

Logical Bayesian Networks

Daan Fierens, Hendrik Blockeel, Jan Ramon, and Maurice Bruynooghe

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan
200A, 3001 Leuven, Belgium
{daanf,hendrik,janr,maurice}@cs.kuleuven.ac.be

Abstract. Several models combining Bayesian networks with logic exist. The two most developed models are Probabilistic Relational Models (PRM's) and Bayesian Logic Programs (BLP's). While PRM's are easier to understand, BLP's are more expressive. However, we argue that BLP's do not always allow modeling problems intuitively. This motivates us to introduce Logical Bayesian Networks (LBN's). We argue that LBN's provide an expressive and intuitive modeling language due to explicitly distinguishing deterministic and probabilistic information and having multiple components. We briefly discuss perspectives for learning LBN's from data.

Keywords: probabilistic-logical models, Bayesian Logic Programs, Bayesian networks

1 Introduction

In this paper we introduce *Logical Bayesian Networks (LBN's)*, an instance of the approach called *Knowledge Based Model Construction* [1]. The idea is that, given a description of a specific problem, a general (probabilistic-logical) knowledge base can be used to generate a specific model. For LBN's, this specific model is a *Bayesian network* [13].

Various models combining Bayesian networks and logic¹ already exist (see the overview by Kersting and De Raedt [11]). The most developed and best known models of this kind are Probabilistic Relational Models (PRM's) by Getoor et al. [7] and Bayesian Logic Programs (BLP's) by Kersting and De Raedt [9, 10]. PRM's are easy to understand and offer an intuitive way of modeling problems. However, it has been shown in the literature that PRM's are less expressive than other models combining Bayesian networks and logic [16], including BLP's [9]. On the other hand, we will argue on the basis of an example that BLP's do not always allow modeling problems intuitively. These observations are our motivation for introducing LBN's. With LBN's we want to combine the intuitiveness of PRM's with the expressiveness of BLP's. We do this by explicitly distinguishing deterministic and probabilistic information and by having separate components to determine different elements (nodes, directed edges and conditional probability distributions (*CPD's*)) of the Bayesian networks generated, like PRM's.

¹ In this paper we use a broad interpretation of 'logic' including e.g. also relational formalisms.

We proceed as follows. In Section 2 we discuss BLP's, explain the motivation for introducing LBN's and illustrate the basic principles of LBN's. In Section 3 we more formally define LBN's. In Section 4 we discuss perspectives for learning LBN's from data. In Section 5 we briefly discuss related work. In Section 6 we conclude. We assume familiarity with the basic concepts of Bayesian networks [13] and Logic Programming [12].

2 Motivation for Introducing Logical Bayesian Networks

Consider the following running example (it is based on the 'school'-example by Getoor et al. [7]).

There are students and courses. We know which students take which courses. Each student has an IQ and each course is either interesting or not. A student taking a certain course, gets a grade for that course. Our belief about the interestingness of a course is influenced by the sum of the IQ's of all students taking the course. Our belief about a students' grade for a course is influenced by the students' IQ.

Suppose that we have a specific situation with courses *ai*, *lp* and *db*, and students *jeff* (taking *ai*), *pete* (taking *lp*) and *rick* (also taking *lp*). We can now use the general knowledge above to determine a Bayesian network modeling this specific situation. The structure of this Bayesian network is shown in Figure 1. It contains exactly the random variables we are interested in and encodes all the conditional dependencies we know about.

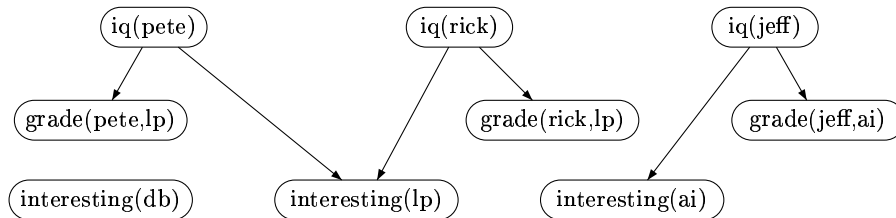


Fig. 1. The structure of the Bayesian network for our running example.

In Section 2.1 we argue, on the basis of the running example, that BLP's do not always allow to model problems intuitively. This is our motivation for introducing Logical Bayesian Networks in Section 2.2.

2.1 Bayesian Logic Programs

First we briefly explain the basics of BLP's [9, 10]. A BLP essentially compactly specifies a Bayesian network. The core of a BLP is a set of *Bayesian clauses*. An

example of a Bayesian clause is `grade(S,C) | iq(S), takes(S,C)`. Predicates used in BLP's are called *Bayesian predicates* and, unlike ordinary logical predicates, have an associated range (e.g. the predicate `iq/1` has range \mathbb{N}). Ground atoms represent random variables (e.g. `iq(jeff)`). The random variables in the Bayesian network are the ground atoms in the least Herbrand model LH of the set of Bayesian clauses (treating these clauses as pure logical clauses). The ground instances of the Bayesian clauses (with respect to LH) encode the conditional dependencies of the Bayesian network: $V \in LH$ depends on $V_p \in LH$ iff V_p is in the body of a ground instance with V in the head. Each Bayesian clause has an associated CPD quantifying the dependency of the head on the body. When there are multiple influences for the same $V \in LH$ (multiple ground instances of clauses with V in the head), a *combining rule* is used to quantify the combined influence.

We now model our running example with a BLP. The Bayesian predicates (with their range) are `student/1` {true,false}, `course/1` {true,false}, `takes/2` {true,false}, `iq/1` \mathbb{N} , `interesting/1` {interesting, uninteresting} and `grade/2` {0,1, ..., 20}. The information about the specific situation considered, can only be represented by the following Bayesian ground facts (these Bayesian facts each have an associated probability distribution (CPD), but we do not show this here).

```
student(pete).      student(rick).      student(jeff).
course(ai).         course(lp).         course(db).
takes(jeff,ai).    takes(pete,lp).    takes(rick,lp).
```

Adding these Bayesian facts to the BLP implies that `student(pete)`, `course(ai)`, ... will all be random variables in the Bayesian network determined by the BLP (since they are all in LH). To represent our general information about `iq/1`, `interesting/1` and `grade/2`, we need the following Bayesian clauses.

```
iq(S) | student(S).
interesting(C) | takes(S,C), iq(S).
interesting(C) | course(C).
grade(S,C) | iq(S), takes(S,C).
```

The first clause guarantees that there is a random variable `iq(S)` in the Bayesian network for every student S . The second clause expresses that “whether a course is interesting depends on the IQ’s of all students taking the course”. The third clause guarantees that there is a random variable `interesting(C)` for every course C (even for a course that is not taken, such as `db`). The fourth clause expresses that “a students’ grade for a course depends on the students’ IQ” (the atom `takes(S,C)` in the body is needed to guarantee that there only is a random variable `grade(S,C)` if student S is taking course C).

There are some disadvantages about the way this BLP models the example.

First, although this BLP seems the most appropriate for our example, it does not generate the Bayesian network that we wanted to obtain (see Figure 1). The Bayesian network for the above BLP contains random variables like `student(jeff)` and `takes(jeff,ai)`. This is strange since we wanted to

model that we are certain that `student(jeff)` and `takes(jeff,ai)` are true and, more important, since the same Bayesian network also contains a random variable `grade(jeff,ai)` that is only meaningful if `student(jeff)` and `takes(jeff,ai)` are true (and not if they are false). In general, BLP’s put deterministic, structural knowledge (such as `takes(jeff,ai)`) inside the Bayesian networks, due to the fact that they do not have ordinary logical predicates to express this kind of knowledge. Note that the Bayesian network that we wanted to obtain (see Figure 1), does not contain random variables like `student(jeff)` or `takes(jeff,ai)`.

Second, the clauses in the above BLP are difficult to interpret. We mean that it is hard to reconstruct from this set of clauses the original description of the running example. This is again because probabilistic and deterministic, structural information are mixed (such as in the clause `grade(S,C) | iq(S), takes(S,C)`) and, also, because some clauses convey information about the dependencies in the Bayesian network (e.g. `interesting(C) | takes(S,C), iq(S)`), some about the random variables (e.g. `iq(S) | student(S)`) and some about both (e.g. `grade(S,C) | iq(S), takes(S,C)`). Of course for larger, real-world problems, interpreting BLP’s only becomes harder.

2.2 Logical Bayesian Networks

The above observations motivated us to develop Logical Bayesian Networks (LBN’s). We start by explaining LBN’s informally. A definition of LBN’s is given in Section 3.

Like most related models, a LBN contains general, probabilistic-logical knowledge. Given a description of the *domain of discourse* for a specific problem (e.g. information about the courses taken by students), a LBN generates a Bayesian network.

Section 2.1 illustrated the need to explicitly distinguish deterministic, structural information and probabilistic information (an idea that can be traced back to Ngo and Haddawy [14]). To do so, LBN’s use two disjunct sets of predicates: the set of *logical predicates* and the set of *probabilistic predicates*. Logical predicates are used to specify the domain of discourse for a specific problem. This is supposed to be deterministic information and can be specified as a normal logic program [12] P_l for the logical predicates. Probabilistic predicates in LBN’s, like predicates in BLP’s, have an associated range and are used to represent random variables (a random variable is a ground atom built from a probabilistic predicate and has a range equal to the range of that predicate).

To get an intuitive modeling language, we believe it is also important that LBN’s, following PRM’s, have separate components to determine different elements of the Bayesian networks generated. As illustrated in Figure 2, a LBN consist of three components: \mathcal{V} determines the random variables, \mathcal{DE} determines the conditional dependencies (more precisely, the directed edges) and \mathcal{DI} determines the conditional probability distributions.

We now show how our running example can be modeled with a LBN. For now, we are mainly interested in *qualitative* probabilistic knowledge. Hence, we

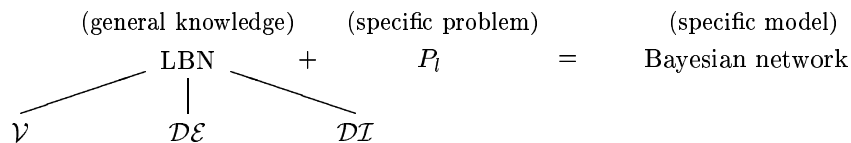


Fig. 2. Together a LBN (containing general knowledge) and a normal logic program P_l (describing a specific problem) determine a Bayesian network.

delay the discussion of the component \mathcal{DI} (containing *quantitative* probabilistic knowledge) to Section 3.

We need the logical predicates `student/1`, `course/1` and `takes/2`. The probabilistic predicates (with their range) are `iq/1` \mathbb{N} , `interesting/1` {interesting, uninteresting} and `grade/2` {0,1, ..., 20}. The logic program P_l describing the specific problem is the following.

```

student(pete).      student(rick).      student(jeff).
course(ai).         course(lp).         course(db).
takes(jeff,ai).    takes(pete,lp).    takes(rick,lp).

```

Here P_l is very simple and consists of ground facts only. In general, P_l can be any logic program for the logical predicates. The component \mathcal{V} contains the following clauses.

```

iq(S) <= student(S).
interesting(C) <= course(C).
grade(S,C) <= takes(S,C).

```

\mathcal{V} determines the random variables in the Bayesian network that is generated for P_l . For instance, the clause `iq(S) <= student(S)` states that “for every student S , there is a random variable `iq(S)` in the Bayesian network”. The component \mathcal{DE} contains the following clauses.

```

interesting(C) | iq(S) <- takes(S,C).
grade(S,C) | iq(S).

```

\mathcal{DE} determines the conditional dependencies in the Bayesian network. The first clause states that “whether a course is interesting depends on the IQ of a student, if the student takes that course”. The second clause states that “the grade of a student for a course depends on the students’ IQ”.

The Bayesian network generated by the above LBN for this P_l has the structure shown in Figure 1 (i.e. the structure that we wanted to obtain for our running example).

The above LBN seems to be a more intuitive model of our running example than the BLP of Section 2.1. As desired, the Bayesian network generated by the LBN does not contain random variables like `student(jeff)`. Also, the clauses in the LBN are easy to interpret and there is a close correspondence between these clauses and the original problem description. The relationship of the clauses in

the BLP with the original description is a lot more obscure. We observed this also in other applications.

An additional advantage of LBN's is that they have non-monotonic negation (for logical predicates). This is useful for specifying the set of random variables. Suppose that we want $\text{grade}(S, C)$ to be a random variable only if student S takes course C and S is not absent from the exam of C . We can simply write the clause $\text{grade}(S, C) \leftarrow \text{takes}(S, C), \text{not}(\text{absent}(S, C))$ in \mathcal{V} . This has no direct counterpart in BLP's as there $\text{absent}(S, C)$ is a Bayesian atom that cannot be negated.

Finally, note as a side-remark that a Bayesian network generated by a LBN can of course be used for answering probabilistic queries about random variables in the network. In a similar vein as for BLP's, the *support network* (i.e. the subnetwork of the entire network that is sufficient to answer the query) can be determined based on the clauses in \mathcal{V} and \mathcal{DE} . For details we refer to the discussion for BLP's by Kersting and De Raedt [9].

3 Defining Logical Bayesian Networks

We define a LBN as a tuple $(\Pi_l, \Pi_p, \mathcal{V}, \mathcal{DE}, \mathcal{DI})$, where Π_l is the set of logical predicates and Π_p is the set of probabilistic predicates (having an associated range)². The semantics of a LBN is that it generates a Bayesian network when supplied with a normal logic program P_l defining the predicates in Π_l . As explained in Section 2.2, P_l describes the domain of discourse for a specific problem. We use the *well-founded semantics* [18]: every normal logic program P_l has a unique well-founded model $WFM(P_l)$.

In Section 3.1 we define the syntax and semantics of the components \mathcal{V} , \mathcal{DE} and \mathcal{DI} , ignoring consistency issues (i.e. conditions that have to be satisfied for the generated Bayesian network to be well-defined). We discuss consistency separately in Section 3.2.

3.1 Defining the Components \mathcal{V} , \mathcal{DE} and \mathcal{DI}

We simply call an atom built from a probabilistic predicate a *probabilistic atom*. Similarly we talk about *logical atoms* and *logical literals*. Remember that a random variable is represented as a ground probabilistic atom. We use $\text{range}(V)$ to denote the range of a probabilistic atom V and $\text{pred}(V)$ to denote the predicate that V is built from. To simplify notation, we use the LBN and P_l in question as implicit arguments of the definitions.

The Random Variables: \mathcal{V}

² A LBN also uses a set of functor symbols and constants, but this is left implicit in the definitions as this is standard in Logic Programming [12].

Definition 1 (\mathcal{V}). \mathcal{V} is a set of range-restricted clauses of the form $pAtom \leq lit_1, \dots, lit_n$, where $n \geq 0$, $pAtom$ is a probabilistic atom and lit_1, \dots, lit_n are logical literals (we call $pAtom$ the head of the clause and lit_1, \dots, lit_n the body of the clause).

A clause is *range-restricted* iff all free variables that occur in the head also occur in a positive literal in the body. As for the semantics, \mathcal{V} determines the set of random variables RV in the Bayesian network generated for a given P_l .

Definition 2 (RV). The set of random variables, RV , is the set of all ground probabilistic atoms that are true in $WFM(P_l \cup \mathcal{V})$, where the clauses in \mathcal{V} are interpreted as standard logic program clauses.

For our running example, RV is the set $\{iq(jeff), iq(pete), iq(rick), interesting(ai), interesting(lp), interesting(db), grade(jeff, ai), grade(pete, lp), grade(rick, lp)\}$.

The Directed Edges: \mathcal{DE}

Definition 3 (\mathcal{DE}). \mathcal{DE} is a set of clauses of the form $pAtom \mid pAtom_1, \dots, pAtom_n \leftarrow lit_1, \dots, lit_m$, where $n, m \geq 0$, $pAtom, pAtom_1, \dots, pAtom_n$ are probabilistic atoms and lit_1, \dots, lit_m are logical literals (we call $pAtom$ the head, $pAtom_1, \dots, pAtom_n$ the body and lit_1, \dots, lit_m the context of the clause).

If the context of a clause is empty, we write the clause as $pAtom \mid pAtom_1, \dots, pAtom_n$. As for the semantics, \mathcal{DE} defines the conditional dependencies (more precisely, the directed edges) of the Bayesian network generated for a given P_l . We can exploit the context to ensure that the network contains only conditional dependencies relevant for the specific problem at hand.

Definition 4 ($Edges$). The set of directed edges, $Edges$, is $\{(V_p, V) \mid \exists a \text{ ground instance } V \mid \text{body} \leftarrow \text{context of a clause in } \mathcal{DE} \text{ such that } V_p \in \text{body and context is true in } WFM(P_l) \text{ and } V, V_p \in RV\}$.

Together RV and $Edges$ constitute the structure of a Bayesian network (a directed graph). This directed graph has a node for each $R \in RV$ and an edge from V_p to V for each $(V_p, V) \in Edges$. For our running example, this directed graph is shown in Figure 1 (Section 2). Below, we need the well-known notion of *parents* of a node in a directed graph. We use $Nb(V)$ to denote the number of parents of $V \in RV$. We assume that we have an (arbitrary) order³ on the parents of a node and use $Parent_i(V)$ to denote the i -th parent of $V \in RV$. For our running example, $Parent_1(interesting(lp))$ is $iq(pete)$ and $Parent_2(interesting(lp))$ is $iq(rick)$.

³ Here we use the alphabetical order, considering atoms as strings.

The Conditional Probability Distributions (CPD's): \mathcal{DI} To complete the generated Bayesian network, we need a CPD for each random variable $V \in RV$. Such a CPD is a function mapping any tuple in $range(Parent_1(V)) \times \dots \times range(Parent_{Nb(V)}(V))$ to a probability distribution on $range(V)$. For instance, a possible CPD for the random variable `interesting(1p)` is the following function⁴.

If $iq(pete) + iq(rick) > 1000$ then 0.7/0.3 else 0.5/0.5 .

However, we cannot define all CPD's individually, be it only that the set of random variables depends on the specific problem instance P_I . We need a mechanism to define a set of CPD's at once. To this end we define a *logical CPD* for each probabilistic predicate.

Definition 5 (Logical CPD). *A logical CPD for a probabilistic predicate p is a function mapping any set of pairs (Var, Val) to a probability distribution on the range of p , where Var is a ground probabilistic atom built from a predicate occurring in the body of a clause in \mathcal{DE} with p in the head and $Val \in range(Var)$.*

An example of a logical CPD for `interesting/1` is the following function (IN stands for the input of the function, a set of pairs according to Definition 5).

If $\sum_{(iq(S), Val) \in IN} Val > 1000$ then 0.7/0.3 else 0.5/0.5 .

As will become clear below, this should be read as: "if the sum of the IQ's of all students taking a certain course is greater than 1000, there is a probability of 0.7/0.3 that the course is interesting/uninteresting, otherwise this is 0.5/0.5".

Logical CPD's in LBN's essentially have the same function as *combining rules* [9] in BLP's. Logical CPD's can be specified in various ways, for instance in Prolog or by *logical decision trees*⁵ such as used by the TILDE-system [3]. Because of space restrictions we do not elaborate this further and express logical CPD's using a kind of pseudo-code as above.

\mathcal{DI} is now simply defined as follows.

Definition 6 (\mathcal{DI}). *\mathcal{DI} is a set containing, for each probabilistic predicate $p \in \Pi_p$, one logical CPD for p .*

We use $LCPD_p$ to denote the logical CPD for p in \mathcal{DI} . It can be used to determine the CPD for any random variable $V \in RV$ for which $pred(V)$ is p .

Definition 7 ($CPD(V)$). *The CPD, $CPD(V)$, for a ground probabilistic atom $V \in RV$ is the function mapping any $(Val_1, \dots, Val_{Nb(V)}) \in range(Parent_1(V)) \times \dots \times range(Parent_{Nb(V)}(V))$ to the probability distribution obtained by applying $LCPD_{pred(V)}$ to the set $\{(Parent_1(V), Val_1), \dots, (Parent_{Nb(V)}(V), Val_{Nb(V)})\}$.*

⁴ Remember that the range of `interesting/1` is {interesting, uninteresting}.

⁵ Friedman et al. [6] show how to represent ordinary CPD's as decision trees. By upgrading this approach to first order logic, we could represent logical CPD's as logical decision trees. In the case of the above logical CPD for `interesting/1`, we have to use an extension of logical decision trees with aggregates [20].

The CPD for `interesting(lp)` is a function of `iq(pete)` and `iq(rick)`. Suppose, for instance, that we want to determine the probability distribution specified by this CPD when `iq(pete)` is 100 and `iq(rick)` is 120. By applying the logical CPD for `interesting/1` to the set $\{(iq(pete),100), (iq(rick),120)\}$ we get as a result 0.5/0.5. Like this we can obtain the entire CPD for `interesting(lp)`. This is the following function.

If $iq(pete) + iq(rick) > 1000$ then 0.7/0.3 else 0.5/0.5 .

3.2 The Generated Bayesian Network

Some conditions have to be satisfied in order for a LBN to generate a well-defined Bayesian network (i.e. a Bayesian network that specifies a unique probability measure), given P_l .

Proposition 1. *If the set of random variables RV is non-empty, the directed graph defined by RV and Edges is acyclic and all random variables have a finite number of ancestors in this directed graph, then the Bayesian network determined by RV , Edges and $CPD(V)$ for each $V \in RV$, is well-defined.*

4 Perspectives for Learning LBN's

The main aim of this paper is to introduce LBN's as a model for knowledge representation. Future work includes developing algorithms for learning LBN's from data. Here we already discuss a high-level strategy. Following the approach for BLP's [10], we use concepts from learning Bayesian networks [13] and Inductive Logic Programming (ILP) [5].

A LBN can be learned from a set of *data cases* each consisting of a normal logic program P_l and a set of assignments of values to random variables. In practice, \mathcal{V} will often be known in advance (when learning from a relational database, for instance, \mathcal{V} corresponds to the relational schema). Otherwise, \mathcal{V} can be learned using traditional, non-probabilistic ILP-algorithms (e.g. CLAUDIEN [10]). Learning \mathcal{DI} is nested inside learning \mathcal{DE} as follows. When \mathcal{DE} is known, the logical CPD's in \mathcal{DI} can be learned in a number of ways, for instance using a logical decision tree learner (e.g. TILDE [3]). To learn the clauses in \mathcal{DE} , we can perform local search using refinement operators [5] to find a set of clauses with a high score (e.g. likelihood [13]). To compute such a score, the optimal logical CPD's (i.e. \mathcal{DI}) for this set of clauses \mathcal{DE} will typically have to be learned.

An algorithm for learning BLP's has been described by Kersting and De Raedt [10]. LBN's have some advantages over BLP's with respect to learnability due to the differences in representation, although it is future work to verify this empirically. First, in LBN's, \mathcal{V} and \mathcal{DE} can be learned separately using respectively traditional, non-probabilistic ILP-techniques and probabilistic techniques (if \mathcal{V} needs to be learned at all). In BLP's this distinction cannot be made since the information from \mathcal{V} and \mathcal{DE} is contained in a single component. Second, LBN's explicit distinction between probabilistic and logical predicates

can also facilitate learning. Knowledge about whether a predicate is probabilistic or logical can be used as language bias reducing the hypothesis space, both for learning the logical CPD's in \mathcal{DI} and for learning the clauses in \mathcal{DE} . In addition, features such as determinacy can often be specified as bias for logical predicates, which is usually more difficult for probabilistic predicates. Note that, according to personal communication with Kristian Kersting, the implementation of BLP's distinguishes probabilistic and logical predicates. This is not described in the literature about BLP's sufficiently to know how this works. Hence, we cannot discuss this further.

5 Related Work

A variety of models combining logic and probabilities exists (see the overview by Kersting and De Raedt [11]). Below, we very briefly mention the most important models and discuss some relations with LBN's. A more detailed study of this matter is future work.

On the one hand are models adding probabilities to Logic Programming [12], like Independent Choice Logic by Poole [15], PRISM by Sato and Kameya [17], Logic Programs with Annotated Disjunctions by Vennekens et al. [19] and Stochastic Logic Programs by Muggleton and Cussens [4]. On the other hand are models combining Bayesian networks with logic. Models of this kind have a stronger relationship to LBN's. The most important of these models are BLP's, PRM's and Probabilistic Logic Programs (PLP's) by Ngo and Haddawy [14]. Some others are $CLP(\mathcal{BN})$ by Santos Costa et al. [16] and Relational Bayesian Networks by Jaeger [8].

LBN's are strongly related to PRM's in that there is a clear correspondence in functionality between the components of a LBN and that of a PRM: \mathcal{V} corresponds to the relational schema, \mathcal{DE} to the dependency structure, \mathcal{DI} to the CPD's and aggregates for the dependency structure and P_i to the relational skeleton (for an explanation of these terms, we refer to Getoor et al. [7]). This guarantees that it is as simple to model problems with LBN's as with PRM's. Moreover, as most other probabilistic-logical models [9, 16], LBN's are more expressive than PRM's. For instance, Blockeel and Bruynooghe [2] show that PRM's cannot model certain dependencies. Also, PRM's do not have functor symbols (useful to model temporal processes, for instance).

LBN's are also strongly related to PLP's (also known as Context-Sensitive Probabilistic Knowledge Bases). PLP's also distinguish deterministic information and probabilistic information. However, LBN's have a number of advantages over PLP's, which are mainly the same as the advantages of BLP's over PLP's stated by Kersting and De Raedt [9]: the ability to handle continuous variables, the separation of qualitative and quantitative information and the closer link to ordinary Bayesian networks.

6 Conclusions

We introduced a new probabilistic-logical model called Logical Bayesian Networks. LBN's are strongly related to other models combining Bayesian networks and logic, most prominently BLP's. LBN's explicitly distinguish probabilistic and deterministic information and have separate components to determine different elements of the Bayesian networks generated. We argued that, as a consequence, LBN's are often more intuitive than BLP's.

Some extensions of LBN's (not discussed due to space restrictions) are the incorporation of logical background knowledge and a more structured representation of logical CPD's. A lot of future work remains. This includes implementing and testing a complete learning algorithm for LBN's and investigating in detail the relations of LBN's with other probabilistic-logical models, with respect to both knowledge representation and learning.

Acknowledgements

Daan Fierens is supported by the GOA/2003/08(B0516) on Inductive Knowledge Bases. Hendrik Blockeel and Jan Ramon are post-doctoral fellows of the Fund for Scientific Research (FWO) of Flanders. The authors would like to thank Kristian Kersting for interesting discussions and the reviewers for useful comments.

References

- [1] F. Bacchus. Using first-order probability logic for the construction of Bayesian networks. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 219–226, 1993.
- [2] H. Blockeel and M. Bruynooghe. Aggregation versus selection bias, and relational neural networks. In *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data, SRL-2003, Acapulco, Mexico, August 11, 2003*, 2003.
- [3] H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
- [4] J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
- [5] P. A. Flach. The logic of learning: a brief introduction to Inductive Logic Programming. In *Proceedings of the CompulogNet Area Meeting on Computational Logic and Machine Learning*, pages 1–17. University of Manchester, 1998.
- [6] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Learning in graphical models*, pages 421–459. MIT Press, 1999.
- [7] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, pages 307–334. Springer-Verlag, 2001.
- [8] M. Jaeger. Relational Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 266–273. Morgan Kaufmann Publishers, 1997.
- [9] K. Kersting and L. De Raedt. Bayesian logic programs. Technical Report 151, Institute for Computer Science, University of Freiburg, Germany, April 2001.

- [10] K. Kersting and L. De Raedt. Towards combining inductive logic programming and Bayesian networks. In *Proceedings of the 11th international conference on Inductive Logic Programming*, pages 118–131, 2001.
- [11] K. Kersting and L. De Raedt. Probabilistic logic learning. In S. Dzeroski and L. De Raedt, editors, *SIGKDD Explorations, special issue on Multi-Relational Data Mining*, volume 5(1), pages 31–48, 2003.
- [12] J. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd edition, 1987.
- [13] R. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, New Jersey, 2003.
- [14] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171(1–2):147–177, 1997.
- [15] D. Poole. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1–2):5–56, 1997.
- [16] V. Santos Costa, D. Page, M. Qazi, and J. Cussens. CLP(BN): Constraint logic programming for probabilistic knowledge. In *Proceedings of 19th Conference on Uncertainty in Artificial Intelligence*, 2003.
- [17] T. Sato and Y. Kameya. PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI97)*, pages 1330–1335, 1997.
- [18] A. Van Gelder, K. Ross, and J. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3), 1991.
- [19] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *Proceedings of the 20th International Conference on Logic Programming*, 2004.
- [20] C. Vens, A. Van Assche, H. Blockeel, and S. Dzeroski. First order random forests with complex aggregates. In *Proceedings of 14th International Conference on Inductive Logic Programming, Porto, Portugal*, 2004.