A Comparison Between two Statistical Relational Models

Lorenza Saitta *, Christel Vrain **
saitta@mfn.unipmn.it, christel.vrain@univ-orleans.fr

* Dip. di Informatica, Università del Piemonte Orientale
Via Bellini 25/G, 15100 Alessandria, Italy
** LIFO, University of Orléans , BP 6759, 45067 Orléans cedex 02, France

Abstract. Statistical Relational Learning has received much attention this last decade. In the ILP community, several models have emerged for modelling and learning uncertain knowledge, expressed in subset of first order logics. Nevertheless, no deep comparisons have been made among them and, given an application, determining which model must be chosen is difficult. In this paper, we compare two of them, namely Markov Logic Networks and Bayesian Programs, especially with respect to their representation ability and inference methods. The comparison shows that the two models are substantially different, from the point of view of the user, so that choosing one really means choosing a different philosophy to look at the problem. In order to make the comparison more concrete, we have used a running example, which shows most of the interesting points of the approaches, yet remaining exactly tractable.

1 Introduction

In the last years there has been an increasing interest in graphical models for reasoning and learning. In particular, several approaches aiming at representing structured domains, described in some subset of First Order Logic, have been proposed, such as, for instance, Markov Logic Networks [RD06], Bayesian Logic Programs [KdR07], Probabilistic Relational Models [FGKP99], Relational Bayesian Networks [Jae97,Jae02]. All these approaches rely on the possible-world semantics of first-order probabilistic logic [Hal96], which assigns a probability distribution over possible worlds [1]. Stochastic Logic Programs [Mug00] are slightly different in the sense that they extend stochastic grammars and allow a distribution to be computed on the Herbrand base; in [Cus01], they have been proven useful for representing undirected Bayes nets.

In this paper we review two statistical relational models, namely Markov Logic Networks and Bayesian Logic Programs, and we attempt a quantitative comparison between them. The reader is assumed to be familiar with both syntax and semantics of First Order Logics, especially with clausal form representation.

Throughout this paper we will use a running example to perform the comparison. Even though it is a very simple one, it shows most of the interesting features of a

[1] Let us notice that Probabilistic Relational Models have been defined both on possible-world semantics and on domain frequency semantics [Get06]

stochastic predicate logic knowledge base, yet remaining tractable, so that exact inferences can be done.

Example 1. Let $\mathcal{K}$ be a knowledge base containing four rules:

$$F_1: \; R(X,Y) :- P(X), Q(Y) \qquad F_2: \; S(X,Y) :- P(X), Q(Y)$$
$$F_3: \; V(X,Y) :- Q(X), Q(Y) \qquad F_4: \; R(X,Y) :- V(X,Y)$$

In $\mathcal{K}$ the two rules F_1 and F_2 share the antecedent but have different consequents, whereas F_1 and F_4 share the consequent, but have different antecedents. Moreover, if $\mathbf{C} = \{a,b\}$ is the set of constants in the universe, the Herbrand base $HB_{\mathcal{K}}$, relative to the knowledge base $\mathcal{K}$, contains the following ground atoms:
$HB_{\mathcal{K}} = \{P(a), P(b), Q(a), Q(b), R(a,a), R(a,b), R(b,a), R(b,b), S(a,a), S(a,b),$
$S(b,a), S(b,b), V(a,a), V(a,b), V(b,a), V(b,b)\}$

The paper is organized as follows. Section 2 is devoted to Markov Logic Networks, whereas we present Bayesian Logic Programs in Section 3. In Section 4 we compare the two frameworks, and in Section 5 some conclusions are drawn.

2 First order logic background

Classically, a *first order language* $\mathcal{L}$ is defined by a countable set **V** of variables, a set **F** of function symbols (including functions with arity 0, also called constants) and a set **P** of predicate symbols.

In this paper, we consider only *function-free first-order languages* that contain no function symbols other that constants, and in the following, **C** denotes the set of constants. With this restriction, a *term* is either a variable or a constant. A *ground term* is a constant. The *Herbrand Universe* of $\mathcal{L}$, denoted by $HU_{\mathcal{L}}$, is the set **C** of constants. A substitution σ is a *variable assignment* that maps each variable X to a variable or to a constant. An *atom* is an expression $p(t_1, \ldots, t_n)$, where p is a predicate symbol and $t_1, \ldots, t_n$ are either constants or variables. A *ground atom* is an atom that contains no variables. The *Herbrand Base* of $\mathcal{L}$, denoted by $HB_{\mathcal{L}}$, is the set of ground atoms.

A *(positive) rule* r is an expression $A \leftarrow B_1, \ldots, B_k$, where $B_1, \ldots B_k$ are atoms. The atom A is called the *head* of r and is denoted by $head(r)$. The right-hand side $B_1, \ldots, B_k$ is called the *body* of r and is denoted by $body(r)$.

A *(positive) program* $\mathcal{P}$ is a set of positive rules. In the following, if $\mathcal{P}$ is a program, $Inst(\mathcal{P})$ is the set of all ground rules obtained by substituting variables in $\mathcal{P}$ by ground terms ($c \in Inst(\mathcal{P})$ when c is ground and there exists a clause c' of $\mathcal{P}$ and a ground substitution σ of variables of c' s.t. $c'\sigma = c$)

In the following, $HB_{\mathcal{P}}$ denote the Herbrand base of the first-order language, underlying the definition of program $\mathcal{P}$.

An *interpretation* I is defined by giving a domain D, $D \neq \emptyset$, and by associating to each constant c, $c \in \mathbf{C}$, an element of D and to each predicate symbol p, $p \in \mathbf{P}$, with arity

n, a function $I_p : D^n \rightarrow \{0,1\}$. A *Herbrand interpretation I* is an interpretation on the Herbrand Universe that interprets each constant by itself.
The semantics of a positive logic program can be defined as in the following:

Definition 1. *Let $\mathcal{P}$ be a positive logic program. The semantics $\mathcal{M}_\mathcal{P}$ of $\mathcal{P}$ is defined as the least fixpoint of the operator $T_\mathcal{P}$ defined for a set a ground atoms I by:*

$$T_\mathcal{P}(I) = \{A \in HB_\mathcal{P} | \exists C \in Inst(\mathcal{P}),\ head(C) = A \text{ and } body(C) \subseteq I\}.$$

$\mathcal{M}_\mathcal{P}$ is the least Herbrand model of $\mathcal{P}$, *i.e.*, it is the set of ground atoms that are logical consequences of $\mathcal{P}$.

3 Markov Logic Networks

Domingos and co-workers have used the concepts of *Markov Networks* (MN) and of *Markov Logic Networks* (MLN) to address the problem of making probabilistic inferences in first order logic [RD06,KD05,DR04].

3.1 Markov Network Representation

A *Markov Network* (**MN**) is a model of the joint probability distribution over a set $\mathbf{X} = (X_1, ..., X_N)$ of variables. More precisely, a Markov network is an undirected graph G, with N nodes (each corresponding to one of the stochastic variables) and M edges, and a set of potential functions Φ_k $(1 \leqslant k \leqslant K)$. Each Φ_k is associated to a different *clique* in the graph. Let $\mathbf{x} = (x_1, ..., x_N) \in \mathcal{X}$ be a set of values that the variables can take on; then, a Markov network is associated to the following probability distribution:

$$Pr(\mathbf{X} = \mathbf{x}) = \frac{1}{Z}\, e^{\sum_{k=1}^{K} w_k f_k(\mathbf{x}^{(k)})} \tag{1}$$

In (1) each function $f_k(\mathbf{x}^{(k)})$ is a *feature* of the k-th clique, and w_k is its *weight*. Moreover, Z is a normalization factor. Even though a feature can be any function, Domingos's approach focuses on binary features, *i.e.*, $f_j(\mathbf{x}^{(\mathbf{k})}) \in \{0,1\}$. Moreover, $w_k = ln\ \Phi_k(\mathbf{x}^{(k)})$.

A Markov Network is derived from a *Markov Logic Network* (**MLN**) and a set $\mathbf{C} = \{a_1, ..., a_C\}$ of constants. A Markov Logic Network $\mathbf{MLN} = \{(F_i, w_i) | 1 \leqslant i \leqslant M\}$ is a set of pairs (F_i, w_i), consisting of a logical formula F_i, belonging to a knowledge base $\mathcal{K}$, and an associated real number w_i. The logical language is function-free.

The association between $\mathbf{MLN}, \mathbf{C}$ and a Markov network **MN** is performed as follows: let $HB_\mathcal{K}$ be the Herbrand base of $\mathcal{K}$. Each element of $HB_\mathcal{K}$ is associated to a node in the graph **MN**, which then represents a binary variable. Now we can ground all formulas F_i $(1 \leqslant i \leqslant M)$ in all possible ways. In **MN** two nodes are connected if they appear together in at least one of the groundings of a formula F_i, for some i.

Example 2. By considering the knowledge base $\mathcal{K}$ of Example 1, we can define an **MLN** as follows:

$$\mathbf{MLN} = \{(F_1, w_1), (F_2, w_2), (F_3, w_3), (F_4, w_4)\},$$

where the F_i's are the rules in $\mathcal{K}$ and $\mathbf{w} = (w_1, w_2, w_3, w_4)$ is a vector of weights.

Using the **MLN** of Example 1 and its Herbrand base $HB_{\mathcal{K}}$, we obtain the Markov network **MN** reported in Figure 1. The network **MN** has $N = 16$ nodes and $M = 31$

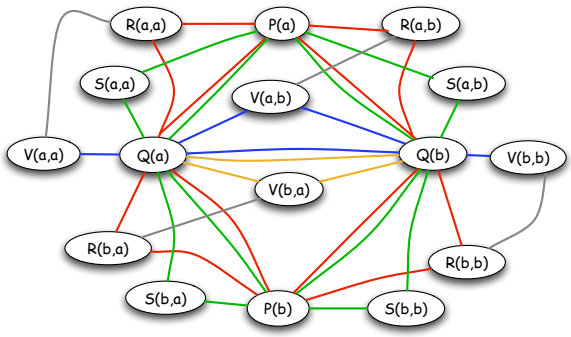

Fig. 1. Markov network corresponding to Example 3.

edges.

Clearly, groundings of the same formula constitute cliques in **MN**. Each grounding of each formula is a feature f_j. To each grounding of the same formula F_i the same weight w_i, specified in **MLN**, is attached. We may notice, however, that in **MN** there may appear spurious cliques, *i.e.*, cliques that do not correspond to a grounding of any formula. In the Markov network **MN** of Figure 1, there are 31 cliques of length 2 (equal to the number of edges in the graph), among which only 6 are maximal, and 16 cliques of length 3 (all maximal ones). In fact, in this case, the length-2 and length-3 cliques derive from different formulas and have different weights associated to them. In particular, the length-2 cliques to be considered are the groundings of formula F_4, and some of formula F_3. Of the sixteen length-3 cliques, only ten are true groundings, and they are reported in Figure 2 (each one with its associated weight). A spurious clique is, for instance, (Q(b), R(a,b), V(a,b)).

Let us now consider the set $\mathcal{X}$ of possible worlds. Given a world $\mathbf{x} \in \mathcal{X}$, all variables associated to the nodes of **MN** have a value in $\{true, false\}$. Correspondingly, each feature f_j, corresponding to the grounding of a formula, has the value *1*, if the corresponding grounding is *true*, 0 otherwise. By reconsidering formula (3) and recalling that $w_k = ln\, \Phi_k(\mathbf{x}^{(k)})$, we obtain the following final form for the probability distribution:

$$Pr(\mathbf{X} = \mathbf{x}) = \frac{1}{Z}\, e^{\sum_{i=1}^{M} w_i\ n_i(\mathbf{x})} = \frac{1}{Z} \prod_{i=1}^{M} e^{w_i\ n_i(\mathbf{x})} \tag{2}$$

In (2) $n_i(\mathbf{x})$ is the number of true groundings of formula F_i in $\mathbf{x}$, or, in other words, the number of features, corresponding to F_i, that are equal to 1 in $\mathbf{x}$. Formula (2) will be the one used in the computation of probabilities. However, it has to be noted that

F_i	Formula	Features	Weight w_i
F_1	R(X,Y) :- P(X), Q(Y)	(P(a), Q(a), R(a,a))	w_1
		(P(a), Q(b), R(a,b))	w_1
		(P(b), Q(a), R(b,a))	w_1
		(P(b), Q(b), R(b,b))	w_1
F_2	S(X,Y) :- P(X), Q(Y)	(P(a), Q(a), S(a,a))	w_2
		(P(a), Q(b), S(a,b))	w_2
		(P(b), Q(a), S(b,a))	w_2
		(P(b), Q(b), S(b,b))	w_2
F_3	V(X,Y) :- Q(X), Q(Y)	(Q(a), V(a,a))	w_3
		(Q(a), Q(b), V(a,b))	w_3
		(Q(b), Q(a), V(b,a))	w_3
		(Q(b), V(b,b))	w_3
F_4	R(X,Y) :- V(X,Y)	(R(a,a), V(a,a))	w_4
		(R(a,b), V(a,b))	w_4
		(R(b,a), V(b,a))	w_4
		(R(b,b), V(b,b))	w_4

Fig. 2. List of features associated to the formulas in the **MLN** of Example 2.

computing the partition function Z is intractable, because it requires a number of steps $O(2^{|\mathbf{MN}|})$. From (2) and Z, it is possible to compute the probability of any possible world $\mathbf{x}$.

For the sake of exemplification, let $\mathbf{w} = (w_1, w_2, w_3, w_4) = (2, 1.5, 0.8, 1.7)$ be the weight vector associated to $\mathcal{K}$. Using formula (3) with the weights introduced before, it is possible to compute, in this simple case, the exact value of $Z = 3.0225... \cdot 10^{14} \simeq 3.023 \cdot 10^{14}$.

From Figure 1 we see that the state space corresponding to the Markov network is the space $\{0,1\}^{16}$, consisting of all the possible truth value assignments to the binary variables associated to the nodes. The probability distribution over the possible worlds can be written as follows:

$$Pr(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_{k=1}^{4} e^{w_k n_k(\mathbf{x})} = \frac{1}{Z} e^{2\ n_1(\mathbf{x})} \cdot e^{1.5\ n_2(\mathbf{x})} \cdot e^{0.8\ n_3(\mathbf{x})} \cdot e^{1.7\ n_4(\mathbf{x})} \tag{3}$$

In equation (3), $n_k(\mathbf{x})$ is the number of features associated to formula F_k that are true in the world $\mathbf{x}$. In the extreme case when all the variables are true (let $\mathbf{x_1}$ be the corresponding world), we have $n_1(\mathbf{x_1}) = n_2(\mathbf{x_1}) = n_3(\mathbf{x_1}) = n_4(\mathbf{x_1}) = 4$, and then:

$$Pr(\mathbf{X} = \mathbf{x_1}) = \frac{1}{Z} e^{4\ (2+1.5+0.8+1.7)} = \frac{e^{24}}{3.023 \cdot 10^{14}} = 8.76 \cdot 10^{-5}$$

On the opposite, when all the variables are false (let $\mathbf{x_0}$ be the corresponding world), we still have $n_1(\mathbf{x_0}) = n_2(\mathbf{x_0}) = n_3(\mathbf{x_0}) = n_4(\mathbf{x_0}) = 4$, because all the rules' premises are false and hence all the groundings of the rules are true; then:

$$Pr(\mathbf{X} = \mathbf{x_1}) = Pr(\mathbf{X} = \mathbf{x_0})$$

The following result could be easily shown:
Proposition. When the knowledge base $\mathcal{K}$ consists only of positive rules without facts, then the probability of the world where all the variables are false is the same as the probability of the world where all all the variables are true, and it is maximal.

As a further example, let us consider the world $\mathbf{x_3}$ in which P(a) = 1, P(b) = 1, Q(a) = 0, Q(b) = 1, R(a,b) = 0, S(a,b) = 1, R(b,b) = 0, S(b,b) = 0, R(b,a) = 1, S(b,a) = 0, R(a,a) = 0, S(a,a) = 0, V(a,a) = 0, V(b,b) = 0, V(a,b) = 1, V(b,a) = 0. In this world, we have $n_1(\mathbf{x_2}) = 2, n_2(\mathbf{x_2}) = 3, n_3(\mathbf{x_2}) = 3, n_4(\mathbf{x_2}) = 3$, and, hence:

$$Pr(\mathbf{X} = \mathbf{x_3}) = \frac{1}{Z}\, e^{2\cdot 2+1.5\cdot 3+0.8\cdot 3+1.7\cdot 3} = 2.94\ 10^{-8}$$

We may notice that, from the point of view of computing probability distribution (2), the translation of the Markov logic network **MLN** into the Markov network **MN** is not useful. As spurious cliques may appear in the transformation, it is not possible to write formula (2) from **MN** only. In fact, even though, in the general case, all cliques contribute to the probability distribution with a potential function, in this case only those that correspond to groundings of formulas in **MLN** must be considered, and these are only a subset of all the cliques occurring in the network; then, a further check is needed to discover which ones they are.

3.2 Inference

Given a Markov Logic Network **MLN** and a set **C** of constants, the central inference problem can be formulated as follows:

"What is the probability that a ground formula φ_1 is true, knowing that another ground formula φ_2 is true?"[2].

In general, to find an answer to this question is intractable. Then, approximate algorithms are to be used. Richardson and Domingos provide such an algorithm for the special case where φ_1 and φ_2 are conjunctions of ground literals. The algorithm proceeds in two phases; the first one returns the minimal subset of the ground Markov network required to compute $Pr(\varphi_1|\varphi_2, \mathbf{MLN}, \mathbf{C})$. The authors observe that the size of the network returned may be further reduced by ignoring any ground formula which is made true by the evidence (*i.e.*, by φ_2), and by removing the corresponding arcs removed from the network.

The second phase of Richardson and Domingos' algorithm performs inference on the network returned in the first one, with the nodes in φ_2 set to their values. Their algorithm implementation uses Gibbs sampling. The basic Gibbs step consists of sampling one ground atom given its Markov blanket. The Markov blanket of a ground atom is the set of ground atoms that appear in some grounding of a formula with it. In a Markov network, the Markov blanket of a node is the set of all its immediate neighbors. Let X be the binary variable associated to a node in **MN**, and let $\mathcal{B}(X)$ be its Markov blanket.

[2] In the following, we use greek letters to denote ground formulas, not to be confused with the rules in $\mathcal{K}$

The probability that X is true, given the state $\mathbf{s}$ of $\mathcal{B}(X)$, can be estimated through Gibbs sampling as follows:

$$Pr(X=1|\mathcal{B}(X)=\mathbf{s}) = \frac{e^{\sum_{i=1}^{M} w_i \cdot n_i"(X=1,\mathcal{B}(X)=\mathbf{s})}}{e^{\sum_{i=1}^{M} w_i \cdot n_i"(X=1,\mathcal{B}(X)=\mathbf{s})} + e^{\sum_{i=1}^{M} w_i \cdot n_i"(X=0,\mathcal{B}(X)=\mathbf{s})}} \quad (4)$$

In (4) $n_i"(X=0,\mathcal{B}(X)=\mathbf{s})$ is the number of true features involving X, for $X=1$ and $\mathcal{B}(X)=\mathbf{s}$. The estimated probability is the fraction of samples in which the conjunction is true, once the Markov chain has converged.

3.3 Learning

One way in which learning occurs is acquiring the weights, given the structure of the Markov logic network. Weights can be learned from databases under the *closed world* assumption: if an atom occurs in the database, it is true, otherwise it is assumed to be false. If there are n possible ground atoms, a database is effectively a vector $\mathbf{x} = (x_1,...,x_k,...,x_n)$, where x_k is the truth value of the k-th ground atom ($x_k = 1$ if the atom appears in the database, and $x_k = 0$ otherwise).

The weights could be learned through maximization of the log-likelihood with respect to the weights themselves, but this process is computationally intractable, and Richardson and Domingos propose a more efficient method, based on the optimization of the pseudo-likelihood:

$$Pr^*_w(\mathbf{X}=\mathbf{x}) = \prod_{k=1}^{n} Pr_w(X_k = x_k|\mathcal{B}_\mathbf{x}(X_k)), \quad (5)$$

where $\mathcal{B}_x(X_k)$ is the Markov blanket of X_k in data $\mathbf{x}$. Optimization of equation (5) is done using the limited-memory BFGS algorithm [LN89].

If the network structure has to be learned (totally or partially), any relational or ILP learner could be used to the purpose.

4 Bayesian Programs

Kersting and De Raedt [KdR07] have proposed an approach based on Bayesian networks, extending the traditional notions of atom, clauses, and interpretation of a clause. The core of their approach is the notion of *Bayesian program*.

4.1 Bayesian Programs

In Bayesian programs, the notion of *Bayesian atom* is introduced: it is built with a predicate and with terms, but it takes a finite set of possible values, instead of simply $true$ or $false$. For instance, in first order logic, the marital status of a person (single, married, divorced, ...) is represented either by introducing a predicate $single(X)$, $married(X)$, ...for each possible value, or by introducing a binary predicate $status(X,Y)$, where Y is instantiated by a constant representing the marital status. Kersting and De Raedt suggest to use a predicate *status*, where $status(X)$ can take a finite set of possible values {*single*, *married*, ...}[KdR07]. Let us notice that in this framework, a classical atom is a particular case of a Bayesian atom with two values $\{true, false\}$.

Definition 2. *A* Bayesian clause *c is an expression $A|A_1,\ldots,A_n$, where A, A_1, $\ldots A_n$ are Bayesian atoms. The atom A is called the* head *of c, written $head(c)$, whereas A_1, $\ldots$, A_n is called the* body *of c, written $body(c)$. When $n = 0$, such a clause is called a Bayesian fact. A* Bayesian program *is a finite set of Bayesian clauses. Moreover, for each clause c there exists a conditional probability distribution $cpd(c)$ that encodes $Pr(head(c)|body(c))$.*

It is assumed that Bayesian clauses are range-restricted, *i.e.*, all the variables that occur in A also occur in $A_1, \ldots, A_n$.

Example 3. By transforming the knowledge base $\mathcal{K}$ of Example 1 into a Bayesian program, we obtain four clauses:

$$\mathcal{P}_{\mathcal{K}} = \{[R(X,Y)|P(X),Q(Y)],[S(X,Y)|P(X),Q(Y)],[V(X,Y)|Q(X),Q(Y)],\\ [R(X,Y)|V(X,Y)]\}$$

Conditional probability distributions must be associated to the rules, in the form of tables.

4.2 Semantics

A directed graph $\mathcal{G}_{\mathcal{P}}$ is associated to a Bayesian logic program $\mathcal{P}$ as follows: the nodes of $\mathcal{G}_{\mathcal{P}}$ are the elements of $\mathcal{M}_{\mathcal{P}}$, and there is an edge from x to y if there exists a clause c of $Inst\text{-}rel(\mathcal{P})$ s.t. $x \in body(c)$ and $y = head(c)$.

An important point that differs from propositional Bayesian networks is that, given a node y, there may be several clauses with $head(c) = y$. In such cases, there is an edge starting from each ground atom occurring in the bodies of these clauses and ending at y, thus losing in the graphical representation the structured information encapsulated in the clauses.

As already mentioned, in a Bayesian logic program $\mathcal{P}$ a conditional probability distribution is associated to each clause. Nevertheless, as several clauses of $Inst\text{-}rel(\mathcal{P})$ may have the same head, it is necessary to define some *combining rule* [NH97] that maps finite sets of conditional probability distributions $P(A|A_{i_1},\ldots,A_{i_{n_i}})$, $i = 1,\ldots,m$, onto a single one $P(A|B_1,\ldots,B_k)$, where $\{B_1,\ldots,B_k\} \subseteq \cup_{i=1}^{m}\{A_{i_1},\ldots,A_{i_{n_i}}\}$. It is shown that if $\mathcal{M}_{\mathcal{P}} \neq \emptyset$, $\mathcal{G}_{\mathcal{P}}$ is acyclic, and each node is influenced by a finite set of random variables, then $\mathcal{P}$ specifies a unique probability distribution over $\mathcal{M}_{\mathcal{P}}$. Two rules that are widely employed in Bayesian networks are *Max* and *Noisy-Or*.

4.3 Inference

Each Bayesian logic program specifies a propositional Bayesian net, where inference can be done using standard available algorithms. The structure of the network follows from the semantics of the logic program, whereas the quantitative aspects are encoded in the conditional probability distributions associated to the nodes and in the combining rule. The stochastic variables associated to the nodes of the net are the atoms occurring in the *least Herbrand model* of the program, namely all ground atoms that are logically

entailed by the program. In order to make inference it is then necessary to provide some ground facts, *i.e.*, the truth values of some of the elements of the Herbrand base.

When the logic program is encoded as a propositional Bayesian net, any inference algorithm can be used. Inference takes the form of answering a query, and this can be done with or without available evidence. More formally, a query is the expression:

$$Q \mid E_1, \ldots, E_m$$

where Q is a random variable and $m \geqslant 0$. Answering the query means to find the conditional probability distribution:

$$Pr(Q \mid E_1, \ldots, E_m)$$

Inference without evidence Answering Q without evidence (the case $m = 0$) amounts to compute the probability distribution $Pr(Q)$ over the possible values assumed by Q. To compute this probability, it is not necessary to consider the whole network, but only the part of it containing the *relevant* nodes, *i.e.*, the nodes corresponding to the variables that influence Q. In order to answer the query (in the probabilistic sense), Kersting et al. [KDRK00] use a *pruned AND-OR tree*, which allows the probabilistic and logic computations to be combined. A pruned AND-OR tree represents all proofs of the query, because all branches leading to failures are pruned. It is interesting to note that each ground atom has a unique pruned AND-OR tree. When the same ground atom A occurs more than once in a tree, all nodes corresponding to it must be merged, so that the tree becomes an AND-OR graph.

For computing the probabilities, to each branch from an OR to an AND node the corresponding conditional probability distribution is associated. If Y is the random variable corresponding to the considered OR node, its conditional probability distribution is computed according to the combining rule for the predicate in Y.

Inference with evidence Evidence takes the form of a set of ground random variables $\{E_1, E_2, ..., E_n\}$ and their associated values $\{e_1, e_2, ..., e_n\}$. The Bayesian network necessary to compute the probability of the query random variable Q is the union of all the pruned AND-OR graphs for the facts $\{Q, E_1, E_2, ..., E_n\}$. This network can be computed incrementally, starting from the graph for Q and adding those corresponding to each piece of evidence in turn. In this way, there is the guarantee that each node occurs only once in the graph. Once the global graph is built up, any Bayesian net inference engine can be used to compute $Pr(Q \mid E_1 = e_1, ..., E_n = e_n)$.

4.4 Learning

Learning a Bayesian program may consist of two parts: learning the clauses, and learning the associated conditional probability distributions. For the first part, ILP algorithms [MdR94] could be helpful. Once the Bayesian program is learned (or given), the probability distributions must be acquired. To this aim, Kerstings and de Raedt suggest that a technique proposed by Koller and Pfeffer [KP97], based on the EM approach, could be

adapted. This algorithm makes two assumptions: (1) different data cases are independent, and (2) the combining rules are decomposable, *i.e.*, they can be expressed using a set of separate nodes corresponding to different influences, which are then combined in another node.

5 Comparison Between the two Models

In this section we will use the example introduced previously to make a comparison between the two models. Given the knowledge base $\mathcal{K}$ of Example 1, let us now associate to the rules of the knowledge base $\mathcal{K}$ some probability distributions, reported in Figures 3 and 4. In addition, it is necessary to provide a combining procedure for those rules that share the consequent. Let us assume here to use the *Max* combination rule. Moreover, let us assume, as an example, that:

$$Pr(P(a)=1)=0.4, \quad Pr(P(b)=1)=0.6, \quad Pr(Q(a)=1)=0.5, \quad Pr(Q(b)=1)=0.7$$

$P(X)$	$Q(Y)$	$R(X,Y)$	$P(X)$	$Q(Y)$	$S(X,Y)$
$true$	$true$	$(0.7,0.3)$	$true$	$true$	$(0.3,0.7)$
$true$	$false$	$(0.5,0.5)$	$true$	$false$	$(0.5,0.5)$
$false$	$true$	$(0.5,0.5)$	$false$	$true$	$(0.5,0.5)$
$false$	$false$	$(0.5,0.5)$	$false$	$false$	$(0.5,0.5)$

Fig. 3. Probability distributions associated to the the first two clauses (F_1 and F_2) in $\mathcal{K}$. The 0.5 values in the last three rows on the table indicate that there is total uncertainty about the consequence of a rule, when the premise is false.

$Q(X)$	$Q(Y)$	$V(X,Y)$	$V(X,Y)$	$R(X,Y)$
$true$	$true$	$(0.6,0.4)$	$true$	$(0.2,0.8)$
$true$	$false$	$(0.5,0.5)$		
$false$	$true$	$(0.5,0.5)$	$false$	$(0.5,0.5)$
$false$	$false$	$(0.5,0.5)$		

Fig. 4. Probability distributions associated to the second two clauses (F_3 and F_4) in $\mathcal{K}$. As in Figure 3, a false premise does not supply any information about the truth of the consequence.

If we want to represent $\mathcal{K}$ and the probability distributions with the graphical model associated to Kersting and de Raedt's Bayesian program $\mathcal{P}_{\mathcal{K}}$, we obtain the graph $\mathcal{G}_{\mathcal{P}}$ depicted in Figure 5. Actually, this graph is reported here only for illustration purposes,

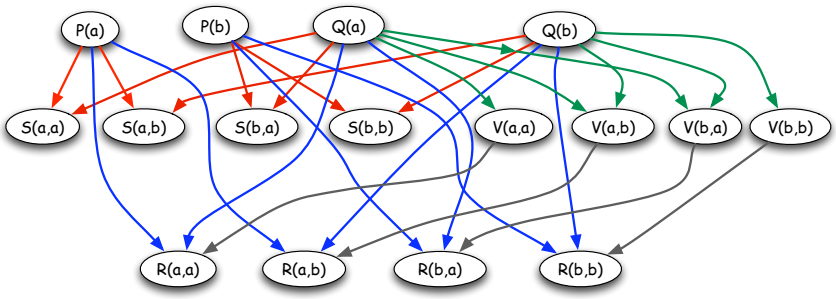

Fig. 5. Global graph associated to the Bayesian logical program reported in Example 3. As it may be seen, nodes $R(a,a), R(a,b), R(b,a)$, and $R(b,b)$ receive edges from the atoms occurring in the bodies of both rule F_1 and rule F_4.

because it is not really built up, as only the part relevant to perform a required inference is actually constructed.

As we may notice, the graph in Figure 5 has the same nodes as the one in Figure 1, but the edges (in addition of being oriented) are different. For what concerns the probability distributions, they can be embedded in the Bayesian model as they are, in the sense that the same tables, reported in Figure 3 and 4 as related to the rules of the knowledge base, can be considered associated to the corresponding edges in the graph, provided that just one node for each ground predicate occurs. The combining rule $\bigotimes$ must be applied when necessary. In our case, for instance:

$$Pr(R(a,b))|P(a),Q(b),V(a,b)) = Pr(R(a,b))|P(a),Q(b)) \bigotimes Pr(R(a,b))|V(a,b))$$

By numbering the node of the graph in Figure 5 from 1 to 16, we can write the usual joint probability distribution over all nodes:

$$Pr(X_1,...,X_{16}) = \prod_{i=1}^{16} Pr(X_i \mid par(X_i)) \tag{6}$$

Expression (6) is the analogue, in the Bayesian framework, of expression (4) in the Markovian framework, and it can be used to perform inference, even without evidence. In the Markovian framework it is not immediate to translate the probability distributions given for $\mathcal{K}$ into the joint distribution (2).

Let us now consider the problem of answering the query $\varphi_1 = R(a,b)$, given that $Pr(P(a) = 1)) = 0.4, Pr(Q(b) = 1) = 0.7, Pr(Q(a) = 1) = 0.5$. In Figure 6 the AND-OR tree associated to the query is reported. Using the probabilities defined in Figures 3 and 4, we can construct the tables in Figures 7 and 8. We define by $R_1(a,b)$ the ground predicate obtained from formula F_1, and by $R_4(a,b)$ the one obtained from formula F_4. The probability $Pr(R(a,b) = false)$ is the complement to 1 of $Pr(R(a,b) = true)$. These probabilities must be computed for each row of the table in Figure 7, and the results are reported in Figure 8.The third column is computed according to the following

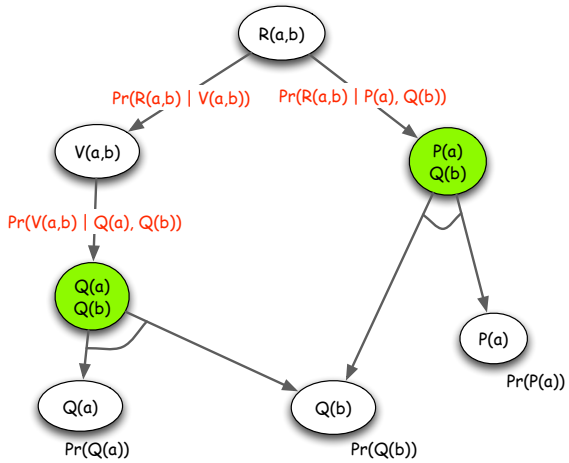

Fig. 6. AND-OR tree associate to the query $Q = R(a,b)$. In the tree, the two occurrences of the node $Q(b)$ have been merged. Nodes $[Q(a)Q(b)]$ and $[P(a)Q(b)]$ are AND nodes, *i.e.*, they are true if all the children are true. Probabilities written on the edges are conditional distributions, associated to the corresponding branches of the tree, whereas probabilities written outside the nodes are a priori probabilities, and are associated to leaves in the tree.

$P(a)$	$Q(a)$	$Q(b)$	$P(a)Q(b)$	$Q(a)Q(b)$	$V(a,b) = (true, false)$
0.4	0.5	0.7	0.28	0.35	$(0.21, 0.14)$
0.4	0.5	0.3	0.12	0.15	$(0.075, 0.075)$
0.4	0.5	0.7	0.28	0.35	$(0.175, 0.175)$
0.6	0.5	0.7	0.42	0.35	$(0.21, 0.14)$
0.6	0.5)	0.7	0.42	0.35	$(0.175, 0.175)$
0.6	0.5	0.3	0.18	0.15	$(0.075, 0.075)$
0.4	0.5	0.3	0.12	0.15	$(0.075, 0.075)$
0.6	0.5	0.3	0.18	0.15	$(0.075, 0.075)$

Fig. 7. The AND-OR tree of Figure 6 is used to compute the probability distribution of the query $R(a,b)$. In this table, the probability distribution of the random variable $V(a,b)$ is computed for each possible combination of $P(a) \in \{1,0\}, Q(a) \in \{1,0\}$, and $P(b) \in \{1,0\}$. The values reported in the first three columns of the table derive from Tables 3 and 4.

combining rule:

$$Pr(R(a,b)=1) = Max\{Pr(R_1(a,b)=1), Pr(R_4(a,b)=1)\}$$

Concerning the Markovian framework, in order to estimate the probability of $R(a,b)$,

$R_1(a,b)=(true,false)$	$R_4(a,b)=(true,false)$	$R(a,b)=true$
(0.7,0.3) 0.28	(0.32,0.68) 0.35	0.196
(0.5,0.5) 0.12	(0.35,0.65) 0.15	0.06
(0.7,0.3) 0.28	(0.35,0.65) 0.35	0.196
(0.5,0.5) 0.42	(0.32,0.68) 0.35	0.21
(0.5,0.5) 0.42	(0.35,0.65) 0.35	0.21
(0.5,0.5) 0.18	(0.35,0.65) 0.15	0.09
(0.5,0.5) 0.12	(0.35,0.65) 0.15	0.06
(0.5,0.5) 0.18	(0.35,0.65) 0.15	0.09

Fig. 8. The query $R(a,b)$ can be answered through two rules, F_1 and F_4 (see Example 1). The results from these two rules must be combined. In the third column only the probability $Pr(R(a,b)=true$ is reported.

we have to sample from its Markov blanket, knowing the state of this last (see, formula (6)). The rows in Figures 7 and 8 correspond to different states of the Markov network, depending on the truth of $P(a), Q(b)$ and $Q(a)$. Let us consider the case, as an example, of $P(a)=1, Q(b)=1$ and $Q(a)=1$, which are then the evidence. First of all, we have to extract, using the first part of Richardson and Domingos' algorithm, the relevant part of the net, which is reported in Figure 9.

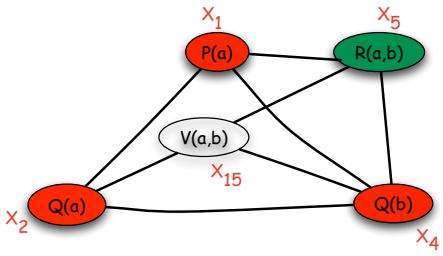

Fig. 9. Part of the Markov network of Figure 1 necessary to compute $Pr(R(a,b)|P(a),Q(a),Q(b),\mathbf{MN},\mathbf{C})$. Nodes $P(a), Q(b)$, and $Q(a)$ are evidence nodes, whereas $R(a,b)$ is the query node.

The probability distribution over the nodes in Figure 9, which is the marginal of distribution (4) with respect to the interested variables, is the following one:

$$Pr(\mathbf{X}' = \mathbf{x}') \equiv Pr(X_1 = x_1, X_2 = x_2, X_4 = x_4, X_5 = x_5, X_{15} = x_{15}) = \frac{1}{Z'} \, e^{w_1 \cdot n'_1(\mathbf{x}')} \, e^{w_3 \cdot n'_3(\mathbf{x}')} \, e^{w_4 \cdot n'_4(\mathbf{x}')},$$

where $\mathbf{x}'$ is a subvector of $\mathbf{x}$ only involving the surviving variables, and n'_i is the number of cliques that are true in the subgraph of Figure 9. In order to proceed further, we have to assume to have estimated a suitable vector of weights $\mathbf{w}$. If this is the case, we may apply Gibbs samplig to the subgraph in Figure 9, obtaining thus an estimate of $Pr(R(a,b))$, using the same procedure exemplified in Section 3.

Let us now give a last example for illustrating the differences that appear when modelling a knowledge base in both frameworks.

Example 4 (Communication Modelling).

We aim at modelling different ways of communicating between people. Let us consider the following Markov Logic Network $\mathbf{MLN} = \{(F_i, w_i) | 1 \leqslant i \leqslant 6\}$, constructed by adding weights to the rules of a knowledge base $\mathcal{K}$:

$$\begin{aligned}
(F_1, 1.8) &= phone(X,Y) :- \ friends(X,Y), communic(X,Y) \\
(F_2, 1.2) &= email(X,Y) :- \ friends(X,Y), communic(X,Y) \\
(F_3, 1.0) &= phone(X,Y) :- \ coworkers(X,Y), communic(X,Y) \\
(F_4, 2.0) &= email(X,Y) :- \ coworkers(X,Y), communic(X,Y) \\
(F_5, 2.0) &= coworkers(X,Z) :- \ coworkers(X,Y), coworkers(Y,Z) \\
(F_6, 0.8) &= friends(X,Y) :- \ coworkers(X,Y)
\end{aligned}$$

Let moreover $\mathbf{C} = \{\texttt{Ann}, \texttt{Bob}, \texttt{Tom}\}$ be a set of constants.

In the framework of Bayesian logic programs, such knowledge can be represented by the same set of formulas, but we can also use the notion of Bayesian atoms to represent it in a more concise way. We can for instance introduce a predicate *comean* taking different values *phone*, *email*, ..., leading thus to

$$\begin{aligned}
commean(X,Y) \ &| \ friends(X,Y), communic(X,Y) \\
commean(X,Y) \ &| \ coworkers(X,Y), communic(X,Y) \\
coworkers(X,Z) \ &| \ coworkers(X,Y), coworkers(Y,Z) \\
friends(X,Y) \ &| \ coworkers(X,Y)
\end{aligned}$$

A conditional probability must be associated to each of the clause. For instance, for the first clause, we could have:

$communic(Y,X)$	$friends(X,Y)$	$Pr(commean(X,Y)\|body)(phone, mail)$
$true$	$true$	$(0.7, 0.3)$
$true$	$false$	$(0.5, 0.5)$
$false$	$true$	$(0.5, 0.5)$
$false$	$false$	$(0.5, 0.5)$

There is still another way of representing this knowledge, namely by introducing a predicate *relation* that can take three values $\{friend, coworker, none\}$ The first two rules can then be written with a single rule. On the other hand, the last two rules must be generalized, thus requiring more probabilities to be given.

$$\begin{aligned} commean(X,Y) &\mid relation(X,Y), communic(X,Y) \\ relation(X,Z) &\mid relation(X,Y), relation(Y,Z) \\ relation(X,Y) &\mid relation(X,Y) \end{aligned}$$

$communic(Y,X)$	$relation(X,Y)$	$Pr(commean(X,Y)\|body)(phone,mail)$
$true$	$friend$	$(0.7,0.3)$
$true$	$coworker$	$(0.2,0.8)$
$true$	$none$	$(0.5,0.5)$
$false$	$friend$	$(0.5,0.5)$
$false$	$coworker$	$(0.5,0.5)$
$false$	$none$	$(0.5,0.5)$

$relation(X,Y)$	$relation(Y,Z)$	$Pr(relation(X,Z)\|body)(phone,mail)$
$true$	$friend$	$(0.7,0.3)$
$true$	$coworker$	$(0.2,0.8)$
$true$	$none$	$(0.5,0.5)$
$false$	$friend$	$(0.5,0.5)$
$false$	$coworker$	$(0.5,0.5)$
$false$	$none$	$(0.5,0.5)$

6 Conclusions

According to Section 4, the two models, Markov Logic Networks and Bayesian Logic Programs, are essentially different, at least for what concerns representation and inference. The Markov framework, being based on computing probabilities according to the numbers of true groundings of formulas in the knowledge base, assigns high probability to many groundings corresponding to false premises, whereas the Bayesian model suspends the judgement, giving a neutral probability assignment to these cases. For the user point of view, using Markov networks is then rather counterintuitive, because when many rules have a false premise, a drift towards neutrality (probability near 1/2 or near to some a-priori value) would be more intuitively acceptable than a high probability assignment.

In Richardson and Domingos' approach, general formulas are considered, but they do not address the semantics of negation. A weight is associated to a formula. All the instantiations of a same formula have the same weight.

In Kersting and de Raedt's approach only Datalog clauses are considered, and they distinguish two types of predicates: logical (which can be either *true* or *false*) and Bayesian (which can assume values in a given set). The definition of Bayesian atoms allows different probabilities to be assigned, depending on the values of the atoms in the bodies. On the other hand, this facility complicates the probability distribution tables.

Given a knowledge base, the Bayesian approach seems more natural and easier to be built up and understood. We may notice that the graphical models may be very large. In the Markovian approach the graph is built up and then it is reduced when evidence is given. In the Bayesian approach the complete graph is not constructed, but, adding the known facts to the knowledge base, only the relevant part of it is actually constructed.

Even though, formally, both Markov and Bayesian networks can be reduced to a common representation, important differences, at the semantic level, may direct the user to the choice of one or the other approach. One difference resides in the parameters that the user must provide, in order to build the model. In the Markov network approach, the user must provide the weight of the formulas in the knowledge base $\mathcal{K}$. Once the weight are provided, any query can be answered. As always with weights, the user may or may not be able to give useful weights; in this case, in addition, he/she may have an idea of the range of probabilities that he/she is expecting for a given query, given the evidence. To provide weights consistent with his/her expectations may be very difficult for the user. In the Bayesian approach, the user must provide the conditional probability distributions, which also may be difficult, even though these distributions are local, and hence more close to the user's understanding.

Both models can avoid asking the user for the parameters, and automated learning can be applied. In this case, the burden on the user is lifted, but still the interpretation of the results poses different challenges.

A final issue regarding the representation is the combination rule, which allows rules with the same consequent to be merged. In the Markov network approach, no explicit combination rule exists, because combination is taken into account in the assignment of values to the cliques' features. In the Bayesian model, the combination rule must be explicitely provided. Globally, the Bayesian model construction has more degrees of freedom.

Concerning inference, both models try to answer a query, given evidence. To answer a query means, in both, to compute the probability of a formula. The Bayesian approach restricts the Bayesian network to the atoms occurring in the least Herbrand model. The facts must be given in the knowledge base. In order to make inference on a given atom, all its proofs must be computed, considering Bayesian rules as logical rules and combining the associated probabilities. Both models are intractable, and must revert to approximations to make inferences. In the Markov network approach, approximate inference is done, by using Gibbs sampling on the Markov blanket of the nodes in the network. Then, the complexity of the inference depends on the size of the relevant part of the Markov network extracted from the global one by Richardson and Domingos' algorithm. In the worst case, the size os this subnetwork may coincide with that of the global one.

In the Bayesian approach it is not clear what approximations are suggested, except those that can be applied to generic Bayesian networks. Also in this case, the computational complexity of the inference is determined by the size of the network corresponding to the least Herbrand universe. The inference process, per se, has the same complexity as the best algorithm that work on Bayes nets.

Finally, learning is the more similar issue in the two approaches. In fact, learning the knowledge base can be done with an ILP learning algorithm in both cases, and then the

weights or the conditional probability distributions can be estimated from a set of data. The Bayesian approach is described as an example of the learning from interpretations framework.

Acknowledgements. Christel Vrain'work is partly supported by the ANR project 071*N*: *Semi-supervised learning of graph structure by statistical and relational approaches: towards the identification of biological networks from heterogeneous data.*

References

Cus01. J. Cussens. Statistical aspects of stochastic logic programs. In Tommi Jaakkola and Thomas Richardson, editors, *Artificial Intelligence and Statistics 2001: Proceedings of the Eighth International Workshop*, pages 181–186. Morgan Kaufmann., 2001.

DR04. P. Domingos and M. Richardson. Markov logic: A unifying framework for statistical relational learning. In *Proc. of the ICML Workshop on Statistical Relational Learning and its connections to other Fields*, 2004.

FGKP99. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI-99)*, pages 1300–1309, Stockholm, Sweden, 1999. Morgan Kaufmann.

Get06. Lise Getoor. An introduction to probabilistic graphical models for relational data. *Data Engineering Bulletin*, 29(1), march 2006.

Hal96. J.Y.. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1996.

Jae97. M. Jaeger. Relational Bayesian networks. In D. Geiger and P. P. Shenoy, editors, *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 266–273, Providence, Rhode Island, USA, 1997. Morgan Kaufmann.

Jae02. M.. Jaeger. Relational bayesian networks: a survey. *Linkping Electronic Articles in Computer and Information Science*, 6, 2002.

KD05. S. Kok and P. Domingos. Learning the structure of markov logic networks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 441–448, New York, NY, USA, 2005. ACM Press.

KdR07. K. Kersting and L. de Raedt. Bayesian logic programming: Theory and tool. MIT Press, 2007.

KDRK00. K. Kersting, L. De Raedt, and S. Kramer. Interpreting bayesian logic programs. In L. Getoor and editors D. Jensen, editors, *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data, Technical Report WS-00-06*. AAAI Press, 2000.

KP97. D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *Prpoc. IJCAI-97*, 1997.

LN89. D.C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

MdR94. S. Muggleton and L. de Raedt. Inductive logic programming:theory and methods. *J. of Logic Programming*, 19:629–679, 1994.

Mug00. S.H. Muggleton. Learning stochastic logic programs. In Lise Getoor and David Jensen, editors, *Proceedings of the AAAI2000 workshop on Learning Statistical Models from Relational Data*. AAAI, 2000.

NH97. L. Ngo and P. Haddaway. Answering queries from context-sensitive probabilistic knowledge base. *Theoretical Computer Science*, 171:147–177, 1997.

RD06. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.