Available online at www.sciencedirect.com
ScienceDirect
International Journal of Approximate Reasoning
49 (2008) 3–17
INTERNATIONAL JOURNAL OF APPROXIMATE REASONING
www.elsevier.com/locate/ijar

Probabilistic logic with independence

Fabio Gagliardi Cozman [a,*], Cassio Polpo de Campos [b], José Carlos Ferreira da Rocha [c]

[a] *Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brazil*
[b] *Escola de Artes, Ciências e Humanidades, USP Leste, Universidade de São Paulo, São Paulo, SP, Brazil*
[c] *Universidade Estadual de Ponta Grossa, Ponta Grossa, PR, Brazil*

Available online 7 September 2007

Abstract

This paper investigates probabilistic logics endowed with independence relations. We review propositional probabilistic languages without and with independence. We then consider graph-theoretic representations for propositional probabilistic logic with independence; complexity is analyzed, algorithms are derived, and examples are discussed. Finally, we examine a restricted first-order probabilistic logic that generalizes relational Bayesian networks.

Keywords: Probabilistic logic; Graph-theoretic models; Sets of probability distributions; Linear and multilinear programming

1. Introduction

In this paper, we discuss formalisms that combine probabilistic and logical reasoning. This promising mixture has old roots and has been rediscovered a few times. Pioneering efforts by Boole [4] and by de Finetti [23] were later generalized [33] and then surfaced in the artificial intelligence literature [55]. First-order probabilistic logics have received contributions from several fields, ranging from philosophy of science to knowledge representation [2,34,39,55]; in particular, significant attention has been devoted to relational probabilistic languages in artificial intelligence research [30,42,58].

We can divide these probabilistic logics in two groups. In one group, probability assessments are rather flexible and independence relations have a subsidiary status. In the second group, probabilistic assessments are required to specify a unique distribution and independence is an essential concept. In this paper, we contribute by exploring probabilistic logics where independence is a central character *and* assessments are not tied to uniqueness concerns.

In Section 2, we review relevant literature on propositional probabilistic logic, hopefully presenting matters from a perspective that may be of interest, as we include discussions on inferential vacuity, on phase transitions, on independence, and on zero probabilities. We then propose graph-theoretic representations inspired

* Corresponding author.
E-mail addresses: fgcozman@usp.br (F.G. Cozman), cassiopc@usp.br (C.P. de Campos), jrocha@uepg.br (J.C. Ferreira da Rocha).

0888-613X/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2007.08.002

by Bayesian networks, derive new complexity results and present algorithms and examples (Sections 3–5). In Section 6, we consider extensions of relational Bayesian networks [42]. Such networks combine fragments of first-order logic and graphs to obtain a reasonably flexible language.

This paper progresses through a hopefully cogent sequence of languages. Even though we are most interested in languages with enough power to represent practical problems, we start from the most unassuming one, propositional probabilistic logic without independence, and move slowly to more expressive languages. This gradually accelerating pace is deliberate, as we hope to present convincing arguments for our basic "design" choices.

2. Satisfiability, inferential vacuity, and independence

Consider first propositional probabilistic logic (an excellent historical and technical review is given by Hansen and Jaumard [36]). We have propositions and Boolean operators (conjunction, disjunction, negation). A *truth assignment* is a vector assigning either true or false to each proposition (assignments are often called *possible worlds* [55]). Satisfaction of a formula ϕ by a truth assignment ω is denoted by $\omega \models \phi$.

Probabilities are defined over truth assignments, and $P(\phi)$ is understood as $\sum_{\omega:\omega\models\phi} P(\omega)$ [29,36]. Our problem, referred to as *probabilistic satisfiability*, is to find whether or not there is a probability distribution satisfying a set of m assessments

$$P(\phi_i \wedge \varphi_i) \geqslant \alpha_i P(\varphi_i), \quad \text{with } \phi_i \text{ and } \varphi_i \text{ formulas in } n \text{ propositions.} \tag{1}$$

Note that Expression (1) is similar to $P(\phi_i|\varphi_i) \geqslant \alpha_i$; however, the former is valid regardless of whether $P(\varphi_i) = 0$ or not, while the latter is usually left undefined when $P(\varphi_i) = 0$. We avoid issues of zero probability in essence by assuming that all assessments are unconditional (Section 2.3 offers additional references on this issue).

2.1. Satisfiability and inference

If there is a distribution satisfying assessments in Expression (1), then the assessments are *satisfiable*. Obtaining the minimum/maximum value for a probability $P(\phi)$, subject to satisfiable assessments, is an *inference*. Given our assumptions, inferences are solved by linear programming. If instead we wish to produce *conditional inferences*, that is, minimum/maximum values for a conditional probability $P(\phi|\varphi)$, we must resort to linear fractional programming (note that linear fractional programs can be transformed into linear programs [10,36]).

Example 1. Consider the following hypothetical facts, inspired by Jaeger's example on birds [41]. (*Note*: in all examples, assessments are expressed through conditional probabilities, even though they should be understood as Expression (1); we also use material implication to express constraints that can be easily transformed to conjunctive normal form.) We have:

$\mathsf{AntarcticBird} \rightarrow \mathsf{Bird}$, $\mathsf{FlyingBird} \rightarrow \mathsf{Bird}$, $\mathsf{Penguin} \rightarrow \mathsf{Bird}$, $\mathsf{FlyingBird} \rightarrow \mathsf{Flies}$, $\mathsf{Penguin} \rightarrow \neg\mathsf{Flies}$, $P(\mathsf{FlyingBird}|\mathsf{Bird}) = 0.95$, $P(\mathsf{AntarcticBird}|\mathsf{Bird}) = 0.01$, $P(\mathsf{Bird}) \geqslant 0.2$, $P(\mathsf{FlyingBird} \vee \mathsf{Penguin}|\mathsf{AntarcticBird}) \geqslant 0.2$, $P(\mathsf{Flies}|\mathsf{Bird}) \geqslant 0.8$. Using fractional linear programming, we obtain $P(\mathsf{Penguin}|\neg\mathsf{AntarcticBird}) \in [0.000, 0.050]$ and $P(\mathsf{FlyingBird}|\mathsf{Bird} \wedge \neg\mathsf{AntarcticBird}) \in [0.949, 0.960]$.

The best exact algorithms for probabilistic satisfiability are based on the revised simplex algorithm; problems with (n,m) up to $(200, 1000)$ have been solved exactly, and many approximations have been proposed [36,37,44]. One might hope that *phase transitions* would be present in probabilistic satisfiability, much as they are present in SAT problems [49], so that hard problems would be concentrated in narrow regions. However, even a preliminary experiment reveals that phase transitions, if they exist at all, do not follow here the easy–hard–easy pattern observed in SAT. The graphs in Fig. 1 show time (wall-clock) to solve random probabilistic satisfiability problems. A problem was generated by fixing the number of propositions n, the ratio $r = m/n$ (m is the number of assessments), and the number of literals per clause k. Each literal in each clause is positive or negative with probability 1/2 [49], and each clause was associated with a randomly selected probability in $[0, 1]$. For each combination of n, r and k, 30 problems were generated and a revised simplex (with column

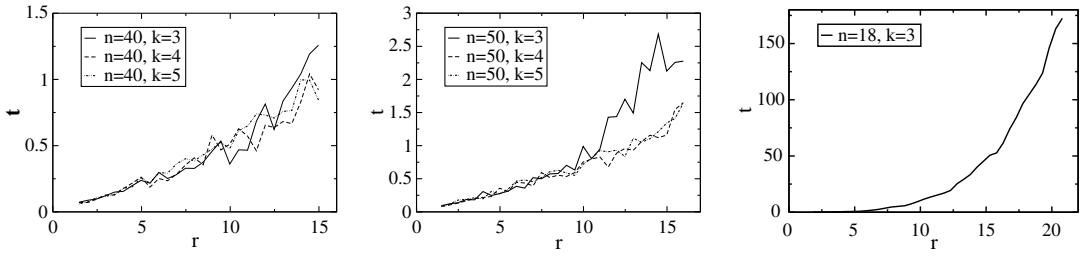

Fig. 1. Left and middle: time (s) to check satisfiability of random problems. Right: time (s) to check satisfiability of *satisfiable* random problems (note that these problems required more effort).

generation based on auxiliary 0/1 programs) was applied [36]. The rightmost graph deals with problems that are known to be satisfiable (the logical part is satisfiable and the probabilistic assessments were generated from a distribution over propositions). Overall, the pattern of all graphs in Fig. 1 suggests that the larger the problem, the harder it is to check its satisfiability.

Some intuition on the easy–hard pattern displayed in Fig. 1 can be derived from related results in the literature. Note first that the solution of probabilistic satisfiability with the revised simplex algorithm requires the solution of an auxiliary maximum weighted satisfiability problem at each iteration of the simplex procedure [36]. It has been reported that maximum weighted satisfiability has an easy–hard pattern rather than an easy–hard–easy pattern [5,72]. Thus it is not entirely surprising that probabilistic satisfiability fails the easy–hard–easy pattern that is so attractive in (usual) satisfiability.

2.2. Inferential vacuity and independence

Computational complexity is not the only reason for concern in probabilistic logic. Another difficulty is *inferential vacuity* — the concern that, due to the flexibility of the language, it is too easy to obtain vacuous inferences. A simple example of inferential vacuity is as follows. Suppose events A and B have no logical relation and $P(A) = 1/2$, $P(B) = 1/2$. Then bounds on $P(A \wedge B)$ are rather vacuous: $P(A \wedge B) \in [0, 1/2]$. This imprecision vanishes if A and B are independent; then $P(A \wedge B) = 1/4$. In fact, standard statistical models typically avoid inferential vacuity by assuming independence relations. We should likewise obtain compact and modular descriptions by adding independence relations to probabilistic logic.

Hence consider a three-place relation that denotes independence of ϕ and ξ given φ, where ϕ, ξ and φ are propositional formulas. The relation is denoted by $(\phi \perp\!\!\!\perp \xi|\varphi)$. We again form 2^n truth assignments for the n propositions of interest, and take probability measures over this set of truth assignments. An assessment $P(\phi) \geqslant \alpha$ is interpreted as before, and a relation $(\phi \perp\!\!\!\perp \xi|\varphi)$ now asserts independence among sets of truth assignments. We interpret conditional independence of ϕ and ξ given φ to mean the constraint $P(\phi \wedge \xi|\varphi) = P(\phi|\varphi)P(\xi|\varphi)$ whenever $P(\varphi) > 0$ over all distributions that satisfy additional assessments. (We note that several other concepts of independence have been proposed in the literature for situations where several distributions may satisfy assessments [8,11,16,21,50,69,70])

We again transform constraints on conditional probabilities into constraints on unconditional probabilities; thus the judgement of conditional independence $(\phi \perp\!\!\!\perp \xi|\varphi)$ is written as the nonlinear constraint:

$$\left(\sum_{\omega \models \phi \wedge \xi \wedge \varphi} P(\omega)\right) \times \left(\sum_{\omega \models \varphi} P(\omega)\right) = \left(\sum_{\omega \models \phi \wedge \varphi} P(\omega)\right) \times \left(\sum_{\omega \models \xi \wedge \varphi} P(\omega)\right). \tag{2}$$

To compute a tight upper bound on $P(\theta)$ for a propositional formula θ, we must find $\max \sum_{\omega \models \theta} P(\omega)$ subject to assessments and independence relations.

We can deal with conditional inferences such as $\max P(\theta|\vartheta)$ by computing $\max P(\theta \wedge \vartheta)/P(\vartheta)$ subject to linear and nonlinear constraints. For this procedure to be meaningful, we must first verify through nonlinear programming that $\min P(\vartheta) > 0$. In this paper, we leave $\max P(\theta|\vartheta)$ undefined whenever $\min P(\vartheta)$ is equal to zero (Section 2.3 comments on this).

In practice, it may be useful to rewrite expressions so as to eliminate ratios (similarly to the Charnes-Cooper transformation [10,36]). To do so, introduce a function q such that $q(\omega) = t\,P(\omega)$ where $t\,P(\vartheta) = 1$; to simplify notation, we use $q(\phi) = \sum_{\omega \models \phi} q(\omega)$ for any formula ϕ. We have to find $\max(t\,P(\theta \wedge \vartheta))$, or rather

$$\begin{aligned} &\max_{q} q(\theta \wedge \vartheta), \\ &\text{subject to : (1)} \quad m \text{ linear constraints } q(\phi_i \wedge \varphi_i) - \alpha_i q(\varphi_i) \geqslant 0, \text{ Expression(1)}; \\ &\qquad (2)\ r \text{ multilinear constraints on } q(\omega) \text{ (multiplying Expression(2) by } t^2); \\ &\qquad (3)\ \text{constraints } q\ (\omega) \geqslant 0 \text{ for all } \omega \text{ and } q(\vartheta) = 1; \\ &\qquad (4)\ \text{constraint } \sum_{\omega} q(\omega) = t. \end{aligned} \tag{3}$$

Program (3) takes us to nonconvex optimization; more precisely, to problems that can be expressed as *multilinear programs*. We have employed Sherali and Adams' algorithm for multilinear programming in our tests [66], because this algorithm iterates over a sequence of linear programs, and can thus benefit from column generation techniques in large problems [36].

Example 2. Consider the constraints and assessments in Example 1, and suppose that AntarcticBird and FlyingBird are independent (ability to fly does not depend on origin). Then $P(\mathsf{Penguin}|\neg\mathsf{AntarcticBird}) \in [0.000, 0.050]$ and $P(\mathsf{FlyingBird}|\mathsf{Bird} \wedge \neg\mathsf{AntarcticBird}) \in [0.950, 0.958]$.

Independence relations yield powerful constraints that reduce inferential vacuity. However, "unstructured" independence relations lead to difficult problems: there are 2^n truth assignments to handle, and additional nonlinear constraints. The techniques discussed so far cannot be directly applied.

2.3. Conditioning and zero probabilities

In this short section, we comment on our strategy regarding conditioning on events of zero probability. Readers who are comfortable with Expressions (1) and (3) may skip this section.

The strategy we have adopted is to avoid conditioning on events of zero probability by interpreting conditional assessments as unconditional constraints. A perhaps more elegant strategy would be to take conditional probability as a primitive notion, thus allowing conditioning on events of zero probability and allowing a direct interpretation of conditional assessments. Theories that take conditional probability as primitive have been advocated by de Finetti [23], Renyi [62], Popper [60], and many other researchers [7,31,38,61,71]. Inference then goes beyond linear programming [7,12,71]. Despite the added complexity of these algorithms, we could have relied on them in this paper. However, matters are not so simple when we move to the definition of independence.

When conditional probability is a primitive notion, a constraint such as Expression (3) does not fully capture the meaning of $(\phi \perp\!\!\!\perp \xi|\varphi)$, as Expression (3) can be satisfied even when $P(\phi|\xi \wedge \varphi) \neq P(\phi|\varphi)$ (this can happen if $P(\xi \wedge \varphi) = 0$). More stringent definitions have been considered in important work [17,35,67,68]. A difficulty here is that these more stringent definitions of independence fail properties that are important in our present setting; in particular, some of the graphoid properties that are the basis of Bayesian network theory [15,67]. Dealing with this issue, in any case a still controversial matter, would take us too far away from our interests in this paper.

When we deal with *sets* of probability measures, we must take into account an additional feature of conditioning. In producing the conditional probability $P(\theta|\vartheta)$, we may find that some possible measures have $P(\vartheta) = 0$ while others have $P(\vartheta) > 0$. In this paper, we have taken a somewhat extreme, but safe, strategy: a conditional probability $P(\theta|\vartheta)$ is defined only if $P(\vartheta) > 0$ for every satisfying probability measure. A theory where conditional probability is primitive would not need such precautions. However, even within the confines of the "standard" theory we might contemplate alternatives. For instance, we might take $P(\theta|\vartheta)$ to belong to the set $\{P(\theta\ \wedge \vartheta)/P(\vartheta)\colon P(\vartheta) > 0\}$ whenever this set is nonempty (this solution is similar to Walley's *regular extension* [70]). In fact, Program (3) does produce valid answers under this definition of conditioning; however, we do not explore this path further in this paper.

3. Graph-theoretic representations: PPL networks

As noted in the previous section, the flexibility of propositional probabilistic logic comes at a price in computational complexity. Besides, a language that is too unstructured may in fact overwhelm its users. In this section, we explore ways to attach "structure" to assessments using graph-theoretic tools.

3.1. PPL networks

Consider a set $\mathbf{X} = \{X_1, \ldots, X_n\}$ of binary variables, each representing a proposition; $\widehat{X}_i$ denotes the corresponding literal. A directed acyclic graph $\mathcal{G}$ is associated with $\mathbf{X}$: each variable is a node in $\mathcal{G}$. If edge $X \to Y$ belongs to $\mathcal{G}$, then X is a *parent* of Y; parents of Y are denoted by $\mathrm{pa}(Y)$. Nodes that can be reached from X through directed edges are *descendants* of X. We assume the graph to be endowed with the following Markov condition, taken literally from the theory of Bayesian networks [9,51,57]: X_i is conditionally independent from its nondescendants nonparents given its parents. This leads to the unique factorization $P(\mathbf{X}) = \prod_i P(X_i \mid \mathrm{pa}(X_i))$ for every distribution satisfying the Markov condition.

For the purposes of probabilistic logic, it makes sense to separate the graph and its Markov condition from the probabilistic assessments and logical constraints. The directed acyclic graph simply indicates independence relations amongst variables; assessments need not directly specify the probabilities $P(X_i|\mathrm{pa}(X_i))$. However, the following is required. Every assessment is either a logical formula ϕ_j that is asserted to be true, or an assessment $P(\phi_j \wedge \varphi_j) \in [\alpha_j P(\varphi_j), \beta_j P(\varphi_j)]$, where ϕ_j and φ_j are propositional formulas *in conjunctive normal form (CNF)* containing only variables in the graph. The reason why we restrict formulas to CNF is discussed in Section 4 and in the proofs of lemmas and theorems.

To understand the advantage of this graph-theoretic framework, consider the size of the multilinear program that must be solved for satisfiability and inference. While "unstructured" probabilistic logic requires manipulation of probabilities for 2^n truth assignments, a graph-theoretic model must only deal with the probabilities in the decomposition $\prod_i P(X_i \mid \mathrm{pa}(X_i))$.

Example 3. Consider Example 1 and take the graph $\mathcal{G}$ in Fig. 2 left (one might adopt a different graph; a possible alternative is also shown in Fig. 2). This graph and its Markov condition imply independence relations; for instance, FlyingBird and Penguin are independent given any conjunction of literals $\widehat{\mathsf{AntarcticBird}}$ and $\widehat{\mathsf{Bird}}$. Multilinear programming, subject to these independence assumptions and previous assessments, produces $P(\mathsf{FlyingBird}|\mathsf{Bird} \wedge \neg\mathsf{AntarcticBird}) \in [0.949,\ 0.960]$ and $P(\mathsf{Penguin}|\neg\mathsf{AntarcticBird}) = 0$. *Note*: the conditional independence of FlyingBird and Penguin stated in the network may seem at first to clash with the *logical dependence* between these propositions (we must have $\neg$ (FlyingBird $\wedge$ Penguin)). But the network is satisfiable: either $P(\mathsf{FlyingBird}|\mathsf{AntarcticBird}) = 0$ or $P(\mathsf{Penguin}|\mathsf{AntarcticBird}) = 0$ or both. If these conclusions seem inadequate, then an edge must be included between FlyingBird and Penguin, thus removing multilinear constraints. Similarly, if we had assessments such that $P(\mathsf{FlyingBird}|\mathsf{AntarcticBird}) > 0$ and $P(\mathsf{Penguin}|\mathsf{AntarcticBird}) > 0$, the network would be unsatisfiable and would have to be modified. So a PPL network is not a finished collection of truths; it is a tool that lets assessments and constraints be critically evaluated through inference.

In short, we have the following model, where assessments follow the pattern of Expression (1).

Definition 4. A *Propositional Probabilistic Logic* (*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; $\mathcal{L}$ is a list of formulas; $\mathcal{A}$ is a list of assessments $\alpha_j P(\varphi_j) \leqslant P(\phi_j \wedge \varphi_j) \leqslant \beta_j P(\varphi_j)$, where ϕ_j and φ_j are formulas; all formulas are in CNF containing variables in $\mathcal{G}$.

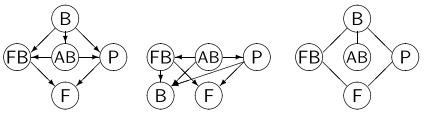

Fig. 2. From left to right: graph $\mathcal{G}$ in Example 3; an alternative graph; constraint network for the first graph ($\mathcal{G}$). Nodes contain only the capital letters of variable names.

We will always assume the Markov condition to be imposed on a PPL network. That is, a PPL network defines a set of probability distributions (over its variables) that must satisfy $\mathscr{A}$, $\mathscr{L}$, and all independence relations imposed by its graph. The formulas of $\mathscr{A}$ and $\mathscr{L}$ are kept separate as this leads to important simplifications when assessing complexity and developing algorithms (Sections 4 and 5). In a PPL network assessments may be (i) unsatisfiable; (ii) satisfiable by a single distribution; (iii) satisfiable by a set of distributions.

3.2. Brief comments on two related graphical languages

The previous work that is closest to this paper is undoubtely Andersen and Hooker's "Bayesian logic" [1]. Indeed several ideas presented here are already stated by Andersen and Hooker; in particular, the need to combine probabilistic satisfiability (where independence is usually ignored) and Bayesian networks (where a single joint probability distribution is always assumed), and the need to resort to nonlinear programming for inference. Their language stays slightly closer to the standard Bayesian network scheme, in that they propose encoding deterministic constraints through probabilities. We instead separate formulas into sets $\mathscr{A}$ and $\mathscr{L}$, so as to obtain sharper complexity results; we also note that the complexity analysis in this paper is new as Andersen and Hooker do not focus on this aspect of their model. An addditional, and significant, difference between the present paper and previous efforts by Andersen and Hooker is the inference framework we propose in Section 5. While Andersen and Hooker strive to retain the linear programming flavor of probabilistic satisfiability, we directly employ results from multilinear programming and variable elimination, in order to produce better elimination orderings during inference. In any case, we are clearly indebted to the seminal ideas advanced by Andersen and Hooker.

There are several languages in the literature that combine (propositional) constraints and probabilistic assessments, and yet guarantee satisfiability by a unique distribution [6,26,46,57]. We present here a very brief summary of a representative proposal, to indicate the main conceptual differences from PPL networks.

The formalism we wish to discuss is Dechter and Mateescu's *mixed networks* [26]. A mixed network consists of a Bayesian network over $\mathbf{X}$, whose joint distribution is denoted by P_B, and a list of logical constraints whose set of valid truth assignments is denoted by Γ. The logical constraints are represented by an undirected graph: two variables are connected if they appear together in a logical constraint (Fig. 2 shows the *constraint network* for Example 3; clearly such constraint networks can be used to describe $\mathscr{L}$ in PPL networks). Dechter and Mateescu combine the Bayesian network and the constraint network to produce a single probability distribution over the truth assignments in Γ:

$$P(\mathbf{X}=\mathbf{x}) = \begin{cases} \prod_i P_B(X_i = x_i | (\mathrm{pa}(X_i) = \pi_i) \wedge (\mathbf{x} \in \Gamma)) & \text{if } \mathbf{x} \in \Gamma, \\ 0 & \text{if } \mathbf{x} \notin \Gamma. \end{cases}$$

Now, a mixed network is *not* a tool that combines logical and probabilistic data with equal status; what we have is a Bayesian network and an elaborate description of a conditioning event Γ. We are not, in any sense, enforcing $P(\Gamma) = 1$. Perhaps what should be stressed is this: an inference conditional on event Γ is *not* the same as an inference under the logical constraint that Γ obtains.

Example 5. Consider two variables X and Y, a directed acyclic graph $X \rightarrow Y$, and the constraint $X \vee \neg Y$. Suppose we have $P(X=1) \leqslant 1/2$, $P(Y=1|X=1) \geqslant 1/2$, $P(Y=1|X=0) \leqslant 1/2$. Then $\max P(Y=1|X \vee \neg Y) = 2/3$. However, $\max P(Y=1) = 1/2$ when subject to probabilistic assessments and to $X \vee \neg Y$ being true. (Note: if we had a Bayesian network $X \rightarrow Y$ with $P(X=1) = P(Y=1|X=1) = P(Y=1|X=0) = 1/2$, then $P(X \vee \neg Y) = 3/4$, inconsistent with logical constraint $X \vee \neg Y$.)

4. The complexity of PPL networks

In this section, we present results on the complexity of PPL networks. We will use in this section results from the theory of *strong extensions of credal networks* [13,22]. A *credal network* consists of a directed acyclic graph where each node is associated with a variable X_i and with constraints on conditional distributions; these constraints define sets of probability measures (such sets are referred to as *credal sets*). The usual Markov condition then creates the network's *strong extension*: a set of Bayesian networks sharing the same graph. We also

need a more formal statement of our decision problem. Given a PPL network, an inference computes the lower (or upper) probability $\underline{P}(\phi|\varphi)$ (or $\overline{P}(\phi \mid \varphi)$) for formula ϕ given the formula φ. The decision problem is:

Definition 6. Given a PPL network $\mathcal{N} = (\mathcal{G}, \mathcal{L}, \mathcal{A})$, `PPL-inference` is the problem of deciding whether there is a probability distribution d (for the node set of $\mathcal{G}$) satisfying all constraints in $\mathcal{L}$ and $\mathcal{A}$.

Let $\mathcal{N} = (\mathcal{G}, \mathcal{L}, \mathcal{A})$ be a PPL network. The *auxiliary* Boolean credal network $\mathcal{N}' = (\mathcal{G}', \mathbf{K}')$ of $\mathcal{N}$ is produced as follows:

- $\mathcal{G}' \supseteq \mathcal{G}$ and has a new node C_{ij} for each non-unitary clause appearing in a formula ϕ_j of $\mathcal{A}$ (both for conditioned and conditioning formulas). Each C_{ij} is connected with new arcs from the variables it contains.
- Constraints of $\mathcal{A}$ are inserted in $\mathbf{K}'$ as follows: $A \in \mathcal{A}$ is inserted into $\mathbf{K}'$ iff A defines an interval of probabilities for a single variable of $\mathcal{G}'$ given a conjunction of its parents, that is, $A \equiv\ \alpha_j P(\varphi_j) \leqslant P(\phi_j \wedge \varphi_j) \leqslant \beta_j P(\varphi_j)$, where ϕ_j is a single literal $\widehat{X}$ and φ_j is a conjunction of X's parents in $\mathcal{G}$.
- CPTs encoding the truth tables of new C_{ij} clauses are inserted into $\mathbf{K}'$ associating these nodes and their parents.

An auxiliary network can be formed in polynomial time: it keeps the nodes of the original PPL network $\mathcal{N}$ and adds a node per non-unitary clause in $\mathcal{A}$. As formulas are in CNF, clauses are disjunctions of literals. Truth tables of these clauses are therefore easy to encode (zero only when all literals are negated and one elsewhere). An important quantity is the *treewidth* [64,65] of the auxiliary network, because constraints in $\mathcal{A}$ may imply new relations between variables that are expressed by new C_{ij} nodes in the auxiliary network (the *treewidth* of a network is the *treewidth* of its underlying graph). If each clause appearing in $\mathcal{A}$ is restricted to a variable and its parents, then all constraints in $\mathcal{A}$ may be encoded in the local credal sets $\mathbf{K}'$ of the auxiliary network, and the treewidth of the PPL network is the same as the treewidth of its auxiliary network. We now present a sequence of results that employ these concepts; proofs of *all* lemmas and theorems are in Appendix A.

Lemma 7. *The treewidth of a PPL network $\mathcal{N} = (\mathcal{G}, \mathcal{L}, \mathcal{A})$ is equal to the treewidth of its auxiliary network $\mathcal{N}' = (\mathcal{G}', \mathbf{K}')$ if all clauses in all constraints of $\mathcal{A}$ are restricted to variables of $\mathcal{N}$ and their parents in $\mathcal{G}$.*

We classify a PPL network either as *bounded treewidth* or *multi-connected*:

Definition 8. A PPL network $\mathcal{N}$ has *bounded treewidth* (*BTW*) if its auxiliary network has treewidth smaller than $O(\log(S))$, where S is the input size needed to specify $\mathcal{N}$; otherwise $\mathcal{N}$ is *multi-connected*.

Logical constraints and assessments can be tested in time similar to Bayesian network inference:

Lemma 9. *Given a BTW (respectively multi-connected) PPL network $\mathcal{N} = (\mathcal{G}, \mathcal{L}, \mathcal{A})$ and a constraint $\phi_j \in \mathcal{L}$, deciding whether a given probability distribution d (for the node set of $\mathcal{G}$) satisfies ϕ_j is in* `P` *(respectively in* `PP`*).*

Lemma 10. *Given a BTW (respectively multi-connected) PPL network $\mathcal{N} = (\mathcal{G}, \mathcal{L}, \mathcal{A})$ and a assessment $A_j \in \mathcal{A}$, deciding whether a given probability distribution d (for the node set of $\mathcal{G}$) satisfies A_j is in* `P` *(respectively in* `PP`*).*

Now, our main theorem:

Theorem 11. `PPL-inference` *is NP-Complete for BTW PPL networks and NP^{PP}-Complete for multi-connected PPL networks.*

The following consequence of our results deserves attention. Note that complexity of inferences depends on the treewidth of the auxiliary network (proof of Theorem 11), and this treewidth does not depend on the number/size of constraints in $\mathcal{L}$ (by construction). Thus, purely logical constraints do not increase complexity of inferences, regardless of the treewidth of their corresponding constraint network. However, note that this observation is valid only for the logical constraints in $\mathcal{L}$, not for probabilistic assessments involving formulas in $\mathcal{A}$ (as they may increase the treewidth of the auxiliary network). Another observation is this. The `CNF-SAT` problem (satisfiability restricted to CNF formulas) is already `NP`-Complete; thus if we include probabilistic assessments into a `CNF-SAT` problem using a *BTW* credal network, our results show that complexity does not increase.

5. A framework for inferences in PPL networks

One can obtain lower/upper probabilities in a PPL network by multilinear programming. A "naive" solution is to write down linear and multilinear constraints for the 2^n truth assignments for variables in the network. In this section, we propose a better strategy to compute inferences, where we exploit the factorization $\prod_i P(X_i \mid \mathrm{pa}(X_i))$ expressed by a PPL network. Instead of focusing on a single algorithm, we present a framework that can be combined with, and benefit from, any Bayesian network inference algorithm.

The basic idea is to run a Bayesian network inference algorithm to produce a compact description of a multilinear program — a program where the probability values $P(X_i|\mathrm{pa}(X_i))$ in the auxiliary network appear as optimization variables. This idea is borrowed from the literature of credal networks [19].

At first, the auxiliary network of the given PPL network is created. This network is in fact a credal network (with Boolean variables) that encodes the dependence structure of the original model. A multilinear program is generated where the optimization variables are probabilities $P(X_i|\mathrm{pa}(X_i))$ of the auxiliary network. The challenge is how to handle probabilities over formulas (more precisely: how to generate the expressions connecting these probabilities to probabilities $P(X_i|\mathrm{pa}(X_i))$ that constitute the factorization). To do so, each constraint in $\mathscr{L}$ is treated as a query in $\mathscr{N}'$. These queries are in fact joint queries in $\mathscr{N}'$ (see proof of Lemma 9 for details). Each assessment of $\mathscr{A}$ either was inserted into $\mathscr{N}'$ (during its construction) as a constraint on local credal sets, or it is also treated as a query in $\mathscr{N}'$. Again, these queries are joint queries in $\mathscr{N}'$, because each $A_j \in \mathscr{A}$ deals with clauses that have become nodes in the construction of the auxiliary network (see proof of Lemma 10 for details).

As we finish these computations, we have a credal network $\mathscr{N}'$ (with Boolean variables) and a set of queries. The only, but important, difference between this setting and a simple belief updating in a credal network is that we must simultaneously satisfy all queries. But we still have a multilinear program: we have only joint queries, and each such joint query is a multilinear function $\sum_{\mathbf{X}\setminus X_Q}\prod_i P(X_i|\mathrm{pa}(X_i))$ over the optimization variables and restricted to some rational interval.

A particularly simple algorithm that computes queries in Bayesian networks, and that can be applied here, is *variable elimination* [25,73]. The purpose of variable elimination is to efficiently compute a summation of products such as $\sum_{\mathbf{X}}\prod_i P(X_i|\mathrm{pa}(X_i))$. In our setting, we must write down symbolically the factorization produced by variable elimination. Every time a summation can be interchanged with a multiplication, we introduce new optimization variables to simplify the expression (similarly to the algorithm in [19]). Applying this idea several times, the initial multilinear function is transformed into a set of smaller multilinear constraints. The drawback is that we introduce new artificial optimization variables that will be part of the final multilinear programming problem.

Here is a step-by-step description of how to generate the multilinear program and make the inference:

- First, construct the auxiliary Boolean credal network $\mathscr{N}'$ from $\mathscr{N}$ and insert into a multilinear program all constraints defined in $\mathscr{N}'$ by its credal sets.
- For each $\phi_j \in \mathscr{L}$, we must have $P(\phi_j) = 1$. We know that $\phi_j = \bigwedge_i C_{ij}$, as ϕ_j is in CNF. Moreover, each clause C_{ij} is a disjunction of literals $\vee_k \widehat{X}_{ijk}$, where X_{ijk} appears in the auxiliary network. Thus,

$$P\left(\bigwedge_i C_{ij}\right) = 1 \iff \forall_i P(\neg C_{ij}) = 0 \iff \forall_i P\left(\bigwedge_k \neg\widehat{X}_{ijk}\right) = 0.$$

  Note that this last assertion is just a joint query in the auxiliary network. So we insert this assertion into the multilinear program and constraints to define these joint queries as factorized probability measures of $\mathscr{N}'$. This is done by running the symbolic procedure mentioned before. Any inference algorithm (adapted to run symbolically) can be used.
- For each $A_j \in \mathscr{A}$, we must have $\alpha_j P(\varphi_j) \leqslant P(\phi_j \wedge \varphi_j) \leqslant \beta_j P(\varphi_j)$, where $\phi_j = \bigwedge_i C_{ij}$, with C_{ij} clause of ϕ_j and $\varphi_j = \bigwedge_i D_{ij}$, with D_{ij} clause of φ_j (formulas ϕ_j and φ_j are in CNF too). By construction of the auxiliary network, clauses C_{ij} and D_{ij} are already represented by nodes in $\mathscr{N}'$. So, we have

$$\alpha_j P\left(\bigwedge_i D_{ij}\right) \leqslant P\left(\bigwedge_i C_{ij} \wedge \bigwedge_i D_{ij}\right) \leqslant \beta_j P\left(\bigwedge_i D_{ij}\right).$$

Each function $P(\cdot)$ of these inequalities is a joint query over variables in $\mathcal{N}'$. So we insert those inequalities into the multilinear program and constraints to define these joint queries as factorized probability measures of $\mathcal{N}'$. Again we need to run the symbolic procedure mentioned before.

- In the final step, the multilinear program must be solved. Its solution determines whether there is a probability distribution that satisfies all assessments.

Thus, for each query, we have run a symbolic version of our preferred algorithm for inference, which outputs a set of multilinear constraints. The overall multilinear program was obtained by taking all multilinear constraints plus the constraints already defined in $\mathcal{N}'$ by the credal sets. Using this multilinear program, we can verify satisfiability or even evaluate the maximum/minimum of $P(\phi|\varphi)$ (subject to all assessments and constraints).

This method can be used with a variety of inference algorithms for Bayesian networks; in particular, it can be used with algorithms that exploit the presence of determinism [6,27]. Another obvious path is to explore approximation algorithms, because exact techniques are certainly limited in the size of solvable problems [40]. We leave such investigations for the future.

6. Moving to first-order: relational networks

In this section, we move beyond propositional languages, briefly looking into elements of first-order logic. To fix terminology: in first-order logic a *formula* is a combination of *constants*, *relations*, *functions*, Boolean operators, *quantifiers* $\exists$ and $\forall$, and *logical variables* (not to be confused with the "random" variables X_i we associate with nodes in networks). A *sentence* is a formula without free logical variables. The semantics is established using a *domain* (a set of *individuals*). An *interpretation* assigns constants, relations and functions in the domain respectively to constants, relations and functions in the vocabulary. More details can be found in several textbooks [52].

Before we move into technical matters, we would like to mention a few notable languages in the literature. We wish to stress the fact, already alluded to several times, that existing languages *either* emphasize flexible assessments without independence judgements, *or* emphasize uniqueness of probabilities with strong assumptions on independence.

We start with Nilsson's influential work on first-order probabilistic logic [55], where he took assessments similar to Expression (1) but with ϕ and φ first-order sentences. Nilsson's logic interprets probabilities as measures over sets of interpretations. More general languages [2,34] attach probabilities also to domains, and allow quantifiers to appear "outside" of probabilistic assessments. Several logics [32,48,56] and applications (for example, in "probabilistic" logic programming [45,47,53]) have been studied in the last 20 years. Most such work, typically referred to as "first-order probabilistic logic," allows general assessments (interval-valued, set-valued, qualitative ones), and ignores independence relations, even when independence can be expressed in the language. Inferential vacuity is a concern, and complexity is typically very high.

A different strategy has received attention recently. Here graph-theoretic models, such as Bayesian networks, are enhanced with elements of first-order logic. Usually the language is relational, the domain is finite, and assumptions of unique names are made to guarantee finiteness of propositionalized models. Often the resulting languages are called "probabilistic relational models" [30,42,54,58]. A formalism that is worth mentioning is the language of *Markov logic* [63]. Markov logic attaches grounded terms to nodes in *undirected graphs*; this creates difficulties because the usual Markov conditions for undirected graphs fail to induce factorization when events of zero probability are present [9].[1] Markov logic postulates a particular factorization, striving to guarantee that a single probability distribution is produced by any set of allowed sentences and assessments. In all of these "probabilistic relational models" and related schemes, we obtain models that are compact and efficient; that guarantee satisfiability and existence of a unique probability distribution;

[1] The same problem appears with chain graphs [9]; that is, the same failure of factorization may happen in the presence of logical constraints. We have entertained the use of chain graphs in probabilistic logic [14]; however, we have concluded this not to be feasible, and in this paper we have reverted to a previous proposal [20].

and that deal explicitly with independence. The drawback is that languages are often restricted in the assessments they allow, and they require strong factorization assumptions to guarantee uniqueness.

After this brief interlude on related work, we can return to our proposal. We wish to focus on probabilistic logics with a balanced mix of point/interval/set-valued assessments and logical constraints, *and* with efficient graph-theoretic representations based on independence relations. We cannot hope to address, in this section, first-order probabilistic logics in their full generality; instead we present a restricted language with some attractive features. The basic idea is to extend PPL networks to relational settings using concepts in Jaeger's relational Bayesian networks [42,43].

Relational Bayesian networks assume a vocabulary $\mathscr{S}$ of relations and a finite domain where each individual has an exclusive name. Each node of a directed acyclic graph is associated with a relation. A relation r is then associated with a *probability expression* F_r that produces the probability of $r(v)$ for any v in a domain $\mathscr{D}$. The probability formula F_r may depend on other relations, the *parents* of r in the graph. If a free variable appears in the parents of r but not in r, then probabilities must be fused in some way. For example, suppose that relation $r(v)$ has parent relation $s(u,v)$; then the probability $p(r(v)|\{s(u,v)\})$ must be specified for arbitrary sets of individuals u. Relational Bayesian networks employ *combination functions*, denoted by $M\{F_1,\ldots,F_n|u;c(u,v)\}$, to combine F_i for all u that satisfy the equality constraint $c(u,v)$. Examples of such functions are Noisy-OR, Max, Min, and Mean. A relational Bayesian network is perhaps best viewed as a first-order "template" for Bayesian networks: given a domain, a relational Bayesian network yields a "propositionalized" Bayesian network that can be used for inference [43].

All of this can be directly turned into a powerful (but very restricted) first-order probabilistic logic, using ideas already tested in PPL networks.

Definition 12. A *Restricted Relational Probabilistic Logic* (*RRPL*) network consists of a tuple $(\mathscr{D},\mathscr{S},\mathscr{G},\mathscr{L},\mathscr{A})$, where $\mathscr{D}$ is a finite set of individuals, $\mathscr{S}$ is a vocabulary with constants and relations, $\mathscr{G}$ is a directed acyclic graph with n nodes, each one identified with a relation r_i in $\mathscr{S}$; $\mathscr{L}$ is a list of sentences in $\mathscr{S}$, restricted to universally quantified conjunctions of disjunctions of possibly negated relations; $\mathscr{A}$ is a list of assessments $\alpha_j P(\varphi_j) \leqslant P(\phi_j \wedge \varphi_j) \leqslant \beta_j P(\varphi_j)$, where ϕ_j and φ_j are formulas without quantifiers in $\mathscr{S}$ such that all logical variables are bound to universal quantifiers "outside" of the inequalities.

We assume throughout the corresponding Markov condition, directly from the theory of relational Bayesian networks [42]: an instantiation of a relation is independent of instantiations of its nondescendants nonparents given instantiations of its parents. Note the restriction on the form of first-order sentences: basically, they must be universal sentences in conjunctive normal form. It is possible to relax the restrictions on assessments by allowing expressions of the generality considered by Jaeger [42]; we have simplified the definition due to lack of space.

Similarly to relational Bayesian networks, a RRPL network is a template for PPL networks: given a finite domain, we can produce a PPL network for inferences.

Example 13. Consider Example 3. The example is perhaps more accurately viewed as an ontology expressing facts about individuals in some domain. Suppose a RRPL network is defined by the graph in Fig. 3, and sentences and assessments: $\forall x$:AntarcticBird$(x) \rightarrow$ Bird(x), $\forall x$:FlyingBird$(x) \rightarrow$ Bird(x), $\forall x$:Penguin$(x) \rightarrow$ Bird(x), $\forall x$:FlyingBird$(x) \rightarrow$ Flies(x), $\forall x$:Penguin$(x) \rightarrow \neg$Flies(x); $\forall x{:}P(\text{FlyingBird}(x)|\text{Bird}(x)) = 0.95$, $\forall x$: P(AntarcticBird(x)|Bird(x)) $= 0.01$, $\forall x{:}P(\text{Bird}(x)) \geqslant 0.2$, $\forall x{:}P(\text{FlyingBird}(x) \vee \text{Penguin}|\text{AntarcticBird}(x)) \geqslant 0.2$, $\forall x{:}P(\text{Flies}|\text{Bird}) \geqslant 0.8$; finally, $\forall x,y{:}P(\text{SameSpecies}(x,y)|\text{AntarcticBird}(x) \wedge \text{AntarcticBird}(y)) \geqslant 0.7$. This last assessment ties together individuals in the domain (note: the assessment is not required to "follow" the direction of edges; we could have assessed a different expression for the probability of SameSpecies). Now suppose we have two individuals, Tweety and Opus. The propositionalized PPL network for this domain is presented in Fig. 3. Using the framework in Section 5 with the variable elimination algorithm, we obtain

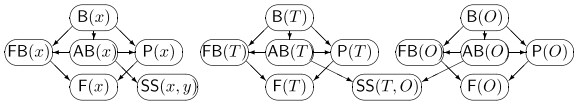

Fig. 3. Left: graph $\mathscr{G}$ in Example 13. Right: propositionalized PPL network in Example 13. Nodes contain only capital letters of relations and individuals.

$P(\mathsf{SameSpecies}(Tweety,Opus)|\mathsf{Penguin}(Tweety)\wedge\mathsf{AntarcticBird}(Opus)) \in [0.7, 1.0]$. Note that this probability is "inherited" from the conditioning on $\mathsf{AntarcticBird}(x) \wedge \mathsf{AntarcticBird}(y)$ in the last assessment (as $P(\neg\, \mathsf{AntarcticBird}(x) \wedge \mathsf{Penguin}(x)) = 0$ for all x).

In Example 13, we have point-valued and interval-valued assessments; we could also have qualitative and ordinal assessments. More importantly in the context of RRPL networks, we need not adopt any artificially "standardized" way to combine expressions (such as the Noisy-OR function).

Example 14. Consider assessments, constraints, and graph $\mathcal{G}$ in Example 13. Suppose another relation, Sick, is introduced to represent illnesses spreading amongst members of the same species. Suppose the node for Sick is added to graph $\mathcal{G}$ as a child of SameSpecies and Flies, and the following probability expression is available: $\forall x,y{:}P(\mathsf{Sick}(x)|\mathsf{Flies}(y) \wedge \mathsf{SameSpecies}(x,y)) = 0.9$. This assessment can be taken literally: as y ranges over the domain, we obtain a number of separate constraints and there is no need to combine them through Noisy-OR — there is no obstacle to us using Noisy-OR or any other function if we judge that to be reasonable, but a combination is not mandated by the language.

We have discussed inference in RRPL networks through propositionalization; the relevant complexity results and algorithms are inherited from the theory of PPL networks, presented in previous sections. An alternative and most promising idea is to conduct *lifted* inference; that is, to do calculations, to the extend that it is possible, with parameterized probabilities and first-order formulas [24,59]. Several algorithms for lifted inference in relational Bayesian networks may be subject to the framework described in Section 5: lifted inference is run "symbolically" (perhaps a few times), thus generating a multilinear program that is subsequently solved. We leave for the future an investigation of the practical feasibility of this idea.

7. Conclusion

Probabilistic logics must accommodate independence relations, both to reduce inferential vacuity and to organize assessments. In this paper we have introduced graph-theoretic models for probabilistic logics with independence, using elements of Bayesian network theory; we then investigated their complexity and presented inference algorithms. We have tried to stay close to one of the main tenets of probabilistic logic: constraints and assessments hold equal status, and verifying their consistency is an important task.

The networks we propose in this paper try to balance the flexibility of traditional probabilistic logic and the efficiency of probabilistic relational models. In fact, while the former is "too loose" and leads to computational challenges and inferential vacuity, the latter are "too strict" in what they assume and demand. Our proposals accept that some structure is necessary in practice, thus adopting graph-theoretic tools; however, our proposals do not assume as many independence relations and structural conditions as needed to obtain a single probability distribution. In fact, once we move to sets of distributions, we can also handle many other types of assessments, such as qualitative, interval-valued, set-valued, and ordinal ones [20].

Two final points should be emphasized. First, our results show that complexity of inferences grows basically as the treewidth of assessments in $\mathcal{A}$, and *not* with the treewidth of a constraint network representing $\mathcal{L}$. The second point to emphasize is the unifying character of credal networks in the framework we have built.

In previous sections, we have explicitly left several topics for future work: an investigation of phase transitions, a proper account of conditioning on events of zero probability, a study of approximate algorithms. Concerning algorithms, it would be particularly useful to develop techniques that exploit logical constraints during inference. We note that there has already been work on the detection of "entailment separation" in structured constraint networks [18]. It is perhaps unrewarding to list all possible future work on probabilistic logic with independence — one might investigate more expressive first-order languages, applications, connections to other fields. Clearly this section closes just the beginning of a longer journey.

Acknowledgements

The work has received partial support from CNPq (Grant 3000183/98-4) and FAPESP (Grant 04/09568-0). We acknowledge partial support from HP Brazil R&D. We thank Barbara Vantaggi for advice on the topic of

zero probabilities, and Thomas Lukasiewicz for important technical help and for encouraging us to produce the paper. We thank Paulo Sérgio Souza Andrade for his excellent implementation of inference algorithms used in Section 2.

Appendix A. Proofs

Proof of Lemma 7. Suppose $\mathcal{G}$ has treewidth k and this value can be achieved by an elimination order $X_1,\ldots,X_n$ of its nodes. Take the moral graph of $\mathcal{G}'$ and a variable elimination order for it that begins with the variables $C_{1j},C_{2j},\ldots,C_{mj}$ (related to all clauses appearing on formulas A_j in $\mathcal{A}$) followed by $X_1,\ldots,X_n$. After eliminating all C_{ij}, the moral graph of $\mathcal{G}'$ becomes exactly the moral graph of $\mathcal{G}$, because the parents of each C_{ij} were already connected in the moral graph of $\mathcal{G}$ (we have supposed in this lemma that each clause involves only a variable and its parents). Thus the width of this elimination order in $\mathcal{G}'$ is equal to the width of $X_1,\ldots,X_n$ in $\mathcal{G}$, that is, k. An alternative path to prove this lemma is to realize that the size of maximum clique of the moral graph of $\mathcal{G}'$ is the same as in graph $\mathcal{G}$, because each C_{ij} is only connected to variables of $\mathcal{G}$ that already form a clique, and nodes C_{ij} have no arcs among each other. □

Proof of Lemma 9. First, construct the auxiliary Boolean credal network $\mathcal{N}'$ from $\mathcal{N}$. Deciding whether ϕ_j is satisfied by d is exactly the question whether d leads to $P_d(\phi_j)=1$, with $\phi_j=\bigwedge_i C_{ij}$ and C_{ij} a clause (as long as ϕ_j is in CNF). Because formulas are in CNF, each C_{ij} is a disjunction of literals $C_{ij}=\vee_k \widehat{X}_{ijk}$, where $\widehat{X}_{ijk}$ is a literal in $\mathcal{N}'$. Thus, $P(C_{ij})=1 \iff P(\neg C_{ij})=0 \iff P(\bigwedge_k \neg\widehat{X}_{ijk})=0$. Note that this last assertion is just a joint query in $\mathcal{N}'$. For a fixed probability distribution d, $\mathcal{N}'$ is a Bayesian network (a single conditional distribution is selected from each credal set). So we just need to evaluate whether d leads to $P_d(\bigwedge_k \neg\widehat{X}_{ijk})=0$ for all clauses of ϕ_j. Thus, for a *BTW* PPL network, each such evaluation takes polynomial time in S (as a *BTW* Bayesian network belief updating is in P), and because of the polynomial number of clauses, the total amount of time is polynomial in S too. For a *multi-connected* network, each evaluation is in PP (the complexity class for general belief updating in Bayesian networks). Because PP is closed under truth-table reductions [28], we can make all the PP evaluations "in parallel" and answer yes if and only if each one answered yes. This completes the proof. □

Proof of Lemma 10. Here, we proceed similarly to Lemma 9. First, construct the auxiliary Boolean credal network $\mathcal{N}'$ from $\mathcal{N}$. Deciding whether A_j is satisfied by d is the same as deciding whether $\alpha_j P_d(\varphi_j) \leqslant P_d(\phi_j \wedge \varphi_j) \leqslant \beta_j P_d(\varphi_j)$, where $\phi_j=\bigwedge_i C_{ij}$, with C_{ij} clause of ϕ_j and $\varphi_j=\bigwedge_i D_{ij}$, with D_{ij} clause of φ_j (this holds because formulas are in CNF). By construction of auxiliary networks, clauses C_{ij} and D_{ij} are already represented by nodes in $\mathcal{N}'$ encoding their truth tables. Thus, we just need to evaluate $p_1=P_d(\bigwedge_i C_{ij} \wedge \bigwedge_i D_{ij})$ and $p_2=P_d(\bigwedge_i D_{ij})$, and test whether p_1 is in the interval $[\alpha_j p_2,\beta_j p_2]$. Note that evaluation of p_1 and p_2 are just joint queries in $\mathcal{N}'$. For a fixed probability distribution d, $\mathcal{N}'$ is a Bayesian network. Thus, for a *BTW* PPL network, both evaluations take polynomial time in S and the total time is polynomial. For a *multi-connected* network, each evaluation is in PP, and we need two adaptive evaluations (as to decide whether p_1 is in that interval we need to previously compute p_2). Because PP equals $\mathrm{P}^{\mathrm{PP}[\log n]}$ (that is, $O(\log n)$ adaptive queries to PP is still in PP) [3,28], the lemma follows. □

Proof of Theorem 11. Consider first *BTW* networks. Pertinence is achieved by Lemmas 9 and 10: given a probability distribution d, verifying satisfaction of each constraint/assessment in $\mathcal{L}$ and $\mathcal{A}$ takes polynomial time in the size of the input. We show hardness with a polynomial time reduction from CNF-SAT: *given a CNF Boolean formula ξ with clauses $C_1,\ldots,C_m$ in the variables* $\mathbf{X}$, *is there an instantiation of* $\mathbf{X}$ *that satisfies ξ*? Construct a PPL network $\mathcal{N}=(\mathcal{G},\mathcal{L},\mathcal{A})$ with no arcs in $\mathcal{G}$ and nodes $\mathbf{X}$, empty $\mathcal{A}$ and $\mathcal{L}$ containing only the formula ξ. It is clear that there is a probability distribution d that satisfies the PPL-inference in $\mathcal{N}$ if and only if there is an instantiation that satisfies the CNF-SAT problem, because we verify in the PPL-inference whether $P(\neg C_j)=\prod_i P(\neg\widehat{X}_{ij})=0$ for all $1 \leqslant j \leqslant m$, where $\widehat{X}_{ij}$ are the literals appearing in C_j. Note that the topology of $\mathcal{G}$ used here is even simpler than a *polytree* (there are no arcs and every node is independent from each other). So, even in this case the PPL-inference is NP-Complete.

Now consider multi-connected networks. Again, pertinence is achieved by Lemmas 9 and 10. Given a probability distribution d, deciding whether each constraint in $\mathscr{L}$ and each assessment in $\mathscr{A}$ is satisfied is in PP. Because PP is closed under truth-table reductions [28], we can make all the PP evaluations (for all formulas in $\mathscr{A}$ and $\mathscr{L}$) "in parallel" and answer yes if and only if each one answered yes. PPL-inference is $\mathtt{NP}^{\mathtt{PP}}$-Hard because a PPL network is an obvious generalization of a Boolean credal network. The belief updating problem in a Boolean credal network is to decide whether $P(\widehat{X}_q) \geqslant r$, where $\widehat{X}_q$ is the query and r is a rational number. To polynomially reduce this problem to a PPL-inference, we just have to encode the local credal sets of the credal network as assessments in $\mathscr{A}$ (this is trivial, as assessments in $\mathscr{A}$ are more general than local credal sets) and insert an additional assessment specifying that $rP(\varphi) \leqslant P(\phi \wedge \varphi) \leqslant P(\varphi)$, with $\phi = \widehat{X}_q$ and φ as a tautology. Now the theorem is immediate. A solution to the PPL-inference solves belief updating in a Boolean credal network. □

References

[1] K.A. Andersen, J.N. Hooker, Bayesian logic, Decision Support Systems 11 (1994) 191–210.
[2] F. Bacchus, Representing and Reasoning with Probabilistic Knowledge: A Logical Approach, MIT Press, Cambridge, 1990.
[3] R. Beigel, N. Reingold, D. Spielman, PP is closed under intersection, Journal of Computer and System Science 50 (2) (1995) 191–2002.
[4] G. Boole, The Laws of Thought, Dover edition, 1958.
[5] B. Borchers, J. Mitchell, S. Joy, A branch-and-cut algorithm for MAX-SAT and weighted MAX-SAT, in: D. Du, J. Gu, P. Pardalos (Eds.), Satisfiability Problem: Theory and Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 35, AMS, 1997, pp. 519–536.
[6] M. Chavira and A. Darwiche. Compiling Bayesian networks with local structure, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence, 2005.
[7] G. Coletti, R. Scozzafava, Probabilistic Logic in a Coherent Setting. Trends in logic, vol. 15, Kluwer, Dordrecht, 2002.
[8] I. Couso, S. Moral, P. Walley, A survey of concepts of independence for imprecise probabilities, Risk, Decision and Policy 5 (2000) 165–181.
[9] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, D.J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, New York, 1999.
[10] F.G. Cozman, Calculation of posterior bounds given convex sets of prior probability measures and likelihood functions, Journal of Computational and Graphical Statistics 8 (4) (1999) 824–838.
[11] F.G. Cozman, Constructing sets of probability measures through Kuznetsov's independence condition, in: Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications, Ithaca, New York, 2001, pp. 104–111.
[12] F.G. Cozman. Algorithms for conditioning on events of zero lower probability, in: International Florida Artificial Intelligence Research Society Conference, Pensacola, FL, 2002, pp. 248–252.
[13] F.G. Cozman, Graphical models for imprecise probabilities, International Journal of Approximate Reasoning 39 (2-3) (2005) 167–184.
[14] F.G. Cozman, C.P. de Campos, J.C. Ferreira da Rocha, Probabilistic logic with strong independence, in: Brazilian Symposium on Artificial Intelligence, 2006.
[15] 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.
[16] F.G. Cozman, P. Walley, Graphoid properties of epistemic irrelevance and independence, Annals of Mathematics and Artificial Intelligence 45 (2005) 173–195.
[17] L. Crisma, The notion of stochastic independence in the theory of coherent probability, Quaderni del Dipartimento di Matematica Applicata Bruno de Finetti 8/91 (1999).
[18] A. Darwiche, A logical notion of conditional independence: properties and application, Artificial Intelligence 97 (1–2) (1997) 45–82.
[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' Symp. (STAIRS), IOS Press, Amsterdam, The Netherlands, 2004, pp. 50–61.
[20] C.P. de Campos, F.G. Cozman, Belief updating and learning in semi-qualitative probabilistic networks, in: F. Bacchus, T. Jaakkola, Conference on Uncertainty in Artificial Intelligence (UAI), Edinburgh, Scotland, 2005.
[21] C.P. de Campos, F.G. Cozman, Computing lower and upper expectations under epistemic independence, International Journal of Approximate Reasoning 44 (3) (2007) 244–260.
[22] C.P. de Campos, F.G. Cozman, The inferential complexity of Bayesian and credal networks, in: International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, United Kingdom, 2005, pp. 1313–1318.
[23] B. de Finetti, Theory of Probability, vols. 1–2, Wiley, New York, 1974.
[24] R. de S. Braz, E. Amir, D. Roth. MPE and partial inversion in lifted probabilistic variable elimination, in: National Conference on Artificial Intelligence (AAAI), 2006.
[25] R. Dechter, Bucket elimination: a unifying framework for reasoning, Artificial Intelligence 113 (1–2) (1999) 41–85.
[26] R. Dechter, D. Larkin, Hybrid processing of beliefs and constraints, in: Conference on Uncertainty in Artificial Intelligence, 2001.

[27] 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.
[28] L. Fortnow, N. Reingold, PP is closed under truth-table reductions, Structure in Complexity Theory Conference (1991) 13–15.
[29] G. Georgakopoulos, D. Kavvadias, C.H. Papadimitriou, Probabilistic satisfiability, Journal of Complexity 4 (1988) 1–11.
[30] L. Getoor, N. Friedman, D. Koller, B. Taskar, Learning probabilistic models of relational structure, in: International Conference on Machine Learning , 2001. pp. 170–177.
[31] A. Gilio, R. Scozzafava, Conditional events in probability assessment and revision, IEEE Transactions on Systems, Man and Cybernetics 24 (12) (1994) 1741–1746.
[32] R. Giugno, T. Lukasiewicz, P-$\mathcal{SHOQ}$(D): A probabilistic extension of $\mathcal{SHOQ}$(D) for probabilistic ontologies in the semantic web, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Proceedings of the 8th European Conf. on Logics in Artificial Intelligence (JELIA), Lecture Notes in Artificial Intelligence, vol. 2424, Springer, Cosenza, Italy, 2002, pp. 86–97.
[33] T. Hailperin, Best possible inequalities for the probability of a logical function of events, American Mathematical Monthly 72 (1965) 343–359.
[34] J.Y. Halpern, Reasoning about Uncertainty, MIT Press, Cambridge, MA, 2003.
[35] P.J. Hammond, Elementary non-Archimedean representation of probability for decision theory and games, in: P. Humphreys (Ed.), Patrick Suppes: Scientific Philosopher, vol. 1, Kluwer, Dordrecht, The Netherlands, 1994, pp. 25–59.
[36] P. Hansen, B. Jaumard, Probabilistic satisfiability, Report G-96-31, Les Cahiers du GERAD, École Polytechique de Montréal, 1996.
[37] P. Hansen, S. Perron, Merging the local and global approaches to probabilistic satisfiability, Report, Cahier du GERAD G-2004-48, 2004.
[38] S. Holzer, On coherence and conditional prevision, Bolletino Unione Matematica Italiana Serie VI 1 (1985) 441–460.
[39] C. Howson, P. Urbach, Scientific Reasoning: the Bayesian Approach, Open Court Publishing Company, Chicago, IL, 1993.
[40] J.S. Ide, F.G. Cozman, IPE and L2U: Approximate algorithms for credal networks, in: Second Starting AI Researcher Symposium (STAIRS), IOS Press, 2004, pp. 118–127.
[41] M. Jaeger, Probabilistic reasoning in terminological logics, in: Principles of Knowledge Representation (KR), 1994, pp. 461–472.
[42] M. Jaeger, Relational Bayesian networks, in: D. Geiger, P.P. Shenoy (Eds.), Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1997, pp. 266–273.
[43] M. Jaeger, Complex probabilistic modeling with recursive relational Bayesian networks, Annals of Mathematics and Artificial Intelligence 32 (2001) 179–220.
[44] D. Jovanović, N. Mladenović, Z. Ognjanović, Variable neighborhood search for the probabilistic satisfiability problem, in: Proceedings of the 6th Metaheuristics International Conference (MIC 2005), Vienna, Austria, 2005.
[45] L.V.S. Lakshmanan, F. Sadri, Probabilistic deductive databases, in: Symposium on Logic Programming, 1994, pp. 254–268.
[46] D. Larkin, R. Dechter, Bayesian inference in the presence of determinism, in: AI and Statistics (AI-STAT), 2003.
[47] T. Lukasiewicz, Probabilistic logic programming, in: European Conference on Artificial Intelligence, 1998, pp. 388–392.
[48] T. Lukasiewicz, Probabilistic logic programming with conditional constraints, ACM Transactions on Computational Logic 2 (3) (2001) 289–339.
[49] D. Mitchell, B. Selman, H. Levesque, Hard and easy distributions of SAT problems, in: Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI), San Jose, CA, 1992, pp. 459–465.
[50] S. Moral, Epistemic irrelevance on sets of desirable gambles, Annals of Mathematics and Artificial Intelligence 45 (1-2) (2005) 197–214.
[51] R.E. Neapolitan, Learning Bayesian Networks, Prentice-Hall, 2003.
[52] A. Nerode, R.A. Shore, Logic for Applications, 2nd ed., Springer-Verlag, New York, 1997.
[53] R. Ng, V.S. Subrahmanian, Probabilistic logic programming, Information and Computation 101 (2) (1992) 150–201.
[54] L. Ngo, P. Haddawy, Answering queries from context-sensitive probabilistic knowledge bases, Theoretical Computer Science 171 (1–2) (1977) 147–177.
[55] N.J. Nilsson, Probabilistic logic, Artificial Intelligence 28 (1986) 71–87.
[56] Z. Ognjanović, Discrete linear-time probabilistic logics: completeness, decidability and complexity, Journal of Logic Computation 16 (2) (2006) 257–285.
[57] J. Pearl, Probabilistic Reasoningin Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.
[58] D. Poole, Probabilistic Horn abduction and Bayesian networks, Artificial Intelligence 64 (1993) 81–129.
[59] D. Poole, First-order probabilistic inference, in: International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 985–991.
[60] K.R. Popper, The logic of Scientific Discovery, Hutchinson, London, 1975.
[61] E. Regazzini, Finitely additive conditional probability, Rendiconti del Seminario Matematico e Fisico di Milano 55 (1985) 69–89.
[62] A. Renyi, On a new axiomatic theory of probability, Acta Mathematics Acadamic Science Hungarian 6 (1955) 85–335.
[63] M. Richardson, P. Domingos, Markov logic networks, Machine Learning 62 (1–2) (2006) 107–136.
[64] N. Robertson, P.D. Seymour, Graph minors – II. Algorithmic aspects of tree-width, Journal of Algorithms 7 (1986) 309–322.
[65] N. Robertson, P.D. Seymour, Graph minors – I. Excluding a forest, Journal of Combinatorial Theory Series B 35 (1983) 39–61.
[66] H.D. Sherali, W.P. Adams, A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, Kluwer Academic Publishers, 1999.
[67] B. Vantaggi, Graphical models for conditional independence structures, in: Second International Symposium on Imprecise Probabilities and Their Applications, Shaker, 2001, pp. 332–341.
[68] B. Vantaggi, Graphical representation of asymmetric graphoid structures, in: Third International Symposium on Imprecise Probabilities and Their Applications, Carleton Scientific, 2003, pp. 560–574.

[69] P. Vicig, Epistemic independence for imprecise probabilities, International Journal of Approximate Reasoning 24 (2000) 235–250.
[70] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, London, 1991.
[71] 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.
[72] W. Zhang, Transitions and backbones of 3-SAT and maximum 3-SAT, in: International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, vol. 2239, Springer, 2001, pp. 153–167.
[73] N.L. Zhang, D. Poole, Exploiting causal independence in Bayesian network inference, Journal of Artificial Intelligence Research (1996) 301–328.