A Behavioral Comparison of Some Probabilistic Logic Models

Stephen Muggleton and Jianzhong Chen

Department of Computing, Imperial College London, London SW7 2AZ, UK
{shm,cjz}@doc.ic.ac.uk

Abstract. *Probabilistic Logic Models* (PLMs) are efficient frameworks that combine the expressive power of first-order logic as knowledge representation and the capability to model uncertainty with probabilities. *Stochastic Logic Programs* (SLPs) and *Statistical Relational Models* (SRMs), which are considered as *domain frequency* approaches, and on the other hand *Bayesian Logic Programs* (BLPs) and *Probabilistic Relational Models* (PRMs) (*possible worlds* approaches), are promising PLMs in the categories. This paper is aimed at comparing the relative expressive power of these frameworks and developing translations between them based on a behavioral comparison of their semantics and probability computation. We identify that SLPs augmented with combining functions (namely *extended SLPs*) and BLPs can encode equivalent probability distributions, and we show how BLPs can define the same semantics as complete, range-restricted SLPs. We further demonstrate that BLPs (resp. SLPs) can encode the relational semantics of PRMs (resp. SRMs). Whenever applicable, we provide inter-translation algorithms, present their soundness and give worked examples.

1 Introduction

Probabilistic Logic Models (PLMs) combine expressive knowledge representation formalisms such as relational and first-order logic with principled probabilistic and statistical approaches to inference and learning. This combination is needed in order to face the challenge of real-world learning and data mining problems in which data are complex and heterogeneous and we are interested in finding useful predictive and/or descriptive patterns.

Probabilistic logic representations and PLMs have varying levels of expressivity. As yet more effort has been put into defining new variants of PLMs than into characterising their relationships. Studying the relative expressive power of various probabilistic logic representations is a challenging and interesting task. On the one hand, there exist some theoretical study of the relations of two probabilistic approaches, i.e., possible-worlds and domain-frequency. On the other hand, it is interesting to see how both logic programming-based approaches and relational model/database-based methods have converged and to analyze relations between them. Essentially the following questions at different levels can be asked:

L. De Raedt et al. (Eds.): Probabilistic ILP 2007, LNAI 4911, pp. 305–324, 2008.

- From a semantical perspective: can we compare the semantics between PLMs in terms of the definition of probability distributions?
- From a theoretical perspective: given an expert domain, can we encode the same knowledge in different PLMs?
- More practically: can we find inter-translations between these PLMs?

This study is based on a *behavioural* approach: we analyze what the respective interests of these formulations are. In particular, how can we semantically compare PLMs and if there exist inter-translations, do the translations have the same features (computation of the probabilities, inference with or without evidence)?

The chapter is organised as follows. In section 2, we shortly introduce logical/relational models and probabilistic models as well as their extension to first-order probabilistic models. In section 3, we present a introduction to the four PLMs we choose for the study. In section 4, we explain the behavioural comparison approach and compare the semanics of the four PLMs based on the catigories of first-order probabilistic models. In the next three sections, we detail probabilistic knowledge encoding and inter-translations between the PLMs of interest. We deal with the concluding remarks and direction of future work in the last section.

2 Preliminaries

2.1 Logical/Relational Models

Knowledge encoded in PLMs are mainly based on two representation languages and their associated system of inference, namely the *Definite Clause Logic* used in logic programming model and the *Relational Algebra* used in the relational model.

Logic Programming Model. *Definite Clause Logic* (DCL), also known as *Horn clause logic*, is the subset of first-order logic whose well formed formulas are universally quantified conjunctions of disjunctions of literals. Each disjunct is called a clause. DCL further requires that clauses are definite, which means that they have exactly one positive literal each (the *head* of the clause). The list of negative literals (if any) is called the body of the clause. A list of definite clauses is called a logic program (LP). We assume our terminology discussed in the paper are within DCL.

The semantics, ie. the knowledge that can be encoded by a logic program L, can be defined with the least Herbrand model of L (noted $LH(L)$). There exists several inference algorithms to compute $LH(L)$ such as the SLD-resolution proof procedure. The success set of L, ie. the set of statements that can be refuted using SLD-resolution is exactly $LH(L)$. It is interesting to note that the use of functors and recursive clauses allows DCL programs to encode potentially infinite semantics. However, the functor-free subset of DCL (called Datalog) has a finite semantics since Datalog programs have finite least Herbrand models.

The use of DCL or Horn clause logic is supported by the wide availability and applicability of the programming language Prolog [1] and most Inductive Logic Programming systems, such as Progol [17].

Relational Model. Knowledge is often stored in relational databases which are formally defined by the so called *relational model* for data. We use the notations of [4] and [6] to introduce the relational model setting. This setting uses the abstract notion of *relational schema* to structure the data encoded in a database. Formally, a *relational schema* is a set of *classes* (also called *relations* or *tables*) $\mathcal{R} = \{R_i\}$. Each class R is associated to a *primary key* $R.K$, a set of *foreign keys* $\mathcal{F}(R) = \{F_k\}$ and a set of *descriptive attributes* $\mathcal{A}(R) = \{A_j\}$. Primary keys are unique identifiers for class instances. Descriptive attributes A_j take their respective values in the finite domains $\mathcal{V}(A_j)$. Each foreign key $F_k \in \mathcal{F}(R)$ establishes a directed relationship from its source class $Dom[F_k] = R$ onto a target class $Range[F_k]$. An instance of such a relational schema is a set of *objects* (also called *entities*, *records* or *tuples*). Instances of relational schemas are also called *databases*. Each object x is an instance of a class R: for each descriptive attribute $A_j \in \mathcal{A}(R)$, x is associated to an attribute value $x.A_j = a_j \in \mathcal{V}(A_j)$. Furthermore, class level foreign keys define binary relationships between objects: for each foreign key $F_k \in \mathcal{F}(R)$, x is associated an object $y = x.F_k$ such that y is an instance of the class $Y = Range[\rho_k]$. Databases that respect the constraints implied by foreign keys are said to respect the *referential integrity*.

Relational Algebra (RA) is a procedural language to perform queries on relational databases. RA is a procedural equivalent to declarative calculi such as the *tuple calculus* and the *domain calculus* that provides mathematical foundation for relational databases. RA is built upon six fundamental operations on sets of tuples: the selection, the projection, the Cartesian product, the set union, the set difference and the renaming of attribute names. These operations can be considered building blocks to define higher level operations such as joins, set intersections and divisions. From a more practical perspective, RA corresponds to SQL (*Structured Query Language*) without the aggregate and group operations. RA's expressive power is equivalent to non-recursive Datalog's (which is not Turing complete).

2.2 Probabilistic Models

Uncertainty is commonly modelled by defining a probability distribution over the domain $Dom(V) = \{v_i\}$ of a *random variable* V. We assume all random variables are discrete in the paper. We note $P(V = v_i) = p_i$. A set $\{v_i, p_i\}$ defines a probability distribution if and only if the $\{p_i\}$ are *normalized* ($\sum_i p_i = 1$).

Bayesian networks (BNs) [19,10] are introduced to make additional independency assumptions between random variables and to calculate the probabilities of the conjunctions. A BN is a Directed Acyclic Graph (DAG) where each vertex/node represents a random variable. Independency assumptions are encoded in the edges between chance nodes. Each variable is independent of its non-descendants given its parents. The parameters of the Bayesian networks

are embedded in Conditional Probability Tables (CPTs) which specify, for each variable V_i, the probability that $V_i = v$ ($v \in Dom(V_i)$) given the values of the parents $Pa(V_i)$. Given these and based on the Bayes theorem, a joint probability distribution of a set of n random variables can be defined using the following chain rule formula:

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i | Pa(V_i))$$

There exists a variety of algorithms to efficiently compute this probability distribution, such as Pearl's message passing, Variable Elimination, graph based Junction Tree, etc.

2.3 First-Order Probabilistic Models

First-Order Probabilistic Reasoning. *First-order logic* alone is not suitable to handle uncertainty, while this is often required to model non-deterministic domains with noisy or incomplete data. On the other hand, *propositional* probabilistic models (such as BNs) can't express relations between probabilistic variables; such frameworks suffer from a lack of structure over the domain, and the knowledge they can encode is fairly limited.

First-order probabilistic models are models which integrate logics and probabilities; they overcome the limits of traditional models by taking the advantages of both logics and probabilities.

Categories of First-Order Probabilistic Models. A well-known categorization of first-order probabilistic models was introduced by Halpern [7], in which two types of first-order probabilistic logic are categorized, ie. probabilities on the domain (or *type 1* probability structure) and probabilities on possible worlds (or *type 2* probability structure).

Type 1 probability structure can represent statements like "The probability that a randomly chosen bird will fly is greater than .9". It provides a type of *domain-frequency* approaches, which semantically illustrates objective and 'sampling' probabilities of domains. Precisely, a type 1 probability structure is a tuple (D, π, μ), where D is a domain, π maps predicate and function symbols in alphabet to predicates and functions of the right arity over D, and μ is a discrete probability function on D. The probability here is taken over the domain D. In the logic programming setting, it is reasonable to consider the Herbrand base over a given signature to have the same function as the domain.

On the other hand, type 2 probability structure may represent statements like "The probability that Tweety (a particular bird) flies is greater than .9". It is a kind of *possible-world* approaches and illustrates the subjective and 'degree-of-belief' semantics of the probabilities of domains. Formally, a type 2 probability structure is a tuple (D, W, π, μ), where D is a domain, W is a set of states or possible worlds, for each state $w \in W$, $\pi(w)$ maps predicate and function symbols in alphabet to predicates and functions of the right arity over D, and μ

is a discrete probability function on W. The probability here is taken over W, the set of states (or possible worlds or logic models). BNs and related models are type 2 approaches.

One of the differences between the two models is that there seems to assume only one possible world in type 1 case, saying that "any bird has a probability of flying greater than 0.9" is like giving the result of some statistical analysis (by counting domain frequency) in the real world.

3 Presentation of PLMs

We choose four promising PLMs to do the comparison.

3.1 Stochastic Logic Programs

Stochastic Logic Programs (SLPs) were first introduced in [14] as a generalization of stochastic grammars.

Syntax. An SLP consists of a set of labelled clauses $p : C$, where p is from the interval $[0, 1]$, and C is a range-restricted[1] definite clause. Later in this report, the labelled clauses $p : C$ will be named *parameterized* clauses or *stochastic* clauses. This original 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 ([2] 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. *Pure* SLPs are introduced in [2], 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. An SLP S 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 S. The probabilities are assigned to atoms according to an SLD-resolution strategy that employs a stochastic selection rule[2].

Three different related distributions are defined in [2], over derivations, refutations and atoms. Given an SLP S with n parameterized clauses and a goal G, it is easy to define a *log-linear probability distribution over the set of derivations*

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

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

[2] 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 [15]).

where x is a derivation of 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 C_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 C_i is used in the derivation x. If we assign the probability 0 to all derivations that are not refutations of the goal G, and 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

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

The computed answer in the SLD-tree is the most general instance of the goal G that is refuted by r, which is also named the *yield atom*. Let $X(y)$ be 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*

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

Given an SLP S, a query G and a (possibly partial) instantiation of G noted G_a, if λ is the vector of log-parameters associated to S and Y the set of yield atoms appearing in the refutations of G_a, we define $\mathbf{P_S^{SLP}}(G_a) = \sum_{y \in Y} p_{\lambda,G}(y)$.

3.2 Bayesian Logic Programs

Bayesian Logic Programs (BLPs) were first introduced in [13], as a generalization of Bayesian networks (BNs) and Logic Programs.

Syntax. A Bayesian logic program has two components – a *logical* one, which is a set of *Bayesian clauses*), and a *quantitative* one, which is 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 | is employed instead of $:-$; Bayesian atoms are assigned a (finite) *domain*, whereas first order logic atoms have binary values. Following the definitions in [13], we assume that atom domains in BLPs are discrete.

In order to represent a probabilistic model, each Bayesian clause c is associated with a conditional probability distribution $cpd(c)$ which encodes the probability that $head(c)$ takes some value, given the values of the Bayesian atoms in $body(c)$, ie. $P(head(c)|body(c))$. This conditional probability distribution is represented with a conditional probability table. As there can be many clauses with the same head (or non-ground heads that can be unified), *combining rules* are introduced 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 of BLPs to BNs is 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 together with the structure defined by the set of ground instances of the Bayesian clauses define a global (possibly infinite) dependency graph.

The semantics of BLPs can be discussed in a *well-defined* BLP. A range restricted BLP B is *well-defined* if:

1. Its least Herbrand model not empty: $LH(B) \neq \emptyset$. There must be at least one ground fact in B.
2. The induced dependency graph is acyclic;
3. Each random variable is only influenced by finite set of random variables.

Any such *well-defined* BLP B defines a unique probability distribution over the possible valuations of a ground query $G_a \in LH(B)$ [13]. The query-answering procedure actually consists of two parts: first, given a ground query and some evidence, the Bayesian network (namely the support network) containing all *relevant* atoms is computed, using *Knowledge Based Model Construction* (KBMC). 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.

Let B be a *well-defined* BLP and G_a a ground query. The Bayesian network constructed with KBMC 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) is denoted $\mathbf{P}^{\mathbf{BLP}}_{\mathbf{B},\mathbf{G_a}}(Q = v)$.

3.3 Statistical Relational Models

Statistical Relational Models (SRMs) were introduced in [6] in order to provide ways to infer statements over the success of some relational databases queries.

Syntax. SRMs are defined with respect to a given relational schema $\mathcal{R}$. Furthermore, SRMs require $\mathcal{R}$ to be *table stratified*, that is there must exist a partial ordering $\prec$ over classes in $\mathcal{R}$ such that for any $R.F \in \mathcal{F}(\mathcal{R})$ and $S = Dom[R.F]$, $S \prec R$ holds. Given such a *table stratified* relational schema $\mathcal{R}$, an SRM ψ is a pair $(\mathcal{S}, \theta)$ that defines a local probability model over a set of variables $\{R.A\}$ (for each class R and each descriptive attribute $A \in \mathcal{A}(R)$) and a set of boolean join indicator $\{R.J_F\}$ (for each foreign key $F \in \mathcal{F}(R)$ with $S = Dom[R.F]$). For each random variable of the form $R.V$, $\mathcal{S}$ specifies a set of parents $Pa(R.V)$ where each parents has the form $R.B$ or $R.F.B$, and θ specifies a CPT $\theta_{R.V} = P(R.V|Pa(R.V))$. $\mathcal{S}$ is further required to be a directed acyclic graph.

Semantics. Any SRM ψ defines a unique probability distribution $\mathbf{P}^{\mathbf{SRM}}_{\psi}$ over the class of so called *inverted-tree-foreign-key-join* queries (or *legal* queries) of a table stratified relational schema $\mathcal{R}$.

A *legal* query Q has form: $\bowtie_Q (\sigma_Q(R_1 \times R_2 \times \ldots \times R_n))$. The set $T = \{t_1, \ldots, t_n\}$ of tuple variables occurring in Q must be *closed* with respect to the *universal foreign key closure* of $\mathcal{R}$ as defined in [6] so that:

$$\bowtie_Q = \{t.F \bowtie s.K \mid t \in T,\ s \in T \textit{ is associated to } t.F\}$$

The select part of Q occurs on some subset of $\mathcal{A}(T)$, $\sigma_Q = \{A_i = a_i \mid A_i \in \mathcal{A}(T)\}$

Given an SRM $\psi = (\mathcal{S}, \theta)$, $\mathcal{S}$ induces a Bayesian network B over the attributes of tuples variables in T (joint indicators included). The parameters of B are set according to θ. $\mathbf{P}_\psi^{\mathbf{SRM}}$ is then defined as the probability distribution induced by B over possible instantiations of attributes of T that correspond to σ_Q of any *legal* query Q over T:

$$\mathbf{P}_\psi^{\mathbf{SRM}}(Q) = \prod_{t.V_i \in Q} \theta(t.V_i | Pa_B(t.V_i))$$

SRMs can thus be used to estimate the probability P_D of success of legal queries against a database D that implements the relational schema $\mathcal{R}$. For any select-join query Q over D, P_D is defined as follows:

$$P_D(Q) = \frac{|\bowtie_Q (\sigma_Q(R_1 \times R_2 \times \ldots \times R_n))|}{|R_1| \times |R_2| \times \ldots \times |R_n|}$$

A *table stratified* database D is said to be a model of an SRM ψ if ψ's estimations are correct, ie. for any legal query Q, $\mathbf{P}_\psi^{\mathbf{SRM}}(Q) = P_D(Q)$. In this case, we note $D \models \psi$.

3.4 Probabilistic Relational Models

Probabilistic Relational Models (PRMs) were introduced in [4], which extends the relational model presented in section 2.1 by introducing *reference slots* and *relational skeletons*. For a given class R, the set of reference slots $\varrho(R) = \{\rho_k\}$ is the union of R's foreign keys $\mathcal{F}(R)$ with the set of foreign keys $R'.F$ that point to $R.K$ (ie. reverse foreign keys). Such a reference slot ρ may thus establish a one-to-many relationship between R and $R' = R.\rho$. A relational skeleton σ is a set of objects respecting the constraints of a given relational schema. The difference between a relational skeleton σ and a complete instance is that the values of some descriptive attributes of objects in σ are unknown. However σ specifies the values for the foreign keys.

Syntax. PRMs with attribute uncertainty consider each class-level descriptive attribute as a random variable. PRMs make some independency assumptions in order to shrink the model size. As with BNs, these independency assumptions are encoded in an dependency structure $\mathcal{S}$ where the vertices represent the descriptive attributes. $\mathcal{S}$ is defined with respect to a given relational structure $\mathcal{R}$. For each descriptive attribute $R.A$ in $\mathcal{R}$, $\mathcal{S}$ specifies a set of parents $Pa(R.A)$. A parent takes either the form $R.A'$ (another descriptive attribute of the same class) or $\gamma(R.\tau.A')$ where $\tau = \rho_{k_1}.\rho_{k_2}.\ldots.\rho_{k_n}$ is a chain of n reference slots and

γ is an aggregate function. Indeed, for a given object $r \in R$, $r.\tau$ is potentially a multi-set of objects of class $R.\tau$. In such a case $\mathcal{S}$ uses an aggregate function γ to map the different values in $r.\tau.A'$ to a single value in the domain $\mathcal{V}(\gamma)$. Aggregate functions can be any of those traditionally used in SQL: *min*, *max*, *average*, etc. A dependency structure $\mathcal{S}$ is said to be *legal* with respect to a given relational skeleton if it is *guaranteed-acyclic* at the object-level: an object's descriptive attribute cannot be its own ancestor.

A PRM quantifies the probabilistic dependencies encoded in $\mathcal{S}$ through a set of parameters $\theta_{\mathcal{S}}$. For each attribute $r.A$, $\theta_{\mathcal{S}}(r.A) = P(r.A|Pa(r.A))$. The CPTs are identical for every objects of the same class. However, as the aggregate functions might compute different values for two different objects, the resulting probability can change from an object to another. A PRM Π is fully defined by a dependency structure $\mathcal{S}$ and its associated CPTs $\theta_{\mathcal{S}}$ (parameters), $\Pi = (\mathcal{S}, \theta_{\mathcal{S}})$.

Semantics. Given a relational skeleton σ, every PRM $\Pi = (\mathcal{S}, \theta_{\mathcal{S}})$, with $\mathcal{S}$ legal w.r.t. σ, defines a coherent probability distribution over $\mathcal{I}^{\sigma}$, the set of possible instances of σ, by the following chain-rule formula

$$\mathbf{P}^{\mathbf{PRM}}_{\mathbf{\Pi},\sigma}(i) = \prod_{x \in \sigma} \prod_{A \in \mathcal{A}(x)} P(i(x.A)|i(Pa(x.A)))$$

where $i(x.A)$ and $i(Pa(x.A))$ are the respective representations of the random variables $x.A$ and $Pa(x.A)$ in the instance i.

4 Behavioural Comparison of Expressive Knowledge Representations

Suppose that A, B represent two Herbrand bases[3] over given signatures Σ, Ω and that p, q represent probability functions over sets. Halpern's two types of probabilistic logic can be characterised as classes of probability functions with the following forms: $p : A \to [0, 1]$ (type 1) and $q : 2^B \to [0, 1]$ (type 2). Here 2^B represents the set of all possible worlds over the Herbrand base B. In this paper the approach taken to establishing relationships between type 1 and type 2 probabilistic logics involves demonstrating the existence of mappings between the logics. Suppose R_1, R_2 denote particular type 1, 2 logics respectively.

We say that R_1 is behaviourally weaker than R_2, or simply $R_1 \preceq_b R_2$ in the case that for every probability function p in R_1 with Herbrand base A there exists a probability function q in R_2 with Herbrand base B and a function f such that $f : A \to 2^B$ where $\forall a \in A \cdot q(f(a)) = p(a)$. Similarly, R_2 is behaviourally weaker than R_1, or simply $R_2 \preceq_b R_1$ when for every q in R_2 with Herbrand base B there exists a probability function p in R_1 with Herbrand base A and a function g such that $g : 2^B \to A$ where $\forall b \in 2^B \cdot p(g(b)) = q(b)$. As usual we say that R_1 is behaviourally equivalent to R_2, or simply $R_1 \equiv_b R_2$, in the case that $R_1 \preceq_b R_2$ and $R_2 \preceq_b R_1$.

[3] As stated before, we treat the Herbrand base of a logic model as its domain.

Halpern's work [7] provides good clarifications about what respective kinds of knowledge can be captured with probabilities on the domain (such as those defined by SLPs and SRMs) and probabilities on possible worlds (BLPs and PRMs). Links between these probabilities are also provided. However, the conclusions that can be drawn from a *behavioral* approach differ from the results obtained in previous *model-theoretical* studies (such as that of [7]): our aim is to provide ways in which knowledge encoded in one framework can be transferred into another framework. We focus on inter-translations, their features and limits.

We have adopted the following methodology:

- We first demonstrate relations between semantics: for a pair of frameworks (say, SLPs and BLPs), we define *equivalent programs* and *equivalent set of queries*. For instance, the fact that a k-ary Bayesian atom G_a takes the value v in a BLP can be represented in an *equivalent* SLP 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. We then say that a k-ary atomic BLP query is equivalent to the associated $(k+1)$-ary atomic SLP query.
- We say that the semantics are *equivalent* when equivalent (set of) queries on equivalent programs infer the same (set of) probability distributions.
- Hence our goal is eventually to provide algorithms that transform a program into an *equivalent* program in another framework, (such algorithms are referred to as *inter-translations*) and to analyze their features and their limits.

From the semantics perspective, we compare the four PLMs in terms of the following categories (as presented in section 2)

- Logic programming (LP) vs. relational models (RM): SLPs and BLPs are LP-based, while SRMs and PRMs are RM-based.
- Possible-world vs. domain-frequency: SLPs and SRMs are type 1 / domain-frequency approaches, in contrast type 2 / possible-world perspective is dominant in BLPs and PRMs.

In addition, SLPs are considered to be grammar-based models, while BLPs, PRMs and SRMs are classified to be graph-based models. In the rest sections, we detail the inter-translations between the PLMs of interests: SLPs-BLPs, SLPs-SRMs and BLPs-PRMs respectively.

5 A Behavioral Comparison of SLPs and BLPs

We first claim that a BLP B and an SLP S 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, ie. $\mathbf{P_S^{SLP}}(G_a) \equiv_b \mathbf{P_{B,G_a}^{BLP}}(G_a = v)$. There is an intuitive and global approach to find an inter-translation: any BLP B can be represented by

a (possibly infinite) Bayesian network BN_B, and the KBMC stage consists in finding the Bayesian variables relevant to the query (hence leading to a finite BN -a subpart of BN_B- that can be queried). Provided that the least Herbrand model of the BLP is finite, BN_B will be finite, and it is possible to use the method in [2] to translate BN_B into SLPs. But this approach cannot be extended to general BLPs.

To solve the problem, we need either restrict BLPs or extend SLPs. Therefore we developed a *standard translation* [20], which exists in two versions: one translates restricted BLPs (which do not make use of combining rules) into SLPs; and the other one translates general BLPs into extended SLPs (which are augmented with combining functions). One remaining drawback is that the standard translations do not handle evidence, that is, some prior knowledge about the domain in BNs. The reason is that SLPs and e-SLPs define semantics on tree structure, whereas KBMC in BLPs permits the union of several trees and the computation of probabilities in singly connected networks.

We summarize the translation approaches and theorems presented in [20] without examples and proofs, and provide some revisions with examples.

5.1 Restricted BLPs and Extended SLPs

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.

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

Given an e-SLP S_e consisting of the SLP S and the combining functions $\{CR_h\}$, and a query Q (consisting of a predicate h), the probability $\mathbf{P}^{\mathbf{eSLP}}_{\mathbf{S_e}}(Q)$ 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, $\mathbf{P}^{\mathbf{eSLP}}_{\mathbf{S_e}}(Q)$ is $\mathbf{P}^{\mathbf{SLP}}_{\mathbf{S}}(Q)$, the probability of S at query Q.
- 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 $\mathbf{P}^{\mathbf{eSLP}}_{\mathbf{S_e}}(Q) = CR_h((r_i,p_i)_{i\in\mathcal{N}_n})$, 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 $\mathbf{P}^{\mathbf{eSLP}}_{\mathbf{S_e}}(Q) = \prod_{i=1}^{n} p_i$, where p_i is the probability of the pruned and-or subtree rooted at the or-node r_i.

5.2 Standard Translation from Restricted BLPs to SLPs

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 clause consisting of the parameter $p_{v_h,v_{b1},...,v_{bn}}$, and the definite clause $head(v_h) \leftarrow b_1(v_{b1}), ..., b_n(v_{bn})$.
- 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* $\mathbf{P}_{\mathbf{S}}^{\mathbf{SLP}}(G(v)) \equiv_b \mathbf{P}_{\mathbf{B},\mathbf{G_a}}^{\mathbf{BLP}}(G_a = v)$.

5.3 Standard Translation from BLPs to e-SLPs

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.
- 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 $\{r_{i,j}\}$ with probabilities in S equal to $\{p_{i,j}\}$. 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:

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

  where CR is the combining rule defined in B.

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* $\mathbf{P}_{\mathbf{S_e}}^{\mathbf{eSLP}}(G(v)) \equiv_b \mathbf{P}_{\mathbf{B},\mathbf{G_a}}^{\mathbf{BLP}}(G_a = v)$.

5.4 Translation from SLPs to BLPs

Let S denote a complete, range-restricted and non-recursive SLP[4].

- For each stochastic clause $p : head \leftarrow b_1, ..., b_n$ in S, 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.

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* $\mathbf{P_S^{SLP}}(G) \equiv_b \mathbf{P_{B,G_a}^{BLP}}(G_a = true)$.

5.5 A Revised Translation from BLPs to SLPs

There exists a potential 'contradictory refutation' problem in BLPs-SLPs translation, which is illustrated in Figures 1, 2 and 3 for an example. The error lies in the potential inconsistent value settings (or substitutions) between atoms in a clause, eg. in clause **d(tom,y)** ← **b(tom,y)**,**c(tom,y)**, **b(tom,y)** may be set to **a(tom,y)** while **c(tom,y)** might be set to a contradictory value **a(tom,n)** simultaneously. To solve the problem, we introduce an extra data structure of list to 'set and remember' values instead of just setting values. Translations from the

[4] A clause C is said to be *non-recursive* iff the head of C is not found in the body of C.

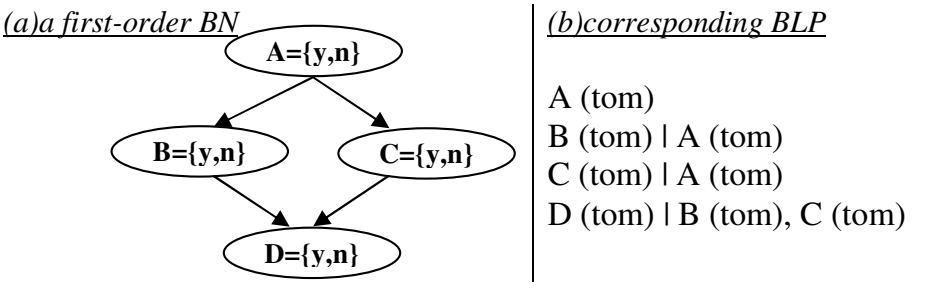

Fig. 1. An example of a first-order BN and corresponding BLP representation

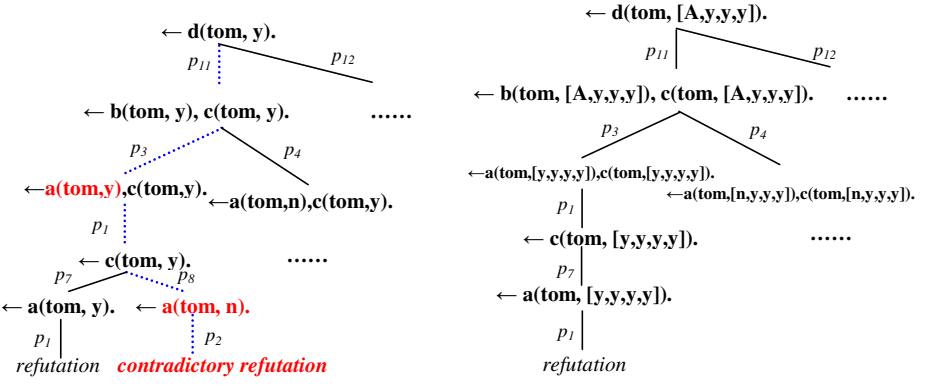

Fig. 2. (a) Stochastic SLD-tree with contradictory refutation (shown in dash lines) and (b) Resolved SSLD-tree without contradictory refutation

p_1: a(tom, y) ← .	p_1: a(tom, [y,B,C,D]) ← .
p_2: a(tom, n) ← .	p_2: a(tom, [n,B,C,D]) ← .
p_3: b(T, y) ← a(T, y).	p_3: b(T, [y,y,C,D]) ← a(T, [y,y,C,D]).
p_4: b(T, y) ← a(T, n).	p_4: b(T, [n,y,C,D]) ← a(T, [n,y,C,D]).
p_5: b(T, n) ← a(T, y).	p_5: b(T, [y,n,C,D]) ← a(T, [y,n,C,D]).
p_6: b(T, n) ← a(T, n).	p_6: b(T, [n,n,C,D]) ← a(T, [n,n,C,D]).
p_7: c(T, y) ← a(T, y).	p_7: c(T, [y,B,y,D]) ← a(T, [y,B,y,D]).
p_8: c(T, y) ← a(T, n).	p_8: c(T, [n,B,y,D]) ← a(T, [n,B,y,D]).
p_9: c(T, n) ← a(T, y).	p_9: c(T, [y,B,n,D]) ← a(T, [y,B,n,D]).
p_{10}: c(T, n) ← a(T, n).	p_{10}: c(T, [n,B,n,D]) ← a(T, [n,B,n,D]).
p_{11}: d(T, y) ← b(T, y), c(T, y).	p_{11}: d(T, [A,y,y,y]) ← b(T, [A,y,y,y]), c(T, [A,y,y,y]).
p_{12}: d(T, y) ← b(T, y), c(T, n).	p_{12}: d(T, [A,y,n,y]) ← b(T, [A,y,n,y]), c(T, [A,y,n,y]).
p_{13}: d(T, y) ← b(T, n), c(T, y).	p_{13}: d(T, [A,n,y,y]) ← b(T, [A,n,y,y]), c(T, [A,n,y,y]).
p_{14}: d(T, y) ← b(T, n), c(T, n).	p_{14}: d(T, [A,n,n,y]) ← b(T, [A,n,n,y]), c(T, [A,n,n,y]).
p_{15}: d(T, n) ← b(T, y), c(T, y).	p_{15}: d(T, [A,y,y,n]) ← b(T, [A,y,y,n]), c(T, [A,y,y,n]).
p_{16}: d(T, n) ← b(T, y), c(T, n).	p_{16}: d(T, [A,y,n,n]) ← b(T, [A,y,n,n]), c(T, [A,y,n,n]).
p_{17}: d(T, n) ← b(T, n), c(T, y).	p_{17}: d(T, [A,n,y,n]) ← b(T, [A,n,y,n]), c(T, [A,n,y,n]).
p_{18}: d(T, n) ← b(T, n), c(T, n).	p_{18}: d(T, [A,n,n,n]) ← b(T, [A,n,n,n]), c(T, [A,n,n,n]).

Fig. 3. Previous and revised translations from the above BLP to an SLP

BLP to an SLP by applying previous method (in [20]) and a revised method (this paper) are shown in Fig. 3(a) and (b) respectively, and the resolved stochastic SLD-tree can be seen in Fig.2(b). More precisely, a revised BLP-SLP translation algorithm is shown in the following steps. Let B denote a restricted BLP and S denote its translation SLP.

- Identify each k-ary Bayesian atom b, which appears in B and has the value domain V_b, to the $(k+1)$-ary (logical) atom $b(v_b)$ having the same k first arguments and a value $v_b \in V_b$ as last argument.
- Construct a list lv_b to replace v_b. The length of lv_b is the number of all Bayesian atoms. Each element of lv_b corresponds to an arbitrary Bayesian atom b' and is set to a fresh variable if $b' \neq b$ or a value $v_{b'} \in V_{b'}$ if $b' = b$.
- For each Bayesian clause $head \mid b_1, ..., b_n$ in B, for each value in the associated CPD, 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 a list lv_h for $head$ as done in step 2, then construct the stochastic clause consisting of the parameter $p_{v_h, v_{b1}, ..., v_{bn}}$, and the definite clause: $head(lv_h) \leftarrow b_1(lv_{b1}), ..., b_n(lv_{bn})$.
- For $lv_h, lv_{b1}, ..., lv_{bn}$, update the value for each element in lists with respect to $v_h, v_{b1}, ..., v_{bn}$ respectively.
- 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 CPD tables. 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).

Note that, in a definite clause (eg. **b(tom,[n,y,C,D])** $\leftarrow$ **a(tom,[n,y,C,D])**), all atoms have the same list values (eg. **[n,y,C,D]**), in which the elements corresponding to the atoms occurred in the clause are value set (eg. **[n,y, ,]** corresponds to atoms **a,b**) and other elements are assigned to be variables (eg. **[, ,C,D]** correspond to atoms **c,d**). The introduction of lists with variables will guarantee atoms to propagate consistent values to their predecessors in stochastic SLD-trees.

6 A Behavioral Comparison of SRMs and SLPs

SRMs naturally have a type 1 semantics with domain frequency over the rows of a database. This section will show how to build SLPs that encode the same class of semantics. Our approach is to find for any SRM ψ, a SLP S whose least Herbrand model is a contraction of a minimal *table stratified* database D such that $\psi \models D$ and such that the probabilities of success of legal queries against D or ψ match the probabilities induced by S on a set of corresponding SLP queries.

Let $\mathcal{R}$ be a *table stratified* relational schema and ψ an associated SRM. In order to translate ψ into an equivalent SLP S, we need to translate the model itself on one hand and the associated set of *legal* queries on the other hand.

For every descriptive attribute $R_i.A_j$ and for each ground instantiation with $\overrightarrow{k}$ as tuple of indices, we assert the parameterised ground fact:

$$\theta^{\overrightarrow{k}}_{R_i.A_j} \ : \ r_{i,j}(a^{k_1}_{i,j}, \overrightarrow{pa}(R_i, A_j)^{\overrightarrow{k}})$$

where $\theta^{\overrightarrow{k}}_{R_i.A_j}$ is the corresponding ψ parameter in which all joint indicator variables in the parents are set to *true*.

For each class R_i, we recursively define the key-validator predicate g_i as follows:

$$\begin{aligned} 1 \ : \ g_i(k_i(\overrightarrow{A_{i_1}}, \overrightarrow{F_i}, \overrightarrow{J_i})) \quad \leftarrow \\ g_{i_1}(F_{i_1}), \ldots, g_{i_1}(F_{i_n}), \\ ga_{i,1}(A_{i,1}), \ldots, ga_{i,n_i}(A_{i,n_i}), \\ gj_{i,1}(J_{i,1}), \ldots, gj_{i,m_i}(J_{i,m_i}). \end{aligned}$$

For each class R_i in ψ we can build a clause that defines the predicate r_i/n_i by using the above predicates and by introducing additional helper predicates[5].

$$\begin{aligned} 1 \ : \ r_i(k_i(\overrightarrow{A_i}, k_j(\overrightarrow{A_{j_1}}, \overrightarrow{F_j}, true(\overrightarrow{A_{i|J_{j,i}}}), \overrightarrow{J_j}), \overrightarrow{A_i}, \overrightarrow{F_i}, \overrightarrow{J_i}), \overrightarrow{A_i}, \overrightarrow{F_i}) \quad \leftarrow \\ g_j(k_j(\overrightarrow{A_{j_1}}, \overrightarrow{F_j}, true(\overrightarrow{A_{i|J_{j,i}}}), \overrightarrow{J_j})), \\ r_{i,1}(A_{i,1}, \overrightarrow{Pa}(A_{i,1})), \ldots, r_{i,n}(A_{i,n}, \overrightarrow{Pa}(A_{i,n})), \\ j_{i,1}(J_{i,1}, \overrightarrow{A_{i|J_{i,1}}}), \ldots, j_{i,m}(J_{i,m}, \overrightarrow{A_{i|J_{i,m}}}). \end{aligned}$$

Let Q be a *legal* query with respect to ψ. The following shows how to build a corresponding SLP query G_Q. Initialise G_Q to an empty formula. For each tuple variable t_j in of class R_i occurring in Q, add a literal to G_Q with predicate symbol r_i and the free variable K_j as primary key. Descriptive attributes arguments are set to the constants $a^l_{i,k}$ corresponding to the values $t_j.a^l_k$ specified in σ_Q or to some new free variable if no value is specified. Foreign key arguments are set bound variables K'_j according to the join $\bowtie_Q$.

Theorem. *The previous procedure translates any SRM ψ into a SLP S that computes the same probability distribution over legal queries; that is, for any legal query Q, the corresponding SLP query G_Q, such that* $\mathbf{P}^{\mathbf{SRM}}_{\psi}(Q) \equiv_b \mathbf{P}^{\mathbf{SLP}}_{\mathbf{S}}(G_Q)$.

7 A Behavioral Comparison of PRMs and BLPs

PRMs naturally have a type 2 semantics with possible worlds which correspond to possible instances of a given relational skeleton. This section will show how to build BLPs that can capture the same class of semantics where unary Bayesian predicates represent descriptive attributes and aggregate functions, binary predicates represent reference slots and constants represent objects of the relational skeleton.

Given a relational skeleton σ and a PRM $\Pi = (\mathcal{S}, \theta_{\mathcal{S}})$, Table 1 defines a translation procedure `prm2blp` that builds a BLP B inferring a probabilistic distribution on $\mathcal{I}^{\sigma}$, the set of complete instances of the relational skeleton σ.

[5] Definition and translation of helper predicates are omitted.

Table 1. The prm2blp translation procedure from PRMs to BLPs

proc prm2blp(σ, $\mathcal{S}$, $\theta_{\mathcal{S}}$):

1. for each class R_i:
   (a) for each descriptive attribute $A_j \in A(R_i)$:
   define the unary Bayesian predicate $\mathtt{p_{i,j}}/1$ with
   $dom(\mathtt{p_{i,j}}/1) = \mathcal{V}(A_j)$
   (b) for each reference slot $\rho_k \in R(R_i)$:
   define the binary Bayesian predicate $\mathtt{r_{i,k}}/2$ with
   $dom(\mathtt{r_{i,k}}/2) = \{true, false\}$
   (c) for each aggregate function γ_i in $\theta_{\mathcal{S}}$:
   define the unary Bayesian predicate $\mathtt{g_i}/1$ with
   $dom(\mathtt{g_i}/1) = \mathcal{V}(\gamma_i)$
2. let B be an empty BLP
3. for each class R_i:
   (a) for each object $o \in \mathcal{O}^{\sigma}(R_i)$:
   i. for each reference slot $\rho_k \in \varrho(R_i)$ and each
   $o' \in \rho_k(o)$:
   assert in B the ground Bayesian fact $\mathtt{r_{i,k}}(\mathtt{o}, \mathtt{o'})$. with associated
   (instantiated) CPT: $[1, 0]$
   (b) for each descriptive attribute $A_j \in A(R_i)$:
   – if $(Pa(R_i.A_j) = \emptyset)$ according to $\mathcal{S}$ then:
   for each object $o \in \mathcal{O}^{\sigma}(R_i)$:
   i. assert in B the ground Bayesian fact $\mathtt{p_{i,j}}(\mathtt{o})$. with
   associated CPT: $\theta_{\mathcal{S}}(R_i.A_j)$
   – else:
   i. let C be a Bayesian clause with head $\mathtt{p_{i,j}}(\mathtt{V})$
   ii. for each $U_l \in \mathbf{U} = Pa(R_i.A_j)$
   • if $U_l = R_i.A_m$ then:
   add the literal $\mathtt{p_{i,m}}(\mathtt{V})$ to the body of C
   • else $U_l = \gamma_k(R_i.\tau.A_m)$ where τ a chain of
   reference slots of the form $\tau = \rho_{k_1}, \rho_{k_2}, \ldots, \rho_{k_n}$:
   ∗ add the literal $\mathtt{g_k}(\mathtt{V})$ to the body of C
   ∗ let $R_{i'} = R_i.\tau$
   ∗ assert in B the following helper Bayesian clause :
   $\mathtt{g_k}(\mathtt{V}) \mid \mathtt{r_{i,k_1}}(\mathtt{V}, \mathtt{V_1}), \ldots, \mathtt{r_{k_{n-1},k_n}}(\mathtt{V_{n-1}}, \mathtt{V_n}), \mathtt{p_{i',m}}(\mathtt{V_n})$.
   iii. let CPT_C be $\theta_{\mathcal{S}}(R_i.A_j|Pa(R_i.A_j))$
   iv. assert C in B
4. build the Combining Rules for each predicate in B by applying the
   build_cr procedure
5. return B

Theorem. *For any PRM Π that is* guaranteed-acyclic *w.r.t. some non-empty relational skeleton σ:*

$$\forall i \in \mathcal{I}^{\sigma} : \mathbf{P}^{\mathbf{PRM}}_{(\sigma,\mathbf{\Pi})}(i) \equiv_b \mathbf{P}^{\mathbf{BLP}}_{\mathtt{prm2blp}(\sigma,\mathbf{\Pi})}(i'),$$

where i' is the ground Bayesian query corresponding to the database instance i.

8 Discussion and Conclusions

The first result we achieved in the study is BLPs $\equiv_b$ e-SLPs. We argue that SLPs augmented with combining functions (namely extended SLPs) and BLPs can encode the same knowledge, in that they encode equivalent probability distributions for equivalent set of queries. 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 BLP's query-answering procedure benefits from different frameworks, say logic programming and Bayesian networks, while inference mechanisms in SLPs are straightforward using only logic programming.

Another finding is also shown in the study, denoting as PRMs $\preceq_b$ BLPs and SRMs $\preceq_b$ SLPs. When considering models within the same probabilistic model category (type 1 or type 2), BLPs (resp. SLPs) can naturally express PRMs (resp. SRMs), i.e., translated models and queries can be forged, which compute the same probability distributions.

We believe this study to be a formal basis for further research. Several learning algorithms have been devised for SLPs [2,15,16], BLPs [11,12], PRMs [4,5] and SRMs [5,6]. Further work thus includes the study of how inter-translations between those frameworks can help devising better learning algorithms for PLMs depending on the kind of knowledge we want to model. For instance, 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). Investigating such extensions could be interesting. We also hope this study provide a bridge to developing an integrated theory of probabilistic logic learning.

As the related studies, one may find an intuitive approach of translating SLPs into BNs, Markov networks and stochastic context free grammars in [2]; a simpler scheme for mapping PRMs to BLPs is contained in [3]; a theoretical comparison of BLPs and Relational Markov Models (RMMs) is presented in [18]; and another approach of analysing the expressive power of different probabilistic logic languages could be found in [9,8] as well as in this volume.

Acknowledgement

We are very grateful to Aymeric Puech and Olivier Grisel for their initial contributions on the topic when they were studying in Imperial College London. This work was supported by the Royal Academy of Engineering/Microsoft Research Chair on 'Automated Microfluidic Experimentation using Probabilistic Inductive Logic Programming'; the BBSRC grant supporting the Centre for Integrative Systems Biology at Imperial College, Grant Reference BB/C519670/1; the BBSRC grant on 'Protein Function Prediction using Machine Learning by Enhanced Novel Support Vector Logic-based Approach', Grant Reference BB/E000940/1;

ESPRIT IST project 'Application of Probabilistic Inductive Logic Programming II (APRIL II)', Grant reference FP-508861.

References

1. Bratko, I.: Prolog for artificial intelligence. Addison-Wesley, London (1986)
2. Cussens, J.: Parameter estimation in stochastic logic programs. Machine Learning 44(3), 245–271 (2001)
3. De Raedt, L., Kersting, K.: Probabilistic Logic Learning. ACM-SIGKDD Explorations: Special issue on Multi-Relational Data Mining 5(1), 31–48 (2003)
4. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Dean, T. (ed.) Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI 1999), Stockholm, Sweden, pp. 1300–1309. Morgan Kaufmann, San Francisco (1999)
5. Getoor, L.: Learning Statistical Models from Relational Data. PhD thesis, Stanford University (2001)
6. Getoor, L., Koller, D., Taskar, B.: Statistical models for relational data. In: Wrobel, S. (ed.) MRDM 2002, University of Alberta, Edmonton, Canada, July 2002, pp. 36–55 (2002)
7. Halpern, J.Y.: An analysis of first-order logics of probability. Artificial Intelligence 46, 311–350 (1989)
8. Jaeger, M.: Type extension trees: A unified framework for relational feature construction. In: Gärtner, T., Garriga, G.C., Meinl, T. (eds.) Working Notes of the ECML 2006 Workshop on Mining and Learning with Graphs (MLG 2006), Berlin, Germany (September 2006)
9. Jaeger, M., Kersting, K., De Raedt, L.: Expressivity analysis for pl-languages. In: Fern, A., Getoor, L., Milch, B. (eds.) Working Notes of the ICML 2006 Workshop Open Problems in Statistial Relational Learning (SRL 2006), Pittsburgh, USA, June 29 (2006)
10. Jensen, F.V.: Introduction to Bayesian Networks. Springer, New York (1996)
11. Kersting, K., De Raedt, L.: Adaptive Bayesian Logic Programs. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, Springer, Heidelberg (2001)
12. Kersting, K., De Raedt, L.: Towards Combining Inductive Logic Programming and Bayesian Networks. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, Springer, Heidelberg (2001)
13. Kersting, K., De Raedt, L.: Bayesian logic programs. In: Cussens, J., Frisch, A. (eds.) Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming, pp. 138–155 (2000)
14. Muggleton, S.H.: Stochastic logic programs. In: de Raedt, L. (ed.) Advances in Inductive Logic Programming, pp. 254–264. IOS Press, Amsterdam (1996)
15. Muggleton, S.H.: Learning stochastic logic programs. Electronic Transactions in Artificial Intelligence 4(041) (2000)
16. Muggleton, S.H.: Learning structure and parameters of stochastic logic programs. In: Matwin, S., Sammut, C. (eds.) ILP 2002. LNCS (LNAI), vol. 2583, Springer, Heidelberg (2003)
17. Muggleton, S.H., Firth, J.: CProgol4.4: a tutorial introduction. In: Dzeroski, S., Lavrac, N. (eds.) Relational Data Mining, pp. 160–188. Springer, Heidelberg (2001)

18. Muggleton, S.H., Pahlavi, N.: The complexity of translating blps to rmms. In: Muggleton, S., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 351–365. Springer, Heidelberg (2007)
19. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, Los Altos (1988)
20. Puech, A., Muggleton, S.H.: A comparison of stochastic logic programs and Bayesian logic programs. In: IJCAI 2003 Workshop on Learning Statistical Models from Relational Data, IJCAI (2003)