0.3
0.4
0.3
Refutation r1
Refutation r2
Refutation r3

Figure 3: Stochastic SLD tree for $?\!- burglary(tom, yes)$.

- The refutation 1 is associated to the set of possible worlds in which $\{burglary(tom) = yes, neighborhood(tom) = bad\}$. $P = 0.4 \times 0.3 = 0.12$
- The refutation 2 is associated to the set of possible worlds in which $\{burglary(tom) = yes, neighborhood(tom) = avg\}$. $P = 0.2 \times 0.4 = 0.08$
- The refutation 3 is associated to the set of possible worlds in which $\{burglary(tom) = yes, neighborhood(tom) = good\}$. $P = 0.1 \times 0.3 = 0.03$.

Hence the probability $P(burglary(tom, yes) \;/\; S)$ is equal to $0.12 + 0.08 + 0.03 = 0.23$. In the BN constructed from the original BLP, we have as well: $P(burglary(tom, yes)) = P(\{possible\ worlds\ where\ burglary(tom) = yes\}) = 0.12 + 0.08 + 0.03 = 0.23$.

Extended SLPs:

Restricted BLPs don't make use of the *or-nodes*, in that all queries can match at most one head. In order to lift the latter result to general BLPs, we have to introduce an extension of SLPs, namely *extended SLPs*.

Definition (Syntax of Extended SLPs): An extended SLP (e-SLP) is an SLP S augmented with a set of *combining functions* CR_h, for all predicates h appearing in the head of some stochastic clause in S. A combining function is a function that maps a set of possible resolvents of h (obtained using one clause in S_h) and associated real numbers in $[0, 1]$ to a real number in $[0, 1]$:

$$CR_h \;:\; ((r_1, p_1), \ldots, (r_n, p_n)) \;\mapsto\; r \in [0, 1]$$

Definition (Proof of Extended SLPs): Given an e-SLP S_e consisting of the SLP S and the combining functions $(CR_h)_h$, and a query Q (consisting of the predicate p with none or more arguments), the probability $P_e(Q/S_e)$ is the probability of the *pruned* and-or tree T rooted at the or-node Q. The probability of a pruned and-or tree is defined by structural induction:

- Base case: if T is a single or-node, $P_e(Q/S_e)$ is the probability $P(Q/S)$.
- If the root of T is an or-node with n branches leading to the resolvents (and-nodes) $(r_i)_{i \in \mathcal{N}_n}$, then $P_e(Q/S_e) = CR_p((r_i, p_i)_{i \in \mathcal{N}_n})$, where p_i is the probability of the pruned and-or subtree rooted at the and-node r_i.

- If the root of T is an and-node leading to the resolvents (or-nodes) $(r_i)_{i\in\mathcal{N}_n}$, then $P_e(Q/S_e) = \prod_{i=1}^{n} p_i$, where p_i is the probability of the pruned and-or subtree rooted at the or-node r_i.

Remark: The probability $P_e(Q/S_e)$ is defined by structural induction in SLD and-or trees (while SLPs' distributional semantics make use of SLD (and-)trees). The computation of the probabilities at the and-nodes is the same as in SLPs' distributional semantics: it is simply a product. At or-nodes, probabilities are computed using the newly added combining functions.

Standard Translation for BLPs:

Definition (Standard Translation of a BLP): Let B denote a BLP. The standard translation of B is the extended SLP S_e defined by the following stochastic clauses and combining functions:

- The stochastic clauses (which form the set S) are obtained in the same way as the stochastic clauses obtained from a restricted BLP (definition 2).
- Let us take a ground predicate h in the head of some clause in S and assume that it can be unified with the heads of some clauses in S_h, leading to the resolvents $(r_{i,j})_{i,j}$ with probabilities in S equal to $(p_{i,j})_{i,j}$. A resolvent can contain several atoms. The clauses in S_h come from z different Bayesian clauses with the same predicate in the head. These original clauses can be indexed with a number that corresponds to the first index $i \in \mathcal{N}_z$ in the name of the resolvents. The second index $j \in \mathcal{N}_{n_i}$ refers to one of the n_i different distributions of values over the Bayesian atoms in the body of the Bayesian clause i. We define CR_h by:

$$CR_h = \sum_{j_1\in\mathcal{N}_{n_1},\ldots,j_z\in\mathcal{N}_{n_z}} CR(h, r_{1,j_1}, \ldots, r_{z,j_z}) \times \prod_{t=1}^{z} p_{t,j_t}$$

where CR is the combining rule defined in B.

Proposition: The theorem 1 stating the equivalence of the semantics for the standard translation of restricted BLPs still holds if the translation is done with the latter rules (using e-SLPs and probabilities defined in definition 4).

Theorem: Given any BLP B, its standard translation S_e obtained as defined above, and a ground Bayesian query G_a which is safe with regards to B. Let us associate to G_a the logical query $G(v)$, $v \in dom(G_a)$. Then: $P_e(G(v)/S_e) = P_{B,G_a}(G_a = v)$.

It is worth saying a word about the computational complexities involved in the standard translation. Although the latter representation is less compact than the original BLP (there is one stochastic clause for each parameter in the CPTs of the BLP), the computation of probabilities is more efficient. Indeed, queries in standard translations can be answered by simply summing the probabilities of refutations in a stochastic SLD-tree. On the other hand, querying the original BLP requires the computation of the pruned and-or tree, the transformation of this tree into a BN, and the initialization of the constructed BN (usually with a message passing algorithm such as Pearl's).

3.2 BN Translations

It is a well-known result that BNs can be formulated in terms of SLPs. The next subsection recalls Cussens' suggestion of encoding. Since Herbrand bases of the BLPs define (possibly infinite) Bayesian networks, a possible way of translating a BLP into an SLP is to encode the corresponding Bayesian net.

From Bayesian Nets to SLPs:

According to Cussens, *unnormalised SLPs can conveniently represent Bayesian nets.*

The encoding is presented throughout an example. Let us take the following Bayesian net (figure 4).

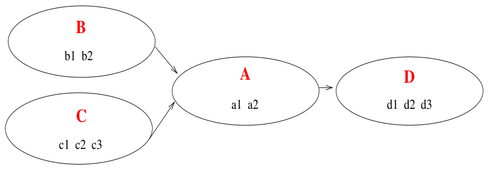


Figure 4: A simple Bayesian net

The following SLP encodes this Bayesian network:

```
1: world(A,B,C,D) :-
       b(B), c(C),
       a(A,B,C),
       d(D,A).

0.05: b(b1).
0.95: b(b2).

0.07: a(a1,b2,c1).
...
```

The first stochastic clause[7] permits the *possible-worlds* approach, since **the probability of the refutation of a ground instance of the predicate symbol *world* is exactly the probability of the associated possible world**.

The next clauses encode the values that are contained in the CPTs. They are unary stochastic clauses *groundatom* associated to the parameter $P(groundatom)$.

From Cussens' point of view the translation from BN to SLP seems obscure in that the directionality of BN is obscured. Indeed, the labels of the clauses in the resulting SLP encode *conditional* probabilities, while the predicate symbols don't seem to make any distinction between the head and the other variables the head depends on. However one can argue that the structure of the BN is encoded in the first stochastic clause, the *query clause*

[7]This clause is obviously different from all other clauses in the SLP. We will call it the ***query clause*** of the SLP.

as defined above. This approach works pretty well and will be used in the next section as the basis for a new translation of BLPs (namely: BN translations).

Lifting this approach to BLPs:

The resulting SLP falls into two parts: a *data* part (consisting of several stochastic clauses which encode the knowledge in the conditional probability tables) and a *query* part (consisting of only one stochastic clause with the parameter 1.0).

Data Part

In order to mimic the global approach that was presented in the previous section, we propose to use the following clauses to encode the knowledge in the conditional probability tables of our example:

$$\begin{array}{lcl}
0.99 & : & holds_1(alarm(A,yes), burglary(A,yes), ... \\
 & & \quad ...\ tornado(A,yes)). \\
0.80 & : & holds_1(alarm(A,yes), burglary(A,yes), ... \\
 & & \quad ...\ tornado(A,no)). \\
... & & ... \\
0.40 & : & holds_2(burglary(A,yes), neighborhood(A,bad)). \\
... & & ... \\
0.99 & : & holds_4(tornado(tom,no)).
\end{array}$$

Note that *alarm*, *burglary*... were predicate symbols in the last *standard* translation, while we consider them as functors from now on.

Query Part

Let us recall that a BLP defines a (possibly infinite) Bayesian network whose chance nodes are the atoms in the least Herbrand model associated to the set of Bayesian clauses (this set can indeed be identified to a DCL (Prolog-like) program). Now there are two options:

• **A)** Either **the least Herbrand model** HM **is finite**: in that case the *query clause* can contain all atoms in HM, and the probability of the refutation of a ground instance of the query clause is exactly the probability of the corresponding possible world.

In the *alarm* example, the Herbrand model HM is finite: it contains exactly $alarm(tom)$, $burglary(tom)$, $neighborhood(tom)$ and $tornado(tom)$.

Thus the query clause will be:

$$\begin{array}{lcl}
1 & : & holds(\\
 & & \quad tornado(tom,A), \\
 & & \quad alarm(tom,B), \\
 & & \quad burglary(tom,C), \\
 & & \quad neighborhood(tom,D)\ ) \leftarrow \\
 & & holds_3(neighborhood(tom,D)), \\
 & & holds_2(burglary(tom,C), neighborhood(tom,D)), \\
 & & holds_4(tornado(tom,A)), \\
 & & holds_1(alarm(tom,B), burglary(tom,C), \\
 & & \qquad tornado(tom,A)).
\end{array}$$

Implementations have been carried out in Prolog, so that it is possible to query the resulting SLP (data part + query clause) by asking Prolog:

$$?-\ query(\ holds(\ tornado(tom,A), alarm(tom,B), burglary(tom,C), neighborhood(tom,D)\ )\ ).$$

The variables A, B, C or D can be replaced by constants (*yes*, *no*, *good*, *bad* or *avg* when appropriate), whenever evidence must be taken into account.

• **B)**... Or **the least Herbrand model** HM **is infinite**: then we don't know what a *possible world* will consist of, since it clearly depends on the query. Furthermore, we don't even know what evidence can be declared (the actual chance nodes of the Bayesian net depend on the query). Thus **we need some additional stage to query the SLP**, which can replace the -cumbersome- construction of the Bayesian net with KBMC.

To construct the query clause, we need to have an idea of the corresponding Bayesian net that would be generated in the BLP with the same query. In KBMC, all *relevant* Bayesian atoms are determined by calculating the and-or trees of the query and the evidence, and by merging these trees. We use a slightly different approach[8]: given a query (assimilated to a ground Bayesian atom), we use the structure of the Bayesian clauses to determine: - all *influencers* of the query: this is a kind of *deductive* approach, since the influencers are the atoms that appear in the refutation of the query, when assimilating the BLP with a LP. - all *influenced atoms* of the query: this is a kind of *abductive* approach, since we will look for the Bayesian clauses whose body contains the query.

4 From SLPs to BLPs

In [Kersting and Raedt, 2000], Kersting and de Raedt show that any logic program can be formulated in terms of BLPs: they assign the domain $\{true, false\}$ to every atom in the Herbrand base of the logic program, and associate the naive conditional probability tables to the clauses, which is defined as follows:

- the probability that the head takes the value *true* given that all atoms in the body have the value *true* is 1.0.
- the probability that the head takes the value *true* given any other distribution of values over the atoms in the body is 0.0.

Kersting and de Raedt claim that this BLP (together with the *noisy-or* or the *max* combining rule) mimics the original logic program. How can we lift this translation mechanism to SLPs?

In order to shift the approach from a possible-worlds to a domain-frequency perspective (which is essentially a *single-world* perspective), the idea is to assign non-zero probabilities to *only one* set of values of the body. Here we propose a way to translate into a BLP: the resulting BLP can compute the same probabilities as the distributional semantics defined in [Muggleton, 2001].

Definition (translation of an SLP):

Let P denote a complete, range-restricted and non-recursive SLP.

[8]This approach is detailed in [Puech, 2003]; this is only a summary.

- For each stochastic clause $p \;:\; head \leftarrow b_1, ..., b_n$ in P, identify each atom to a Bayesian atom whose domain is $\{true, false\}$.
- Construct the Bayesian clause having the same head, the same body, and the following conditional probability table:

			head	
b_1	...	b_n	true	false
true	true	true	p	$1-p$
true	true	false	0	1
≀	≀	≀	0	1
false	false	false	0	1

- To complete the definition of the BLP, we need to define a *combining rule* CR. Suppose that we have to combine n conditional probability tables CPT_i $(1 \leq i \leq n)$. Each CPT_i defines the probabilities $P(head \mid \mathcal{B}_i)$, where $\mathcal{B}_i$ is the set of ground Bayesian atoms in the body of the associated clause. Thus to define $CR(\ (CPT_i)_{1 \leq i \leq n}\ )$, and by using normalization, we only have to set the values of $P(head = true \mid \cup_{i=1}^{n} \mathcal{B}_i)$ for all possible instantiations of the ground Bayesian atoms in $(\cup_{i=1}^{n} \mathcal{B}_i)$. The value of $P(head = false | \cup_{i=1}^{n} \mathcal{B}_i) = 1 - P(head = true | \cup_{i=1}^{n} \mathcal{B}_i)$ can then be deduced.
- For each possible instantiation $(\cup_{i=1}^{n} Inst_i)$ of $(\cup_{i=1}^{n} \mathcal{B}_i)$, we take the **sum** $\sum_{i=1}^{n} P(head = true \mid \mathcal{B}_i = Inst_i)$ and assign it to $P(head = true \mid \cup_{i=1}^{n} \mathcal{B}_i)$. Since the SLP is complete, this sum will never be greater than 1, and the CR is well defined.

We will now examine two examples.

Example (Unbiased coin): Let us recall the coin example. The SLP is complete, range-restricted and not recursive.

$0.5 \;:\; coin(0) \leftarrow$
$0.5 \;:\; coin(1) \leftarrow$

In the coin problem, the very simple BLP that we construct doesn't make use of any combining rule. It only contains 2 Bayesian clauses. The BLP will be written:

```
coin(0).
coin(1).
```

coin(0)	
true	false
0.5	0.5

coin(1)	
true	false
0.5	0.5

If we query this BLP with, for example, "$? - coin(1)$.", the KBMC will result in the Bayesian net containing the single chance node *coin(1)*, whose probability of being *true* will be 0.5 as required by the distributional semantics in [Muggleton, 2001].

Let us take a more complex example.

Example:

Let P be the complete, range-restricted SLP consisting of the following stochastic clauses:

0.4	:	$s(X) \leftarrow p(X), q(X).$	0.3	:	$q(a).$
0.6	:	$s(X) \leftarrow r(X).$	0.7	:	$q(b).$
0.3	:	$p(a).$	0.2	:	$r(a).$
0.7	:	$p(b).$	0.8	:	$r(b).$

If we follow the method presented above, we obtain the BLP:

```
s(X) | p(X), q(X).              q(a).
s(X) | r(X).                    q(b).
p(a).                           r(a).
p(b).                           r(b).
```

		s(X)	
p(X)	q(X)	true	false
true	true	0.4	0.6
true	false	0.0	1.0
false	true	0.0	1.0
false	false	0.0	1.0

	s(X)	
r(X)	true	false
true	0.6	0.4
false	0.0	1.0

p(a)	
true	false
0.3	0.7

The CPTs for p(b), q(a), q(b), r(a) and r(b) are not detailed.

Now, if we query this BLP with "$? - \;s(a)$.", the resulting Bayesian network will contain 4 chance nodes: $p(a)$, $q(a)$ and $r(a)$ directly influence $s(a)$. The combining rule gives the combined conditional probability table as follows.

			s(a)	
p(a)	q(a)	r(a)	true	false
true	true	true	1.0	0.0
true	true	false	0.4	0.6
true	false	true	0.6	0.4
true	false	false	0.0	1.0
false	true	true	0.6	0.4
false	true	false	0.0	1.0
false	false	true	0.6	0.4
false	false	false	0.0	1.0

By using any standard inference algorithm in this BN (e.g. Pearl's message passing), we obtain the probability $P(s(a) = true) \;=\; 0.156$, which is the result we were looking for (since the distributional semantics gives: $P(s(a) \;/\; P) \;=\; (0.4 \times 0.3 \times 0.3 + 0.6 \times 0.2) \;=\; 0.156$).

Remark: The latter example is different from Cussens' example, in that we avoid the double-refutation of $p(X)$ in the first stochastic clause. This is to prevent the query $s(a)$ from not being safe with regards to the translation of the SLP. It is necessary that the query be safe with regards to the translation of the SLP, according to the next theorem:

Theorem: Given a complete, range-restricted and non-recursive SLP S, its translation into a BLP B obtained as defined above , and a ground query G. Let us associate to G the Bayesian atom G_a, whose domain is $\{true, false\}$, and which is itself associated to a chance node in the Bayesian net BN_{B,G_a}. If G_a is safe with regards to B then: $P(G/S) = P_{B,G_a}(G_a = true)$.

5 Conclusion and Further Work

We have demonstrated relations between SLPs' and BLPs' semantics, and we have shown that SLPs augmented with combining functions (namely *extended SLPs*) and BLPs can encode the same knowledge, in that they encode equivalent distributions of probabilities with regards to the latter relations. Since SLPs need to be augmented with combining rules in order to be as expressive as BLPs, and BLPs are able to encode complete, range-restricted and non-recursive SLPs, we are tempted to conclude that BLPs are more expressive than strict SLPs.

However, SLPs' and BLPs' formalisms are more or less intuitive, depending on the kind of knowledge we want to model. It should be noted that BLPs' query-answering procedure is cumbersome because of KBMC and the necessity of using different frameworks (computational logic, Bayesian nets), while inference mechanisms in SLPs are straightforward.

We believe this paper to be a formal basis for further studies. In the perspective of inductive learning, inter-translations of e-SLPs and BLPs can be used to extend learning techniques designed for BLPs to the learning of e-SLPs (and vice-versa). We think it could be interesting to investigate the interests of such extensions.

So far e-SLPs have been introduced in a fairly general way; the definition of suitable constraints on the combining functions in e-SLPs is also a precondition for their *learnability*.

Acknowledgments

The first author was supported by the Engineering and Physical Sciences Research Council (EPSRC). The second author acknowledges support from the ESPRIT IST project *Application of Probabilistic Inductive Logic Programming* (APRIL, IST-2001-33053). We would like to thank Luc De Raedt and Kristian Kersting for helpful discussions on this topic. We would also like to thank anonymous reviewers for their comments and constructive criticism.

References

[Cussens, 1999a] J. Cussens. Integrating probabilistic and logical reasoning. In *Electronic Transaction on Artificial Intelligence*, 1999. Machine Intelligence Workshop (MI16), special issue, (submitted).

[Cussens, 1999b] James Cussens. Loglinear models for first-order probabilistic reasoning. In Kathryn Blackmond Laskey and Henri Prade, editors, *Proceedings of the 15th Annual Conference on Uncertainty in AI (UAI'99)*, pages 126–133. Morgan Kaufmann, 1999.

[Cussens, 2000] J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2000.

[Halpern, 1989] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1989.

[Kersting and Raedt, 2000] Kristian Kersting and Luc De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.

[Muggleton and de Raedt, 2001] S. Muggleton and L. de Raedt. *Application of Probabilistic Inductive Logic Programming, Final and Periodic Progress Report.* European Union IST programme Assessment, Project 33053, 2001.

[Muggleton, 2000] S. Muggleton. Learning stochastic logic programs. *Electronic Transactions in Artificial Intelligence*, 4(041), 2000.

[Muggleton, 2001] S. Muggleton. Stochastic logic programs. *Journal of Logic Programming*, 2001. Accepted subject to revision.

[Puech, 2003] A. Puech. Parameter estimation in extended slps. Technical report, Imperial College London, September 2003.