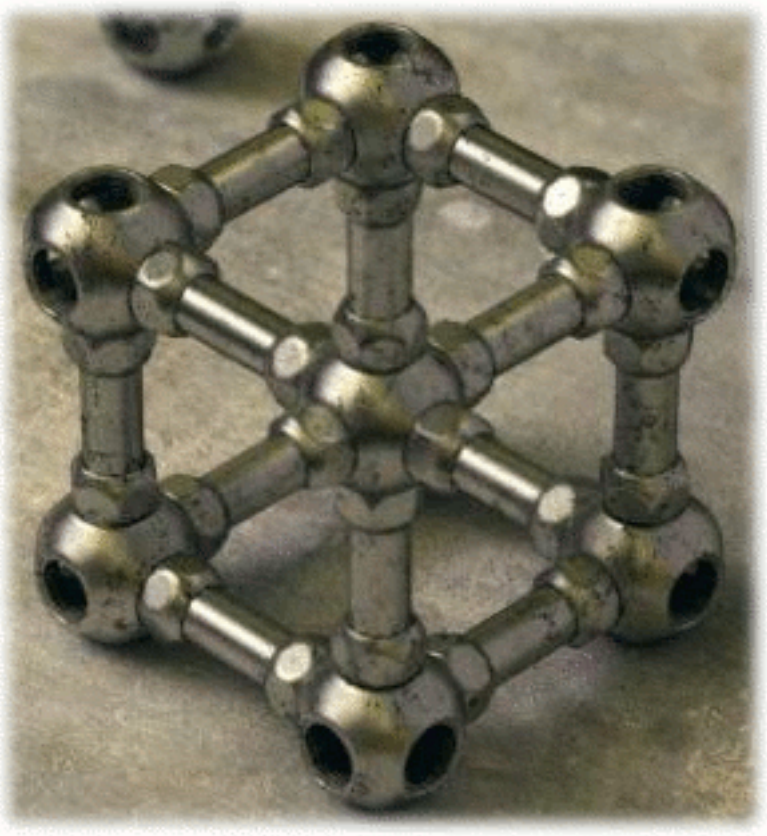
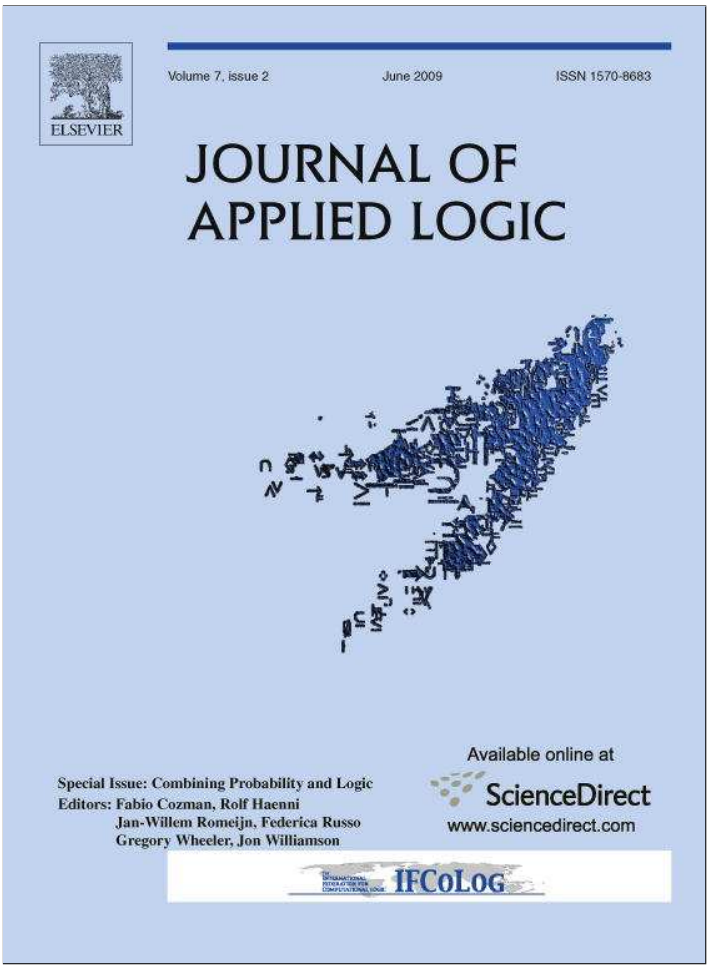


Assembling a consistent set of sentences in relational probabilistic logic with stochastic independence [1]

Author(s):

Cassio Polpo de Campos
Fabio Gagliardi Cozman
José Eduardo Ochoa Luna

[1]This work was supported by Fapesp Project LogProb, grant 2008/03995-5, São Paulo, Brazil.



Available online at www.sciencedirect.com
ScienceDirect
Journal of Applied Logic 7 (2009) 137–154
JOURNAL OF APPLIED LOGIC
www.elsevier.com/locate/jal

Assembling a consistent set of sentences in relational probabilistic logic with stochastic independence

Cassio Polpo de Campos [a], Fabio Gagliardi Cozman [b,*], José Eduardo Ochoa Luna [b]

[a] *Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, SP, Brazil*
[b] *Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brazil*

Received 4 July 2007; received in revised form 13 November 2007; accepted 19 November 2007

Available online 1 February 2008

Abstract

We examine the representation of judgements of stochastic independence in probabilistic logics. We focus on a relational logic where (i) judgements of stochastic independence are encoded by directed acyclic graphs, and (ii) probabilistic assessments are flexible in the sense that they are not required to specify a single probability measure. We discuss issues of knowledge representation and inference that arise from our particular combination of graphs, stochastic independence, logical formulas and probabilistic assessments.

Keywords: Probabilistic logic; Stochastic independence; Graph-theoretic models; Linear and multilinear programming

1. Introduction

In this paper we wish to discuss probabilistic logics that combine a relational fragment of first-order logic with probabilistic assessments and judgements of stochastic independence expressed through directed acyclic graphs. We wish to focus on probabilistic logics where a set of well-formed formulas, assessments and judgements may be *inconsistent*, if they are not satisfied by any measure, or *consistent*, if they are satisfied by *one or more* measures. In allowing such possibilities we depart from the current literature, where we find that probabilistic logics *either* represent stochastic independence judgements through graphs and produce a single satisfying measure, *or* they do not represent independence judgements at all and allow for inconsistency and sets of satisfying measures. To adequately position our proposals within the existing literature, we devote the relatively long Section 2 to a commentary on past and contemporary work on probabilistic logic.

The strategy we pursue is to add elements of first-order logic to the *PPL networks* we have recently introduced [16]; while PPL networks only allow a proposition per node in a graph, we now allow quantified relational formulas to be attached to nodes. We focus on finite and known domains; this is clearly an important restriction, but one that

* Corresponding author.
E-mail addresses: cassiopc@usp.br (C.P. de Campos), fgcozman@usp.br (F.G. Cozman), eduardo.ol@gmail.com (J.E.O. Luna).

1570-8683/$ – see front matter
doi:10.1016/j.jal.2007.11.002

holds in many applied situations, for instance in support systems for diagnosis or in automated planning. Section 3 presents our proposal and comments on inference algorithms. Even though the worst-case complexity of inferences is high, in practice the graph topology can be exploited to obtain conceptual and computational gains. Section 4 presents an extended example in knowledge representation.

In Section 5 we offer some methodological hints on how to build knowledge bases that avoid inconsistency. First, there are logical inconsistencies and probabilistic inconsistencies to avoid. Second, the usual Markov condition on directed graphs, which we adopt, may in some circumstances impose stochastic independence on two sentences that are in fact logically dependent, causing some events to receive probability zero—even though we do not have an inconsistency, we do have a pathological situation where the qualitative pieces of the language force zero probabilities to appear. The suggestions in Section 5 should be useful to avoid such pitfalls.

2. Probabilistic logics and independence

This section reviews significant efforts in the literature of probabilistic logic. We also use this review to discuss our design choices regarding conditioning and credal set semantics, and to present our perspective on Markov conditions and on "syntactic" independence. To the best of our knowledge, a historical review containing both "classical" work on probabilistic logics and more recent work on "graph-theoretic" logics is not available, so we attempt to sew these trends together in this section.[1]

2.1. Assessments and credal sets in "classical" probabilistic logics

Probabilistic logic is not a new field; its origins are generally attributed to Boole, who looked into assessments of the form $P(\phi) = \alpha$, where ϕ is a formula in propositional logic [7]. Boole's attempts to manipulate such assessments were not entirely successful; more than a century later, Hailperin presented a solution based on linear programming [37]. Every probability $P(\phi)$ is then a linear combination of probabilities over truth assignments ω: $P(\phi)$ is interpreted as $\sum_{\omega \models \phi} P(\omega)$, where $\omega \models \phi$ denotes satisfaction of formula ϕ by ω [40]. An *inference* is the calculation of $\min / \max P(\theta)$ for some formula θ, and these are linear programs. The computational problem is not negligible, because the number of assignments is exponential on the number of propositions. Problems of moderate size have been tackled by various algorithms [40], but it seems that inferences do not display the phenomenon of "phase transition" that has been so valuable in propositional satisfiability [16, Section 2.1].

Example 1. Take propositions A, B, C and assessments $P(B) = P(A \to \neg C) = P(B \to C) = 2/3$. Then $P(A) \in [1/3, 1]$ and $P(A \wedge B) \in [0, 2/3]$.

A somewhat different tradition in probabilistic logic stems from the work of de Finetti [21]. Here the emphasis is on subjective assessments over events that may be related through propositional formulas. The "fundamental theorem" of de Finetti in essence describes the linear constraints that must be constructed for inference [20]. The application of linear programming to that result appeared slightly later than Hailperin's reformulation of Boole's probabilistic logic [8]. These ideas have been actively pursued by several researchers, often under the banner of *coherency-based reasoning* [12,59,81]. A noteworthy characteristic of the coherency-based framework is that conditioning is based on *full conditional measures* [25,57]: the conditional probability $P(\phi|\theta)$ is taken as a primitive concept, satisfying the properties of a normalized nonnegative measure plus the condition $P(\phi \wedge \theta) = P(\phi|\theta)P(\theta)$. Consequently, $P(\phi|\theta)$ is well defined even if $P(\theta) = 0$. The result is a promising theory where an assessment $P(\phi|\theta) = \alpha$ can be given meaning even if $P(\theta) \leqslant \beta$ is the only available assessment about θ. However, for our present purposes, there are difficulties still to be sorted out regarding the definition of independence in the coherency-based framework [17,80].

In this paper we adopt a rather conservative position concerning conditioning: we take that an assessment such as $P(\phi|\theta) \geqslant \alpha$ is just a shorthand for the inequality $P(\phi \wedge \theta) \geqslant \alpha P(\theta)$. The judgement of stochastic independence of ϕ and φ is interpreted as the constraint $P(\phi \wedge \varphi) = P(\phi)P(\varphi)$, and the judgement of stochastic conditional independence

[1] We have structured this section so that its material is complementary to our paper on Propositional Probabilistic Logic (PPL) networks [16]; some repetitions were kept so as to make the present paper self-contained.

of ϕ and φ given θ is interpreted as the constraint $P(\phi \wedge \varphi \wedge \theta)P(\theta) = P(\phi \wedge \theta)P(\varphi \wedge \theta)$. Thus we effectively strip from the language any genuine assessment of conditional probability. We however allow *conditional inferences* that aim at answering: "Among all probability measures over assignments that satisfy assessments, what is the smallest value of $P(\phi|\theta)$ that is well defined?". That is, we seek $\min_{P(\theta)>0} P(\phi \wedge \theta)/P(\theta)$. (Alternative interpretations for conditional inferences, as well as additional discussion of conditioning on events of zero probabilities, can be found elsewhere [16, Section 2.3].)

First-order probabilistic logics have been studied by several philosophers of science [43,58]. In particular, Gaifman and Snir consider probability a function over sentences of a first-order language [30]. Gaifman and Snir adopt a *model-based* semantics by interpreting $P(\phi) \geqslant \alpha$ as $P(\text{Mod}(\phi)) \geqslant \alpha$, where $\text{Mod}(\phi)$ is the set of all interpretations for the sentence ϕ (Gaifman and Snir consider a fixed domain containing the natural numbers). Another notable effort is the probabilistic logic of Keisler [52,53] and Hoover [42], who explore a *domain-based* semantics where probability measures are taken over the *domain* of individuals, and not over the interpretations. More recently, considerable research on first-order probabilistic logic has been sparked by Nilsson's probabilistic logic [70]. The perspective is a little diverse from most predecessors: instead of just taking his language to be a medium for expressing facts about probability, Nilsson treats it as a generalization of logic, where truth values are real numbers instead of just 0 and 1. Nilsson considers assessments such as $P(\phi) = \alpha$, where ϕ is a sentence, and model-based semantics with probability measures over interpretations. Inferences are solved by linear programming; Nilsson also offers a maximum entropy method that always produces a single probability during inference.

The literature on propositional probabilistic logics grew quickly since Nilsson's proposal [1,26,29,31,36,51,79]; in particular, general assessments of the form $P(\phi|\theta) \in [\alpha, \beta]$ have been studied at length, together with exact and approximate inference algorithms. The developments in propositional approaches are reviewed by Hansen and Jaumard [40]. First-order probabilistic logics of various kinds have also appeared, exploring model-based semantics, domain-based semantics, or both [4,9,38,39]; again, assessments of the form $P(\phi|\theta) \in [\alpha, \beta]$, where ϕ and θ are now first-order formulas, are the most common object of study. Fragments of these logics have been applied to many topics, for instance logic programming [62,68] and description logics [34,44].

In most logics just discussed, consistency means existence of *at least one* measure satisfying assessments; there may be more than one satisfying measure. As sets of probability measures are often referred to as *credal sets* [61], we say that these logics have *credal set semantics*. Two obvious reasons to have a credal set semantics are flexibility and realism: we can accommodate incomplete, vague and imprecise assessments in the language. A third reason is the commendable desire to be mathematically general by allowing satisfaction of assessments by more than a measure. Yet another argument for credal set semantics is that we may want inferences to take into account assessments actually announced by a subject, and nothing more. Finally, we should also want probabilistic logic to offer a deductive system where assessments are collected and their consequences are then explored. We should *not* require various checks to be executed beforehand; for instance it would be awkward to list all logically impossible events from the outset so as to assign probability zero to them. For all these reasons, we adopt credal set semantics throughout, where our measures are over the set of interpretations.

2.2. The quest for guaranteed uniqueness in first-order probabilistic logics

In dealing with probabilistic logics with credal set semantics, one has to face the fact that a small number of assessments may lead to *inferential vacuity*; that is, the situation where inferences of interest yield very wide probability intervals. Inferential vacuity can be mitigated, or even eliminated, using judgements of stochastic independence.

Example 2. In Example 1, suppose $P(A) = 2/3$, so that assessments now imply $P(A \wedge B) \in [1/3, 2/3]$. If additionally A and B are stochastically independent, then $P(A \wedge B)$ is precisely $4/9$.

It is natural to imagine that stochastic independence should be inserted into probabilistic logics not just to avoid inferential vacuity, but also to help "decompose" large knowledge bases. There has been in fact active research on "graph-theoretic" first-order probabilistic logics that rely on the theory of Bayesian networks to do so.

A Bayesian network consists of a directed acyclic graph with n nodes, where each node is associated with a variable X_i. The graph encodes judgements of stochastic conditional independence through a Markov condition:

every variable X_i is independent of its nondescendants given its parents $\text{pa}(X_i)$. This Markov condition leads to a unique factorization of the joint distribution: $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{pa}(X_i))$ [14,71].[2]

The first attempts to marry Bayesian networks and elements of first-order logics seem to have been motivated by the desire to broaden the scope of Bayesian networks, and not to extend first-order logics. The goal was to specify Bayesian networks over sets of individuals, by replicating basic blocks of propositions or variables, in scenarios of interest to statistics [77] and artificial intelligence [82]. These replication rules were sometimes diagrammatic [2,56,63,77], and sometimes explicitly based on first-order logic [5,35,45,69,72]. These latter proposals spent significant effort in dealing with the following difficulty. Suppose a directed acyclic graph is specified where each node is associated with a relation, and suppose binary relation s is the sole parent of unary relation r. Attempting to mimic the Bayesian network scheme, we have to assess $P(r(x)|s(x, y))$. We can easily interpret the free variable x that appears both in the conditioned and conditioning formulas: if we were to "propositionalize" the assessment, we would replicate it for each individual x in the domain. But it is harder to interpret the free variable y that appears only in the conditioning formula, and various proposals have been made on how to specify the probability of $r(x)$ conditional on an arbitrary number of "propositionalized" parents.

We later employ a particular proposal for extending Bayesian networks with elements of first-order logic, Jaeger's *relational Bayesian networks* [45,49,50]. A relational Bayesian network specifies a *single* measure over grounded relations, using a directed acyclic graph where each node is a relation. Every relation is then associated with a *probability formula* that indicates how to compute the probability of the relation for any individual in its domain. Section 3.2 contains additional discussion on relational Bayesian networks.

Many graph-theoretic first-order probabilistic logics have appeared in the last decade; this surge of interest is possibly due to the connections made with machine learning applications. The availability of vast amounts of relational data does seem to justify the interest in statistical models with relations among individuals [28,32,41]. Graph-theoretic first-order probabilistic logics have found application in inductive logic programming, and in fields such as speech and image understanding. We do not attempt to provide a complete list of all current applied work related to graph-theoretic first-order probabilistic logics, as this would take us unnecessarily far afield. Note that, while some probabilistic logics are "directly extending Bayesian networks", others are preoccupied in "generating Bayesian networks from rules" [13,18,27,54,55,65,75]. An excellent review of these (and related) languages has been published by Milch and Russell [66].

It seems worth emphasizing that all of these graph-theoretic logics guarantee, using various assumptions, that assessments are always satisfied by a unique probability measure. At this point we should pause and note that a rough division of existing first-order probabilistic logics in two classes is possible. First, there are logics with credal set semantics and where stochastic independence is not represented. Second, there are logics where graphs and stochastic independence are used to produce a single measure for any given set of well-formed formulas. With notable exceptions [3,78], few logics have tried to combine features of these two classes of probabilistic logic. This combination is exactly what we try in this paper.

2.3. *Undirected graphs, factorization, and Markov conditions*

A promising approach to probabilistic logic is to encode judgements of stochastic independence through undirected graphs; that is, to resort to the theory of Markov networks [74,76]. As logical relations do not seem to carry any particular "direction", the use of undirected edges seems adequate. Besides, Markov networks can contain cycles, thus removing the irritating "acyclicity" condition of Bayesian networks.

However it is not easy to mix logical sentences and Markov conditions for undirected graphs, as the usual Markov condition fails in the presence of zero probabilities [14]. This fact prevents us from drawing a graph and hoping that it will produce a factorization of the probability measure over interpretations (note that getting such a factorization is in the end the main point of graph-theoretic first-order logics). These difficulties with the Markov condition are circumvented in *Markov logic*, a notable language for its generality [74]. In Markov logic *any* formula ϕ_i can be

[2] A Bayesian network may have "deterministic" parts containing zero/one probabilities [60], and conditioning events may be described using logical formulas [24]; the presence of determinism may then be exploited computationally during inference. These situations do not really characterize a probabilistic logic and are not examined further in this paper (we comment on it elsewhere [16, Section 3.2]).

associated with a weight w_i, and the joint distribution over all grounded relations is assumed to be proportional to $\exp(\sum w_i \phi_i)$; that is, the joint distribution factorizes *as if* a Markovian decomposition condition were in place.[3]

In this paper we wish to follow the usual strategy where a Markov condition with intuitive content is adopted. Then judgements of stochastic independence are obtained from such condition, and finally a factorization of the joint distribution is produced. For this reason, we insist on directed acyclic graphs. It is possible that undirected graphs will ultimately be the tool of choice for probabilistic logic, and Markov logic makes a strong case in that direction; however we feel that more thought should be given to appropriate Markov conditions before the final verdict is agreed upon.

2.4. "Syntactic" independence

In Markov logic, the graph is never explicit; the structure of the graph is obtained syntactically from the logical formulas. In this paper we proceed differently: judgements of stochastic independence are obtained from a graph that must be explicitly specified. Indeed we let the graph be specified separately from the logical formulas. One reason to do so is to increase the flexibility of the language. However, the following example suggests that syntactic extraction of a graph from formulas may fail even in seemingly "obvious" situations.

Example 3. Take propositions A, B, C, formulas $A \to \neg C$ and $B \to C$, and assessments $P(A) = P(B) = 2/3$. It might seem that we could syntactically extract the following graph from the formulas: $A \to C \leftarrow B$. The Markov condition for this graph would then imply that A and B are stochastically independent. However the logical sentences imply $\neg(A \wedge B)$; thus $P(A|B) = 0 \neq P(A) = 2/3$. The result is that the formulas, assessments and the Markov condition are inconsistent without an edge connecting A and B; however this edge can be determined only by analysis of the logical formulas.

Hence, we should not in general expect that "syntactic independences" extracted from logical sentences cohere with stochastic independence, at least for directed acyclic graphs. Our reaction is to separate stochastic independences, logical sentences and probabilistic assessments.

Obviously one may find it advantageous to generate a graph just by reading syntactic independences from the available logical sentences; that is a fine strategy for us, but it may cause the resulting knowledge base to become inconsistent if not done properly. We leave for the future the development of algorithms that can determine all stochastic independences that are implied by a given set of logical sentences (we return to this issue at the end of Section 5).

2.5. PPL networks

We have recently investigated the following graph-theoretical scheme for propositional probabilistic logic [16]. Consider a triplet $(\mathcal{G}, \mathcal{L}, \mathcal{A})$ containing: a graph $\mathcal{G}$ where nodes are associated with propositions; a set of propositional formulas $\mathcal{L}$; and a set of probabilistic assessments $\mathcal{A}$ over propositional formulas. Assume a Markov condition: a proposition is stochastically independent of its nondescendants given its parents. We call such a triplet a *Propositional Probabilistic Logic* (*PPL*) network [16].

Example 4. Take propositions A, B, C. Consider the graph $\mathcal{G}$ as $A \to C \leftarrow B$. The Markov condition for $\mathcal{G}$ imposes stochastic independence of A and B. Now take the set of logical sentences $\mathcal{L} = \{A \to \neg C\}$, and the set of assessments $\mathcal{A} = \{P(B) = 2/3, P(B \to C) = 2/3\}$. An inference is $P(A) \in [0, 1/2]$.

Assessments in $\mathcal{A}$ may be interval-/set-valued, and they may involve logical sentences themselves. We explicitly separate $\mathcal{L}$ from $\mathcal{A}$ because, at least in the propositional case, the complexity of inferences depends on $\mathcal{L}$ and $\mathcal{A}$ in different ways (in the first-order case this distinction does not seem apparent, and it is an open problem whether it

[3] To express a logical constraint to the effect that ϕ_i holds, the associated weight w_i must be set to infinity. The presence of infinite weights produces a uniform distribution over the interpretations that satisfy formulas with infinite weights [74]. Consequently, once we have strict logical formulas, probabilistic assessments are automatically discarded. This is different from our proposals, as we wish to satisfy all logical formulas and probabilistic assessments.

holds in some form). Note that if a proposition is not drawn explicitly in the graph, then it is stochastically independent of all others (it is stochastically independent of its nondescendants given its parents).

PPL networks are directly related to Andersen and Hooker's *Bayesian logic* [3]. Their logic is based on directed acyclic graphs associated with the Markov condition and with assessments that may contain propositional sentences. We have added an explicit treatment of complexity issues, leading to the separation of $\mathcal{L}$ and $\mathcal{A}$ already alluded to; our inference algorithms for PPL networks [16] are also significantly different from Andersen and Hooker's.

In PPL networks: (i) assessments are associated with formulas; (ii) stochastic independence is part of the language; (iii) graphs are used to represent stochastic independences in a modular fashion; (iv) graphs are not extracted syntactically from logical formulas and assessments; (v) formulas and assessments may be inconsistent, or they may be satisfied by one or more probability measures. The main insight in the analysis of PPL networks is that the complexity of inference depends on the interactions between graph, sentences and assessments [16]. Complexity depends on the sparseness of the graph and on the "closeness" (in the graph, enlarged with arcs that represent logical formulas) of propositions in sentences and assessments. The situation is similar to Bayesian networks: the sparser the graph, the better.

3. Networks for relational probabilistic logic

In this section we propose a language with first-order constructs, where one can express logical formulas, probabilistic assessments and judgements of stochastic independence. As announced in Section 1, the language combines directed acyclic graphs and credal set semantics. The strategy we pursue is simple: we expand the language of PPL networks as much as possible by introducing relational features into it, while we try to keep the inference algorithms within the reach of existing methods. However, despite the simplicity of approach, the number of ingredients we mix together is relatively large:

(1) We start with Jaeger's relational Bayesian networks (reviewed in Section 3.2). Our first step is to allow the probability formulas in these networks to return *sets of probability measures* rather than just single measures. For binary variables, particularly for indicators of logical formulas, these sets of probability measures are just interval-valued probabilities.[4]
(2) Our second step is to allow logical sentences, possibly associated with probabilities, to be added to such "interval-valued relational Bayesian networks". These sentences and assessments are additional constraints on the probability measures, just like the constraints in the sets $\mathcal{L}$ and $\mathcal{A}$ in PPL networks; they are separately specified and are not produced syntactically from one another. Note that such "free-form" constrains are not allowed in standard relational Bayesian networks.

We work throughout with a vocabulary containing relations, Boolean operators (disjunction, conjunction, negation), quantifiers (existential and universal), and logical variables; as usual a sentence is a formula without free logical variables. We do not allow functions over individuals, even though there may be functions in the definition of probability distributions (combination functions). The semantics of logical formulas is established by selecting a set of *individuals*, called a *domain* $\mathcal{D}$. We adopt the unique name assumption for all individuals, and restrict the discussion to finite and known domains; this is a serious limitation that must be lifted in the future. An *interpretation* assigns relations in the domain to relations in the vocabulary [67].

3.1. Typed and functional relations

Suppose we have a relation $grade(x, y, z)$, and a domain containing 8 individuals: $\{Mary, John, Math1, Phil1, Eng1, A, B, C\}$. There are then 512 distinct instantiations. However if $x \in \{Mary, John\}$, $y \in \{Math1, Phil1, Eng1\}$ and $z \in \{A, B, C\}$, we only have 18 instantiations. The following definition even allows a relation to be restricted to a few combinations of its arguments; for example, *Mary* has a grade only for *Phil1* while *John* has grades for *Math1* and *Phil1*.

[4] We have already discussed a few situations where combination functions are partially specified in our previous publication [16].

Let r be a n-ary relation with arguments $x_1, \ldots, x_n$, with $x_i \in \mathcal{D}_i$, where $\mathcal{D}_i \subseteq \mathcal{D}$, and let $\mathcal{R} \subseteq \times_i \mathcal{D}_i$. The relation $r(x_1, \ldots, x_n)$ is *strongly typed* by $\mathcal{R}$ if arguments $x_1, \ldots, x_n$ are restricted to $\mathcal{R}$ (that is, there is a set $\mathcal{R}$ of tuples describing the valid arguments for r). The set $\mathcal{D}_i$ is the *type* of x_i in r; the set $\mathcal{R}$ the *strong type* of r. If instantiations of logical variables $(x_1, \ldots, x_n)$ are restricted to be in the set $\mathcal{R}(y_1, \ldots, y_m)$, where $y_1, \ldots, y_m$ are logical variables, then $\mathcal{R}(y_1, \ldots, y_m)$ is a *parameterized strong type*.

Types are not fundamental elements of the language per si, but they turn out to be crucial in order to control complexity of inferences. Typing schemes can also be found in relational Bayesian networks and in Markov logic.

We say that a relation is *functional* if there is only one possible instantiation for some of its arguments for each instantiation of the remaining arguments. For example, there is only one grade (either A, B or C) for each pair (x, y) with appropriate types. In this case we abuse notation by writing $grade(x, y)$, so that, for each instantiated (x, y), $grade(x, y)$ is an individual.

3.2. Relational Bayesian networks with interval-valued combination functions

A relational Bayesian network specifies a single measure over grounded relations in a finite known domain. We will consider sentences and assessments that require a credal set semantics; we will then need networks that can handle sets of probability measures. In this section we take the first step in that direction, by allowing the combination functions used in relational Bayesian networks to be interval-valued.[5]

It is necessary first to describe in slightly more detail the inner workings of relational Bayesian networks [45,49,50]. A relational Bayesian network can be viewed as a tuple $(\mathcal{D}, \mathcal{S}, \mathcal{G}, \mathcal{F})$, where $\mathcal{D}$ is a finite domain, $\mathcal{S}$ is a vocabulary containing relations, $\mathcal{G}$ is a directed acyclic graph where each node is a relation in $\mathcal{S}$, and $\mathcal{F}$ is a list of *probability formulas*, one for each node of $\mathcal{G}$.

Probability formulas are defined inductively as follows. First, a real number in $[0, 1]$ is a probability formula. Second, the *indicator function* of a formula ϕ is also a probability formula (the indicator function of ϕ is a random variable that takes value 1 if ϕ holds, and value 0 otherwise). Third, the convex combination $f_1 f_2 + (1 - f_1) f_3$, where f_1, f_2 and f_3 are probability formulas, is a probability formula. An example of probability formula is $0.9r(x) + 0.2(1 - r(x))$, where we use $r(x)$ to refer to its indicator. Note that this probability formula can be understood as "if $r(x)$ holds then 0.9 else 0.2".

The fourth and last kind of probability formula consists of a triplet containing a *combination function* g, a list of probability formulas $f_1, \ldots, f_n$, and a strong type $\mathcal{R}$ over the logical variables in the probability formulas $f_1, \ldots, f_n$. A combination function g takes a list of real numbers in $[0, 1]$ and returns a real number in $[0, 1]$. A probability formula $g(f_1, \ldots, f_n | \mathcal{R})$ is interpreted as follows. For each element of $\mathcal{R}$, each probability formula f_i is instantiated. As we run through all probability formulas $f_1, \ldots, f_n$ and all elements of $\mathcal{R}$, we produce a list of real numbers that is then processed by function g. Note that $\mathcal{R}$ may be a parameterized strong type over a subset of the logical variables in $f_1, \ldots, f_n$ (thus allowing *recursive* relational Bayesian networks to be built [49]). We assume without further discussion that parameterized strong types are specified as suggested by Jaeger [45,49,50], by a combination of quantifiers, additional relations and equalities. Note that probability formulas in essence determine the structure of $\mathcal{G}$ by indicating the dependences amongst relations; it is required that these dependences be acyclic.

Once a probability formula is specified for each relation, a Bayesian network can be easily generated for domain $\mathcal{D}$. The *propositionalization* of a relational Bayesian network is intuitively simple: each grounded relation becomes a node in a directed acyclic graph; the edges are produced by analyzing the probability formulas. For each probability formula f that specifies the probability of a relation r, there are edges into a particular grounded $r(a)$ from every grounded relation that inductively appears in f [45,49]. The probability formula is then used to specify the probability of $r(a)$ given its parents. For example, if $P(r(x))$ is given by $g(0.4s(x, y) | \forall y : y \neq x)$, then all grounded instances of $s(x, y)$, for $y \neq x$, are parents of $r(x)$ for every x. We denote the resulting directed acyclic graph by $\mathcal{G}'_{\mathcal{D}}$.

A Markov condition is assumed on the graph $\mathcal{G}'_{\mathcal{D}}$: every grounded relation is stochastically independent of grounded relations that are its nondescendants given the grounded relations that are its parents. Inference is a matter of processing the propositionalized Bayesian network.

[5] The expression "interval-valued Bayesian" is somewhat contradictory, as the Bayesian philosophy would require a single measure to be always specified [6], and perhaps a neutral term such as "interval-valued relational probabilistic networks" would be appropriate. We keep the former term to emphasize our reliance on Jaeger's relational Bayesian networks.

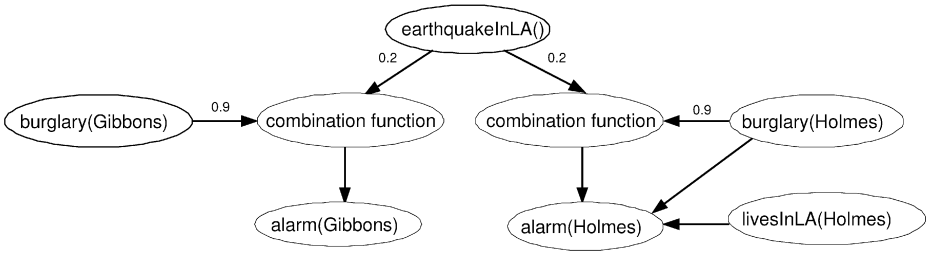


Fig. 1. Propositionalized graph $\mathcal{G}'_{\mathcal{D}}$ for the *Holmes* example; in Example 5 the combination functions are NoisyOR ones; in Example 6 the combination functions are CumulativeSynergy ones.

Example 5. Consider the well-known "Holmes example" [49, Sections 1, 2], expressed as a relational Bayesian network.[6] If v does not live in LA, then she sounds the alarm with probability 0.9 in case there is a burglary; if v lives in LA, then she sounds the alarm depending on whether there is a burglary and whether there is an earthquake. This is expressed approximately as follows:

```
P(alarm(v)) = if (livesInLA(v))
                  NoisyOR(
                       (if burglary(v) then 0.9 else 0.0),
                       (if earthquakeInLA() then 0.2 else 0.0)
                       | for v);
              else
                  (if burglary(v) then 0.9 else 0.0);
```

where the combination function $\mathit{NoisyOR}(p_i\colon i = 1, \ldots, n)$ returns, by definition, $1 - \prod_{i=1}^{n}(1 - p_i)$. (Other examples of combination functions are Mean, Maximum, Minimum.)

Suppose we have a domain containing two individuals, *Holmes* and *Gibbons*. Then the probability of `alarm`(*Holmes*) depends on `burglary`(*Holmes*) and additionally depends on `earthquakeInLA`() if `livesInLA`(*Holmes*) holds. Suppose only the relation `livesInLA`(*Gibbons*) holds; then the Bayesian network in Fig. 1 is obtained through propositionalization. □

To obtain interval-valued relational Bayesian networks, the only modification that is necessary is to allow intervals in $[0, 1]$ to replace real numbers in the definition of probability formulas. That is, an *interval-valued probability formula* is obtained inductively as follows. First, a real interval in $[0, 1]$ is a probability formula. Second, the indicator function of a formula ϕ is also a probability formula. Third, the convex combination $f_1 f_2 + (1 - f_1) f_3$, where f_1, f_2 and f_3 are probability formulas and interval arithmetic is used, is a probability formula. Fourth, an *interval-valued combination function* takes a list of real intervals in $[0, 1]$ and returns a real interval in $[0, 1]$; probability formulas are built from combination functions as described previously.

We continue to assume that a set of probability formulas is propositionalized as before, yielding a graph $\mathcal{G}'_{\mathcal{D}}$ from the graph $\mathcal{G}$ and the domain $\mathcal{D}$. That graph $\mathcal{G}'_{\mathcal{D}}$ is subject to the Markov condition, so that propositionalization now yields an interval-valued Bayesian network.

Example 6. Consider a modification of the original *Holmes* problem described in Example 5, where the prior probabilities are imprecisely known:

$$P\big(\texttt{earthquakeInLA}()\big) \in [0.01, 0.1], \qquad P\big(\texttt{burglary}(v)\big) \in [0.001, 0.01],$$
$$P\big(\texttt{livesInLA}(v)\big) \in [0.05, 0.15].$$

Also, suppose the probability formula for `alarm`(v) is:

[6] The Holmes example is encoded in Jaeger's Primula system, distributed at http://www.cs.auc.dk/~jaeger/Primula.

```
P(alarm(v)) = if (livesInLA(v))
                    CumulativeSynergy(
                          (if burglary(v) then 0.9 else 0.0),
                          (if earthquakeInLA() then 0.2 else 0.0),
                          (leak in [0.0, 0.1])
                          | for v);
              else
                 (if burglary(v) then 0.9 else 0.0);
```

Note that instead of a *NoisyOR* combination function described previously, we are using a new function called *CumulativeSynergy* that we now introduce. We define this function through *link* probabilities p_i and a *leak* probability α as follows:

$$\begin{aligned} &p(A|B_1,\ldots,B_n) = \alpha && \text{if } p_i = 0 \text{ for all } i,\\ &p(A|B_1,\ldots,B_n) = p_i && \text{if only } p_i > 0,\\ &p(A|B_1,\ldots,B_n) \geqslant \max\{p_i\} && \text{otherwise.} \end{aligned} \tag{1}$$

In words: if all parents of A are false, then the probability of A is the leak probability α; if only a parent of A is true, then the probability of A is the link probability p_i of this parent; otherwise, the probability of A is larger than any of the link probabilities for parents of A that are true. The cumulative synergy function does not define a single probability measure. Instead, it tries to capture a weak form of the following property of the *NoisyOR* function [15]: The more propositions B_i are true, the larger is $P(A|B_1,\ldots,B_n)$.

Fig. 1 is again obtained for a domain containing only *Holmes* and *Gibbons*, and where `livesInLA`(*Gibbons*) holds. With the algorithms discussed later (Section 3.4), we obtain

$$P(\texttt{alarm}(\textit{Holmes})) \in [0.0001, 0.0253],$$
$$P(\texttt{alarm}(\textit{Holmes})|\texttt{earthquakeInLA}()) \in [0.0108, 0.0388].$$

Note that inferences produce rather small intervals—even though only a few assessments are stated, the presence of stochastic independence greatly constrains the probabilities, obviating difficulties with inferential vacuity. □

3.3. Relational Probabilistic Logic networks

We now introduce free-form logical sentences and probabilistic assessments into our networks. We do it at once:

Definition 7. A *Relational Probabilistic Logic* (*RPL*) network consists of a tuple $(\mathcal{D}, \mathcal{S}, \mathcal{G}, \mathcal{F}, \mathcal{L}, \mathcal{A})$, where:

(1) $\mathcal{D}$ is a domain;
(2) $\mathcal{S}$ is a vocabulary containing (typed/functional) relations;
(3) $\mathcal{G}$ is a directed acyclic graph where each node is either a sentence or an open formula over $\mathcal{S}$;
(4) $\mathcal{F}$ is a set of interval-valued probability formulas where each probability formula specifies the probability of an open formula in a node of $\mathcal{G}$ and refers only to parents of that node;
(5) $\mathcal{L}$ is a list of sentences over $\mathcal{S}$;
(6) $\mathcal{A}$ is a list of assessments specified by tuples $(\phi_j, \varphi_j, \alpha_j, \beta_j)$, where ϕ_j, φ_j are sentences over $\mathcal{S}$ and $\alpha_j \leqslant \beta_j$ are numbers in $[0, 1]$, interpreted as $\alpha_j P(\varphi_j) \leqslant P(\phi_j \wedge \varphi_j) \leqslant \beta_j P(\varphi_j)$.

The graph $\mathcal{G}'_{\mathcal{D}}$ (obtained from $\mathcal{G}$ and $\mathcal{D}$ as indicated in the previous section) is subject to a Markov condition: sentences are conditionally independent of nondescendants given instantiations of its relations and parents. □

Even though we could have expressed some of the components of an RPL network through others, we feel that it is easier to appreciate the expressivity and the complexity of the language by keeping all components separately. Inferences may become very complex in a language this flexible; naturally, a sparse graph is likely to demand less

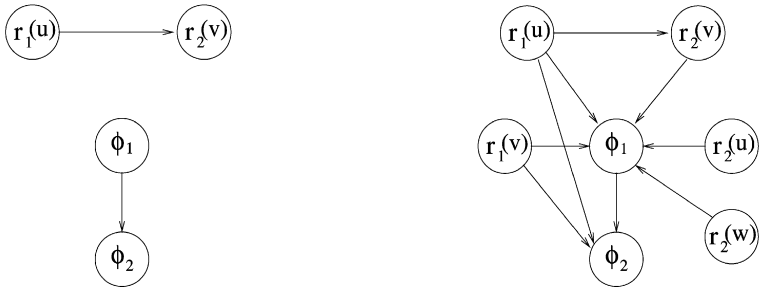


Fig. 2. RPL network. Left: independence graph with logical (based on instantiated relations) and sentence nodes. Right: graph with dependencies between sentences and instantiations of their relations.

computational effort than a dense graph with many associated first-order sentences. We now discuss an approach to inference that operates by propositionalization, using the Markov condition in Definition 7 to generate a PPL network from an RPL network.

The first step in propositionalizing an RPL network is to produce graph $\mathcal{G}'_{\mathcal{D}}$ from graph $\mathcal{G}$ and domain $\mathcal{D}$. The second step is to process $\mathcal{L}$ and $\mathcal{A}$ by linking sentences to instantiated relations as follows:

- For each sentence ϕ (or φ) in $\mathcal{L}$ or in $\mathcal{A}$:
  (1) Insert a sentence node for ϕ in the graph and an instantiated relation node $r_\phi(x)$ for each possible instantiation x of relations appearing in ϕ. These relation nodes will be parents of ϕ in the graph. Only instantiate r_ϕ according to its type and (possibly) functional property.
  (2) Verify whether ϕ or $\neg\phi$ holds, given each configuration of its parents, and encode the truth table for sentence ϕ as a conditional probability table of node ϕ given its parents. For an assessment in $\mathcal{A}$ we define conjunctions of instantiated parents of ϕ and specify the probability of ϕ given each such conjunction (equal to 1 when ϕ holds, and equal to 0 otherwise).

Denote by $\mathcal{G}'_p$, $\mathcal{L}_p$ and $\mathcal{A}'_p$ respectively the graph and the list of sentences and assessments after propositionalization. Note that $\mathcal{G}'_p$ may have many more dependencies among sentences than $\mathcal{G}$, because instantiations of relations may appear as parents of various sentences containing that relation. In $\mathcal{A}'_p$, assessments were created in both steps of the algorithm (when open formulas were replaced using $\mathcal{F}$ and then when conditional probability tables were created to encode truth tables of sentences given instantiated relations). $(\mathcal{G}'_p, \mathcal{L}_p, \mathcal{A}'_p)$ is a PPL network and $\mathcal{D}, \mathcal{S}, \mathcal{F}$ are now useless for inference as all necessary information is already available in the PPL network.

Example 8. Fig. 2 (left) shows a graph linking a few sentences. Suppose we have the domain $\mathcal{D} = \{u, v, w\}$ and the relations $\{r_1(x) : x \neq w\}$ and $r_2(x)$. Suppose further $\mathcal{L} = \{\phi_1 \doteq \forall x\ r_1(x) \wedge r_2(x), \phi_2 \doteq \exists x\ r_1(x)\}$, $\mathcal{A} = \{P(\phi_1) \leqslant 0.6, P(\phi_2) = 0.3, P(\phi_2 \wedge \phi_1) \geqslant 0.3 \cdot P(\phi_1)\}$, and $\mathcal{F}$ is empty. Note that the graph contains arcs between instantiated relations (from $r_1(u)$ to $r_2(v)$). So, during propositionalization, we do not insert new nodes for these two cases. Fig. 2 (right) shows the propositionalized network. We also encode in a conditional probability table for ϕ_2 that $P(\phi_2|r_1(u), r_1(v)) = 1$, $P(\phi_2|\neg r_1(u), r_1(v)) = 1$, $P(\phi_2|r_1(u), \neg r_1(v)) = 1$, because ϕ_2 holds when at least one of $r_1(u), r_1(v)$ holds.

Clearly, some relations are easier to treat than others:

- *typed* or *strongly-typed* relations: create a node for each instantiation of the arguments that comply with typing restrictions.
- *functional* relations: create a node for each instantiation of the arguments except for the dependent argument y; the domain of y now represents the set of categories for that node. For instance, suppose $r(x, y)$ is functional. Then, for each x, there is a single node $r(x)$ with values in the domain of y.

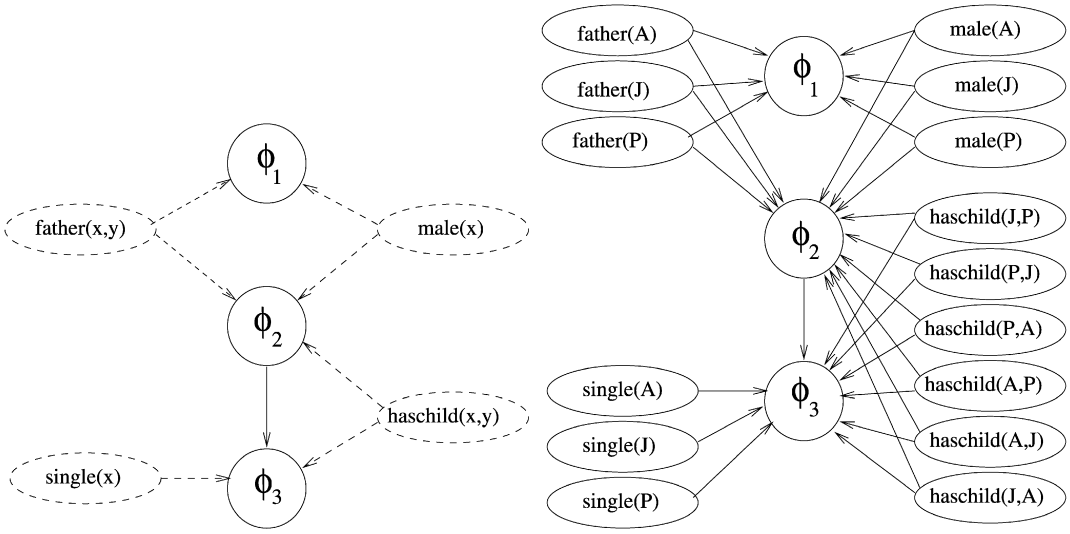


Fig. 3. Left: graph for Family example. Right: propositionalized graph.

Example 9. Suppose we have a knowledge base called *Family*, with domain $\{John, Peter, Ann\}$ for all arguments of relations, and where $\mathcal{L}$ contains:

$$\forall x, y\ father(x, y) \rightarrow male(y) \qquad (\phi_1)$$
$$\forall x, y\ hasChild(x, y) \wedge male(x) \rightarrow father(x, y) \qquad (\phi_2)$$
$$father(John, Peter), hasChild(Peter, Ann)$$
$$male(Peter), single(Peter)$$

and the assessment $0.8 \leqslant P(\phi_3) \leqslant 0.9$ in $\mathcal{A}$, where ϕ_3 is

$$\forall x, y\ \neg hasChild(x, y) \rightarrow single(x).$$

Suppose further that relation *father* is functional, relation *hasChild* is strongly-typed to $\{hasChild(x, y) : x \neq y\}$, and the only dependence between formulas is $\phi_2 \rightarrow \phi_3$. Fig. 3 (left) shows the network for this example; dotted arcs indicate in which sentences a relation appears. These relations are replaced by their instantiations in the propositionalized graph, as shown in Fig. 3 (right). The propositionalized graph is somewhat dense (even though almost bipartite) because all individuals (*Peter*, *John* and *Ann*) may appear in arguments of all relations. Note that relation *father* is functional, so there is only three nodes for it (with categories in $\{Ann, Peter, John\}$) instead of nine. The relation *hasChild* has a domain restriction, which implies six nodes for it (also instead of nine). In the graph, we use initial letters of names to simplify notation. □

3.4. Inferences using multilinear programming

An inference in an RPL network is the task of finding lower/upper bounds for the probability of a sentence θ. After propositionalization, we can apply the same techniques we have developed for PPL networks. We sketch the techniques and suggest [16] for a detailed description.

A propositionalized RPL network (that is, a PPL network) is a Bayesian network where probability values are not precisely known. That is, it is a *credal network* [16]. To produce an inference, each sentence/assessment is treated as a sub-query in this network: a symbolic version of an inference algorithm in Bayesian networks is run [19] to produce a collection of multilinear optimization problems. The fact that joint distributions factorize according to the graph topology is the key to produce multilinear programs of manageable size.

To generate the multilinear program, work with a sentence/assessment at a time:

- For each $\phi_j \in \mathcal{L}_p$, enforce $P(\phi_j) = 1$. As ϕ_j is already a node in the network, perform a symbolic inference for the marginal probability of ϕ_j in the propositionalized network, which creates a set of multilinear constraints that

define $P(\phi_j)$ in terms of the other network nodes. Then insert the constraint $P(\phi) = 1$ and this set of constraints into the multilinear problem.
- For each $A_j \in \mathcal{A}'_p$, enforce $\alpha_j P(\varphi_j) \leqslant P(\phi_j \wedge \varphi_j) \leqslant \beta_j P(\varphi_j)$, where ϕ_j and φ_j already have corresponding nodes in the network. Run two symbolic inferences for the probability of φ_j and for the probability of $\phi_j \wedge \varphi_j$ (this is just a joint query in a Bayesian network). These symbolic queries produce two sets of multilinear constraints. Insert these constraints into the multilinear problem, together with the constraint that $\alpha_j P(\varphi_j) \leqslant P(\phi_j \wedge \varphi_j) \leqslant \beta_j P(\varphi_j)$.
- Include a node for θ (the query) in the network, having as parents all interpretations of relations occurring in θ. Then perform the symbolic inference for the marginal probability of θ using the inserted node. Generated constraints must be included in the multilinear problem.
- In the final step, the multilinear program must be solved. Maximize and minimize the probability of θ. If there is a solution, it will satisfy all assessments. Otherwise no feasible solution exists for this multilinear problem and the RPL network is inconsistent.

Thus, for each sentence, we have run a symbolic version of our preferred algorithm for inference, which produces a set of multilinear constraints. The overall multilinear program is obtained by taking all multilinear constraints. Using this multilinear program, we can verify satisfiability or even evaluate the maximum/minimum of $P(\theta)$ [19].

4. Extended example: The university domain

In this section we present an extended example that illustrates interesting aspects of RPL networks. We use a well-known example in knowledge representation, the "university domain" [28,32,33,41]. The challenge is to represent relationships amongst professors, students, courses and registrations; we simplify matters here by dropping professors and registrations from the discussion.

A student is typically registered in several courses. If we assume that a student's grade depends on the student's intelligence and the course's difficulty, a conditional probability distribution might be defined as in Table 1.

Suppose there are two students, $\mathcal{D}_1 = \{Mary, John\}$; two courses, $\mathcal{D}_2 = \{Math1, Phil1\}$; three grades, $\mathcal{D}_3 = \{A, B, C\}$; and the qualitative descriptions $\mathcal{D}_4 = \{High, Low\}$. Relations of interest are $takes(x, y)$, with $x \in \mathcal{D}_1$, $y \in \mathcal{D}_2$; $intelligence(x, y)$, with $x \in \mathcal{D}_1$, $y \in \mathcal{D}_4$; $difficulty(x, y)$, with $x \in \mathcal{D}_2$, $y \in \mathcal{D}_4$; and $grade(x, y, z)$, with $x \in \mathcal{D}_1$, $y \in \mathcal{D}_2$, $z \in \mathcal{D}_3$. Suppose *intelligence* and *difficulty* are functional. Finally, suppose *grade* is strongly typed in that $grade(John, Math1, z)$, $grade(John, Phil1, z)$, and $grade(Mary, Phil1, z)$ are functional (z is uniquely defined).

Consider the following assessments:

$$P\big(takes(John, Phil1)\big) \leqslant 0.5 \qquad (A_1)$$
$$P\big(\exists\, x\ \forall\, y\ grade(x, y, A)\big) \leqslant 0.001 \qquad (A_2)$$
$$0.1 \leqslant P\big(\exists\, x\ \forall\, y\ takes(x, y)\big) \leqslant 0.15 \qquad (A_3)$$
$$0.05 \leqslant P\big(\exists\, y\ \forall\, x\ takes(x, y)\big) \leqslant 0.1 \qquad (A_4)$$

Assessments (A_1) to (A_4) express:

- *John* will attend *Phil1* with probability no larger than a half.
- It is very unlikely that student will achieve grade A in all courses.
- With low probability there exists a student who takes all courses.
- With low probability there exists a course taken by all students.

Table 1
Probability of a particular grade (A, B or C) given difficulty of course and intelligence of student

difficulty	*intelligence*	A	B	C
High	High	0.5	0.4	0.1
High	Low	0.1	0.5	0.4
Low	High	0.8	0.1	0.1
Low	Low	0.3	0.6	0.1

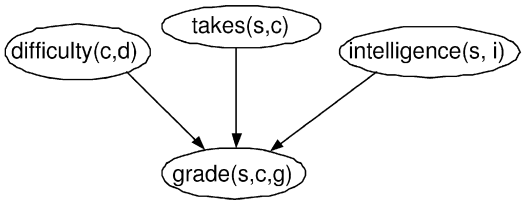


Fig. 4. Simplified graph for the university domain.

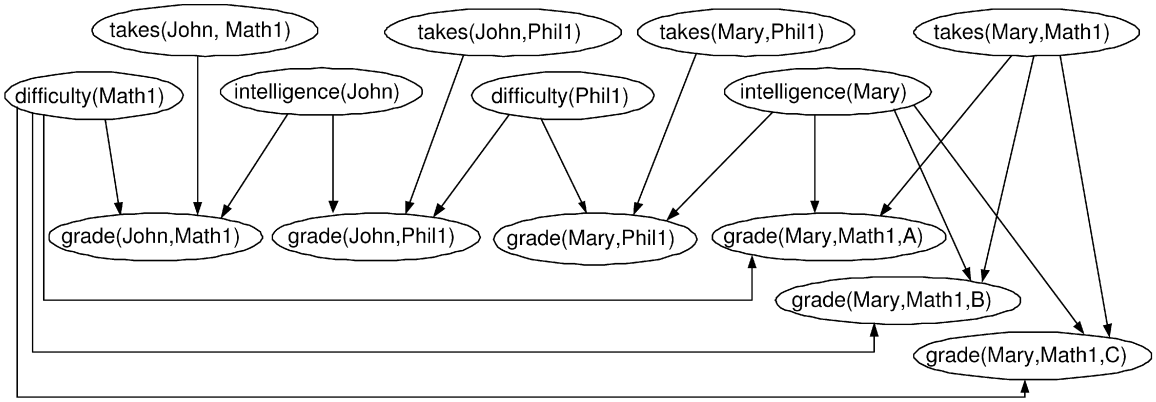


Fig. 5. The graph for the university domain after processing free variables (nodes not displayed are stochastically independent of all others).

Denote by ϕ_i the logical sentence that appears in assessment A_i (for example, ϕ_2 is $\exists\, x\, \forall\, y\, \mathit{grade}(x, y, A)$). Consider additionally the following logical sentences:

$$\forall\, x, y\, \big(\mathit{takes}(x, y) \iff \exists\, z\, \mathit{grade}(x, y, z)\big) \quad (\phi_5)$$
$$\mathit{takes}(\mathit{John}, \mathit{Math1}) \quad (\phi_6)$$
$$\mathit{takes}(\mathit{Mary}, \mathit{Phil1}) \quad (\phi_7)$$
$$\mathit{intelligence}(\mathit{John}, \mathit{Low}) \quad (\phi_8)$$
$$\mathit{intelligence}(\mathit{Mary}, \mathit{High}) \quad (\phi_9)$$
$$\mathit{difficulty}(\mathit{Math1}, \mathit{High}) \quad (\phi_{10})$$
$$\mathit{difficulty}(\mathit{Phil1}, \mathit{Low}) \quad (\phi_{11})$$

Finally, we define graph $\mathcal{G}$ for the domain, depicted in Fig. 4. This graph contains nodes with free variables. The first part of the propositionalization procedure is then to use these nodes and probability functions as templates. We assume that there is no particular restriction in probability functions (that is, they do not constrain conditional probabilities of our relations) so the expanded graph $\mathcal{G}'_{\mathcal{D}}$ is as in Fig. 5 and no additional probabilistic assessments need to be created during the template processing. We emphasize that the graph in Fig. 5 is not yet the final propositionalized graph, because nodes for logical sentences and probabilistic assessments are still not included.

At this point, it may seem that some relations are independent of others. However they may still have logical dependencies that will become clear in the final propositionalized graph (for example, in an extreme case where no graph is specified, everything is stochastically independent). Hence, we finish the propositionalization of the whole network: nodes for sentences are included, as well as dependencies between these sentences and their instantiated relations. The propositionalized graph $\mathcal{G}'_p$ is presented in Fig. 6, where:

- Sentences ϕ_1 and ϕ_6 to ϕ_{11} were not included in the graph because there already exist nodes for such sentences. For example, ϕ_1 is already node $\mathit{takes}(\mathit{John}, \mathit{Phil1})$, ϕ_6 is the node $\mathit{takes}(\mathit{John}, \mathit{Math1})$, and so on.
- We could have removed the nodes $\mathit{takes}(\mathit{John}, \mathit{Math1})$, $\mathit{takes}(\mathit{Mary}, \mathit{Phil1})$, $\mathit{intelligence}(\mathit{John})$, $\mathit{intelligence}(\mathit{Mary})$, $\mathit{difficulty}(\mathit{Math1})$, $\mathit{difficulty}(\mathit{Phil1})$ corresponding to ϕ_6 to ϕ_{11}, respectively, if we noted that they are implied by the sentences of $\mathcal{L}$. We kept the nodes in the graph to illustrate their dependencies and to be compatible with the propositionalization procedure.

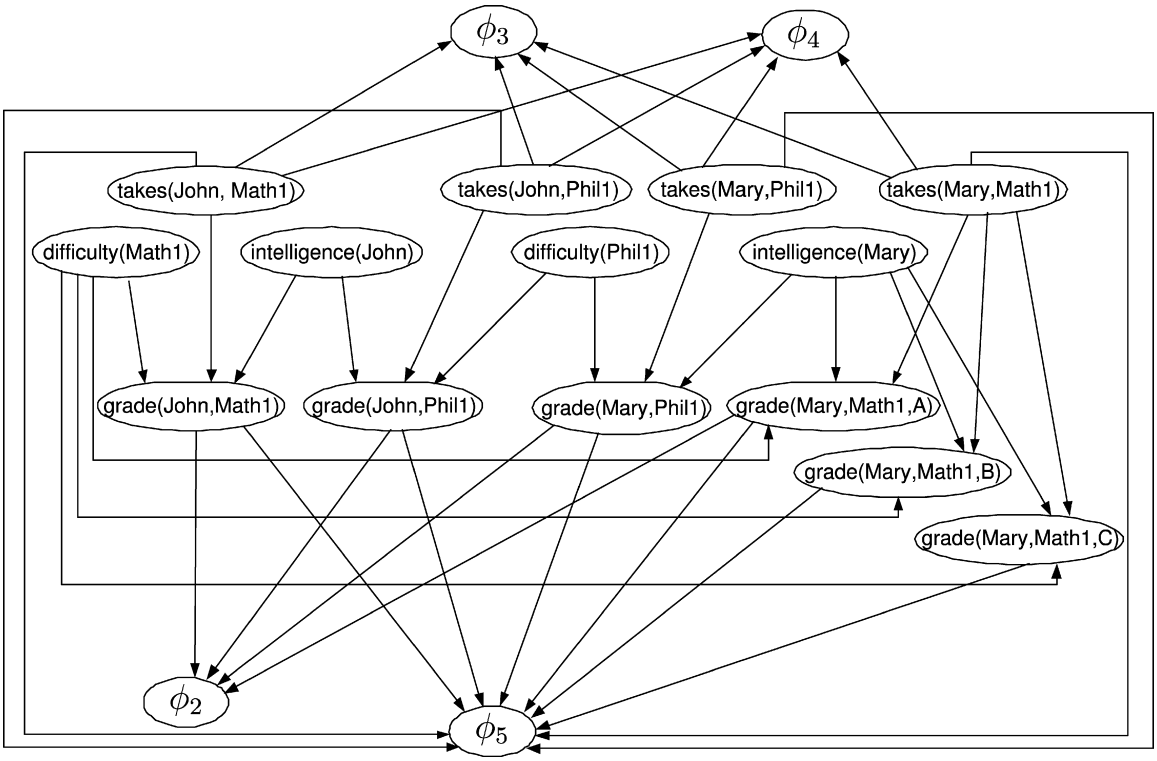


Fig. 6. A propositionalized version of the university domain.

- Sentences ϕ_3 and ϕ_4 have as parents all *takes* nodes, while ϕ_5 has all *takes* and *grade* nodes as parents. Sentence ϕ_2 has as parents only *grade* nodes that are possible instantiations of ϕ_2 (thus nodes $grade(Mary, Math1, B)$ and $grade(Mary, Math1, C)$ are not parents of ϕ_2).
- Concerning relation nodes, we have functional relations *intelligence* and *difficulty*, so they have only one argument. The same thing happens to nodes $grade(John, Phil1)$, $grade(John, Math1)$ and $grade(Mary, Phil1)$. As $grade(Mary, Math1, z)$ is not functional, the *grade* relation when restricted to these arguments still needs three arguments.
- Because a relation may be functional only for certain combinations of arguments, a relation may appear in the propositionalized network with different numbers of arguments. This idea can make the propositionalized network considerably smaller than it could be.

Inferences are produced by multilinear programs generated from the propositionalized network. It is interesting to comment on some inferences. First, we note that the whole network is inconsistent, because A_2 indicates that the probability of obtaining grade A on all courses is very low, but Table 1 and the other constraints in the database lead to a higher probability for this event. Section 5 comments on inconsistency when building such a knowledge base and some ways to spot the sources of trouble.

Removing the assessment A_2, we have a consistent network. Inferences are performed inserting query nodes for the desired query sentences and then using the multilinear techniques detailed in Section 3. To illustrate, some results are:

(1) The probability of *John* receiving grade *A* in *Phil1* given he takes all courses:

$$P\big(grade(John, Phil1, A) | \forall\ y\ takes(John, y)\big) \in [0.0, 0.03].$$

(2) The probability that *Phil1* is taken only by smart students:

$$P\big(\forall\ x\ takes(x, Phil1) \rightarrow intelligence(x, High)\big) \in [0.9, 1.0].$$

(3) The probability of having a student that achieves grade A in all courses:

$$P\big(\exists\, x\ \forall\, y\ \mathit{grade}(x, y, A)\big) \in [0.003, 0.04].$$

This last inference shows why assessment A_2 has led to inconsistency, as A_2 clashes with the inference. This last example also illustrates how inferences are processed, because the node ϕ_2 in the graph of Fig. 6 is exactly the node we should insert to perform this inference. The query is then the maximum and minimum values for the marginal probability $P(\phi_2)$.

5. Assembling consistent knowledge bases

The flexibility of our approach calls for some prudence when assembling a knowledge base, so as to avoid various kinds of inconsistency. In this section we offer a few methodological thoughts on this issue. We feel that there are preferred ways to insert elements into $\mathcal{G}$, $\mathcal{A}$, $\mathcal{L}$ or $\mathcal{F}$. First we note that $\mathcal{F}$ is processed separately from the others during propositionalization, that is, first we use probability functions of $\mathcal{F}$ to expand the graph, and only after that we look at other sentences. Thus, specifying the template part of $\mathcal{G}$ (nodes associated to open formulas) and $\mathcal{F}$ are natural initial steps to model the problem. This is similar to the specification of a relational Bayesian network.

At any rate, suppose $\mathcal{G}, \mathcal{L}, \mathcal{A}$ begin empty, and one inserts a node, edge, sentence, or assessment into one of these sets at a time, as in an incremental construction of the network. Remember that elements not present in $\mathcal{G}$ are meant to be independent of all others by the Markov condition. At some point, suppose one inserts a node in $\mathcal{G}$. In this case, an inference on the sentence just inserted can indicate whether the new node is logically consistent; moreover, it can indicate whether its probability must be set to zero to make the whole model consistent. Addition of arcs into $\mathcal{G}$ offers no reason for concern, as they imply removal of nonlinear constraints during inference. We must only verify that no directed cycle is created by arcs.

Suppose now one inserts a logical constraint into $\mathcal{L}$. It may happen that the new constraint makes the whole network inconsistent, or that the new constraint lowers the probability of some events to zero. The second situation may be somewhat perplexing; it tends to happen when a logical dependence clashes with a stochastic independence. It is possible to avoid these situations with a simple idea: each time a new constraint is inserted into $\mathcal{L}$, we must run an inference to check which probability values are admissible for this new constraint. So, before inserting the logical constraint ϕ, we verify whether $\mathcal{L} \cup \{\phi\}$ is satisfiable or not. If not, the model is inconsistent. Now, if consistency is maintained, we evaluate $P(\phi)$. If $\max P(\phi) < 1$, then we find that the problem will become unsatisfiable.

Consider the insertion of a probabilistic assessment $\alpha P(\varphi) \leqslant P(\phi \wedge \varphi) \leqslant \beta P(\varphi)$. First we evaluate $P(\varphi)$. If $\min P(\varphi) > 0$, then we must have $P(\phi|\varphi) \in [\alpha, \beta]$ (an inference can verify that), for otherwise the whole network becomes inconsistent. If $\min P(\varphi) = 0$, then the network together with this new assessment certainly has a feasible solution, although this solution may be undesirable as it may force events to take probability zero.

After verifying that an addition to the network leads to an undesirable state, a possible path is to look for other constraints that, if removed, would turn the network into a consistent state again (model checking can be used to do so [10,11,64]). We must test the feasibility of each constraint, or even choose k of them at a time—we leave for future work a thorough exploration of strategies to correct inconsistencies. This same idea could be useful to find which constraints have forced some probability value down to zero.

We believe that it is more natural to start by specifying the graph. Suppose instead the user starts by defining all the logical sentences first, and then moves to create the dependence graph. The difficulties just mentioned may all happen at once, and it becomes hard to find where the problem is. That is, if the graph is unspecified while $\mathcal{L}$ is filled in, there are no stochastic independences to take into account; so when the graph is finally built, several clashes between the existing sentences and stochastic independences may happen at once.

One might try to specify $\mathcal{L}$ first and then to extract a graph "syntactically" from the formulas—much like Markov logic, was discussed in Section 2.2. We have avoided this strategy in this paper, but it may well be a most useful path to creating consistent knowledge bases. However we feel that significant future work is yet to be done before algorithms for syntactic extraction of stochastic independence become fully understood (particularly for directed graphs).

Finally, it may happen that $\mathcal{G}$ and $\mathcal{L}$ are given at once. Then it becomes important to determine which logical dependences clash with stochastic independences indicated by the graph. A complete and efficient algorithm that performs such check is left for the future.

6. Conclusion

We have presented a few ideas on the intersection between relational logics and probabilities. At the risk of belaboring the argument, we can summarize our intentions as follows: we wish to provide tools where one can combine logical formulas, probabilities, and stochastic independence. We resort to graphs so as to obtain factorizations of joint distributions; given the difficulties in producing a sensible Markov condition for undirected graphs, we employ directed acyclic graphs. To enhance the flexibility of the language, and to ensure that it truly offers a tool to reason about probabilities, we resort to credal set semantics. We hope this flexibility (with all its consequences) is a positive aspect of our approach. Obviously we could work with even more flexible languages, but then it would become difficult to see how to exploit stochastic independence for inference. In any case, the flexibility of the language requires some methodological care when assembling knowledge bases. We feel that we have enriched the language of PPL networks as much as possible while still keeping the essence of inference algorithms untouched; further extensions will likely require substantially new inference algorithms.

Our inference algorithms are based on propositionalization; for this to work, we assume the domain to be known and finite, with the unique name assumption. This is a serious limitation of this proposal, one we would like to lift in future work. In any case, we feel that the current language can already be applied in a number of fields, as we have tried to show with necessarily abstracted examples.

Clearly, much remains to be done. Besides moving to infinite domains [46–48], it is important to investigate whether it is possible to conduct inference directly on the first-order representation, without propositionalization [22,23,73]. A much larger piece of future work is to consider even more expressive languages and their trade-offs between expressibility and complexity, as relatively little is known on the interaction between fragments of first-order logic and probabilistic assessments. Given this wealthy of material, no shortage of work lies ahead.

Acknowledgements

The work has received partial support from CNPq (grant 3000183/98-4) and FAPESP (grant 04/09568-0). The third author is supported by CAPES.

References

[1] M. Abadi, J.Y. Halpern, Decidability and expressiveness for first-order logics of probability, Information and Computation 112 (1) (1994) 1–36.
[2] R.G. Almond, Hypergraph grammars for knowledge based model construction, Technical Report StatSci 23, MathSoft Inc., 1994.
[3] K.A. Andersen, J.N. Hooker, Bayesian logic, Decision Support Systems 11 (1994) 191–210.
[4] F. Bacchus, Representing and Reasoning with Probabilistic Knowledge: A Logical Approach, MIT Press, Cambridge, 1990.
[5] F. Bacchus, Using first-order probability logic for the construction of Bayesian networks, in: Conference on Uncertainty in Artificial Intelligence, 1993, pp. 219–226.
[6] J.O. Berger, Statistical Decision Theory and Bayesian Analysis, Springer-Verlag, 1985.
[7] G. Boole, The Laws of Thought, Dover, 1958.
[8] G. Bruno, A. Gilio, Applicazione del metodo del simplesso al teorema fondamentale per le probabilità nella concezione soggettivistica, Statistica 40 (1980) 337–344.
[9] E. Charniak, R. Goldman, A semantics for probabilistic quantifier-free first-order languages with particular application to story understanding, in: Proc. Int. Joint Conference on Artificial Intelligence, 1989, pp. 1074–1079.
[10] E.M. Clarke, E.A. Emerson, A.P. Sistla, Automatic verification of finite state concurrent systems using temporal logic, ACM Transactions on Programming Languages and Systems 8 (2) (1986) 244–263.
[11] E.M. Clarke, O. Grumberg, D. Peled, Model Checking, MIT Press, 1999.
[12] G. Coletti, R. Scozzafava, Probabilistic Logic in a Coherent Setting, Trends in Logic, vol. 15, Kluwer, Dordrecht, 2002.
[13] V.S. Costa, D. Page, M. Qazi, J. Cussens, CLP(BN): Constraint logic programming for probabilistic knowledge, in: U. Kjaerulff, C. Meek (Eds.), Conference on Uncertainty in Artificial Intelligence, Morgan-Kaufmann, San Francisco, CA, 2003, pp. 517–524.
[14] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, New York, 1999.
[15] F.G. Cozman, Axiomatizing noisy-OR, in: European Conference on Artificial Intelligence, 2004.
[16] F.G. Cozman, C.P. de Campos, J.C.F. da Rocha, Probabilistic logic with independence, International Journal of Approximate Reasoning, in press (available online 7 September 2007, doi: 10.1016/j.ijar.2007.08.002).
[17] F.G. Cozman, T. Seidenfeld, Independence for full conditional measures, graphoids and Bayesian networks, Technical report, PMR, Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 2007.
[18] P.C.G. da Costa, K.B. Laskey, Of Klingons and starships: Bayesian logic for the 23rd century, in: Conference on Uncertainty in Artificial Intelligence, 2005.

[19] C.P. de Campos, F.G. Cozman, Inference in credal networks using multilinear programming, in: E. Onaindia, S. Staab (Eds.), Proceedings of the Second Starting AI Researchers' Symposium (STAIRS), IOS Press, Amsterdam, 2004, pp. 50–61.
[20] B. de Finetti, Foresight: its logical laws, its subjective sources, in: H.E. Kyburg Jr., H.E. Smokler (Eds.), Studies in Subjective Probability, Wiley, New York, 1964.
[21] B. de Finetti, Theory of Probability, vols. 1–2, Wiley, New York, 1974.
[22] R. de S. Braz, E. Amir, D. Roth, Lifted first-order probabilistic inference, in: International Joint Conference in Artificial Intelligence (IJCAI), 2006.
[23] R. de S. Braz, E. Amir, D. Roth, MPE and partial inversion in lifted probabilistic variable elimination, in: AAAI, 2006.
[24] R. Dechter, R. Mateescu, Mixtures of deterministic-probabilistic networks and their AND/OR search space, in: M. Chickering, J. Halpern (Eds.), Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2004, pp. 120–129.
[25] L.E. Dubins, Finitely additive conditional probability, conglomerability and disintegrations, Annals of Statistics 3 (1) (1975) 89–99.
[26] R. Fagin, J.Y. Halpern, N. Megiddo, A logic for reasoning about probabilities, Information and Computation 87 (1990) 78–128.
[27] D. Fierens, H. Blockeel, M. Bruynooghe, J. Ramon, Logical Bayesian networks and their relation to other probabilistic logical models, in: Int. Conference on Inductive Logic Programming, 2005.
[28] N. Friedman, L. Getoor, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: International Joint Conference on Artificial Intelligence, 1999, pp. 1300–1309.
[29] A.M. Frisch, P. Haddawy, Anytime deduction for probabilistic logic, Artificial Intelligence 69 (1994) 93–122.
[30] H. Gaifman, M. Snir, Probabilities over rich languages, testing and randomness, The Journal of Symbolic Logic 47 (3) (1982) 495–548.
[31] G. Georgakopoulos, D. Kavvadias, C.H. Papadimitriou, Probabilistic satisfiability, Journal of Complexity 4 (1988) 1–11.
[32] L. Getoor, N. Friedman, D. Koller, B. Taskar, Learning probabilistic models of relational structure, in: International Conference on Machine Learning, 2001, pp. 170–177.
[33] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: S. Džeroski, N. Lavrac (Eds.), Relational Data Mining, Springer-Verlag, 2001, pp. 307–335.
[34] R. Giugno, T. Lukasiewicz, P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA), Cosenza, Italy, September 2002, in: Lecture Notes in Artificial Intelligence, vol. 2424, Springer, 2002, pp. 86–97.
[35] S. Glesner, D. Koller, Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 1995, pp. 217–226.
[36] R. Haenni, Modeling uncertainty with propositional assumption-based systems, in: S. Parsons, A. Hunter (Eds.), Applications of Uncertainty Formalisms, in: Lecture Notes in Artificial Intelligence, vol. 1455, Springer, 1998, pp. 446–470.
[37] T. Hailperin, Boole's Logic and Probability: a Critical Exposition from the Standpoint of Contemporary Algebra, Logic, and Probability Theory, North-Holland, Amsterdam, 1976.
[38] J.Y. Halpern, An analysis of first-order logics of probability, Artificial Intelligence 46 (1990) 311–350.
[39] J.Y. Halpern, Reasoning about Uncertainty, MIT Press, Cambridge, MA, 2003.
[40] P. Hansen, B. Jaumard, Probabilistic satisfiability, Technical Report G-96-31, Les Cahiers du GERAD, École Polytechique de Montréal, 1996.
[41] D. Heckerman, C. Meek, D. Koller, Probabilistic models for relational data, Technical Report MSR-TR-2004-30, Microsoft Research, Redmond, WA, 2004.
[42] D.N. Hoover, Probability logic, Annals of Mathematical Logic 14 (1978) 287–313.
[43] C. Howson, P. Urbach, Scientific Reasoning: The Bayesian Approach, Open Court Publishing Company, Chicago, IL, 1993.
[44] M. Jaeger, Probabilistic reasoning in terminological logics, in: Principles of Knowledge Representation (KR), 1994, pp. 461–472.
[45] M. Jaeger, Relational Bayesian networks, in: D. Geiger, P. Pundalik Shenoy (Eds.), Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1997, pp. 266–273.
[46] M. Jaeger, Convergence results for relational Bayesian networks, in: Symposium on Logic in Computer Science (LICS), 1998, pp. 44–55.
[47] M. Jaeger, Reasoning about infinite random structures with relational Bayesian networks, in: Knowledge Representation (KR), Morgan Kaufman, San Francisco, CA, 1998.
[48] M. Jaeger, On the complexity of inference about probabilistic relational models, Artificial Intelligence 117 (2) (2000) 297–308.
[49] M. Jaeger, Complex probabilistic modeling with recursive relational Bayesian networks, Annals of Mathematics and Artificial Intelligence 32 (2001) 179–220.
[50] M. Jaeger, Relational Bayesian networks: a survey, Linkoping Electronic Articles in Computer and Information Science 6 (2002).
[51] B. Jaumard, P. Hansen, M.P. de Aragão, Column generation methods for probabilistic logic, ORSA Journal on Computing 3 (2) (1991) 135–148.
[52] H.J. Keisler, Hyperfinite model theory, in: R.O. Gandy, J.M.E. Hyland (Eds.), Logic Colloquium 76, North-Holland, Amsterdam, 1977, pp. 5–110.
[53] H.J. Keisler, Probabilistic quantifiers, in: J. Barwise, S. Feferman (Eds.), Model-Theoretic Logic, Springer, New York, 1985, pp. 509–556.
[54] K. Kersting, U. Dick, Balios—the engine for Bayesian logic programs, in: European Conference on Principles and Practice of Knowledge Discovery in Databases, 2004, pp. 549–551.
[55] K. Kersting, L. De Raedt, S. Kramer, Interpreting Bayesian logic programs, in: AAAI-2000 Workshop on Learning Statistical Models from Relational Data, 2000.
[56] D. Koller, A. Pfeffer, Object-oriented Bayesian networks, in: Conference on Uncertainty in Artificial Intelligence, 1997, pp. 302–313.
[57] P. Krauss, Representation of conditional probability measures on Boolean algebras, Acta Mathematica Academiae Scientiarum Hungaricae 19 (3–4) (1968) 229–241.
[58] H.E. Kyburg Jr., C.M. Teng, Uncertain Inference, Cambridge University Press, 1997.

[59] F. Lad, J. Dickey, M. Rahman, The fundamental theorem of prevision, Statistica 50 (1) (1990) 15–38.
[60] D. Larkin, R. Dechter, Bayesian inference in the presence of determinism, in: AI and Statistics (AI-STAT), 2003.
[61] I. Levi, The Enterprise of Knowledge, MIT Press, Cambridge, MA, 1980.
[62] T. Lukasiewicz, Probabilistic logic programming with conditional constraints, ACM Transactions on Computational Logic 2 (3) (2001) 289–339.
[63] S. Mahoney, K.B. Laskey, Network engineering for complex belief networks, in: Conference on Uncertainty in Artificial Intelligence, 1996.
[64] K.L. McMillan, Symbolic model checking—an approach to the state explosion problem, PhD thesis, SCS, Carnegie Mellon University, 1992.
[65] B. Milch, B. Marthi, D. Sontag, S. Russell, D.L. Ong, A. Kolobov, BLOG: Probabilistic models with unknown objects, in: IJCAI, 2005.
[66] B. Milch, S. Russell, First-order probabilistic languages: into the unknown, in: Int. Conference on Inductive Logic Programming, 2007.
[67] A. Nerode, R.A. Shore, Logic for Applications, second ed., Springer-Verlag, New York, 1997.
[68] R. Ng, V.S. Subrahmanian, Probabilistic logic programming, Information and Computation 101 (2) (1992) 150–201.
[69] L. Ngo, P. Haddawy, Answering queries from context-sensitive probabilistic knowledge bases, Theoretical Computer Science 171 (1–2) (1997) 147–177.
[70] N.J. Nilsson, Probabilistic logic, Artificial Intelligence 28 (1986) 71–87.
[71] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.
[72] D. Poole, Probabilistic Horn abduction and Bayesian networks, Artificial Intelligence 64 (1993) 81–129.
[73] D. Poole, First-order probabilistic inference, in: International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 985–991.
[74] M. Richardson, P. Domingos, Markov logic networks, Machine Learning 62 (1–2) (2006) 107–136.
[75] T. Sato, Y. Kameya, Parameter learning of logic programs for symbolic-statistical modeling, Journal of Artificial Intelligence Research 15 (2001) 391–454.
[76] B. Taskar, P. Abbeel, D. Koller, Discriminative probabilistic models for relational data, in: Conference on Uncertainty in Artificial Intelligence, Edmonton, Canada, 2002.
[77] A. Thomas, D. Spiegelhalter, W. Gilks, BUGS: A program to perform Bayesian inference using Gibbs sampling, in: J. Bernardo, J. Berger, A. Dawid, A. Smith (Eds.), Bayesian Statistics, vol. 4, Oxford University Press, 1992.
[78] H. Thone, U. Guntzer, W. Kießling, Towards precision of probabilistic bounds propagation, in: VIII Uncertainty in Artificial Intelligence Conference, 1992, pp. 315–322.
[79] L. van der Gaag, Computing probability intervals under independency constraints, in: P.P. Bonissone, M. Henrion, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence 6, Elsevier Science, 1991, pp. 457–466.
[80] B. Vantaggi, Graphical models for conditional independence structures, in: Second International Symposium on Imprecise Probabilities and Their Applications, Shaker, 2001, pp. 332–341.
[81] P. Walley, R. Pelessoni, P. Vicig, Direct algorithms for checking consistency and making inferences from conditional probability assessments, Journal of Statistical Planning and Inference 126 (1) (2004) 119–151.
[82] M.P. Wellman, J.S. Breese, R.P. Goldman, From knowledge bases to decision models, Knowledge Engineering Review 7 (1) (1992) 35–53.