Semantics and derivation for Stochastic Logic Programs

Stephen Muggleton
Department of Computer Science,
University of York,
York, YO1 5DD,
United Kingdom.

Abstract

Stochastic Logic Programs (SLPs) are a generalisation of Hidden Markov Models (HMMs), stochastic context-free grammars, and undirected Bayes' nets. A pure stochastic logic program consists of a set of labelled clauses p:C where p is in the interval $[0, 1]$ and C is a first-order range-restricted definite clause. SLPs have applications both as mechanisms for efficiently sampling from a given distribution and as representations for probabilistic knowledge. This paper discusses a distributional semantics for SLPs. Additionally two derivational mechanisms are introduced and analysed. These are 1) stochastic SLD (Selection-function-Linear-resolution-for-Definite-clauses) refutation and 2) a program for enumerating the least Herbrand distributional model of an SLP in descending order. It is shown that as with logic programs unfolding transformations preserve the models of an SLP. Lastly, we discuss how Bayes' Theorem might be used to devise algorithms for learning SLPs.

1 Introduction

Representations of uncertain knowledge can be divided into a) procedural descriptions of sampling distributions (eg. stochastic grammars (Lari and Young, 1990) and Hidden Markov Models (HMMs)) and b) declarative representations of uncertain statements (eg. probabilistic logics (Fagin and Halpern, 1989) and Relational Bayes' nets (Jaeger, 1997)). Stochastic Logic Programs (SLPs) (Muggleton, 1996) were introduced originally as a way of lifting stochastic grammars (type a representations) to the level of first-order Logic Programs (LPs). Later Cussens (Cussens, 1999) showed that SLPs can be used to represent undirected Bayes' nets (type b representations). SLPs are presently used (Muggleton, 1999b) to define distributions for sampling within Inductive Logic Programming (ILP) (Muggleton, 1999a).

LP	SLP
Definite clause	Labelled definite clause
Interpretation	Distributional interpretation
Model	Distribution model
Closed world model	i) Normalised distribution ii) NF distribution
SLD refutation	Stochastic SLD refutation
Enumeration of success-set	Descending enumeration of distribution

Figure 1: Correspondence between LPs and SLPs

Previous papers describing SLPs have concentrated on their procedural (sampling) interpretation. This paper aims at providing semantics and proof techniques for SLPs analogous to those of LPs. Table 1 shows the correspondences established in this paper between LPs and SLPs.

The paper is organised as follows. Section 2 introduces standard definitions for LPs. Existing syntax and proof techniques for SLPs are given in Section 3 followed by the new distributional semantics. Incomplete SLPs are shown to have multiple consistent distributional models. In Section 4 two distinct "completion" techniques, Normalised and Negation-by-Failure (NF) are discussed which lead to unique distributional models for SLPs. In Section 5 an algorithm for stochastic sampling of SLD refutations is shown to be sound and complete with respect to Normalised models. A program is then given in Section 6 which allows the enumeration of ground atomic consequences of an SLP in descending probability order. The program can be adapted for either the Normalised or NF models of SLPs. In Section 7 it is shown that unfold transformations preserve the semantics of SLPs. Section 8 concludes and discusses further work including the prospects of algorithms for learning SLPs from data.

2 LPs

The following summarises the standard syntax, semantics and proof techniques for LPs (see (Lloyd, 1987)).

2.1 Syntax of LPs

A variable is denoted by an upper case letter followed by lower case letters and digits. Predicate and function symbols are denoted by a lower case letter followed by lower case letters and digits. A variable is a term, and a function symbol immediately followed by a bracketed n-tuple of terms is a term. In the case that n is zero the function symbol is a constant and is written without brackets. Thus $f(g(X), h)$ is a term when f, g and h are function symbols, X is a variable and h is a constant. A predicate symbol immediately followed by a bracketted n-tuple of terms is called an atomic formula, or atom. The negation symbol is: $\neg$. Both a and $\neg a$ are literals whenever a is an atom. In this case a is called a positive literal and $\neg a$ is called a negative literal. A clause is a finite set of literals, and is treated as a universally quantified disjunction of those literals. A finite set of clauses is called a clausal theory and is treated as a conjunction of those clauses. Literals, clauses, clausal theories, True and False are all well-formed-formulas (wffs). A wff or a term is said to be ground whenever it contains no variables. A Horn clause is a clause containing at most one positive literal. A definite clause is a clause containing exactly one positive literal and is written as $h \leftarrow b_1, .., b_n$ where h is the positive literal, or *head* and the b_i are negative literals, which together constitute the *body* of the clause. A definite clause for which all the variables in the head appear at least once in the body is called range-restricted. A non-definite Horn clause is called a goal and is written $\leftarrow b_1, .., b_n$. A Horn theory is a clausal theory containing only Horn clauses. A definite program is a clausal theory containing only definite clauses. A range-restricted definite program is a definite program in which all clauses are range-restricted.

2.2 Semantics of LPs

Let $\theta = \{v_1/t_1, .., v_n/t_n\}$. θ is said to be a substitution when each v_i is a variable and each t_i is a term, and for no distinct i and j is v_i the same as v_j. Greek lowercase letters are used to denote substitutions. θ is said to be ground when all t_i are ground. Let E be a wff or a term and $\theta = \{v_1/t_1, .., v_n/t_n\}$ be a substitution. The instantiation of E by θ, written $E\theta$, is formed by replacing every occurrence of v_i in E by t_i. $E\theta$ is an instance of E.

A first-order language L is a set of wffs which can be formed from a fixed and finite set of predicate symbols, function symbols and variables. A set of ground literals I is called an L-interpretation (or simply interpretation) in the case that it contains either a or $\neg a$ for each ground atom a in L. Let M be an interpretation and $C = h \leftarrow B$ be a definite clause in L. M is said to be an L-model (or simply model) of C iff for every ground instance $h' \leftarrow B'$ of C in L $B' \subseteq M$ implies $h' \in M$. M is a model of Horn theory P whenever M is a model of each clause in P. P is said to be satisfiable if it has at least one model and unsatisfiable otherwise. Suppose L is chosen to be the smallest first-order language involving at least one constant and the predicate and function symbols of Horn theory P. In this case an interpretation is called a Herbrand interpretation of P and the ground atomic subset of L is called the Herbrand Base of P. I is called a Herbrand model of Horn theory P when I is both Herbrand and a model of P. According to Herbrand's theorem P is satisfiable iff it has a Herbrand model. Let F and G be two wffs. We say that F entails G, or $F \models G$, iff every model of F is a model of G.

2.3 Proof for LPs

An inference rule $I = F \rightarrow G$ states that wff F can be rewritten by wff G. We say $F \vdash_I G$ iff there exists a series of applications of I which transform F to G. I is said to be *sound* iff for each $F \vdash_I G$ always implies $F \models G$ and *complete* when $F \models G$ always implies $F \vdash_I G$. I is said to be *refutation complete* if I is complete with G restricted to False. The substitution θ is said to be the unifier of the atoms a and a' whenever $a\theta = a'\theta$. μ is the most general unifier (mgu) of a and a' if and only if for all unifiers γ of a and a' there exists a substitution δ such that $(a\mu)\delta = a\gamma$. The resolution inference rule is as follows. $((C \setminus \{a\}) \cup (D \setminus \{\neg a'\}))\theta$ is said to be the resolvent of the clauses C and D whenever C and D have no common variables, $a \in C$, $\neg a' \in D$ and θ is the mgu of a and a'. Suppose P is a definite program and G is a goal. Resolution is linear when D is restricted to clauses in P and C is either G or the resolvent of another linear resolution. The resolvent of such a linear resolution is another goal. Assuming the literals in clauses are ordered, a linear resolution is SLD when the literal chosen to resolve on is the first in C. An SLD refutation from P is a sequence of such SLD linear resolutions, which can be represented by $D_{P,G} = \langle G, C_1, .., C_n \rangle$ where each C_i is in P and the last resolvent is the empty clause (ie. False). The answer substitution is $\theta_{P,G} = \theta_1\theta_2..\theta_n$ where each θ_i is the substitution corresponding with the resolution involving C_i in $D_{P,G}$. If P is range-restricted then $\theta_{P,G}$ will be ground. SLD resolution is known to be both sound and refutation complete for definite programs. Thus for a range-restricted definite program P and ground atom a it can be shown that $P \models a$ by showing that $P, \leftarrow a \vdash_{SLD}$ False. The Negation-by-Failure (NF) inference rule says that $P, \leftarrow a \nvdash_{SLD}$ False implies $P \vdash_{SLDNF} \neg a$.

3 SLPs

3.1 Syntax of SLPs

An SLP S is a set of labelled clauses $p{:}C$ where p is a probability (ie. a number in the range $[0,1]$) and C is a first-order range-restricted definite clause[1]. The subset S_p of clauses in S with predicate symbol p in the head is called the definition of p. For each definition S_p the sum of probability labels π_p must be at most 1. S is said to be complete if $\pi_p = 1$ for each p and incomplete otherwise. $P(S)$ represents the definite program consisting of all the clauses in S, with labels removed.

Example 1 Unbiased coin. *The following SLP is complete and represents a coin which comes up either heads or tails with probability 0.5.*

$$S_1 = \left\{ \begin{array}{l} 0.5 : coin(head) \leftarrow \\ 0.5 : coin(tail) \leftarrow \end{array} \right\}$$

S_1 is a simple example of a sampling distribution[2].

Example 2 Pet example. *The following SLP is incomplete.*

$$S_2 = \left\{ \begin{array}{ll} 0.3 : likes(X,Y) \leftarrow & pet(Y,X), pet(Z,X), \\ & cat(Y), mouse(Z) \end{array} \right\}$$

S_2 shows how statements of the form $\Pr(P(\vec{x})|Q(\vec{y})) = p$ can be encoded within an SLP, in this case $\Pr(\text{likes(X,Y)}|\ldots)) = 0.3$.

3.2 Proof for SLPs

A Stochastic SLD (SSLD) refutation is a sequence $D_{S,G} = \langle 1{:}G, p_1{:}C_1, .., p_n{:}C_n \rangle$ in which G is a goal, each $p_i{:}C_i \in S$ and $D_{P(S),G} = \langle G, C_1, .., C_n \rangle$ is an SLD refutation from $P(S)$. SSLD refutation represents the repeated application of the SSLD inference rule. This takes a goal $p{:}G$ and a labelled clause $q{:}C$ and produces the labelled goal $pq{:}R$, where R is the SLD resolvent of G and C. The answer probability of $D_{S,G}$ is $Q(D_{S,G}) = \prod_{i=1}^{n} p_i$. The incomplete probability of any ground atom a with respect to S is $Q(a|S) = \sum_{D_{S,(\leftarrow a)}} Q(D_{S,(\leftarrow a)})$. We can state this as $S \vdash_{SSLD} Q(a|S) \leq Pr(a|S) \leq 1$, where $Pr(a|S)$ represents the conditional probability of a given S.

Remark 3 Incomplete probabilities. *If a is a ground atom with predicate symbol p and the definition S_p in SLP S is incomplete then $Q(a|S) \leq \pi_p$.*
Proof. *Suppose the probability labels on clauses in S_p are $p_1, .., p_n$ then $Q(a|S) = p_1 q_1 + .. + p_n q_n$ where each q_i is a sum of products for which $0 \leq q_i \leq 1$. Thus $Q(a|S) \leq p_1 + .. + p_n = \pi_p$.*

3.3 Semantics of SLPs

In this section we introduce the "normal" semantics of SLPs. Suppose L is a first-order language and D_p is a probability distribution over the ground atoms of p in L. If I is a vector consisting of one such D_p for every p in L then I is called a distributional L-interpretation (or simply interpretation). If $a \in L$ is an atom with predicate symbol p and I is an interpretation then $I(a)$ is the probability of a according to D_p in I. Suppose L is chosen to be the smallest first-order language involving at least one constant and the predicate and function symbols of Horn theory $P(S)$. In this case an interpretation is called a distributional Herbrand interpretation of S (or simply Herbrand interpretation).

Definition 4 *An interpretation M is a distributional L-model (or simply model) of SLP S iff $Q(a|S) \leq M(a)$ for each ground atom a in L*[3].

Again if M is a model of S and M is Herbrand with respect to S then M is a distributional Herbrand model of S (or simply Herbrand model).

Example 5 Models.

$$S = \left\{ \begin{array}{l} 0.5{:}p(X) \leftarrow q(X) \\ 0.5{:}q(a) \leftarrow \end{array} \right\}$$

$Q(p(a)|S) = 0.25$ and $Q(q(a)|S) = 0.5$. L has predicate symbols p, q and constant a, b.

$$I_1 = \left\langle \begin{array}{l} \{1{:}p(a), 0{:}p(b)\} \\ \{1{:}q(a), 0{:}q(b) \end{array} \right\rangle$$

I_1 is a model of S.

$$I_2 = \left\langle \begin{array}{l} \{0.1{:}p(a), 0.9{:}p(b)\} \\ \{0.5{:}q(a), 0.5{:}q(b) \end{array} \right\rangle$$

I_2 is not a model of S.

Suppose S, T are SLPs. As usual we write $S \models T$ iff every model of S is a model of T.

[1] Cussens (Cussens, 1999) considers a less restricted definition of SLPs.

[2] A more complex sampling distribution example might define a probability distribution over a language by attaching probability labels to productions of a grammar. The grammar could be encoded as a range-restricted definite program.

[3] It might seem unreasonable to define semantics in terms of proofs in this way. However, it should be noted that $Q(a|S)$ represents a potentially infinite summation of the probabilities of individual SSLD derivations. This is analogous to defining the satisfiability of a first-order formula in terms of an infinite boolean expression derived from truth tables of the connectives

4 Completion of SLPs

If we intend to use an incomplete SLP S either for sampling or for a predictive task then, as with LPs, it is necessary to apply some form of completion to S in order to identify a unique model. In the case of LPs there is a unique least Herbrand model. The situation is somewhat different with SLPs since there is no analogue of the model intersection property (Proposition 6.1 in (Lloyd, 1987)). We suggest two alternative completion strategies, Normalised and NF completion. These would appear to have advantages to sampling and uncertain knowledge representations respectively.

4.1 Normalised completion

Suppose S is an incomplete SLP, B_S is the Herbrand Base of $P(S)$ and B_p is the subset of B_S with predicate symbol p. For each predicate symbol p in S let $Q_p = \sum_{a \in B_p} Q(a|S)$. Now for any ground atom a in B_S with predicate symbol p the normalised Herbrand interpretation of S is defined as follows.

$$S_{\text{Norm}}(a) = \frac{Q(a|S)}{Q_p}$$

Theorem 6 *S_{Norm} is a Herbrand model of S.*
Proof. *According to Definition 4 $S_{Norm}(a)$ is a Herbrand model of S iff $S_{Norm}(a)$ is both a Herbrand interpretation and $Q(a|S) \leq S_{Norm}(a)$ for each ground atom a in B_S. $S_{Norm}(a)$ is a Herbrand interpretation since Q_p acts as a normalising constant for each subset B_p of B_S. Furthermore $Q(a|S) \leq S_{Norm}(a) = \frac{Q(a|S)}{Q_p}$ since $Q_p \leq 1$. This completes the proof.*

4.2 NF completion

Negation-by-Failure (NF) completion is analogous to the use of NF in LPs. In this case we alter the semantics of SLPs introduced in Section 3.3 in two ways. Firstly we drop the requirement that each D_p in an interpretation I should be a **probability** distribution. Secondly we introduce the requirement that if a is a ground atom then $I(\neg a) = 1 - I(a)$. Otherwise the definitions in Section 3.3 remain as they are. This gives us what we will call the NF semantics of SLPs.

Now for any ground literal l whose atom a is in B_S the NF Herbrand interpretation of S is defined as follows.

$$S_{\text{NF}}(l) = \begin{cases} Q(a|S) & \text{if l=a} \\ 1 - Q(a|S) & \text{otherwise} \end{cases}$$

Example 7

$$S = \left\{ \begin{array}{l} 0.2 : fish(X) \leftarrow legs(X, 0) \\ 0.1 : reptile(X) \leftarrow legs(X, 0) \\ 0.9 : legs(eel, 0) \leftarrow \end{array} \right\}$$

a	$S_{NF}(a)$	$S_{NF}(\neg a)$
legs(eel,0)	*0.9*	*0.1*
fish(eel)	*0.18*	*0.82*
reptile(eel)	*0.09*	*0.91*

Figure 2: Probabilities assigned to ground atoms in Example 7

The table in Figure 7 gives the probability assigned to various literals by S_{NF}.

5 Stochastic SLD algorithm

The following algorithm either randomly constructs and returns an SLD refutation or fails.

Algorithm 8 Algorithm for sampling SSLD refutations. *Given incomplete SLP S and goal $G = \leftarrow a$ where a is an atom.*

1. *Partial-refutation := $\langle 1{:}G \rangle$.*
2. *Let p be the predicate symbol of a and S_p the definition of p in S.*
3. *Let $r{:}R$ be the last SSLD-resolvent of Partial-refutation.*
4. *If R is the empty clause then return Partial-refutation.*
5. *With probability $1 - \pi_p$ Fail.*
6. *Randomly choose a clause $q{:}C$ from S_p with probabilities proportional to the distribution of labels in S_p.*
7. *If R and C have an SLD resolvent then append $q{:}C$ to Partial-refutation and Goto 3.*
8. *Fail.*

We now show some properties concerning the correctness of Algorithm 8.

Theorem 9 *Algorithm 8 returns the SSLD refutation $D_{S,G} = \langle 1{:}G,\ p_1{:}C_1,\ ..,p_n{:}C_n \rangle$ with probability $Q(D_{S,G}) = \prod_{i=1}^{n} p_i$.*
Proof. *By induction on n. In the base case $n = 1$ and $D_{S,G} = \langle 1{:}G, p_1{:}C_1 \rangle$. The algorithm would return this by traversing steps 1-7 and then step 3 terminating on 4. Since the transition between steps 5 and 6 would be traversed with probability π_p and $p_1{:}C_1$ would be chosen with probability $\frac{p_1}{\pi_p}$ the overall probability of returning $D_{S,G}$ would be p_1, which proves the base case. The hypothesis is that the theorem is true for $n \leq k$. Now we prove that the theorem is consequently true for $n = k + 1$. Thus we assume that we have correctly iterated steps 3-7 k times without exiting. Thus Partial-refutation is now $\langle 1{:}G,\ p_1{:}C_1,$*

.., p_k:$C_k\rangle$ when we Goto step 3. Once more steps 5,6 would be traversed with probability π_p and p_{k+1}:C_{k+1} would be chosen with probability $\frac{p_{k+1}}{\pi_p}$. Thus the overall probability of returning $D_{S,G} = \langle 1$:G, p_1:C_1, .., p_{k+1}:$C_{k+1}\rangle$ would be $p_{k+1}\prod_{i=1}^{k} p_i$. This proves the step and completes the proof.

Corollary 10 *If a is a most general atom $p(v_1, .., v_m)$ in which the v_i are all distinct then Algorithm 8 Fails with probability $1 - Q_p$.*
Proof. *Follows trivially from Theorem 9 and the definition of Q_p.*

Note that if the failure probability $1 - Q_p$ of an SLP approaches 1 Algorithm 8 will rarely succeed.

6 Descending enumeration program

Algorithm 8 is unlike normal SLD-refutation-based Prolog interpreters in that it does not backtrack, and does not give a systematic enumeration of proofs for the given goal. The program in this section is more like a standard Prolog interpreter in these ways. The program returns ground atoms from the SLP S in descending order of their probability according to either the Normalised or NF interpretation of S (note that the order imposed by probabilities in these two interpretations is identical). In many applications, such as diagnosis, it is expected that such a descending enumeration would be informative to the user.

The task of the program might at first seem trivial. However, any ground atom a may have an arbitrarily large (even infinite) number of associated refutations. In order to output a in the series we need to be able to decide, after having considered only a finite number of its refutations, whether a has the highest probability of all remaining atoms. Thus suppose R is a finite subset of the refutations of the general goal $G =\leftarrow g$ on SLP S. Let P_R be the sum of the probabilities of all refutations in R and $Q_R = 1 - P_R$. Suppose θ_a, θ_b are among the ground answer substitutions in R where $a = g\theta_a$ and $b = g\theta_b$. Let p_a, p_b be the summed probabilities of the refutations in R with answer substitutions θ_a, θ_b respectively. Clearly there exist probabilities q_a, q_b such that $p_a + q_a = Q(a|S)$ and $p_b + q_b = Q(b|S)$.

Remark 11 *$p_a \geq p_b + Q_R$ implies $Q(a|S) \geq Q(b|S)$.*
Proof. *$p_a \geq p_b + Q_R$ implies $p_a + q_a \geq p_b + Q_R$ since $q_a \geq 0$. $p_a + q_a \geq p_b + Q_R$ implies $p_a + q_a \geq p_b + q_b$ since $Q_R \geq q_b$. Thus $Q(a|S) \geq Q(b|S)$ since $p_a + q_a = Q(a|S)$ and $p_b + q_b = Q(b|S)$. This completes the proof.*

In the following descending enumeration program the depth d of an SSLD refutation is simply its sequence length minus 1, R_d is the set of all SSLD refutations of the goal G of depth at most d and $Q_d(a|S)$ is the summed probability of refutations in R_d with answer substitution θ_a, where $a = g\theta_a$.

Program 12 Descending enumeration of distribution.

Given SLP S and goal $G =\leftarrow g$
$d := 0$ and SeenAtoms := $\emptyset$
Forever
  Let AtomQueue be the list of all $\langle c, Q_d(c|S)\rangle$
    pairs sorted in descending order of $Q_d(c|S)$
    where c is not in SeenAtoms.
  Let $P_{R,d}$ be the sum of probabilities of all
    refutations in R_d.
  Let $Q_{R,d} := 1 - P_{R,d}$.
  While not empty AtomQueue
    $\langle a, p_a\rangle$:= Pop(AtomQueue)
    If (empty AtomQueue and $p_a \geq Q_{R,d}$) or
      ($\langle b, p_b\rangle$:=Head(AtomQueue) and
      $p_a \geq p_b + Q_{R,d}$) then
        Print a
        SeenAtoms := SeenAtoms $\cup\{a\}$
    Else Break
  EndWhile
  $d := d + 1$
Loop

The following theorem concerns the correctness of Program 12.

Theorem 13 *Given any SLP S and goal $G =\leftarrow g$ if Algorithm 12 prints atom c after having printed atom a then $Q(a|S) \geq Q(c|S)$.*
Proof. *Assume false. Thus assume Algorithm 12 prints atom c after having printed atom a and $Q(c|S) > Q(a|S)$. Now consider the point in the program at which a is printed. To have reached this point it must be the case that either i) $\langle a, p_a\rangle$ was alone in the queue and $p_a \geq Q_{R,d}$ or ii) the second element of the queue was $\langle b, p_b\rangle$ and $p_a \geq p_b + Q_{R,d}$. But if i) is the case then either c is in SeenAtoms which contradicts the assumption or $Q_d(c|S) = p_c = 0$ in which case according to Remark 11 $Q(a|S) \geq Q(c|S)$, which again contradicts the assumption. Thus ii) must be the case. However in this case again either c is in SeenAtoms, which has already been discounted, or $c = b$, or or c is later in the queue than b, or $Q_d(c|S) = p_c = 0$. The last three cases contradict Remark 11. This contradicts the assumption and completes the proof.*

The author has no proof for the "only if" counterpart of this theorem.

7 Unfold transformations

In this section we discuss how unfold program transformations can be applied to SLPs.

Definition 14 Unfold. *Let S be an SLP and p:$C \in S$ be a labelled clause. Let $U_{p:C,S} = \{pq$:$R|q$:$D \in$*

S, R resolvent of $C, D\}$. Now $S' = S \setminus \{p{:}C\} \cup U_{p:C,S}$ is called the unfold of S on $p{:}C$.

As with logic programs unfold tranformations preserve the semantics of SLPs.

Theorem 15 *Let S be an SLP and $p{:}C \in S$ be a labelled clause. Every model of SLP S is a model of S', the unfold of S on $p{:}C$.*

Proof. *It is sufficient to note that for every SSLD refutation $D_{S,G} = \langle 1{:}G, p_1{:}C_1, p_i{:}C, p_{i+1}{:}D, .., p_n{:}C_n\rangle$ from S there is a corresponding SSLD refutation $D_{S',G} = \langle 1{:}G, p_1{:}C_1, p_ip_{i+1}{:}R, .., p_n{:}C_n\rangle$ from S' with the same answer probability. From this it follows that $Q(a|S) = Q(a|S')$ for all atoms a and thus S has the same models as S' according to Definition 4.*

The following is a simple example involving unfolding an SLP defining an exponential decay distribution over the natural numbers.

Example 16

$$S = \left\{ \begin{array}{l} 0.5 : nat(0) \leftarrow \\ 0.5 : nat(s(X)) \leftarrow nat(X) \end{array} \right\}$$

We now perform an unfold on the second labelled clause to give the following SLP.

$$S' = \left\{ \begin{array}{l} 0.5 : nat(0) \leftarrow \\ 0.25 : nat(s(0)) \leftarrow \\ 0.25 : nat(s(s(X))) \leftarrow nat(X) \end{array} \right\}$$

S' represents the same distribution over the natural numbers as S.

8 Conclusion

This paper is the first to provide a rigorous semantics for SLPs. SLPs have been used in Inductive Logic Programming to allow efficient sampling from the Herbrand Base. SLPs differ from Probabilistic Logic Programs (Ng and Subrahmanian, 1992) and other Probabilistic Logics (Hailperin, 1984; Fagin and Halpern, 1989) which base their semantics on distributions over models. However examples 2 and 7 show that SLPs can be used for representing uncertain first-order knowledge of the form $\Pr(P(\vec{x})|Q(\vec{y})) = p$. The author believes that the NF completion semantics in this paper is particularly approriate in this context.

SLPs are generalisation of the variant of logic programs given in (Sato, 1995). They have also been shown to be a generalisation of HMMs, stochastic context-free grammars and undirected Bayes' nets.

The author believes that the semantics and derivation algorithms for SLPs given in this paper will be useful in the context of learning SLPs from data. One approach that might be taken to this task involves finding the SLP S which has maximum posterior probability given the examples E. This approach is familiar from ILP. The posterior probability can be expressed using Bayes theorem as follows.

$$p(S|E) = \frac{p(S)p(E|S)}{p(E)}$$

$p(S)$ is based on a prior over SLPs, $p(E|S) = p(e_1|S)..p(e_m|S)$ if the e_i are chosen randomly and independently and $p(E)$ is a normalising constant. Note that $p(e_i|S) = S_{NF}(e_i)$ in the case of the NF completion semantics in this paper.

In conclusion, SLPs combine high expressivity with efficient reasoning. We hope that in time SLPs will come to be viewed as an important representation not only for sampling distributions but also for uncertain reasoning in general.

Acknowledgements

The author would like to thank Wray Buntine, David Page, Koichi Furukawa and James Cussens for discussions on the topic of Stochastic Logic Programming. Many thanks are due to my wife, Thirza and daughter Clare for the support and happiness they give me. This work was supported partly by the Esprit RTD project "ALADIN' (project 28623), EPSRC grant "Closed Loop Machine Learning", BBSRC/EPSRC grant "Protein structure prediction - development and benchmarking of machine learning algorithms" and EPSRC ROPA grant "Machine Learning of Natural Language in a Computational Logic Framework".

References

Cussens, J. (1999). Loglinear models for first-order probabilistic reasoning. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, pages 126–133, San Francisco. Kaufmann.

Fagin, R. and Halpern, J. (1989). Uncertainty, belief and probability. In *Proceedings of IJCAI-89*, San Mateo, CA. Morgan Kauffman.

Hailperin, T. (1984). Probability logic. *Notre Dame Journal of Formal Logic*, (25):198–212.

Jaeger, M. (1997). Relational bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA. Kaufmann.

Lari, K. and Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

Lloyd, J. (1987). *Foundations of Logic Programming*. Springer-Verlag, Berlin. Second edition.

Muggleton, S. (1996). Stochastic logic programs. In de Raedt, L., editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press.

Muggleton, S. (1999a). Inductive logic programming: issues, results and the LLL challenge. *Artificial Intelligence*, 114(1–2):283–296.

Muggleton, S. (1999b). Learning from positive data. *Machine Learning*. Accepted subject to revision.

Ng, R. and Subrahmanian, V. (1992). Probabilistic logic programming. *Information and Computation*, 101(2):150–201.

Sato, T. (1995). A statistical learning method for logic programs with distributional semantics. In Sterling, L., editor, *Proceedings of the Twelth International conference on logic programming*, pages 715–729, Cambridge, Massachusetts. MIT Press.