Independence in Relational Languages with Finite Domains

José E. Ochoa Luna [1], **Cassio P. de Campos** [2], **Fabio G. Cozman**[1]

[1]Escola Politécnica – University of São Paulo
São Paulo – SP – Brazil

[2]Escola de Artes, Ciências e Humanidades – University of São Paulo
São Paulo – SP – Brazil.

eduardo.ol@gmail.com, cassiopc@usp.br, fgcozman@usp.br

Abstract. *We introduce a restricted first-order language that combines logical sentences and probabilistic assessments in finite domains. In this preliminary effort to construct a general device for knowledge representation, we present the language and an inference algorithm based on propositionalization and reduction to multilinear programming.*

Resumo. *Este artigo introduz uma linguagem de primeira-ordem restrita que combina sentenças lógicas e asserções probabilísticas em domínios finitos. Neste esforço preliminar para construir um instrumento geral de representação de conhecimento, apresentamos a linguagem e um algoritmo de inferência baseado em proposicionalização e redução para programação multilinear.*

1. Introduction

Combinations of logic and probabilities hold a great deal of promise; however, they also present enormous challenges. While comprehensive probabilistic logics can claim a special place in knowledge representation with regard to expressivity, such logics pose serious difficulties concerning complexity. In fact, a rather general combination of first-order logic and probability theory has already been defined by [Halpern 2003], together with results suggesting how difficult inference may be in such a general language.

We look for a language that combines features of first-order logic and probability theory while still offering some hope of tractability. We start by constraining ourselves to finite and known domains. We then work towards the use of independence relations as powerful constraints on the possible models of sentences. In doing so, we intend to start an investigation towards first-order probabilistic logics that exploit independence relations for inference in a clever way. Our preliminary ideas in this direction are reported in this paper.

The remainder of the paper is organized as follows. In Section 2, we discuss probabilistic logics and their linear and multilinear inference algorithms, particularly for PPL networks — a recently introduced model for propositional probabilistic logic with independence constraints expressed in graphical form. In Section 3 we present our main contributions, focusing on restricted aspects of first-order logic, and deriving solutions for probabilistic entailment. We conclude with a brief discussion of our results, indicating some open questions and future work.

2. Probabilistic Logic and PPL Networks

The combination of logic and probabilities has been extensively studied in the last decades. This promising mixture has old roots and has been rediscovered a few times. Nilsson's probabilistic logic [Nilsson 1986] has been a significant influence, as well as the very general languages proposed by [Halpern 2003].

We may, in rough terms, divide existing languages that claim to be "first-order (or subsets thereof) probabilistic logics" in two groups. In one group, probability assessments are rather flexible and independence relations are mostly ignored. Nilsson's and Halpern's logics are examples, as are several other formal languages [Lukasiewicz 2001, Giugno and Lukasiewicz 2002, Ognjanović 2006] and programming languages based on probabilistic logic [Ng and Subrahmanian 1992, Lakshmanan and Sadri 1994, Lukasiewicz 1998]. In the second group, probabilistic assessments are required to specify a unique distribution and independence is an essential concept; several of these languages are known as probabilistic relational models [Jaeger 1997, Ngo and Haddawy 1997, Lukasiewicz 1998, Jaeger 2001, Getoor et al. 2001, Poole 2003, de Salvo Braz et al. 2006]. The second group of languages is more restricted than the first in the kinds of logical sentences and probabilistic assessments that are allowed. Often the languages in the second group are based on graphical models such as Bayesian networks or Markov random fields [Cowell et al. 1999, Pearl 1988], and an attempt is made to transfer the excellent computational properties of these graphical models to the languages.

In this paper we wish to develop a language that is restricted in some important respects (finite and known domain, decidable fragment of first-order logic based on relations) but that still allows quite general sentences to be expressed. We also wish to consider a language where probabilistic assessments can include independence relations — and we hope to exploit these relations to reduce the computational effort necessary for inference.

In the remainder of this section we present some basic elements of probabilistic logic. We do not attempt to review the (rather large) literature on first-order probabilistic logic; rather, our intention is to indicate the main algorithmic tools one can actually use to handle progressively general probabilistic logic. Thus we start with propositional logic without independence, and move to languages based on graphical models. These tools will be used later when we deal with our restricted first-order logic.

Basically, in a probabilistic logic knowledge base each formula is associated with a probability interval. The conditional probability of a formula given other formula may be specified as well. The probability that a formula is true is the sum of the probabilities of all the models (or possible worlds, interpretations, truth assignments) in which the formula is true. The inference problem (probabilistic entailment) is either to determine whether the knowledge base is satisfiable (that is, whether or not there is a probability distribution over all models, that satisfies the logical sentences and the probabilistic assessments) or to find the probability of a given formula.

Consider first *propositional* probabilistic logic *without* independence relations. To write a general linear programming program for inference, we follow the notation of [Chandru and Hooker 1999]. Suppose there are n atomic propositions, which give rise to

$N = 2^n$ possible worlds. Let the vector p denote a probability distribution over all possible worlds. Suppose there are m formulas with probability assessments. Their lower and upper bounds are $\underline{\pi} = (\underline{\pi}_1, \ldots, \underline{\pi}_m)$ and $\overline{\pi} = (\overline{\pi}_1, \ldots, \overline{\pi}_m)$, respectively (we may have $\underline{\pi} = \overline{\pi}$). In addition, there are lower bounds $\underline{\rho}$ and upper bounds $\overline{\rho}$ for conditional probabilities $P(G_1|H_1), \ldots, P(G_{m'}|H_{m'})$, where G_i and H_i are formulas. To evaluate a range for the probability of a formula G_0, where c is the indicator vector of which worlds satisfy G_0, the linear model can be written as

$$\begin{aligned} \min / \max \quad & c^T p \\ s.t. \quad & \underline{\pi} \leq Ap \leq \overline{\pi} \\ & \overline{B}p \geq 0 \\ & \underline{B}p \leq 0 \\ & e^T p = 1, p \geq 0, \end{aligned} \tag{1}$$

where e is a vector of ones, the matrix A has dimension $N \times m$ (each line is an indicator function of those possible worlds that satisfy the line's formula), and matrices $\overline{B}$ and $\underline{B}$ specify conditional probabilities. That is, $a_{ij} = 1$ if G_i is true in world j, and $a_{ij} = 0$ otherwise. Similarly,

$$b_{ij} = \begin{cases} 1 - \rho & \text{if } G_i \wedge H_i \text{ are true in world } j \\ -\rho & \text{if } G_i \text{ is false and } H_i \text{ is true in world } j \\ 0 & \text{otherwise} \end{cases}$$

where ρ is $\overline{\rho}$ for $\overline{B}$ and $\underline{\rho}$ for $\underline{B}$. The main algorithmic difficulty here is the size of the matrices that must be manipulated; the most efficient algorithms to date employ the *Column Generation* method to avoid explicit generation of matrices [Hansen and Jaumard 1996].

Independence of events and variables is an extremely powerful source of probabilistic knowledge. Therefore it is desirable to incorporate independence assumptions in the knowledge base. In principle, independence relations should be useful, because they might allow the probabilities in the linear program above to be written in factorized form; that is, they should lead to a reduction on the size of matrices manipulated during inference. However, the problem is that probabilistic logic becomes significantly harder when independence relations are present because they introduce nonlinear constraints that destroy the linearity of the Program (1) [Chandru and Hooker 1999].

However, it is possible to exploit independence relations if they are introduced in an organized fashion — similarly to the way they are expressed in Bayesian networks. We now review a recent proposal for the combination of propositional logic and probability, the *PPL networks* [Cozman et al. 2006, Cozman et al. 2007]. In this model, independence relations are represented through a directed acyclic graph $\mathcal{G}$, where each node represents a proposition/variable X_i in $\mathbf{X}$. The parents of X_i are denoted by $pa(X_i)$. Each variable X_i is assumed independent of its nondescendants nonparents given its parents (the Markov condition). This leads to the unique factorization

$$P(\mathbf{X}) = \prod_i P(X_i|pa(X_i)). \tag{2}$$

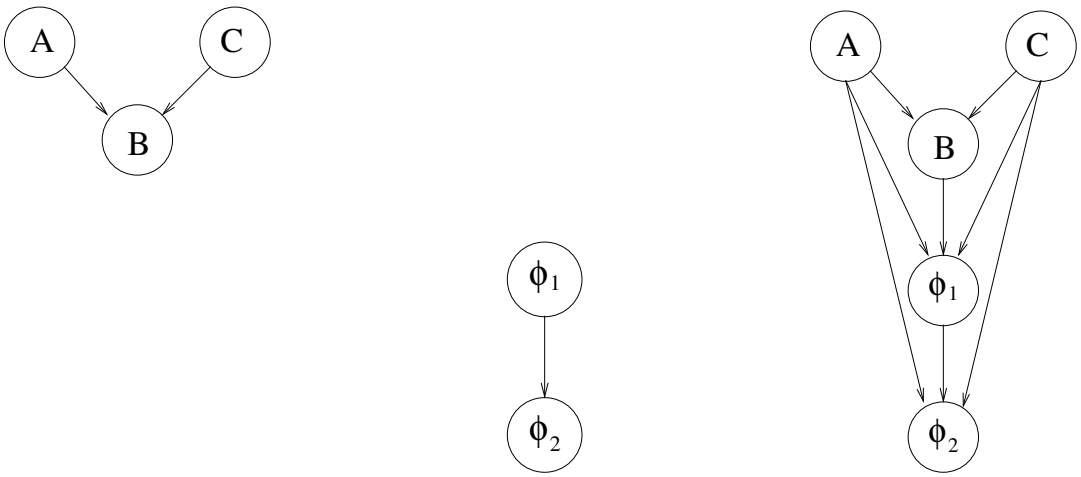
Thus our knowledge base is now composed of logical formulas, probabilistic assessments, and independence relations encoded in the graph $\mathcal{G}$. Note that this is far more

general than a standard Bayesian network, because we accept general logical formulas (in propositional logic) in the knowledge base, and we do not require that all probabilities in Expression (2) are specified precisely.

The complete model for PPL networks is as follows. A PPL network consists of a triple $(\mathcal{G}, \mathcal{L}, \mathcal{A})$, where $\mathcal{G}$ is a directed acyclic graph with n nodes, each one identified with a variable X_i; $\mathcal{L}$ is a list of formulas; $\mathcal{A}$ is a list of assessments $\alpha_j P(\varphi_j) \leq P(\phi_j \wedge \varphi_j) \leq \beta_j P(\varphi_j)$, where ϕ_j and φ_j are formulas; all formulas must contain variables in $\mathcal{G}$. In this paper we do not demand the formulas to be in *conjunctive normal form (CNF)*, although there is a performance gain in doing so.

Inference in a PPL network relies on an augmented graph $\mathcal{G}'$, which has $\mathcal{G}$ as its subgraph and has an additional node for each distinct formula appearing in $\mathcal{L}$ or in an assessment of $\mathcal{A}$. These nodes are called *formula nodes* and have as parents the variables that define them. The truth-table of their associated formulas are used to specify their conditional probability tables (CPTs). With this augmented graph, every constraint on probability values of formulas specified in $\mathcal{L}$ or $\mathcal{A}$ can be encoded as joint queries in the network, because each formula in $\mathcal{L}$ is just a network query, as it is each sub-formula in assessments of $\mathcal{A}$ (they appear now as formula nodes in $\mathcal{G}'$).

Note that the factorization (2) leads to a reduction on the number of optimization variables necessary to produce an inference (when compared to Expression (1)). There is no need to deal with all 2^n optimization variables anymore, as we can expressed any variable using equation (2). On the other hand, multilinear constraints for specifying independence relations are introduced. Approximation techniques [Nilsson 1986], nonlinear methods [Andersen and Hooker 1994] and more recently exact multilinear programming algorithms [Cozman et al. 2006] must then be employed.



(a) Independence graph with variable nodes.

(b) Independence graph with formula nodes.

(c) Augmented graph including variable and formula nodes and their relations.

Figure 1. Example of Propositional Probabilistic network.

The augmented graph $\mathcal{G}'$ will have nodes associated with propositions/variables and with formulas. Figure 1(a) shows variable nodes A, B and C, and their dependencies ($A \rightarrow B$ and $C \rightarrow B$). Figure 1(b) shows formula nodes $\phi_1(A, B, C)$ and $\phi_2(A, C)$ and their

dependencies ($\phi_1 \rightarrow \phi_2$). We note that in previous publications [Cozman et al. 2006], nodes in PPL networks are only associated with propositions, and there are no edges "between" formulas (this assumption was adopted to reduce the complexity of inference). Here we remove this restriction. Thus there may exist dependencies between variables (propositions) or between formulas. For instance, the arc from ϕ_1 to ϕ_2 indicates a dependence between them. We may also have $P(\phi_2 \mid \phi_1) \geq 0.3$, or even $P(\phi_1) \leq 0.4$. Any assessment (conditional or not) between variables and formulas is valid. The only mandatory dependencies are between formulas and variables they contain, and are introduced during the construction of augmented graph $\mathcal{G}'$ (Figure 1(c)).

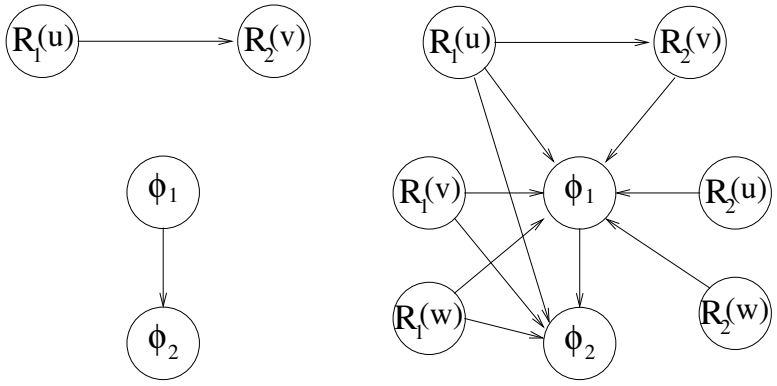
3. A Probabilistic Relational Language with Independence

The networks described in the last section are restricted to propositional logic; this is obviously a very limited language for knowledge representation. We now propose to lift PPL networks to a setting where first-order constructs, such as relations and quantifiers, are present. The task is not entirely obvious: on one hand, the underlying graph of a PPL network represents *independence relations* amongst variables; on the other hand, sentences in first-order logic represent *hard dependencies* amongst variables. We must be careful to create a language that accommodates both formalisms while avoiding clashes.

We propose a restricted first-order model where *formulas* are combinations of *constants, relations*, Boolean operators, *quantifiers* $\exists$ and $\forall$, and *logical variables*. A *sentence* is a formula without free logical variables. The semantics is established using a *domain* (a set of *individuals*), which as stated before, is assumed finite and known. An *interpretation* assigns individuals to constants and relations in the domain respectively to constants and relations in the vocabulary [Nerode and Shore 1997]. More precisely, our model is defined as tuple $(\mathcal{D}, \mathcal{S}, \mathcal{G}, \mathcal{L}, \mathcal{A})$, where $\mathcal{D}$ is a finite set of individuals, $\mathcal{S}$ is a vocabulary with constants and relations, $\mathcal{L}$ is a list of sentences in $\mathcal{S}$ where quantifiers $\forall$ and $\exists$ are allowed; $\mathcal{G}$ is a directed acyclic graph with n nodes, each one can be an interpretation of a relation in S or a formula in $\mathcal{L}$; $\mathcal{A}$ is a list of assessments $\alpha_j P(\varphi_j) \leq P(\phi_j \wedge \varphi_j) \leq \beta_j P(\varphi_j)$ based on formulas in the graph. The graph G imposes a Markov condition on its elements. Dependence relations are allowed among formulas, logical variables and interpretations of relations, thus extending the set of allowed arcs in *PPL networks*.

Figure 2(a) shows a typical network structure for a couple of formulas. For example, we have $\phi_1 = \forall x\ R_1(x) \wedge R_2(x)$ and $\phi_2 = \exists x\ R_1(x)$, where R_1 and R_2 are relations. We could also have assessments such as $P(\phi_1) \leq 0.6$, $P(\phi_2) = 0.3$ or $P(\phi_2 \mid \phi_1) \geq 0.3$. Furthermore, we allow arcs between interpretations, as $R_1(u) \rightarrow R_2(v)$ (suppose the domain is composed by u, v and w). Note that arcs between relations/interpretations and their formulas does not appear in the graph of Figure 2(a), although they are included in the corresponding augmented graph (see Figure 2(b)).

We shall be concerned with the question of determining the probability of an arbitrary sentence S given a set $\mathcal{L}$ of sentences and their probabilities. That is, we consider the probabilistic entailment of S given $\mathcal{L}$. We resort to a *propositionalization* scheme that transform a relational first-order probabilistic network into a PPL network. We note that propositionalization has been used in connection with probabilistic relational models (Section 2), but in that previous work propositionalization usually consisted of replicating



(a) Independence graph with logical (based on interpretations) and formula nodes.

(b) Augmented graph with dependencies between formulas and interpretations of their relations.

Figure 2. Example of relational first-order probabilistic network.

the network structure for all individuals in a domain. Here we use a different approach: we add dependencies among instances of variables and probabilistic relational first-order formulas. Clearly, complexity increases exponentially by the domain size and amount of possible interpretations. However, the graph structure and the Markov condition allow us alleviate the problem by providing a factorization of probabilities.

Propositionalization is carried out as follows. According to domain and interpretations, we explode each logical relation in nodes, creating a node in the augmented version of $\mathcal{G}$ for each possible interpretation of the given relation. This is possible as we assume that the domain is finite and known. Unfortunately this procedure increases dependencies among formulas and variables/relations, because all propositionalized nodes of a given relation may participate as parents of a formula containing the relation (this happens, for example, when the relation appears quantified in the formula). All interpretations in a given formula must appear as its parents in the graph. Note that each relation increases the size of conditional probability tables in formula nodes approximately by 2^m times, where m is the number of valuations or number of possible assignments when predicates are binaries.

To further illustrate the propositionalization technique, suppose we have a knowledge base called *Family*, containing:

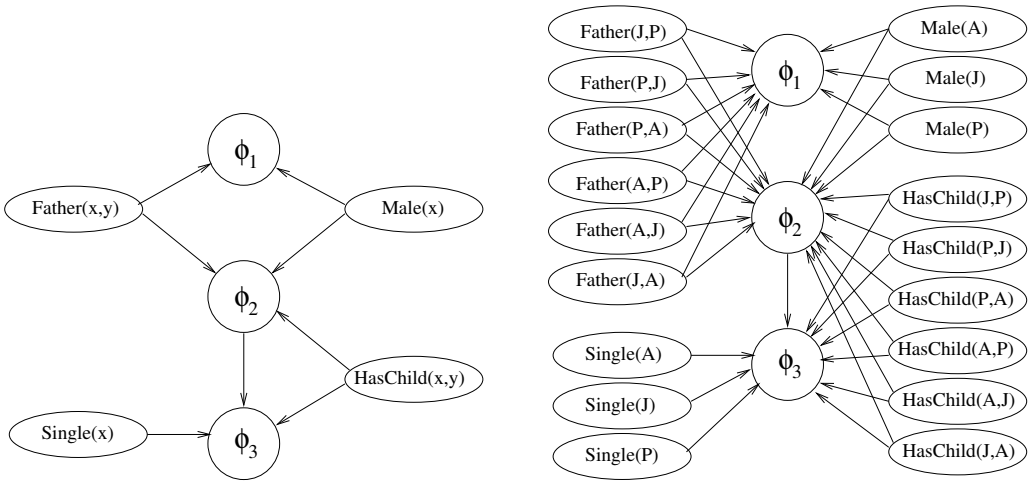
$$\forall\ x, y\ Father(x, y) \rightarrow Male(x) \qquad (\phi_1)$$
$$\forall\ x, y\ HasChild(x, y) \wedge Male(x) \rightarrow Father(x, y) \qquad (\phi_2)$$
$$0.8 \leq P(\forall\ x, y\ \neg HasChild(x, y) \rightarrow Single(x)) \leq 0.9 \qquad (\phi_3)$$

and the following assertions:

$$Father(Joao, Peter)$$
$$HasChild(Peter, Ann)$$
$$Male(Peter)$$
$$Single(Peter)$$

Suppose further that the only dependence between formulas is $\phi_2 \rightarrow \phi_3$. Figure 3(a) shows the network for this example, including extra nodes to show relations and

their formulas. Note that these relations will in fact be replaced by interpretations in the propositionalized graph, as shown in Figure 3(b). We use initial letters of names to simplify notation.



(a) Network of Family example. (b) Propositionalized graph.

Figure 3. Example of network propositionalization.

After propositionalization is finished, inference is just the process of finding lower/upper probabilities for a given formula, much as it is done in PPL networks. We can again use multilinear programming and exploit the factorization defined by the Markov condition. A good strategy is to resort to credal networks so as to construct a multilinear program out of the PPL network [de Campos and Cozman 2004]. To do so, we must run a symbolic inference algorithm that produces a compact description of a multilinear program, where the probability values $P(\phi_i|pa(\phi_i))$ appear as optimization variables. To handle probabilities over formulas, the probabilistic assessments are treated as joint queries in the network. Each joint query is a multilinear function $\sum_{X\setminus X_Q} \prod_i P(\phi_i|pa(\phi_i))$ over the optimization variables and restricted to some rational interval. We can choose among several algorithms to symbolically factor the summation into smaller multilinear constraints. Then these multilinear constraints are put together in a single multilinear program. The important difference from this setting to a simple query is that we must, at the same time, satisfy a collection of joint queries that are defined as multilinear constraints. The resulting multilinear program is the solved; in our implementation we use an optimized version of Sherali-Adams' branch-and-bound method [Sherali and Adams 1999].

4. Conclusion

In this paper we have extended the *PPL networks* formalism for combining logic and probabilities to a restricted relational first-order language. We emphasized the use of independence, and inference is performed by a nonlinear program formulation. We have used a graph-theoretical tool with two types of nodes to represent independence: variables/interpretations and formulas. This structure allows us to obtain a compact model and a compact multilinear program for inference. One of the interesting features of the model is the possibility to use already known algorithms for probabilistic entailment without complex reductions or transformations.

Given its generality, the model proposed is promising as a knowledge representation tool, even though inferences may be computationally intractable. Nevertheless, the model can still be helpful as a testbed for several probabilistic logic scenarios. When the domain size is not too large, the use of independence relations and the multilinear program formulation may achieve great simplifications.

Future work may investigate suitable applications that have natural formulations with relational first-order probabilistic logic with independence and where the size of domain and number of interpretations is manageable. Other logics should also be studied, with the goal of finding a probabilistic logic that is as general as possible, but that is still associated with practical inference procedures.

Acknowledgements

This work has been supported by FAPESP grant 2004/09568-0; the first author is supported by a CAPES scholarship; the third author is partially supported by CNPq grant 3000183/98-4.

References

Andersen, K. A. and Hooker, J. N. (1994). Bayesian logic. *Decision Support Systems*, 11:191–210.

Chandru, V. and Hooker, J. (1999). *Optimization methods for logical inference*. John Wiley & Sons Inc.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York, 1999.

Cozman, F. G., de Campos, C. P., and da Rocha, J. C. F. (2006). Probabilistic logic with strong independence. In *Brazilian Symposium on Artificial Intelligence*.

Cozman, F. G., de Campos, C. P., and da Rocha, J. C. F. (2007). Probabilistic logic with independence. *International Journal of Approximate Reasoning*, accepted for publication.

de Campos, C. P. and Cozman, F. G. (2004). Inference in credal networs using multilinear programming. In Onaindia, E. and Staab, S., editors, *Proceedings of the Second Starting AI Researchers' Symposium (STAIRS)*, pages 50–61, Amsterdam, The Netherlands. IOS Press.

de Salvo Braz, R., Amir, E., and Roth, D. (2006). Lifted first-order probabilistic inference. In *International Joint Conference in Artificial Intelligence (IJCAI)*.

Getoor, L., Friedman, N., Koller, D., and Taskar, B. (2001). Learning probabilistic models of relational structure. In *International Conference on Machine Learning*, pages 170–177.

Giugno, R. and Lukasiewicz, T. (2002). P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In Flesca, S., Greco, S., Leone, N., and Ianni, G., editors, *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 2424, pages 86–97, Cosenza, Italy. Lecture Notes in Artificial Intelligence, Springer.

Halpern, J. Y. (2003). *Reasoning about uncertainty*. MIT Press, Cambridge, Massachusetts.

Hansen, P. and Jaumard, B. (1996). Probabilistic satisfiability. Technical Report G-96-31, Les Cahiers du GERAD, École Polytechique de Montréal.

Jaeger, M. (1997). Relational Bayesian networks. In Geiger, D. and Shenoy, P. P., editors, *Conference on Uncertainty in Artificial Intelligence*, pages 266–273, San Francisco, California. Morgan Kaufmann.

Jaeger, M. (2001). Complex probabilistic modeling with recursive relational bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32:179–220.

Lakshmanan, L. V. S. and Sadri, F. (1994). Probabilistic deductive databases. In *Symposium on Logic Programming*, pages 254–268.

Lukasiewicz, T. (1998). Probabilistic logic programming. In *European Conference on Artificial Intelligence*, pages 388–392.

Lukasiewicz, T. (2001). Probabilistic logic programming with conditional constraints. *ACM Transactions on Computational Logic*, 2(3):289–339.

Nerode, A. and Shore, R. A. (1997). *Logic for Applications (2nd ed.)*. Springer-Verlag, New York.

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

Ngo, L. and Haddawy, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171(1–2):147–177.

Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence*, 28:71–87.

Ognjanović, Z. (2006). Discrete linear-time probabilistic logics: Completeness, decidability and complexity. *Journal of Logic Computation*, 16(2):257–285.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

Poole, D. (2003). First-order probabilistic inference. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 985–991.

H.D. Sherali and W.P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer Academic Publishers, 1999.