

Logical Bayesian Networks and Their Relation to Other Probabilistic Logical Models

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

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan
200A, 3001 Leuven, Belgium

{daanf, hendrik, maurice, janr}@cs.kuleuven.ac.be

Abstract. Logical Bayesian Networks (LBNs) have recently been introduced as another language for knowledge based model construction of Bayesian networks, besides existing languages such as Probabilistic Relational Models (PRMs) and Bayesian Logic Programs (BLPs). The original description of LBNs introduces them as a variant of BLPs and discusses the differences with BLPs but still leaves room for a deeper discussion of the relationship between LBNs and BLPs. Also the relationship to PRMs was not treated in much detail.

In this paper, we first give a more compact and clear definition of LBNs. Next, we describe in more detail how PRMs and BLPs relate to LBNs. Like this we not only see what the advantages and disadvantages of LBNs are with respect to PRMs and BLPs, we also gain more insight into the relationships between PRMs and BLPs.

Keywords: Probabilistic-logical models, Bayesian networks, knowledge representation, Bayesian Logic Programs, Probabilistic Relational Models.

1 Introduction

Probabilistic logical models are models combining aspects of probability theory with aspects of Logic Programming, first-order logic, or relational languages. In recent years a variety of such models has been introduced in the literature (see the overview by Kersting and De Raedt [26]). An important class of such models are those based on the principle of *Knowledge Based Model Construction* (KBMC) [2]. The idea of KBMC is that a general probabilistic logical knowledge base can be used to generate a specific propositional probabilistic model (when given a specific problem). We focus on the case where the propositional model is a Bayesian network [32]. The most developed and best known models of this kind are Probabilistic Relational Models by Getoor et al. [14] and Bayesian Logic Programs by Kersting and De Raedt [23, 24].

We recently introduced Logical Bayesian Networks (LBNs) as yet another model for knowledge based model construction of Bayesian networks [12]. In the original description, we introduced LBNs as a variant of BLPs. In designing LBNs, focus was specifically on introducing all necessary language components

to make knowledge representation with LBNs as simple as possible. First, LBNs cleanly separate deterministic, logical knowledge and probabilistic knowledge. Second, LBNs have different language components to determine different parts of a Bayesian network (the nodes in the graph, the directed edges and the conditional probability distributions).

In this paper, we first give a new, more compact and clear but essentially equivalent definition of LBNs. Next, we compare LBNs with PRMs, which was done only very briefly in [12]. Then we compare LBNs to BLPs. We approach this comparison differently than in [12] by explicitly using LBNs as a reference point and go more into detail. Such a comparison not only teaches us more about LBNs, but also about the mutual relations between PRMs and BLPs.

For several probabilistic logical models techniques for learning from data have been developed. At recent ILP conferences a substantial number of papers (and invited lectures) have been presented on this topic (e.g. [24, 22, 37, 11]). Our paper complements this work in that we do not discuss learning directly, but focus on the knowledge representation used by the different learning systems.

We proceed as follows. In Section 2 we review LBNs, give a new, more compact definition of LBNs and discuss the methodology behind their design. In Section 3 we compare LBNs with Probabilistic Relational Models and Bayesian Logic Programs. In Section 4 we conclude. We assume familiarity with the basic concepts of Bayesian networks [32] and Logic Programming [29].

2 Logical Bayesian Networks

We review Logical Bayesian Networks (LBNs) [12] by means of an example. Then we formally define the syntax and declarative semantics of LBNs. Finally, we discuss the methodology behind the design of LBNs.

2.1 Logical Bayesian Networks by Example

Consider the following running example (based on the ‘university’-example by Getoor et al. [14]).

There are students and courses. We know which students take which courses. Each student has an IQ and a final ranking and each course has a difficulty level. A student taking a certain course, gets a grade for that course. The grade of a student for a course depends on the IQ of the student and the difficulty of the course. The final ranking of a student depends on his grades for all the courses he’s taking.

LBNs explicitly distinguish deterministic, logical knowledge and probabilistic knowledge. To do so, LBNs use two disjoint sets of predicates: the set of *logical predicates* and the set of *probabilistic predicates* (an idea introduced by Ngo and Haddawy [33]). Logical predicates are used to specify logical background knowledge describing the domain of discourse for the world considered (this is supposed to be deterministic information). Probabilistic predicates in LBNs (like

predicates in Bayesian Logic Programs [23]) have an associated range and are used to represent random variables. Precisely, a random variable is represented as a ground atom built from a probabilistic predicate and has a range equal to the range of that predicate. Note that it is debatable whether ‘predicates’ is the right name since these ‘predicates’ behave more like (typed) functors than like ordinary predicates (similarly logical atoms behave more like terms than like ordinary atoms). The main reason for calling them predicates is because like this we stay in line with the terminology of Bayesian Logic Programs (since we introduced LBNs as a variant of Bayesian Logic Programs we believe this to be important).

LBNs have four components. The first one is a set of clauses called the *random variable declarations*. The second one is a set of clauses called the *conditional dependency clauses*. The third one is a set of *logical Conditional Probability Distributions (logical CPDs)*, quantifying the conditional dependencies determined by the conditional dependency clauses. The fourth one is a set of normal logic clauses for the logical predicates used to specify deterministic background information.

We now illustrate some of these notions on our running example. The logical predicates are *student/1*, *course/1* and *takes/2*, the probabilistic predicates are *iq/1*, *diff/1*, *ranking/1* and *grade/2* (having as associated range for example respectively {low,high}, {low,middle,high}, {A,B,C} and {A,B,C}). The random variable declarations are:

```
random(iq(S)) <- student(S).
random(ranking(S)) <- student(S).
random(diff(C)) <- course(C).
random(grade(S,C)) <- takes(S,C).
```

Here *random/1* is a special-purpose logical predicate. The first clause, for instance, should be read as: “*iq(S)* is a random variable if *S* is a student”. The conditional dependency clauses are:

```
ranking(S) | grade(S,C) <- takes(S,C).
grade(S,C) | iq(S), diff(C).
```

The first clause should be read as: “the ranking of *S* depends on the grade of *S* for *C* if *S* takes *C*” and the second “the grade of *S* for *C* depends on the iq of *S* and the difficulty of *C*”. We do not mention anything about the logical CPDs here, leaving this issue for the next section.

The semantics of a LBN is that it defines a mapping from specific problems (or worlds) to Bayesian networks. We use a normal logic program [29] to describe the specific problem. For our running example this could look as follows (the meaning is obvious):

```
student(john).      student(pete).
course(ai).         course(db).
takes(john,ai).    takes(john,db).    takes(pete,ai).
```

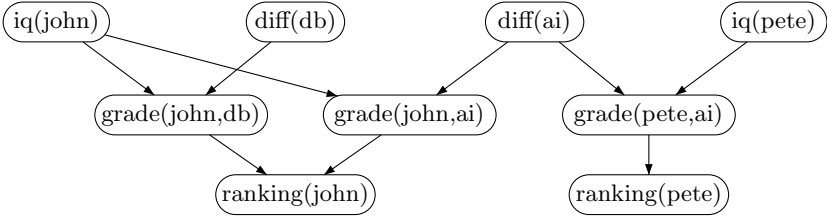


Fig. 1. The structure of the Bayesian network induced for our running example

The structure of the Bayesian network induced by the above LBN given this logic program is shown in Figure 1.

2.2 Syntax of Logical Bayesian Networks

We now define the syntax of LBNs. In the next section we define the semantics.

Remember that LBNs use two disjoint sets of predicates: the logical predicates and the probabilistic predicates having an associated range (we use these sets implicitly in our definitions). We call an atom built from a probabilistic predicate a *probabilistic atom* (it has the same range as the predicate). Similarly we talk about *logical atoms* and *logical literals*. Remember that a random variable is represented as a ground probabilistic atom.

Definition 1 (random variable declaration). A random variable declaration is a range-restricted clause of the form

$$\text{random}(pAtom) \leftarrow lit_1, \dots, lit_n.$$

where $n \geq 0$, $pAtom$ is a probabilistic atom and lit_1, \dots, lit_n are logical literals.

A clause is *range-restricted* iff all free variables that occur in the head also occur in a positive literal in the body.

Definition 2 (conditional dependency clause). A conditional dependency clause is a clause 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.

As will become clear in the next section, these clauses need not be range-restricted. If $m = 0$, we write the clause as $pAtom \mid pAtom_1, \dots, pAtom_n$.

Definition 3 (logical CPD). A logical CPD for a probabilistic predicate p is a function mapping a set of ground probabilistic atoms to a conditional probability distribution on the range of p .

When referring to the logical CPD for a ground probabilistic atom, we mean the logical CPD for the predicate that atom is built from.

Logical CPDs in LBNs play the same role as *combining rules* in Bayesian Logic Programs [23]. This means that a logical CPD not only quantifies a dependency indicated by a single conditional dependency clause but also combines the influences of multiple conditional dependency clauses with the same head.

In [12] we argued that one way to specify a logical CPD is as a *logical decision tree* [5, 43]. Working this out in detail is beyond the scope of this paper.

Definition 4 (Logical Bayesian Network). *A Logical Bayesian Network is a tuple $(\mathcal{V}, \mathcal{D}, \mathcal{B}, \mathcal{L})$ with \mathcal{V} a set of random variable declarations, \mathcal{D} a set of conditional dependency clauses, \mathcal{B} a set of normal logic clauses for the logical predicates and \mathcal{L} a set of logical CPDs, one for each probabilistic predicate.*

The above definitions differ slightly from the original definitions in [12]. First, we use a slightly different notation for the random variable declarations (using *random/1*). Second, we explicitly introduced in our definitions the normal clauses \mathcal{B} describing deterministic background knowledge (in [12] this was left implicit). Third, we tried to make the definition of logical CPD easier.

2.3 Declarative Semantics of Logical Bayesian Networks

The semantics of a LBN is that it defines a mapping from specific problems or worlds (described by a normal logic program P_l defining the logical predicates) to Bayesian networks. In other words, a LBN induces a ground Bayesian network. We use the *well-founded semantics* [41]: every normal logic program P_l has a unique well-founded model $WFM(P_l)$ (for a program without negation, this semantics is equivalent to the least Herbrand semantics).

Definition 5 (Induced Bayesian Network). *The Bayesian Network induced by a LBN $(\mathcal{V}, \mathcal{D}, \mathcal{B}, \mathcal{L})$ given a normal logic program P_l is the Bayesian network determined by the directed graph containing*

- a node (random variable) V iff V is a ground probabilistic atom and random (V) is true in $WFM(P_l \cup \mathcal{B} \cup \mathcal{V})$,
- an edge from a node V_{parent} in the graph to a node V_{child} in the graph iff there is a ground instance $V_{child} \mid body \leftarrow context$. of a clause in \mathcal{D} such that $V_{parent} \in body$ and $context$ is true in $WFM(P_l \cup \mathcal{B})$,

and where the CPD for a node V is obtained by applying the logical CPD for V in \mathcal{L} to the set of ground probabilistic atoms that are parents of V in the graph.

Obviously the Bayesian network induced by a LBN given a logic program P_l is only well-defined (i.e. specifies a unique probability measure) under certain conditions¹.

¹ These conditions are similar to the conditions for a Bayesian Logic Program to be well-defined, see [23].

Proposition 1. *The Bayesian network induced by a LBN given P_l is well-defined iff the directed graph induced is non-empty and acyclic, each node in the graph has a finite number of ancestors and the CPD associated to each node is conditioned only on the parents of that node.*

2.4 Discussion

The language of LBNs was designed from the point of view of knowledge representation. We explicitly tried to unravel the different types of knowledge that one might want to represent and tried to reflect these different types of knowledge in the different components of LBNs. This can be seen on two levels.

First, as LBNs define a mapping from specific worlds to Bayesian networks and a Bayesian network is determined by its nodes, directed edges and CPDs, LBNs have a first component to determine the nodes, a another one to project a set of directed edges on these nodes and yet another one to determine the CPDs.

Second and more general, LBNs explicitly distinguish deterministic and probabilistic knowledge (under the form of two sets of predicates). In Section 3.2 we go into detail about the problems that arise when this distinction is not made.

LBNs have a number of advantages as compared to models offering a language as ‘uniform’ as possible (i.e. with as few language components as possible, as was for instance the original motivation behind Bayesian Logic Programs [27, 21]). First, LBNs are very easy to understand. Second, as argued in [12] knowledge representation with LBNs is very easy. Third, LBNs can be used to gain insight into other probabilistic logical models by investigating how the language components of these models map to the components of LBNs. We illustrate this last point in the next section.

3 Comparing Logical Bayesian Networks to Other Probabilistic Logical Models

We now compare LBNs to Probabilistic Relational Models (Section 3.1) and Bayesian Logic Programs (Section 3.2). We also briefly review other related models (Section 3.3).

3.1 Probabilistic Relational Models

Introduction. Probabilistic Relational Models (PRMs) [13, 15, 14, 16] are based on the entity-relationship model and consist of three components. The relational schema describes the set of classes and their attributes. The dependency structure defines the set of parents that an attribute conditionally depends on. Associated to the dependency structure is a quantitative component: a set of aggregate functions and CPDs. The semantics of a PRM is that it induces a Bayesian network on the so-called relational skeleton. The latter specifies all the objects for all the classes and the values of the (primary and foreign) key-attributes for all objects but leaves the values of all other attributes (‘descriptive’

attributes) unspecified. The Bayesian network then specifies a probability distribution on these unspecified values. Algorithms for learning the dependency structure and the CPDs have been developed [13, 15, 14, 16].

The graphical representation of the dependency structure for our running example is shown in Figure 2 (it is similar to the example in [14]).

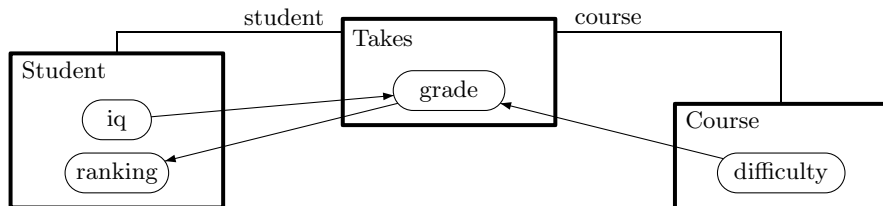


Fig. 2. The dependency structure of the PRM for our running example. Rectangles represent classes, ovals represent descriptive attributes, lines represent relationships through foreign keys and arrows represent conditional dependencies.

Discussion. LBNs can be seen as the counterpart of PRMs in a Logic Programming based language. First, the distinction logical vs. probabilistic predicates in LBNs corresponds to the distinction key-attributes vs. descriptive attributes in PRMs (as key-attributes in PRMs are supposed to be deterministic and are used to specify the objects in the domain of discourse and their relations)². Second, there is a one-to-one correspondence as to functionality between the components of PRMs and those of LBNs. In LBNs we use the random variable declarations to determine the nodes in the Bayesian network, where PRMs use the relational schema. In LBNs we use the conditional dependency clauses to determine the directed edges, where PRMs use dependency structure. In LBNs we use logical CPDs to determine the CPDs in the Bayesian network, where PRMs use a combination of aggregate functions with ordinary CPDs.

Due to this correspondence between the components of LBNs and PRMs, it is trivial to translate any PRM to an equivalent LBN. As a consequence, LBNs can help to clarify the relationships between PRMs and probabilistic logical models based on concepts of Logic Programming [23, 10, 38, 33, 42, 34, 35].

The main advantage of LBNs over PRMs is that LBNs are more flexible and more expressive. This is the result from the transition from the entity-relationship language of PRMs to the full Logic Programming language of LBNs.

LBNs are more flexible than PRMs. First, in LBNs the knowledge that determines the random variables and the dependencies (i.e. the knowledge specified by the logical predicates) can be anything. In PRMs this knowledge can only be knowledge about class-membership and relations (i.e. knowledge contained in the relational skeleton). For example, suppose that we want to specify that only undergraduate students get a final ranking. In LBNs we simply write

² Extensions of PRMs exist where key-attributes do not have to be deterministic: PRMs with ‘structural uncertainty’ [15, 14, 16].

`random(ranking(S)) <- undergrad(S)`. In PRMs we can only specify this if we adapt the relational schema of our running example by explicitly making a new (sub)class for undergraduate students [14]. As the conditions we want to specify get more complex, this process of adapting the relational schema of the PRM becomes more and more cumbersome. In LBNs, this can be handled in a much more uniform way. Second, in the same way it is easier to specify deterministic background knowledge in LBNs than in PRMs (in LBNs we can simply use the normal clauses in \mathcal{B}).

LBNs are also more expressive than PRMs. PRMs do not have functor symbols. Functor symbols are needed to elegantly represent temporal processes such as Hidden Markov Models, or more generally, to represent recursive concepts. For a further discussion we refer to the remarks on recursion and PRMs in [38]. Also, PRMs have no concept of negation. One application of negation is dealing with exceptions, e.g. expressing that a student has a grade for a course if he was taking that course *unless* he was absent on the exam. This cannot be expressed directly in PRMs. We further discuss negation in Section 3.2.

3.2 Bayesian Logic Programs

Introduction. Bayesian Logic Programs (BLPs) combine Bayesian networks with definite Logic Programming. BLPs were defined in [23, 24, 22, 25]. Recently, a modified definition has been given in [10, 28]. We now discuss the original (and probably best known) definition. We come to the new definition later on in this section.

The core of a BLP is a set of Bayesian clauses. An example of such a clause is:

```
grade(S,C) | iq(S), diff(C), takes(S,C).
```

All predicates in BLPs are ‘Bayesian’ predicates having an associated range (like probabilistic predicates in LBNs). Ground atoms represent random variables. The semantics of a BLP is that it induces a Bayesian network. The random variables 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 encode directed edges: there is an edge from $V_{parent} \in LH$ to $V_{child} \in LH$ iff V_{parent} is in the body of a ground instance with V_{child} in the head. As a quantitative component BLPs use CPDs and combining rules. Algorithms for learning the Bayesian clauses and the CPDs have been developed [24, 22, 25].

To model our running example with a BLP, we need the following Bayesian clauses (*student/1*, *course/1*, *takes/2*, *iq/1*, *ranking/1*, *diff/1* and *grade/2* are all Bayesian predicates):

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

```

ranking(S) | grade(S,C), takes(S,C).
student(john).      student(pete).
course(ai).         course(db).
takes(john,ai).    takes(john,db).    takes(pete,ai).
    
```

The first four clauses are essentially needed to specify the random variables, the fifth and the sixth clause to specify the dependencies³ and the Bayesian ground facts to specify the domain of discourse.

Discussion. The most important difference between BLPs as defined above and LBNs is that BLPs do not have standard logical predicates. In the philosophy of BLPs logical predicates are a special kind of Bayesian predicates with range {true,false}. This leads to a number of problems from a knowledge representation point of view:

1. *Compared to the Bayesian network induced by a LBN or PRM, the network induced by a BLP typically contains more nodes and, as a consequence, has CPDs that cannot be filled in meaningfully.* The reason is that for instance `student(john)` in the above BLP is not a logical fact stating that *john* is a student, but a Bayesian fact stating that *student(john)* is a random variable (with an associated CPD which we do not show here). The Bayesian network induced by the above BLP is shown in Figure 3a (only partially because of space restrictions). As a reference, Figure 3b shows the corresponding part of the Bayesian network induced by a LBN (this is a fragment of Figure 1). Note that network induced by the LBN does not contain e.g. *student(john)* as a random variable.

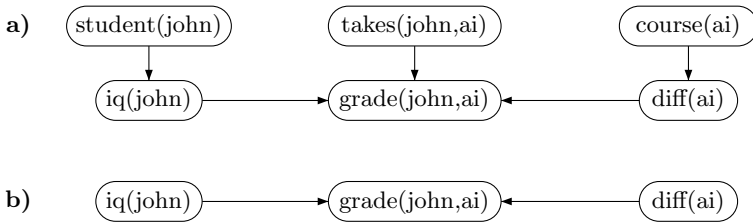


Fig. 3. Part of the structure of the Bayesian network induced for the running example a) by a BLP, b) by a LBN. The former typically contains more nodes than the latter.

In the network of Figure 3b (for LBNs), the node *iq(john)* needs a CPD that is unconditioned, for example the following table:

$p(iq(john))$	
low: 0.4	high: 0.6

³ In the clause `grade(S,C) | iq(S), diff(C), takes(S,C).`, the atom `takes(S,C)` is needed to ensure that *grade(S,C)* is a random variable *only* if *S* takes *C*.

In the network of Figure 3a (for BLPs), the node $iq(john)$ needs a CPD that is conditioned on $student(john)$, for example the following table:

$student(john)$	$p(iq(john) student(john))$	
true	low: 0.4	high: 0.6
false		?

The problem here is that no meaningful probability distribution can be filled in for the case where $student(john)$ is false (the question mark). The reason is that in our example we wanted to model that something has an iq only if it is a student. So if $student(john)$ is false, the random variable $iq(john)$ is meaningless and should not even exist. The same problem appears when trying to specify a CPD for the dependence of $diff(ai)$ conditioned on $course(ai)$ and also for $grade(john, ai)$ conditioned on $takes(john, ai)$. To summarize, the Bayesian networks induced by BLPs contain CPDs that cannot be filled in meaningfully while for LBNs (or PRMs) this problem does not exist.

One might think that the approach taken by BLPs is ‘more general’ than the approach taken by LBNs in that BLPs allow knowledge about the $student/1$ predicate to be non-deterministic and LBNs do not. This is wrong, however. LBNs leave the user the freedom to decide for each application which predicates should be logical and which probabilistic. As such the user could for instance decide to make $student/1$ a probabilistic predicate if needed, accepting the above problems with meaningless entries in CPDs (essentially, the same approach is taken by PRMs with structural uncertainty [15, 14, 16]). Our point, however, is that *if* $student/1$ is deterministic we can make it a logical predicate in LBNs, avoiding the above problems. In BLPs, this is not possible since it has been decided by design that all predicates are probabilistic. In other words, in BLPs we cannot express the fact that certain knowledge is deterministic, while in LBNs we can.

As a more practical side-remark, note that larger CPDs typically result in slower inference [7]. As such, inference in networks induced by BLPs is expected to be slower than for LBNs or PRMs.

2. *Since BLPs do not have logical atoms, no negated atoms are allowed.* One of the possible applications of negation is default reasoning (dealing with exceptions [6]). For instance, suppose we want to express that a student has a grade for a course if he was taking that course *unless* he was absent on the exam (being absent is considered as an exception). In LBNs, we would simply write:

```
random(grade(S,C)) <- takes(S,C), not(absent(S,C)).
```

In BLPs, however, we cannot express this since no negated atoms are allowed. Note that the above form of negation, which cannot be captured by BLPs, is *non-monotonic negation* [1]. *Classical negation*, in contrast, can be simulated by BLPs inside the CPDs [23]. For instance, to model that someone is male if and only if he is not female, we can write a Bayesian clause $male(X) \mid female(X)$ with the following CPD:

$female(X)$	$p(male(X) female(X))$
true	true: 0.0 false: 1.0
false	true: 1.0 false: 0.0

3. In addition to the previous remarks (but less important because somewhat subjective), *it is more difficult to read and write clauses in a BLP than clauses in a LBN*. This is because clauses in a BLP have a double meaning:
- They should be seen as a definite logic program to find the random variables in the Bayesian network (through the least Herbrand model). In this reading, each atom in each clause should be seen as a standard logical atom.
 - At the same time they should be seen as statements about conditional dependencies between random variables. In this reading, each atom should be seen as a random variable (or set of random variables).

This is not the case in LBNs. First, each clause is either a random variable declaration or a conditional dependency clause. Second, each atom in each clause is either a standard logical atom or a random variable. Moreover, both distinctions are clearly visible in the syntax of LBNs.

BLPs Redefined. The above problems are all caused by the fact that BLPs as defined originally [23, 24, 22, 25] do not have standard logical predicates. BLPs have recently been redefined [10, 28] and now indeed distinguish logical predicates and Bayesian predicates. In the new definition only ground Bayesian atoms (in the least Herbrand model of the BLP) become random variables in the induced Bayesian network. Ground logical atoms are kept out of the network. This is also the way BLPs are implemented [28].

Note, that at the time LBNs were first published [12], all literature about BLPs [23, 24, 22, 25] still used the original definition, i.e. the one without this distinction. Also, the ‘new’ literature about BLPs [10, 28] does not give any reasons why this redefinition is needed (in fact, it does not even mention that it is different from the original definition). In our discussion above we tried to show these reasons by explicitly pointing out the problems with the original definition. As such this paper can contribute to the understanding of BLPs.

This redefinition obviously brings BLPs closer to LBNs. The main remaining difference is that LBNs use one set of clauses to specify the random variables in the Bayesian network and a separate set of clauses to specify the directed edges, whereas BLPs use the same set of clauses for both purposes. While this might make LBNs slightly easier to read than BLPs (especially for people acquainted with PRMs), it is not an essential difference.

3.3 Other Probabilistic Logical Models

A variety of probabilistic logical models has been described in the literature (see the overview by Kersting and De Raedt [26]). On a high level, these models can be divided into two classes.

Models of the first class combine Bayesian networks with logic and mainly follow the knowledge based model construction approach. We already discussed LBNs, Probabilistic Relational Models and Bayesian Logic Programs. Some others models of this class are Relational Bayesian Networks [19], Probabilistic Logic Programs (also known as Context-Sensitive Probabilistic Knowledge Bases) [33], MIA (the ‘meta-interpreter approach’, which is the origin of some ideas incorporated in LBNs) [4], $\text{CLP}(\mathcal{BN})$ [38], Hierarchical Bayesian Networks [17] and Markov Logic Networks [11, 36] (the latter are based on Markov networks). Learning algorithms exist for Probabilistic Relational Models [13, 15, 14, 16], Bayesian Logic Programs [24, 22, 25], $\text{CLP}(\mathcal{BN})$ [38], Hierarchical Bayesian Networks [17] and Markov Logic Networks [36].

Models of the second class integrate probabilities into Logic Programming, staying as close as possible to pure Logic Programming. The most important of these models are Probabilistic Horn Abduction [34], Independent Choice Logic [35], PRISM [39, 40], Stochastic Logic Programs [8, 9, 30, 31] and Logic Programs with Annotated Disjunctions [42]. Learning algorithms exist for the last three models [20, 40, 9, 31, 37].

4 Conclusions

We reviewed Logical Bayesian Networks introduced in [12]. We have given more compact and clear but essentially equivalent definitions of syntax and semantics of LBNs than in [12]. We carried out a more detailed comparison of LBNs with Probabilistic Relational Models and Bayesian Logic Programs, hereby clarifying and motivating the difference between the original definition of Bayesian Logic Programs [23, 24, 22, 25] and their recent redefinition [10, 28].

A lot of future work remains. As for knowledge representation, comparing LBNs to other probabilistic logical models is promising given the wide variety of such models. As for learning, we are currently working on learning logical CPDs in LBNs under the form of first order logical probability trees (Tilde [5, 43]). In a next step, algorithms for learning the conditional dependency clauses of LBNs can be developed.

The methodology behind the design of LBNs can also be followed for other graphical models than Bayesian networks. Languages for knowledge based model construction of dependency networks [18], Markov networks [36] or neural networks [3] can all be defined having the same components as LBNs: a component for determining the nodes in the graph, one for the edges and one for the quantitative local models (CPDs for dependency networks, potential functions for Markov networks, activation functions for neural networks).

Acknowledgements

Daan Fierens is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen). 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 and the reviewers for useful comments.

References

- [1] J. Alferes, L. Pereira, and T. Przymusiński. ‘Classical’ negation in Nonmonotonic Reasoning and Logic Programming. *Journal of Automated Reasoning*, 20:107–142, 1998.
- [2] F. Bacchus. Using first-order probability logic for the construction of Bayesian networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI-1993)*, pages 219–226, 1993.
- [3] C. M. Bishop. *Neural Networks for Pattern Recognition*. University Press, Oxford, 1999.
- [4] H. Blockeel. Prolog for Bayesian networks: a Meta-Interpreter Approach. In *Proceedings of the 2nd International Workshop on Multi-Relational Data Mining (MRDM-2003)*, pages 1–13, 2003.
- [5] H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
- [6] R. Brachman and H. Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, 2004.
- [7] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.
- [8] J. Cussens. Stochastic logic programs: Sampling, inference and applications. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pages 115–122, San Francisco, CA, 2000. Morgan Kaufmann.
- [9] J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
- [10] L. De Raedt and K. Kersting. Probabilistic Inductive Logic Programming. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT-2004)*, pages 19–36, October 2004. Invited paper.
- [11] P. Domingos. Learning, logic, and probability: A unified view. In *Proceedings of 14th International Conference on Inductive Logic Programming (ILP-2004)*, Porto, Portugal, page 359, 2004. Invited paper.
- [12] D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon. Logical bayesian networks. In *Proceedings of the 3rd Workshop on Multi-Relational Data Mining (MRDM-2004)*, Seattle, WA, USA, pages 19–30, 2004.
- [13] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-1999)*, pages 1300–1309, 1999.
- [14] 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.
- [15] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Proc. 18th International Conf. on Machine Learning (ICML-2001)*, pages 170–177. Morgan Kaufmann, San Francisco, CA, 2001.
- [16] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

- [17] E. Gytodimos and P. Flach. Hierarchical Bayesian Networks: an Approach to Classification and Learning for Structured Data. In *Proceedings of the ECML/PKDD - 2003 Workshop on Probabilistic Graphical Models for Classification*, pages 25–36, 2003.
- [18] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.
- [19] M. Jaeger. Relational Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-1997)*, pages 266–273. Morgan Kaufmann Publishers, 1997.
- [20] Y. Kameya and T. Sato. Efficient EM learning with tabulation for parameterized logic programs. In *Proceedings of the 1st International Conference on Computational Logic (CL-2000)*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 269–294, 2000.
- [21] K. Kersting and L. De Raedt. Bayesian logic programs. In *Proceedings of the tenth international conference on Inductive Logic Programming, work in progress track*, 2000.
- [22] K. Kersting and L. De Raedt. Adaptive Bayesian Logic Programs. In *Proceedings of the 11th International Conference on Inductive Logic Programming (ILP-2001)*, pages 104–117, 2001.
- [23] K. Kersting and L. De Raedt. Bayesian logic programs. Technical Report 151, Institute for Computer Science, University of Freiburg, Germany, April 2001.
- [24] 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 (ILP-2001)*, pages 118–131, 2001.
- [25] K. Kersting and L. De Raedt. Basic principles of learning bayesian logic programs. Technical Report 174, Institute for Computer Science, University of Freiburg, Germany, June 2002.
- [26] 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.
- [27] K. Kersting, L. De Raedt, and S. Kramer. Interpreting bayesian logic programs. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, 2000.
- [28] K. Kersting and U. Dick. Balios - The Engine for Bayesian Logic Programs. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2004)*, pages 549–551, September 2004. Demonstration paper.
- [29] J. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd edition, 1987.
- [30] S. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.
- [31] S. Muggleton. Learning stochastic logic programs. In L. Getoor and D. Jensen, editors, *Proceedings of the AAAI2000 workshop on learning statistical models for relational data*, 2000.
- [32] R. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, New Jersey, 2003.
- [33] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171(1–2):147–177, 1997.
- [34] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.
- [35] D. Poole. The Independent Choice Logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1–2):5–56, 1997.

- [36] M. Richardson and P. Domingos. Markov Logic Networks. Technical report, Department of Computer Science, University of Washington, 2004.
- [37] F. Riguzzi. Learning logic programs with annotated disjunctions. In *Proceedings of 14th International Conference on Inductive Logic Programming (ILP-2004)*, Porto, Portugal, 2004.
- [38] 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 (UAI-2003)*, 2003.
- [39] T. Sato and Y. Kameya. PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-1997)*, pages 1330–1335, 1997.
- [40] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.
- [41] A. Van Gelder, K. Ross, and J. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3), 1991.
- [42] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *Proceedings of the 20th International Conference on Logic Programming (ICLP-2004)*, 2004.
- [43] C. Vens, A. Van Assche, H. Blockeel, and S. Džeroski. First order random forests with complex aggregates. In R. Camacho, R. King, and A. Srinivasan, editors, *Proceedings of the 14th International Conference on Inductive Logic Programming*, pages 323–340. Springer, 2004.