International Journal of Approximate Reasoning 85 (2017) 178–195

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

The effect of combination functions on the complexity of relational Bayesian networks ☆

Denis Deratani Mauá [a,*], Fabio Gagliardi Cozman [b]

[a] Instituto de Matemática e Estatística, Universidade de São Paulo, Brazil
[b] Escola Politécnica, Universidade de São Paulo, Brazil

ARTICLE INFO

Article history:
Received 23 December 2016
Received in revised form 29 March 2017
Accepted 30 March 2017
Available online 4 April 2017

Keywords:
Relational Bayesian networks
Complexity theory
Probabilistic inference

ABSTRACT

We study the complexity of inference with Relational Bayesian Networks as parameterized by their probability formulas. We show that without combination functions, inference is PP-complete, displaying the same complexity as standard Bayesian networks (this is so even when the domain is succinctly specified in binary notation). Using only maximization as combination function, we obtain inferential complexity that ranges from PP-complete to PSPACE-complete to PEXP-complete. And by combining mean and threshold combination functions, we obtain complexity classes in all levels of the counting hierarchy. We also investigate the use of arbitrary combination functions and obtain that inference is EXP-complete even under a seemingly strong restriction. Finally, we examine the query complexity of Relational Bayesian Networks (i.e., when the relational model is fixed), and we obtain that inference is complete for PP.

1. Introduction

Bayesian networks provide an intuitive language for the probabilistic description of concrete domains [22]. Jaeger's Relational Bayesian Networks, here referred to as RBNs, extend Bayesian networks to abstract domains, and allow for the description of relational, context-specific, deterministic and temporal knowledge [20,21]. There are many languages that also extend Bayesian networks into relational representations [11,12,16,23,24,30]; RBNs offer a particularly general and solid formalism.

RBNs constitute a specification language containing a small number of constructs: relations, probability formulas, combination functions, and equality constraints. Combination functions are a particularly important modeling feature, as they provide a way of aggregating information from different elements of the domain.

It should not be surprising that the inferential complexity of Bayesian networks specified by RBNs depends on the choice of constructs allowed. However, few results have been produced on the relation between the expressivity of such constructs and the complexity of inference.

In this paper, we examine the effect of combination functions on the complexity of inferences with (Bayesian networks specified by) RBNs. We first argue that, without combination functions, RBNs simply offer a language that is similar to *plate models*, a well-known formalism to describe models with simple repetitive structure [17,27]. We show that without

☆ This paper is part of the Virtual special issue on the Eighth International Conference on Probabilistic Graphical Models, edited by Giorgio Corani, Alessandro Antonucci, Cassio De Campos.

* Corresponding author.
E-mail address: denis.maua@usp.br (D.D. Mauá).

http://dx.doi.org/10.1016/j.ijar.2017.03.014
0888-613X/

Table 1
Summary of complexity of inference parameterized by the combination functions allowed, how domain is specified, the maximum arity of relations, and the maximum nesting level of combination expressions.

Combination functions	Domain spec.	Bounded arity?	Bounded nesting?	Complexity of inference
none	unary	yes	–	PP-complete
none	binary	yes	–	PP-complete
max	binary	yes	yes	PEXP-complete
max	unary	no	yes	PEXP-complete
max	unary	yes	no	PSPACE-complete
max	unary	yes	yes	$\text{PP}^{\Sigma_k^p}$-complete
threshold, mean	unary	yes	yes	C_kP-hard[a]
polynomial	unary	yes	yes	EXP-complete

[a] Membership when threshold and mean are used depend on further constrains explained in Section 4.3.

combination functions inference is PP-complete, irrespective of the encoding of the domain; this matches the complexity of inference in standard (propositional) Bayesian networks. When we allow combination functions and the associated equality constraints into the language, matters complicate considerably. When either the domain is specified in binary or the arity of relations is unbounded, inference is PEXP-complete even when the only combination function is maximization. If we place a bound on arity, and specify the domain in unary notation (or equivalent, as an explicit list of elements), and allow only maximization as combination function, then inference is PSPACE-complete. This is mostly generated by the ability to nest an unbounded number of maximization expressions. In fact, by further restricting the number of nesting of combination functions, we obtain the same power as a probabilistic Turing machine with access to an oracle in the polynomial hierarchy. We then look at a combination of mean and a threshold: the former allows probabilities to be defined as proportions; the latter allows the specification of piecewise functions. We argue that threshold and mean combined are as powerful as maximization, and thus all previous results hold. And by a suitable constraint on the use of threshold and mean, we show that we can obtain complexity in every class of the counting hierarchy. We also look at the complexity of inference when the combination function is given as part of the input. The challenge here is to constrain the language so as to obtain non-trivial complexity results. We show that requiring polynomial-time combination functions is too weak a condition in that it leads to EXP-complete inference. On the other hand, requiring polynomially long probability formulas brings inference down to PP-completeness. These results are summarized in Table 1.

We also investigate the complexity of inference when the RBN is assumed fixed. This is equivalent to the idea of compiling a probabilistic model [6,9]. We show that complexity is either polynomial if probability formulas can be computed in polynomial time (which includes the case of no combination expressions) or PP-complete, when the combination functions can be computed in polynomial time (which includes the cases of maximization, threshold and mean).

The paper begins with a brief review of RBNs (Section 2), and key concepts from complexity theory (Section 3). Our contributions regarding inferential complexity appear in Section 4. The complexity of inference without combination functions appear in Section 4.1. Relational Bayesian networks allowing only combination by maximization are analyzed in Section 4.2, while networks allowing mean and threshold are analyzed in Section 4.3. General polynomial-time computable combination formulas are examined in Section 4.4. Query complexity is discussed in Section 5. We justify our use of decision problems (instead of functional problem) and discuss how our results can be adapted to provide completeness for classes in the functional counting hierarchy in Section 6. A summary of our contributions and open questions are presented in Section 7.

2. Relational Bayesian networks

2.1. Bayesian networks

A Bayesian network is a compact description of a probabilistic model over a propositional language [7,22]. It consists of two parts: an acyclic directed graph $G = (V, A)$ over a finite set of random variables $X_1, \ldots, X_n$, and a set of conditional probability distributions, one for each variable and each configuration of its parents. The parents of a variable X in V are denoted by $\text{pa}(X)$. In this paper, we consider only 0/1-valued random variables, hence each conditional distribution $\mathbb{P}(X|\text{pa}(X))$ can be represented as a table.

The semantics of a Bayesian network is obtained by the directed Markov property, which states that every variable is conditionally independent of its non-descendants given its parents. For categorical random variables, this assumption induces a single joint probability distribution by

$$\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} \mathbb{P}(X_i = x_i | \text{pa}(X_i) = \pi_i)\,,$$

where π_i is the vector of values for $\text{pa}(X_i)$ induced by assignments $\{X_1 = x_1, \ldots, X_n = x_n\}$. Bayesian networks can represent complex propositional domains, but lack the ability to represent relational knowledge.

2.2. Definition

RBNs can specify Bayesian networks over repetitive scenarios where entities and their relationships are to be represented. To do so, RBNs borrow several notions from function-free first-order logic with equality [14], as we now quickly review. Consider a fixed vocabulary consisting of disjoint sets of names of relations and logical variables (note that we do *not* allow functions nor constants). We denote logical variables by capital letters (e.g., X, Y, Z), names of relations by lowercase Latin letters (e.g., r, s, t), and formulas by Greek letters (e.g., α, β). Each relation name r is associated with a nonnegative integer $|r|$ describing its *arity*. An *atom* has the form $r(X_1, \ldots, X_{|r|})$, where r is a relation name, and each X_i is a logical variable. An atomic equality is any expression $X = Y$, where X and Y are logical variables. An *equality constraint* is a Boolean formula containing only disjunctions, conjunctions and negations of atomic equalities. For example,

$$\big((X = Y) \wedge (X \neq Z)\big) \vee (Y = Z)\,,$$

where $(X \neq Y)$ is a syntactic sugar for $\neg(X = Y)$ (we will adopt this notation throughout the text).

The local conditional probability models in an RBN are specified using *probability formulas*; each probability formula is one of the following:

(PF1) A rational number $q \in [0, 1]$; or
(PF2) An atom $r(X_1, \ldots, X_{|r|})$; or
(PF3) A convex combination $F_1 \cdot F_2 + (1 - F_1) \cdot F_3$ where F_1, F_2, F_3 are probability formulas; or
(PF4) A *combination expression*, that is, an expression of the form

$$\mathsf{comb}\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}\,,$$

where comb is a word from a fixed vocabulary of names of combination functions (disjoint from the sets of relations symbols and logical variables), $F_1, \ldots, F_k$ are probability formulas, $Y_1, \ldots, Y_m$ is a (possibly empty) list of logical variables, and α is an equality constraint containing only those logical variables and the logical variables appearing in the subformulas. We later define combination functions.

The logical variables $Y_1, \ldots, Y_m$ in a combination expression are said to be *bound* by that expression; there might be no logical variables bound by a particular combination expression, in which case we write $\mathsf{comb}\{F_1, \ldots, F_k | \emptyset; \alpha\}$.

We define *free* logical variables in a probability formula F as follows. A rational number has no (free) logical variables. All logical variables in atom $r(X_1, \ldots, X_{|r|})$ are free. A variable is free in $F_1 \cdot F_2 + (1 - F_1) \cdot F_3$ if it is free in one of F_1, F_2 and F_3. Finally, a logical variable is free in $\mathsf{comb}\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}$ if its is free in any of $F_1, \ldots, F_k$ or α, and it is not bound by comb (i.e., it is not one of $Y_1, \ldots, Y_m$).

An example of a probability formula is

$$\mathsf{mean}\{0.6 \cdot r(X) + 0.7 \cdot \mathsf{max}\{1 - s(X, Y) | X; X = X\} | Y, Z; Y \neq X \wedge Z \neq X\}\,.$$

In this formula X is free (even though X is bound by the combination expression headed by max), and Y and Z are bound. We often write $F(X_1, \ldots, X_n)$ to indicate that $X_1, \ldots, X_n$ are the free logical variables in F.

Similarly to Bayesian networks, an RBN consists of two parts: an acyclic directed graph where each node is a relation symbol, and a collection of probability formulas, one for each node. The probability formula F_r associated with node r contains exactly $|r|$ free logical variables and mentions only the relation symbols $s \in \mathrm{pa}(r)$.

2.3. Semantics

Roughly speaking, one can interpret an RBN with graph $G = (V, A)$ as a rule that takes a set $\mathcal{D}$ of elements, called a *domain*, and produces an *auxiliary Bayesian network*, as follows. First, generate all groundings $r(a)$ for $a \in \mathcal{D}^{|r|}$ and $r \in V$. These groundings are the nodes of the Bayesian network. For each grounding $r(a)$, consider the probability formula at a; that is, $F_r(a)$. This formula depends on many groundings $s(a, b)$ of parents of r; those are the parents of $r(a)$ in the Bayesian network. And for each configuration π of these (grounded) parents, take $\mathbb{P}(r(a) = 1 | \mathrm{pa}(r(a)) = \pi) = F_r(a)$ where $F_r(a)$ is evaluated at π. We formalize this procedure later, right after we present an example:

Example 1. Consider a simple model of student performance, inspired by the "University World" often employed in descriptions of Probabilistic Relational Models [16,22]. Suppose a student is taking courses; for each pair student/course, we have approval or not. We have relations difficult and committed, respectively indicating whether a course is difficult and whether a student is committed, and relation approved, indicating whether a student gets an approval grade in a course. The probabilistic relationships between these relations are captured by the graph in Fig. 1, where we used two additional relations student and course to "type" elements of the domain. The idea is that these relations are fully specified by the evidence during inference (that is, every element is typed), so its associated probabilities are not important; we can take for example:

$$F_{\mathsf{student}}(Y) = 1/2 \text{ and } F_{\mathsf{course}}(X) = 1/2\,.$$

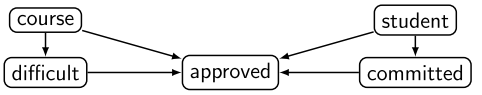

Fig. 1. Relational Bayesian network modeling student performance.

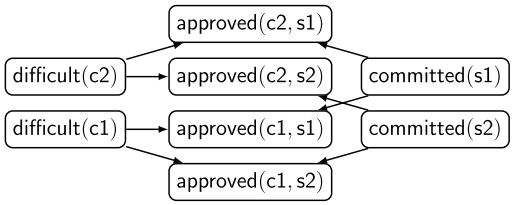

Fig. 2. Bayesian network obtained by grounding the RBN in Example 1 and removing nodes with evidence.

We need probability formulas for difficult and committed:

$$F_{\mathsf{difficult}}(X) = 0.3 \cdot \mathsf{course}(X)\,, \qquad F_{\mathsf{committed}}(Y) = 0.6 \cdot \mathsf{student}(Y)\,.$$

Also, we have a probability formula for approved:

$$F_{\mathsf{approved}}(X, Y) = 0.8 \cdot [0.6 \cdot (1 - \mathsf{difficult}(X)) + 0.4 \cdot \mathsf{committed}(Y)] \cdot \mathsf{course}(X) \cdot \mathsf{student}(Y)\,.$$

Now suppose we have domain $\mathcal{D} = \{\mathsf{s1}, \mathsf{s2}, \mathsf{c2}, \mathsf{c2}\}$. Note that we have 32 groundings (four unary relations and one binary relation), so we obtain a Bayesian network with 32 nodes/random variables. In this network, we have

$$\mathbb{P}(\mathsf{difficult}(a) = 1 | \mathsf{course}(a) = s) = \begin{cases} 0.0 & \text{if } s = 0 \ (a \text{ is not a course}), \\ 0.3 & \text{otherwise;} \end{cases}$$

and likewise for all other groundings of relations. Now suppose that the relation student is given; say we have s1 and s2 as students, and c1 and c2 as courses (hence not students). We denote this evidence $\mathcal{E}$. Given $\mathcal{E}$, we can focus on a smaller Bayesian network obtained by conditioning on this "typing" evidence; such a network is presented in Fig. 2. □

The probability formulas in the previous example did not use combination expressions. Combination expressions are slightly more difficult to interpret; their semantics are given by combination functions. A *combination function* is any function that maps a multiset of finitely many numbers in $[0, 1]$ into a single rational number in $[0, 1]$. Some examples of combination functions are:

- Maximization: $\max\{q_1, \ldots, q_k\} = q_i$, where i is the smallest integer such that $q_i \geq q_j$ for $j = 1, \ldots, k$, and $\max\{\} = 0$;
- Minimization: $\min\{q_1, \ldots, q_k\} = 1 - \max\{1 - q_1, \ldots, 1 - q_k\}$;
- Arithmetic mean: $\mathsf{mean}\{q_1, \ldots, q_k\} = \sum_{i=1}^{k} q_i/k$, and $\mathsf{mean}\{\} = 0$;
- Noisy-or: $\mathsf{noisy\text{-}or}\{q_1, \ldots, q_k\} = 1 - \prod_{i=1}^{k}(1 - q_i)$, and $\mathsf{noisy\text{-}or}\{\} = 0$;
- Threshold: $\mathsf{threshold}\{q_1, \ldots, q_k\} = 1$ if $\max\{q_1, \ldots, q_k\} \geq 1/2$ and $\mathsf{threshold}\{q_1, \ldots, q_k\} = 0$ otherwise.

The combination function threshold, which does not appear in previous RBNs, is useful to encode case-based functions such as piecewise linear functions. It may seem that the comparison with a fixed value $1/2$ is too restrictive; however, with a small effort we can compare any probability formula with any constant using this combination function. To see this, suppose we want to compare a probability formula F with a number $\gamma \in [0, 1]$; that is, we want to determine whether $F \geq \gamma$. If $\gamma > 1/2$, then use $\mathsf{threshold}\{(1/(2\gamma)) \cdot F\}$. And if $\gamma < 1/2$, then use $\mathsf{threshold}\{\gamma' F + (1 - \gamma')\}$ where $\gamma' = 1/(2(1 - \gamma))$. Thus we can generate all sorts of piecewise and case-based functions with threshold. We exploit this versatility to encode decision problems related to counting in some of our proofs.

We can now explain the semantics of RBNs, which is given by interpretations of a domain $\mathcal{D}$. An *interpretation* is a function μ mapping each relation symbol $r \in \mathcal{R}$ into a relation $r^\mu \subseteq \mathcal{D}^{|r|}$, and each equality constraint α into its standard meaning α^μ. An interpretation μ induces a mapping from a probability formula F with n free logical variables into a function $F^\mu(a)$ from $\mathcal{D}^n$ to $[0, 1]$ as follows. If $F = q$, then F^μ is the constant function q. If $F = r(X_1, \ldots, X_n)$, then $F^\mu(a) = 1$ if $a \in r^\mu$ and $F^\mu(a) = 0$ otherwise. If $F = F_1 \cdot F_2 + (1 - F_1) \cdot F_3$ then $F^\mu(a) = F_1^\mu(a) F_2^\mu(a) + (1 - F_1^\mu(a)) F_3^\mu(a)$. Finally, if $F = \mathsf{comb}\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}$, then $F^\mu(a) = \mathsf{comb}(\mathcal{Q})$, where this latter comb is the corresponding combination function and $\mathcal{Q}$ is the multiset containing a number $F_i^\mu(a, b)$ for every $(a, b) \in \alpha^\mu$ (even if F_i does not depend on every coordinate).

For example, consider a domain $\mathcal{D} = \{\mathsf{a}, \mathsf{b}\}$. The probability formula $F = \max\{0.1, 0.2, 0.3 | Y; Y = Y\}$ is interpreted as $\max\{0.1, 0.2, 0.3, 0.1, 0.2, 0.3\}$, and the probability formula $F(X, Y) = \max\{q_1, q_2 | \emptyset; X \neq Y\}$ is interpreted as $F^\mu(\mathsf{a}, \mathsf{a}) = F^\mu(\mathsf{b}, \mathsf{b}) = \max\{\} = 0$, and $F^\mu(\mathsf{a}, \mathsf{b}) = F^\mu(\mathsf{b}, \mathsf{a}) = \max\{q_1, q_2\}$.

Finally: Given a finite domain $\mathcal{D}$, an RBN with graph $G = (V, A)$ is associated with a probability distribution over interpretations μ defined by

$$\mathbb{P}_{\mathcal{D}}(\mu) = \prod_{r \in V} \mathbb{P}_{\mathcal{D}}(r^{\mu} | \{s^{\mu} : s \in \text{pa}(r)\}) ,$$

where

$$\mathbb{P}_{\mathcal{D}}(r^{\mu} | \{s^{\mu} : s \in \text{pa}(r)\}) = \prod_{a \in r^{\mu}} F_r^{\mu}(a) \prod_{a \notin r^{\mu}} (1 - F_r^{\mu}(a)) .$$

We can now rephrase, in precise terms, the more informal explanation given before Example 1. In essence, an RBN is a template model that for each domain $\mathcal{D}$ generates a Bayesian network where each node r is a multidimensional random variable taking values r^{μ} in the set of $2^{N^{|r|}}$ possible interpretations r^{μ}, where $N = |\mathcal{D}|$. Alternatively, we can associate with every relation symbol $r \in \mathcal{R}$ and tuple $a \in \mathcal{D}^{|r|}$ a random variable $r(a)$ that takes value 1 when $a \in r^{\mu}$ and 0 when $a \notin r^{\mu}$. An RBN and a domain $\mathcal{D}$ produce an *auxiliary Bayesian network* over these random variables, where the parents of a node $r(a)$ are all nodes $s(a, b)$ such that s is a parent of r in the RBN and $(a, b) \in \mathcal{D}^{|s|}$, and the conditional distribution associated with $r(a)$ is given by interpretations of the probability formula F_r at a. This Bayesian network induces a joint distribution

$$\mathbb{P}\big(\{r(a) = \rho_{r(a)}\}\big) = \prod_{r \in V} \prod_{a \in \mathcal{D}^{|r|}} \mathbb{P}\big(r(a) = \rho_{r(a)} | \text{pa}(r(a)) = \pi\big) = \mathbb{P}_{\mathcal{D}}(\mu) ,$$

where $a \in r^{\mu}$ if $\rho_{r(a)} = 1$ and $a \notin r^{\mu}$ if $\rho_{r(a)} = 0$, and π is a configuration consistent with those values. Following terminology from first-order logic, we often say that this Bayesian network is the "grounding" of the RBN on domain $\mathcal{D}$. Often, it is possible to generate a smaller grounding, where the parents of $r(a)$ are only those relevant to evaluate $F_r(a)$ (which might not depend on all $s(a, b)$, $s \in \text{pa}(r)$, $a, b \in \mathcal{D}^{|s|}$). Given this equivalence between an RBN endowed with a domain and an auxiliary Bayesian network, we refer to probabilities such as $\mathbb{P}(r(a) = 1)$ to denote $\mathbb{P}(a \in r^{\mu}) = \sum_{\mu : a \in r^{\mu}} \mathbb{P}_{\mathcal{D}}(\mu)$, and similarly for more complex events (note that the dependency of the distribution on the domain is left implicit in the first case).

The next example illustrates the modeling power of different combination functions.

Example 2. Consider again the university domain in Example 1, and suppose that in addition to students and courses we also have teachers in the domain. At the end of the academic year students vote for the best teacher of the year. An award is then given to the most voted teacher. We use a binary relation taught to indicate whether a course was taught by a teacher in that year. This relation will usually be given as evidence, so its probability is not important; for simplicity we specify

$$F_{\text{taught}}(X, Z) = 1/2 .$$

We use an unary relation awarded to indicate whether someone is the recipient of the award. The probability that a teacher is awarded is linearly related to the proportion p of students that failed in at least one course he/she taught, with two regimes:

$$\mathbb{P}(\text{awarded}(Z) = 1 | p) = \begin{cases} 0.425 - 0.4p, & \text{if } p \geq 0.7, \\ 0.25 - 0.15p, & \text{if } p < 0.7. \end{cases}$$

We encode this as,

$$F_{\text{awarded}}(Z) = T(Z) \cdot [0.025 + 0.4 \cdot (1 - F_1(Z))] + (1 - T(Z)) \cdot [0.1 + 0.15 \cdot (1 - F_1(Z))] ,$$

where $T(Z) = \text{threshold}\{5/7 \cdot F_1(Z)\}$ denotes whether the proportion of failed students is above 0.7, $F_1(Z) = \text{mean}\{F_2(Y, Z) | Y; Y = Y\}$ is the proportion of students that failed some course taught by Z, and $F_2(Y, Z)$ is a formula indicating whether Y failed a course taught by Z:

$$F_2(Y, Z) = \max\{(1 - \text{approved}(X, Y)) \cdot \text{taught}(X, Z) | X : X = X\} .$$

Since there can be exactly one winner of the award, we introduce a relation singleWinner of arity zero (i.e., a proposition), and specify:

$$F_{\text{singleWinner}} = \max\{\text{awarded}(Z) | Z; Z = Z\} \cdot \text{threshold}\{\gamma \cdot \text{mean}\{1 - \text{awarded}(Z) | Z; Z = Z\} | \emptyset\} ,$$

where $\gamma = (N - 1)/N$ and $N = |\mathcal{D}|$ is the size of the domain (the number of courses, students and teachers). The graph of the corresponding RBN is shown in Fig. 3. □

This example uses the max combination function to encode an existential quantifier; we will later resort to this trick in our proofs.

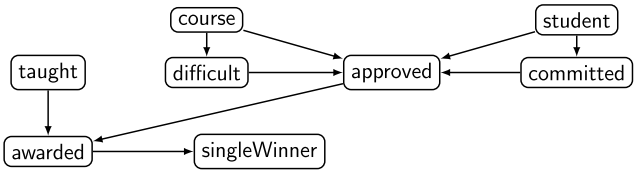

Fig. 3. Relational Bayesian network modeling the university domain in Example 2.

3. The complexity of counting

We assume familiarity with standard complexity classes such as P, NP, PSPACE and EXP, and with the concept of oracle Turing machines [1,29]. As usual, we denote the class of languages decided by a Turing machine in complexity class C with oracle in complexity class O as C^O. We also assume that problems are reasonably encoded using a binary alphabet (say, 0 and 1). A decision problem consists in computing a 0/1-valued function on binary strings. We call an input of a decision problem a YES instance if its output is 1, otherwise we call a NO instance.

The *polynomial hierarchy* is the collection of complexity classes Σ_k^P, Δ_k^P and Π_k^P, for every nonnegative integer k. These decision problems are defined inductively by $\Sigma_0^P = P$, $\Sigma_k^P = NP^{\Sigma_{k-1}^P}$, $\Delta_k^P = P^{\Sigma_{k-1}^P}$, and $\Pi_k^P = co\Sigma_k^P$. The complexity class PH contains all decision problems in the polynomial hierarchy: $PH = \bigcup_k \Sigma_k^P$ (over all integers k). We thus have the following inclusion:

$$P \subseteq \begin{matrix} NP \\ coNP \end{matrix} \subseteq P^{NP} \subseteq \begin{matrix} \Sigma_2^P \\ \Pi_2^P \end{matrix} \subseteq \cdots \subseteq \begin{matrix} \Sigma_k^P \\ \Pi_k^P \end{matrix} \subseteq \Delta_k^P \subseteq \cdots \subseteq PH\,.$$

A well-known result by Meyer and Stockmeyer [28] states that if $\Pi_k^P = \Sigma_k^P$ for some k then $PH = \Sigma_k^P$. In particular, if $P = NP$ then $PH = P$. This suggests that the levels Σ_k^P of the hierarchy are distinct, and that using an oracle in a higher level brings in more computational power.

A *probabilistic Turing machine* is a standard non-deterministic Turing machine that accepts if at least half of the computation paths accept, and rejects otherwise. The complexity class PP contains decision problems which are accepted by a probabilistic Turing machine in polynomial time. The class PP is known to be closed under complement and union [4]. The class PEXP is defined analogously with polynomial-time replaced with exponential-time.

The *counting hierarchy* augments the polynomial hierarchy with oracle machines based on PP. It can be defined as the smallest collection of classes containing P and such that if C is in the collection then NP^C, $coNP^C$, and PP^C are also in the collection [34,36]. We define the class CH as the union over all nonnegative integers k of the classes C_kP, where $C_kP = PP^{C_{k-1}}$ and $C_0P = \Sigma_0^P = P$ [36]. Toda's celebrated result shows that any problem in PH can be solved in polynomial time by a deterministic Turing machine with an oracle PP [32]. Hence, the class CH contains the entire polynomial hierarchy. If fact, we have the following hierarchy:

$$PH \subseteq P^{PP} \subseteq \begin{matrix} NP^{PP} \\ coNP^{PP} \end{matrix} \subseteq PP^{PP} \subseteq C_2P \subseteq \cdots \subseteq CH \subseteq PSPACE \subseteq EXP \subseteq PEXP\,.$$

A *many-one reduction* from a problem A to a problem B is a polynomial-time function f that maps inputs of A into inputs of B, and such that x is a YES instance of A if and only if $f(x)$ is a YES instance of B. For any complexity class C in the counting hierarchy, we say that a problem A is C-hard if for any problem B in C there is a many-one reduction from A to B. If A is also in C, then A is said to be C-complete.

4. Inferential complexity of relational Bayesian networks

Our goal in this paper is to examine the complexity of the decision variant of the following (single-query) inference problem:

Input: An RBN with graph (V, A), a domain $\mathcal{D}$, a list of random variables $r(a), r_1(a_1), \ldots, r(a_n)$, and a rational $\gamma \in [0, 1]$.
Decision: Is $\mathbb{P}(r(a) = 1 | r_1(a_1) = 1 \wedge \cdots \wedge r_n(a_n) = 1) \geq \gamma$?

The conditioning event $r_1(a_1) = 1 \wedge \cdots \wedge r_n(a_n) = 1$ is called *evidence*, and also denoted as $\{r_1(a_1) = 1, \ldots, r_n(a_n) = 1\}$ and $\{r_1(a_1) = 1\}, \ldots, \{r_n(a_n) = 1\}$. We have assumed implicitly in the formulation of the problem above that $\mathbb{P}(r_1(a_1) = 1, \ldots, r_n(a_n) = 1) > 0$ (i.e., that evidence has positive probability). A different strategy would be to define instances where this condition is not observed as NO instances (or as YES instances). In this case, to prove membership in some complexity class we would have to first show how to decide $\mathbb{P}(r_1(a_1) = 1, \ldots, r_n(a_n) = 1) > 0$ in that class. For simplicity, we assume that this is always true; it is possible however to show that the complexity of inference is not changed by this simplifying assumption.

If the domain is a singleton, then the RBN is actually a Bayesian network, since probability formulas can be efficiently converted into tables. Moreover, any Bayesian network can be polynomially encoded as a RBN with a singleton domain. Hence, inference in RBNs is PP-hard, as this is the case for Bayesian networks [8, Theorems 11.3 and 11.5].

For domains with many elements, complexity might depend on how the domain $\mathcal{D}$ is encoded. One possibility is to assume that the domain is some set $\{1, 2, \ldots, N\}$ specified by the integer N (but note that there is no ordering on the elements implied). The integer N can be encoded in different ways. One is this:

Assumption 3. The domain is $\{1, 2, \ldots, N\}$, where N is encoded in binary notation.

Under the previous assumption, the encoding of the domain contributes with $\mathcal{O}(\log N)$ bits to the input size. Depending on the constructs allowed, this alternative can produce probabilities which require exponential effort to be written down, as they might need $\mathcal{O}(N)$ bits. To see this, consider an RBN with a proposition r such that $F_r = \text{noisy-or}\{1/2|X; X = X\}$. Then $\mathbb{P}(r = 0) = 1 - [1 - \prod_{i=1}^{N} 1/2] = 2^{-N}$.

Another possibility is to still assume that the domain is $\{1, \ldots, N\}$, but to encode N using unary notation. In this case, the encoding of the domain contributes with $\mathcal{O}(N)$ symbols to the input. Thus, this specification is equivalent (up to some constant factor) to encoding the domain by an explicit list with every element it contains. So we might adopt:

Assumption 4. The domain is specified as an explicit list of elements.

Another parameter that affects inferential complexity is the arity of relations. If we place no bound on the arity of relations, then an exponential effort might be required to write down probabilities. For example, consider an RBN with graph (r)⟶(s) where r is a k-ary relation with probability formula $F_r(X_1, \ldots, X_k) = 1/2$, and s is proposition with probability formula $F_s = \text{noisy-or}\{r(X_1, \ldots, X_k)|X_1, \ldots, X_k; X_1 = X_1\}$; then inference $\mathbb{P}(s = 0)$ requires an exponential number of bits when 4. We will thus often adopt:

Assumption 5. The arity of relations is bounded.

Without further restrictions, the complexity of inference is dominated by the complexity of evaluating combination expressions. For instance, if the combination function in a combination expression is an arbitrary Turing machine (with no bounds on time and space) then inference is undecidable. Thus, we need to constrain the use of combination functions in probability formulas to make the study of inferential complexity more interesting.

4.1. RBNs without combination functions

We start our analysis with the simplest case, where no combination function is available. That is, we adopt:

Assumption 6. Every probability formula is either a rational number, or an atom, or a convex combination of probability formulas.

Under the assumption above, every logical variable in a probability formula is free; also, in the acyclic directed graph of relations of an RBN no node can have an arity larger than any of its children.

If the domain is given as an explicit list of elements (or by an integer in unary notation), then we can "ground" an RBN into an auxiliary Bayesian network, and run inference there (consider Example 1). This process generates a polynomially-large Bayesian network if the relations have bounded arity (if necessary, we can insert auxiliary nodes so as to bound the number of parents of any given node). Because inference in Bayesian networks is PP-complete, we have that

Theorem 7. *Inference is PP-complete when the domain is specified as a list, the relations have bounded arity and there is no combination function.*

Interestingly, the complexity of inference remains the same *even* if the domain is specified by its size in binary notation (note that in this case the auxiliary Bayesian network has a size that is exponential in the input):

Theorem 8. *Inference is PP-complete when the domain is specified by its size in binary, the relations have bounded arity and there is no combination function.*

Proof. Hardness follows from the fact that an RBN can efficiently encode any propositional Bayesian network using only convex combinations of numbers and atoms (as in Example 1). To prove membership, consider the auxiliary Bayesian network that is generated by "grounding" a given RBN. This network may be exponentially large on the input, when the domain is given in binary notation. We show that we only need to generate a polynomially-large fragment of the auxiliary Bayesian network to decide whether $\mathbb{P}(r(a) = 1|r_1(a_1) = 1 \wedge \cdots \wedge r_n(a_n) = 1) \geq \gamma$. Note that to compute that conditional probability,

all we need is the fragment of the Bayesian network that contains the ancestors of nodes $r(a), r_1(a_1), \ldots, r_n(a_n)$ [15]. We argue that the number of ancestors of $r(a)$ is polynomial in the input size; the same reasoning applies to $r_1(a), \ldots, r_n(a_n)$. If the probability formula F_r is either a number or an atom, then $r(a)$ has no ancestors and we are done. Otherwise assume that F_r is a convex combination; then it may recursively refer to several atoms that are parents of r. These parents may in turn have several parents, and so on. Since there are not combination functions, we can always rename logical variables such the probability formulas of every ancestor of r contain only logical variables in the probability formula F_r. Thus, if r has arity k, then there can be at most 2^k ancestors of $r(a)$, as this is the number of subsets of the logical variables in F_r. Since we assumed k is bounded by a constant, this number of ancestors is also bounded by a constant. Thus, we can build a Bayesian network consisting of these nodes and the edges between them, together with the associated probability values. Inference must then be run in this sub-network, and this can be done by a polynomial-time probabilistic Turing machine [8, Theorems 11.3 and 11.5]. □

According to Theorem 8, the complexity of inference in combination-function-free RBNs matches the complexity of inference in Bayesian networks (if relation arity is bounded). That is, even though an RBN may represent exponentially large Bayesian networks, only a polynomial number of nodes of this network is relevant for inference (and these nodes can be collected in polynomial time).

4.2. RBNs with max combination functions

We now consider RBNs where information is only aggregated by maximization. That is, we assume that

Assumption 9. The only combination function used is max.

As we have seen, without combination functions, inference in RBNs has the same computational power as inference in Bayesian networks, even when the domain is given by an integer in binary notation. This is not the case when we allow combination functions, even one as simple as maximization:

Theorem 10. *Inference is* PEXP*-complete when the domain is specified by its size in binary, the relations have bounded arity and the only combination function is max.*

Proof. Membership follows since an RBN satisfying the above assumptions can be "grounded" into an exponentially large Bayesian network. To see this, note that if we have a domain of size 2^m, and a bound of arity k, then each relation produces at most $(2^m)^k = 2^{km}$ (grounded) random variables. Inference can be decided in that network using a probabilistic Turing machine with an exponential bound on time, using the same scheme normally employed to decide an inference for a Bayesian network [8].

To show hardness, we simulate an exponential-time probabilistic Turing machine using an RBN such that we can "count" the number of accepting paths by a single inference. We do so by resorting to a standard Turing machine encoding described by Grädel [18, Theorem 3.2.4]. There are other possible encodings we might use, but this one is quite compact and relatively easy to grasp.

So, take a Turing machine that can decide a PEXP-complete language. That is, take a PEXP-complete language $\mathcal{L}$ and a nondeterministic Turing machine such that $\ell \in \mathcal{L}$ if and only if the machine halts within time 2^p with half or more of the computation paths accepting, where n is the length of ℓ and p denotes a polynomial in n.[1] Denote by σ a symbol in the alphabet of the machine (which includes blank and boundary symbols in addition to 0 and 1s), and by q a state. As usual, we describe a configuration of the machine by a string $\sigma^1\sigma^2\cdots\sigma^{i-1}(q\sigma^i)\sigma^{i+1}\cdots\sigma^{2^p}$, where each σ^j is a symbol in the tape, and $(q\sigma^i)$ indicates that the machine is at state q and its head is at cell i. The initial configuration is $(q_0\sigma_0^1)\sigma_0^2\cdots\sigma_0^n$ followed by $2^p - n$ blanks, where q_0 is the initial state. Let q_a and q_r be, respectively, states that indicate acceptance or rejection of the input string $\sigma_0^1\cdots\sigma_0^n$. The transition function δ of the machine takes a pair (q, σ) consisting of a state and a symbol in the tape, and returns triples (q', σ', m): the next state q', the symbol σ' to be written in the tape (we assume that a blank is never written by the machine), and an integer m in $\{-1, 0, 1\}$. Here -1 means that the head is to move left, 0 means that the head is to stay in the current cell, and 1 means that the head is to move right. We make the important assumption that, if q_a or q_r appear in some configuration, then the configuration is not modified anymore (the transition function goes from this configuration to itself from that point on). This is necessary to guarantee that the number of accepting computations is equal to the number of ways in which the machine can run from the input.

Grädel's encoding uses quantified Boolean formulas to represent the Turing machine. As shown by Jaeger [20, Lemma 2.4], probability formulas can encode conjunction and negation, respectively, by multiplication and inversion (and then disjunction and implication are easily obtained). Logical equalities such as $X = Y$ are obtained by the formula

[1] In the usual definition, a probabilistic Turing machine accepts when the majority of computation paths accept; however, given such a machine we can always build a machine that decides the same language, and such that the machine accepts if and only if at least half of the computation paths accept.

$F(X,Y) = \max\{1|\emptyset; X = Y\}$, and quantified formulas such as $\exists Y \psi(X,Y)$ are obtained as $F(X) = \max\{F_\psi(X,Y)|Y; Y = Y\}$, where $F_\psi(X,Y)$ encodes $\psi(X,Y)$. Thus we can easily reproduce Grädel's logical encoding of the machine by writing logical formulas as probability formulas. For example, consider a relation less (meaning "less than"), used to order the elements of the domain, and consequently represent the ordering on the computation steps and on the positions of the tape. Since the elements of the domain are ordered, we can interpret them as integers. Define $F_{\mathsf{less}}(X,Y) = 1/2$. We must impose on less the axioms of linear orderings. To constrain less to be irreflexive, we introduce a proposition irref with probability formula

$$F_{\mathsf{irref}} = 1 - \max\{\mathsf{less}(X,X)|X; X = X\}.$$

We have that $F^{\mu}_{\mathsf{irref}} = 1$ if and only if less^{μ} is irreflexive. Note that the above probability formula simply encodes the quantified Boolean formula $\neg\forall X \mathsf{less}(X,X)$ in the language of RBNs. Similarly, we enforce transitivity by a relation trans with probability formula

$$F_{\mathsf{trans}} = 1 - \max\{\mathsf{less}(X,Y)\cdot\mathsf{less}(Y,Z)\cdot(1-\mathsf{less}(X,Z))|X,Y,Z; X = X\}.$$

To understand this expression, note that $\mathsf{less}(a,b)\wedge\mathsf{less}(b,c)\rightarrow\mathsf{less}(a,c)$ holds if and only if $1-[\mathsf{less}(a,b)\cdot\mathsf{less}(b,c)\cdot(1-\mathsf{less}(a,c))] = 1$. Finally, we impose trichotomy (that is, if $X \neq Y$ then $\mathsf{less}(X,Y)\vee\mathsf{less}(Y,X)$) by a relation trich with probability formula

$$F_{\mathsf{trich}} = 1 - \max\{(1-\mathsf{less}(X,Y))\cdot(1-\mathsf{less}(Y,X))|X,Y; X \neq Y\}.$$

Using the relation less we can encode a relation first indicating the smallest element of the domain:

$$F_{\mathsf{first}}(X) = 1 - \max\{\mathsf{less}(Y,X)|Y; Y = Y\}.$$

We can also encode the relation successor(X,Y), which is true if and only if Y is the smallest element larger than X:

$$F_{\mathsf{successor}}(X,Y) = \mathsf{less}(X,Y)\cdot(1-\max\{\mathsf{less}(X,Z)\cdot\mathsf{less}(Z,Y)|Z; Z = Z\}).$$

While probability formulas can encode any quantified Boolean formula using only max as combination function, they are less readable than their logical counterparts. So for the rest of this proof, we will use logical expressions instead of probability formulas.

Introduce a unary relation state_q for each state q and define $F_{\mathsf{state}_q}(X) = 1/2$. Atom $\mathsf{state}_q(X)$ denotes that the machine is in state q at computation step X. Also, introduce a binary relation symbol_σ for each symbol σ in the alphabet, and specify $F_{\mathsf{symbol}_\sigma}(X,Y) = 1/2$. Atom $\mathsf{symbol}_\sigma(X,Y)$ denotes that σ is written in the Yth position of the tape in the Xth computation step. Finally, introduce a binary relation head with $F_{\mathsf{head}}(X,Y) = 1/2$, meaning that the machine head is in the Yth position of the tape in the Xth computation step. We must guarantee that at any computation step the machine is in a single state, each tape position has a single symbol, and the head is at a single tape position. We do so by introducing a proposition r and a probability formula F_r that encodes the logical expression:

$$\forall X\Bigg[\bigvee_q\Bigg(\mathsf{state}_q(X)\wedge\bigwedge_{q'\neq q}\neg\mathsf{state}_{q'}(X)\Bigg)\wedge\bigvee_\sigma\Bigg(\forall Y\Bigg[\mathsf{symbol}_\sigma(X,Y)\wedge\bigwedge_{\sigma'\neq\sigma}\neg\mathsf{symbol}_{\sigma'}(X,Y)\Bigg]\Bigg)$$
$$\wedge\quad\Bigg(\exists Y\Bigg[\mathsf{head}(X,Y)\wedge\forall Z\Big((Z\neq Y)\rightarrow\neg\mathsf{head}(X,Z)\Big)\Bigg]\Bigg)\Bigg],$$

We introduce a proposition start with a probability formula $F_{\mathsf{start}}(X)$ imposing the initial configuration of the machine, and a proposition compute with a probability formula F_{compute} encoding the transitions of the machine. The formula F_{start} is simple: it fixes the first element using the construction $\forall X[\mathsf{first}(X)\rightarrow\mathsf{state}_{q_0}(X)\wedge\mathsf{head}(X,X)]$. The formula F_{compute} is more elaborate; we only sketch the construction by Grädel. First, F_{compute} encodes the conjunction of two logical formulas, one affecting the tape positions that do not change, and the other actually encoding the transitions. The first formula is

$$\forall X,Y,Z,W\Bigg[\bigwedge_\sigma\Big(\mathsf{symbol}_\sigma(X,Y)\wedge(Z\neq Y)\wedge\mathsf{head}(X,Z)\wedge\mathsf{successor}(X,W)\rightarrow\mathsf{symbol}_\sigma(W,Y)\Big).$$

The second formula is

$$\forall X,Y,W\Bigg[\bigwedge\nolimits_{q,\sigma}\Big(\mathsf{state}_q(X)\wedge\mathsf{head}(X,Y)\wedge\mathsf{symbol}_\sigma(X,Y)\wedge\mathsf{successor}(X,W)\Big)$$
$$\rightarrow\Big(\bigvee\nolimits_{(q',\sigma',m)\in\delta(q,\sigma)}\mathsf{state}_{q'}(W)\wedge\mathsf{symbol}_{\sigma'}(W,Y)\wedge\mathrm{HEADMOVE}\Big)\Bigg],$$

where HEADMOVE is the formula $\exists Z[(Z = Y+m)\wedge\mathsf{head}(W,Z)]$, and $Z = Y+m$ is syntactic sugar for a construction that can be accomplished with the successor relation.

We are still to impose that the constraints we included are in fact enforced by any interpretation. We do so by using evidence $\{\mathsf{irref} = 1\}$, $\{\mathsf{trans} = 1\}$, $\{\mathsf{trich} = 1\}$. We also need to impose that the input is represented on the tape. So use

evidence $\{\mathsf{first}(1) = 1\}$, and $\{\mathsf{successor}(i-1, i) = 1\}$ for $i = 2, \ldots, n$, to order n elements of the domain. And use evidence $\{\mathsf{symbol}_{\sigma_0^i}(1, i) = 1\}$ for $i = 1, \ldots, n$. We also need to guarantee that at the first computation step the tape contains only blanks after the input. So insert a proposition blanks with a probability formula F_{blanks} encoding the logical formula

$$\forall X \left[\mathsf{first}(X) \rightarrow \forall Y, Z \left[\left(\mathsf{symbol}\ (X, Y) \wedge \mathsf{successor}(Y, Z) \right) \rightarrow \mathsf{symbol}\ (X, Z) \right] \right],$$

and use evidence $\{\mathsf{symbol}\ (1, n+1) = 1\}$ and $\{\mathsf{blanks} = 1\}$. To conclude the simulation of the machine, we must be able to determine when the machine reaches the accepting state q_a. To do so, introduce a proposition accept with probability formula

$$F_{\mathsf{accept}} = \max\{\mathsf{state}_{q_a}(X) | X; X = X\}.$$

If we now take a domain of size $N = 2^p$ (recall that N is written in binary), we have that an interpretation is consistent with the evidence and with $\{\mathsf{accept} = 1\}$ if and only if the corresponding computation of the machine accepts the input. Thus, the number of accepting paths is at least half of the total number of computation paths if and only if

$$\mathbb{P}(\mathsf{accept} = 1 | \mathcal{E}) \geq 1/2,$$

where $\mathcal{E}$ is the evidence. □

If we replace the assumption of binary encoding of the domain with a unary encoding and remove the constraint on the arity of relations, we obtain the same complexity.

Theorem 11. *Inference is* PEXP*-complete when the domain is specified as a list and the only combination function is max (and there is no bound on the arity of relations).*

Proof. Membership follows as we can build an exponentially-large auxiliary Bayesian network and run inference in it. To show hardness, observe that we can represent 2^n elements using a binary domain (say, $\mathcal{D} = \{0, 1\}$) and an n-ary relation (i.e., every $a \in \mathcal{D}^n$ is understood as distinct element). Thus, the proof of Theorem 10 can be replicated by interpreting every logical variable as a vector of logical variables of length n, and using a domain with elements 0 and 1. □

If we then bound the arity of relations, we obtain:

Theorem 12. *Inference is* PSPACE*-complete when the domain is specified as a list, the relations have bounded arity and the only combination function is max.*

Proof. We show membership by devising a polynomial-space algorithm. Let k be a bound on the arity of the relations; then an interpretation μ assigns a relation consisting of at most $|\mathcal{D}|^k$ values to each of the $n = |V|$ relation symbols. This takes polynomial space. We show that probability formulas can be evaluated (for a fixed interpretation μ) in polynomial space by induction in the number of nested subformulas. If a formula has no combination function, then only constant space is required to evaluate it; otherwise, assume that every subformula takes polynomial space. Convex combinations of such subformulas again take only polynomial space. So consider a probability formula $F = \max\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}$, where each F_i^μ is by assumption computed using polynomial space. Evaluating F^μ takes at most $\mathcal{O}(m \cdot k \cdot |\alpha| \cdot f)$ space (required to count the assignments of the logical variables and decide the maximum over $F_1^\mu, \ldots, F_k^\mu$ for each assignment), where $|\alpha|$ is the size of the constraint α, and f is an upper bound on space required to compute a subformula F_i^μ. Thus, evaluating any probability formula (for a fixed interpretation) takes polynomial space. The probability of an interpretation can be computed in polynomial space by a multiplication of all the terms involved – we only need an accumulator to store intermediate values of the multiplication and a counter over the relations $r \in V$ and the tuples $a \in \mathcal{D}^{|r|}$, all of which can be implemented in space $\mathcal{O}(|V| \cdot |r| \lg |\mathcal{D}|)$, which is a polynomial in the input size. The desired probability can be computed in polynomial space by enumerating every possible interpretation and adding up the relevant parts (possibly followed by normalization), and dividing the probability of target and evidence by the probability of evidence (this takes polynomial space). Deciding whether this probability exceeds the given threshold takes polynomial space.

Hardness is shown by a many-one reduction from QBF, which is PSPACE-complete [29]:

Input: A quantified Boolean formula in the form

$$Q_1 X_1 Q_2 X_2 \ldots Q_n X_n [\psi_1 \wedge \cdots \wedge \psi_m],$$

where each Q_i is either $\exists$ or $\forall$, and each ψ_i is a quantifier-free 3-clause over logical variables $X_1, \ldots, X_n$.
Decision: Is the formula satisfiable?

Our strategy for the reduction is to encode the quantified Boolean formula by a sequence of "nested" max and min combinations, each representing a quantified subformula of φ; a clause is encoded by an equality constraint. Recall that we can encode quantified Boolean formulas using the probability formula that only contain max as combination function, and that by combining a max and convex combination we obtain min.

Denote by $\mathbf{Y}_i$ the logical variables appearing in ψ_i (there are at most 3 such variables). For each clause ψ_i, $i = 1, \ldots, m$, introduce a relation clause_i and specify $F_{\mathsf{clause}_i}(\mathbf{Y}_i, Z) = \max\{1|\emptyset; \alpha\}$, where Z represents an assignment "true", and α is an equality constraint that is satisfied if and only if ψ_i is satisfied by the corresponding assignment $\mathbf{Y}_i$. For example, if ψ_1 is $(\neg X_1 \vee X_2 \vee X_3)$ and ψ_2 is $(X_1 \vee \neg X_4 \vee X_3)$ then relation clause_1 is associated with the probability formula $F_{c_1}(X_1, X_2, X_3, Z) = \max\{1|\emptyset; X_1 \neq Z \vee X_2 = Z \vee X_3 = Z\}$, and relation clause_2 is associated with the probability formula $F_{c_2}(X_1, X_4, X_3, Z) = \max\{1|\emptyset; X_1 = Z \vee X_4 \neq Z \vee X_3 = Z\}$. Now introduce a relation sat with parents $\{\mathsf{clause}_i : i = 1, \ldots, m\}$, and probability formula

$$F_{\mathsf{sat}}(Z) = \mathsf{opt}_1 \left\{ \cdots \mathsf{opt}_n \left\{ \prod_{i=1}^{m} \mathsf{clause}_i(\mathbf{Y}_i, Z) \middle| X_n \right\} \cdots \middle| X_1 \right\},$$

where opt_j is max if $Q_j = \exists$ and opt_j is min if $Q_j = \forall$. Specify the domain as $\mathcal{D} = \{f, t\}$, representing false and true assignments, respectively. Then $\mathbb{P}(\mathsf{sat}(t) = 1) \geq 1/2$ if and only if the QBF problem is a YES instance. □

The previous result suggests that nesting combination functions boost the computational power of inference in RBNs. However, using an unbounded number of nestings does not seem necessary for modeling realistic domains. It is interesting then to look at the complexity of inference when only a bounded number of nestings is allowed. To this end, we define the nesting level of a probability formula:

Definition 13. A probability formula $F = q$ or $F = r(X_1, \ldots, X_n)$ has nesting level zero. A probability formula $F = F_1F_2 + (1 - F_2)F_3$ has nesting level equal to the highest nesting level of F_1, F_2, F_3. Finally, a probability formula $F = \mathsf{comb}\{F_1, \ldots, F_k|Y_1, \ldots, Y_m; \alpha\}$ has nesting level equal to the highest nesting level over all probability formulas F_i in it, plus one.

We consider:

Assumption 14. The nesting level of any probability formula is bounded by a constant $k \geq 0$.

Limiting the nesting level brings inference down to the counting hierarchy:

Theorem 15. *Inference is* $\text{PP}^{\Sigma_k^{\text{P}}}$*-complete when the only combination function is max, the relations have bounded arity and probability formulas have nesting level at most k.*

Proof. Membership follows as we can solve inference by a non-deterministic Turing machine that "guesses" an interpretation and performs binary search to evaluate the interpretation of each formula with polynomially many calls to an oracle Σ_k^{P}. To see this, fix an interpretation μ. Then the interpretation of a formula of nesting level 0 is an arithmetic expression, which can be compared against a given threshold in deterministic polynomial time. So consider the computation of $F^{\mu}(a)$ where $F = \mathsf{comb}\{F_1, \ldots, F_\ell|Y_1, \ldots, Y_m; \alpha\}$ is a formula of nesting level j (hence $F_1, \ldots, F_\ell$ have nesting level at most $j-1$). By induction hypothesis, assume that we can decide if $F_i(a, b) \geq \gamma_i$ with a Σ_{j-1}^{P} machine, for $i = 1, \ldots, \ell$. We can decide $F^{\mu}(a) \geq \gamma$ by "guessing" a configuration b of $Y_1, \ldots, Y_m$, then in deterministic polynomial time verifying if $F_i(a, b) \geq \gamma$ holds for some $i = 1, \ldots, \ell$. More generally, the interpretation of a formula F of nesting level j can be decided by a machine Σ_j^{P} by first solving an arithmetic expression in deterministic polynomial time (to find a threshold for each subformula), and then deciding each combination expression of level j. Finally, the probability $\mathbb{P}_{\mathcal{D}}(\mu)$ is computed in deterministic polynomial time as the product of $F_r(a)$ or $1 - F_r(a)$ (note that the number of tuples $a \in r^{\mu}$ is polynomially bounded). To obtain the desired probability note that deciding $\mathbb{P}(r(a) = 1|\mathcal{E}) \geq \gamma$ is equivalent to deciding $\mathbb{P}(\{r(a) = 1\} \wedge \mathcal{E}) + \gamma\mathbb{P}(\neg\mathcal{E}) \geq \gamma$. The latter can be computed by accepting paths that are either consistent with $\{r(a) = 1\} \wedge \mathcal{E}$ with weight 1 or consistent with $\neg\mathcal{E}$ with weight γ.

Let $\exists^{\geq q} X_1, \ldots X_n \varphi$ denote a (quantified) Boolean formula that is satisfied if and only if φ is satisfied by at least q assignments to $X_1, \ldots, X_n$ that satisfy φ. We obtain hardness by reduction from the following $\text{PP}^{\Sigma_k^{\text{P}}}$-complete problem [36][2]:

[2] Wagner defined complete problems for this class as CNF formulas (with no bound on clause length) and with the starting quantifier fixed; we adopt this modified form (with the ending quantifier fixed to $\exists$) so that we can we use clauses of length at most 3; note that any formula in CNF can be converted into an equisatisfiable 3-CNF formula with the inclusion of new existentially quantified variables.

Input: A formula in the form

$$\exists^{\geq q}\mathbf{X}_0\exists\mathbf{X}_1\forall\mathbf{X}_2\cdots\exists\mathbf{X}_k\,[\psi_1\wedge\cdots\wedge\psi_m]\,,$$

if k is odd, or in the form

$$\exists^{\geq q}\mathbf{X}_0\forall\mathbf{X}_1\exists\mathbf{X}_2\cdots\exists\mathbf{X}_k\,[\psi_1\wedge\cdots\wedge\psi_m]\,,$$

if k is even, where q is a rational given in binary notation, each $\mathbf{X}_i$ is a tuple of logical variables (not appearing in other tuples), and each ψ_i is a 3-clause over $\mathbf{X}_0,\ldots,\mathbf{X}_k$.
Decision: Is the formula satisfiable?

We will use a nesting of maxs and mins to encode the quantifiers over $\mathbf{X}_1,\ldots,\mathbf{X}_k$, and represent the variables in $\mathbf{X}_0$ as relations. Note that we can encode Boolean disjunctions such as $X\vee\neg Y\vee Z$ using the probability formula $\max\{x, 1-y, z|\emptyset;\}$, where x, y and z are propositions (i.e., 0-arity relations) representing the corresponding Boolean variables. For each X_j in $\mathbf{X}_0$, introduce a proposition x_j with no parents and with probability formula $F_{x_j}=1/2$. For each clause, $i=1,\ldots,m$, introduce relations $c_i(\mathbf{Y},Z)$, $d_i(\mathbf{Y},Z)$ and $e_i(\mathbf{Y},Z)$, where $\mathbf{Y}$ are the variables in $\mathbf{X}_1,\ldots,\mathbf{X}_k$ that appear in ψ_i (there are at most 3 such variables). The parents of c_i are d_i and e_i; d_i has no parents; the parents of e_i are the relations x_j corresponding to counting variables in the ith clause. Each relation d_i encodes whether the assignment of $\mathbf{Y}$ satisfies the respective clause, while the relation e_i encodes whether the clause is satisfied by some assignment of the counting variables (and setting the quantified variables so that their values do not affect satisfiability). Let n be the number of variables in $\mathbf{X}_0$, and define $F_{ij}=1-x_j$ if X_j appears negated in the ith clause, $F_{ij}=x_j$ if X_j appears nonnegated in the i the clause, $F_{ij}=0$ if X_j does not appear in the ith clause, and $F_{ij}=1$ if X_j appears both negated and nonnegated in the ith clause. Specify:

$$F_{d_i}(\mathbf{Y},Z)=\max\{1|\emptyset;\alpha\}\,,$$
$$F_{e_i}(\mathbf{Y},Z)=\max\{F_{i1},\ldots,F_{in}|\emptyset;\neg\alpha\}\,,$$
$$F_{c_i}(\mathbf{Y},Z)=\max\{d_i(\mathbf{Y},Z),e_i(\mathbf{Y},Z)|\emptyset;Z=Z\}\,,$$

where the constraint α is satisfied if and only if the respective assignment of $\mathbf{Y}$ satisfies the ith clause. For example, for the clause $(\neg X_1\vee X_2\vee X_3)$, where X_1 and X_2 are in $\mathbf{X}_0$ and X_3 is not, we introduce relations $c(X_3,Z)$, $d(X_3,Z)$, $e(X_3,Z)$, and specify $F_d(X_3,Z)=\max\{1|\emptyset;X_3=Z\}$, $F_e(X_3,Z)=\max\{1-x_1,x_2|\emptyset;\neg(X_3=Z)\}$, $F_c(X_3,Z)=\max\{d(X_3,Z),e(X_3,Z)\}$. Finally, introduce a unary relation sat whose probability formula is

$$F_{\text{sat}}(Z)=\text{opt}_1\left\{\cdots\text{opt}_k\left\{\prod_{i=1}^{m}c_i(\mathbf{Y}_i,Z)\middle|\mathbf{X}_k;\tau\right\}\cdots\middle|\mathbf{X}_1;\tau\right\}\,,$$

where opt_j is max if its corresponding quantifier is $\exists$ and opt_j is min otherwise (so e.g. opt_k is always max), $\mathbf{X}_j$ is the tuple of variables referred to by the jth quantifier in the input, and τ is some tautology. Finally, specify domain $\mathcal{D}=\{f,t\}$ and decide yes if and only if $\mathbb{P}(\text{sat}(t)=1)\geq q/2^n$. □

Note that for nesting level $k=0$, the proof above reduces a $\#\text{SAT}_{\geq}$ problem (complete for PP [2]) and thus provides an alternative proof that inference is PP-complete in RBNs without combination functions. For a nesting level $k=1$, Theorem 15 shows that inference is PP^{NP}-complete, which suggests that **the use of combination functions, even as simple as max, adds computational power to inference, even when combination functions are not nested**. The same problem suggests that nesting increases complexity, and should be used carefully by the model maker.

4.3. RBNs with mean and threshold combination functions

As we have seen in the previous section, the max combination function adds a computational power similar to quantified Boolean formulas. As discussed in Section 2, a different flavor is added by the combination function mean: it allows defining probabilities as proportions. We now consider RBNs whose probability formulas use mean and threshold; we leave the study of mean alone for the future.

By combining threshold and mean we can encode quantified Boolean formulas. To see this, consider a quantified formula $\varphi=\forall X\psi$ and the corresponding probability formula $F=\text{threshold}\{0.5\cdot\text{mean}\{G|X;X=X\}|\emptyset\}$, where (by inductive hypothesis) G encodes ψ. Then $F^{\mu}=1$ if and only if μ satisfies φ. Also, given a formula $\varphi=\exists X\phi$, we can build a probability formula $F=1-\text{threshold}\{0.5\cdot\text{mean}\{1-G|X;X=X\}\emptyset\}$, where G encodes ϕ, such that $F^{\mu}=1$ if and only if μ satisfies φ. Hence, RBNs with threshold and mean can decide QBF, which makes inference PSPACE-hard. This suggests that we need to place some constraint on how these functions are used, similar to what we have done with max. However, simply limiting the nesting level is not sufficient, as a formula that uses k nestings of threshold exhibits a different complexity than a formula with k nestings of mean; and a formula that mixes both combination functions might have yet another complexity. To keep things simple, we adopt:

Assumption 16. A probability formula is either a rational number, or an atom, or a convex combination, or a combination expression of the form $\mathsf{threshold}\{F_1 \cdot \mathsf{mean}\{G_1, \ldots, G_k | Y_1, \ldots, Y_m; \alpha\} + F_2 | \emptyset\}$, where F_1 and F_2 are probability formulas not containing combination expressions, and $G_1, \ldots, G_k$ are probability formulas.

It should be clear that the above assumption is equivalent to assuming that only mean and threshold are used, and that they interact in a very particular way.

We thus have:

Theorem 17. *Inference is $\mathsf{c}_k\mathsf{P}$-complete when the domain is specified as a list, the relations have bounded arity, Assumption 16 holds and the nesting level is at most $2k$.*

Proof. We show membership by constructing a probabilistic Turing machine that "guesses" an interpretation μ, and then performs binary search to evaluate the interpretation of each probability formula with polynomially many calls to an oracle $\mathsf{c}_{k-1}\mathsf{P}$. We can show that the latter is true inductively on the nesting level of formulas. So fix an interpretation μ. The interpretation of a formula of nesting level 0 is an arithmetic expression, and can thus be decided if greater than a given threshold in deterministic polynomial time. So consider the computation of $F^{\mu}(a)$ where $F = \mathsf{threshold}\{F_1 \cdot \mathsf{mean}\{G_1, \ldots, G_\ell | Y_1, \ldots, Y_m; \alpha\} + F_2 | \emptyset\}$ is a formula of nesting level j (hence F_1 and F_2 have nesting level 0, and $G_1, \ldots, G_\ell$ have nesting level at most $j-2$). By induction hypothesis, assume that we can decide if $G_i(a, b) \geq \gamma_i$ with a $\mathsf{c}_{j-2}\mathsf{P}$ machine, for $i = 1, \ldots, \ell$. Then we can decide $F^{\mu}(a) \geq \gamma$ by with a probabilistic Turing machine with oracle $\mathsf{c}_{j-1}\mathsf{P}$. The probability $\mathbb{P}_{\mathcal{D}}(\mu)$ is computed in deterministic polynomial time as the product of $F_r(a)$ or $1 - F_r(a)$ (note that the number of tuples $a \in r^{\mu}$ is polynomially bounded). The desired probability is obtained by counting paths that are either consistent with $\{r(a) = 1\} \wedge \mathcal{E}$ or consistent with $\neg\mathcal{E}$ (with weight γ), in order to decide if $\mathbb{P}(\{r(a) = 1\} \wedge \mathcal{E}) + \gamma\mathbb{P}(\neg\mathcal{E}) \geq \gamma$.

We prove hardness by reduction from $\#_k\mathsf{SAT}$, which is $\mathsf{c}_k\mathsf{P}$-complete [36]:

Input: A Boolean formula of the form

$$\exists^{\geq q_1}\mathbf{X}_1 \cdots \exists^{\geq q_k}\mathbf{X}_k\,[\psi_1 \wedge \cdots \wedge \psi_m]\,,$$

where each $\mathbf{X}_i$ is a tuple of logical variables, and each ψ_i is a 3-clause.
Decision: Is the formula satisfiable?

Denote by n the number of variables in $\mathbf{X}_1$. For each variable X_j in $\mathbf{X}_1$, introduce a zero-arity relation x_j with $F_{x_j} = 1/2$. For each 3-clause ψ_i introduce relation a $c_i(\mathbf{Y}, Z)$, where $\mathbf{Y}$ are the variables *not* in $\mathbf{X}_1$ that appear in the ith clause, and Z defined a "true" assignment. Define $F_{ij} = 0$ if X_j is in $\mathbf{X}_1$ or if X_j does not appear in the ith clause, $F_{ij} = 1 - x_j$ if X_j is in $\mathbf{X}_1$ and appears negated in the ith clause, $F_{ij} = x_j$ if X_j is $\mathbf{X}_1$ and appears nonnegated in the i the clause, and $F_{ij} = 1$ if X_j is in $\mathbf{X}_1$ and appears both negated and nonnegated in the ith clause. Specify:

$$F_{c_i}(\mathbf{Y}, Z) = 1 - \mathsf{th}\{0.5 \cdot \mathsf{mean}\{1 - F_{i1}, \ldots, 1 - F_{in} | \emptyset; \neg\alpha\} | \emptyset; Z = Z\}\,,$$

where α is an equality constraint over $\mathbf{Y}$ and Z that is satisfied if and only if ψ_i is satisfied by some variable in $\mathbf{X}_2, \ldots, \mathbf{X}_k$, and th is short for threshold. A moment's reflection should convince the reader that $F^{\mu}_{c_i}(a) = 1$ if and only if the corresponding assignment to variables in $\mathbf{Y}$ satisfy α (hence ψ_i), or if the interpretation μ assigns values to x_j which satisfy ψ_i. To encode the quantified formula and the conjunction of clauses, introduce $w(Z)$ and specify

$$F_w(Z) = \mathsf{threshold}\Big\{\mathsf{mean}\big\{F_2(\mathbf{X}_2, Z) | \mathbf{X}_2; Z = Z\big\} | \emptyset; Z = Z\Big\}\,,$$

where for $i = 2, \ldots, k$ we define

$$F_i(\mathbf{X}_2, \ldots, \mathbf{X}_{i-1}, Z) = \mathsf{threshold}\Big\{\gamma_{1,i} \cdot \mathsf{mean}\big\{F_{i+1}(\mathbf{X}_2, \ldots, \mathbf{X}_i, Z) | \mathbf{X}_i; Z = Z\big\} + \gamma_{2,i} | \emptyset; Z = Z\Big\},$$

$\gamma_{1,i} = 1/(2q_i)$ and $\gamma_{2,i} = 0$ if $q_i \geq 1/2$, $\gamma_{1,i} = 1/(2(1 - q_i))$ and $\gamma_{2,i} = 1 - 1/(2(1 - q_i))$ if $q_i < 1/2$, and

$$F_{k+1}(\mathbf{X}_2, \ldots, \mathbf{X}_k, Z) = \prod_{i=1}^{m} c_i(\mathbf{Y}_i, Z)\,.$$

Finally, return YES if $\mathbb{P}(w(t) = 1) \geq q_1/2^n$ and return NO otherwise. □

The proof above proves PP-completeness for formulas with nesting level 0, and thus also contains Theorem 7 as a special case.

4.4. RBN with polynomial-time combinations functions

We now consider RBNs with arbitrary combination functions given as part of the input. First, note that one might take an arbitrarily complex combination function to begin with, and then the complexity of inference would be entirely washed over by the complexity of evaluating combination functions (as already noted in Section 4). To prevent such situations, we might adopt:

Assumption 18. Any combination function $\mathsf{comb}\{q_1, \ldots, q_k\}$ is polynomial-time computable in the size of a reasonable encoding of its arguments $q_1, \ldots, q_k$.

The max, mean and threshold combination functions all satisfy this assumption. Thus, given the previous results, inference with polynomial-time combination functions can still produce problems with complexity ranging from PP to PEXP, passing through every class in CH and PSPACE, depending on the maximum nesting level allowed and on whether the arity of relations is bounded. Allowing arbitrary polynomial-time combination functions, however, allows for exponential behavior *even* when the relations have bounded arity and probability formulas have nesting level at most k, due to idiosyncrasies of combination functions. To see this, consider the probability formula

$$F(X_1, \ldots, X_n, Z) = \mathsf{comb}\{G(X_1, Z) + 2^{-1}G(X_2, Z) + \cdots + 2^{-n}G(X_n, Z) | X_1, \ldots, X_n; Z = Z\}, \quad (1)$$

where $G(X, Z) = \max\{1|\emptyset; X = Z\}$. To evaluate such a function, one may face an exponentially large multiset $\{q_1, \ldots, q_{2^n}\}$; if it is indeed necessary to go through all these elements, then the overall effort is exponential even if the function itself is polynomial on the size of the multiset. We have that:

Theorem 19. *Inference is* EXP*-complete when the relations have bounded arity and the combination functions are polynomial-time computable.*

Proof. To prove membership, first note that there are polynomially many groundings for all relations in the RBN (since relations have bounded arity). Thus there are exponentially many truth assignments for these groundings; go over each one of them, computing (and adding) the probabilities produced by evaluating polynomially many combination functions (each evaluation with effort at most exponential). A conditional probability can be compared against a threshold by encoding it as a linear combination on the probability of query and evidence as in the proof of Theorem 15.

To show hardness we only need to consider an combination function that encode an arbitrary decision problem that is complete for EXP. So introduce a relation w whose probability formula F_w is $F(X_1, \ldots, X_n, Z)$ as in Expression (1). Select some EXP-complete problem, and assume that comb encodes this problem (that is, it gets a string $x_1x_2\cdots x_n$ as input, and runs an unavoidably exponentially long computation to decide if that string is in the language). Note that such a combination function is allowed, because comb can select any of the 2^n replicas in its multiset, each encoding a binary number with n bits, and spend exponential time $O(2^{\mathrm{poly}(n)})$, which is polynomial in the encoding of the multiset (that takes $O(2^n \cdot n)$ bits). Now specify the domain as $\mathcal{D} = \{0, 1\}$, and consider the free variables $X_1, \ldots, X_n$ of this formula as specifying the input to the EXP-complete problem encoded by comb; the variable Z specifies a bit set. Verifying if $\mathbb{P}(w(x_1, \ldots, x_n, 1) = 1) \geq 1/2$ then solves the EXP-complete problem, and requires only polynomial-sized input and output. □

An alternative to assuming that the combination functions are polynomial-time computable is to adopt:

Assumption 20. Any probability formula $\mathsf{comb}\{F_1, \ldots, F_k | Y_1, \ldots, Y_m; \alpha\}$ is polynomial-time computable in the size of a reasonable encoding of its description.

One way to satisfy this assumption is to assume that 5 and that the number of logical variables bound in any probability formula is bounded by a constant; the latter is essentially the same as assuming a bound on the *quantifier depth* as defined by Jaeger [21]. Indeed, if the bound on the number of bound logical variables is say M, then there are at most $|\mathcal{D}|^M$ tuples to test with the equality constraint; for each one of them, recursively obtain the values to be used in the combination function, and then compute the latter with polynomial effort.

In any case, inference in RBNs where the probability formulas are polynomial-time computable exhibit a computational power equivalent to that of a Bayesian network:

Theorem 21. *Inference is* PP*-complete when the relations have bounded arity and the probability formulas are polynomial-time computable.*

Proof. An RBN without combination functions and a domain with a single element suffice to specify any Bayesian network, hence PP-hardness obtains. To show membership, note that there is a polynomial number of groundings, as relations have bounded arity; once a truth assignment is guessed nondeterministically for all these groundings, the computation of their

joint probability can be done in polynomial time (by assumption). Hence by adding up the result of all these truth assignments (in fact, a suitably scaled result), we obtain the desired inference. Again, conditional probabilities can be compared against a threshold by comparing a linear inequality on the probabilities of query and evidence. □

These results can be interpreted as follows. If we want to use arbitrary combination functions, and we only accept polynomial-time functions, then the problem becomes exponentially hard in the worst-case and fails to reveal many interesting cases. If on the other hand we take arbitrary functions but assume their interpretations to be polynomial-time computable, then RBNs are essentially a syntactic sugar for representing large Bayesian networks, with no additional computational power. It remains an open question to constrain RBNs so that arbitrary combination functions can be used, while still leading to interesting complexity results.

5. Query complexity

The results proved so far required RBNs whose size (in any reasonable encoding) could grow arbitrarily large. In many applications, however, the relational description of the model is rather succinct; or the model is constantly used with different domains and evidence scenarios. Thus, there is interest in studying the complexity of inference under the following assumption:

Assumption 22. The size of the RBN is bounded by a constant.

An equivalent assumption (as far as complexity is concerned) is to consider that the RBN is fixed; thus the complexity is analyzed in terms of domain size and evidence. For example, we might have a fixed model of interaction in a social network, and we might want to examine complexity of inference as a function of the number of individuals (given in unary or binary) and some information about certain individuals (given as evidence). This scenario was called *query complexity* in [8, Chapter 6.9] and later formalized in [7]. This is also related to the concept of compilation of an RBN [6,9].

Fixing the RBN, we have:

Theorem 23. *Inference is in* P *when the size of the* RBN *is bounded by a constant, the domain is specified as a list and the probability formulas are polynomial-time computable.*

Proof. As the relations have bounded arity and the number of relations is fixed, we can enumerate the interpretations in polynomial time. For each interpretation, we enumerate the random variables $r(a)$, for $r \in V$ and $a \in \mathcal{D}^{|r|}$ and compute its probability formula $F_r(a)$. As by assumption the latter takes polynomial time, the probability of the interpretation (obtained by the product of $F_r(a)$ and $1 - F_r(a)$) takes polynomial time. We can add up all probabilities consistent with query and evidence, and compare the sum against a threshold in polynomial time. □

And we have:

Theorem 24. *Inference is* PP*-complete when the size of the* RBN *is bounded by a constant, the domain is specified as a list and the combination functions are polynomial-time computable.*

Proof. Membership follows as we can "guess" an interpretation and then go through every relation r in the network, and every grounding $r(a)$; for each grounding we evaluate the probability formula $F_r(a)$ by enumerating every multiset in a combination expression; since the RBN is fixed this multiset contains at most a polynomial number in the size of the domain (where the degree of the polynomial can be as high as the number of variables in a combination expression, but this is considered fixed). Computing the combination function is then polynomial in its input.

To show hardness, consider the #SAT$_{\geq}$ problem:

Input: A Boolean formula $\exists^{\geq q} X_1, \ldots, X_n\, [\psi_1 \wedge \cdots \wedge \psi_m]$, where each ψ_i is a 3-clause and q is an integer.
Decision: Is the formula satisfiable?

This problem is PP-complete [36]. Introduce binary relations pos and neg indicating whether variable Y occurs in clause X, respectively, nonnegated or negated. Specify $F_{\mathsf{pos}}(X, Y) = F_{\mathsf{neg}}(X, Y) = 1/2$. Introduce also unary relations true and false indicating whether a variable is assigned the value "true" or "false", respectively. Specify $F_{\mathsf{true}}(X) = 1/2$ and $F_{\mathsf{false}}(X) = 1 - \mathsf{true}(X)$. To encode the clauses, introduce a unary relation clause with probability formula

$$F_{\mathsf{clause}}(X) = \max\{\mathsf{pos}(X, Y) \cdot \mathsf{true}(Y), \mathsf{neg}(X, Y) \cdot \mathsf{false}(Y) | Y; Y = Y\}.$$

Intuitively, $\mathsf{clause}(X)$ is true if and only if there is a variable Y that either occurs in clause ψ_X nonnegated and Y is assigned value "true", or Y occurs in ψ_X negated and Y is assigned "false". Introduce a unary relation isClause, representing whether X is a clause, and specify $F_{\mathsf{isClause}}(X) = 1/2$. Finally, introduce proposition sat with $F_{\mathsf{sat}} = \min\{\mathsf{clause}(X) \cdot \mathsf{isClause}(X) | X;$

$X = X\}$. Let the domain $\mathcal{D}$ be $\{1, \ldots, N\}$, where $N = \max\{n, m\}$. For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, include evidence $\{\mathsf{pos}(i, j) = 1\}$ and if literal X_j occurs in ψ_i, and $\{\mathsf{pos}(i, j) = 0\}$ otherwise; include $\{\mathsf{neg}(i, j) = 1\}$ if $\neg X_j$ occurs in ψ_i and $\{\mathsf{neg}(i, j) = 0\}$ otherwise; and include evidence $\{\mathsf{isClause}(i) = 1\}$ and for $j > n$ (if any) $\{\mathsf{isClause}(j) = 0\}$. Let $\mathcal{E}$ be this evidence. Then $\mathbb{P}(\mathsf{sat} = 1|\mathcal{E}) \geq q/2^{2n}$ if and only if the corresponding Boolean formula is satisfiable. □

Note that the previous result shows that query complexity is PP-complete when we allow only max, mean or threshold, or any combination of these. Moreover, bounding the nesting level does not seem to have any effect on complexity.

6. On the functional complexity of inference

Strictly speaking, the inference problem is a functional problem, where one is interested in computing the value of a probability, and not on deciding whether this value exceeds a given threshold. This has led many researchers to study the complexity of the functional version of inference problem, usually within the framework of #P-completeness theory.

The complexity class #P contains the *integer-valued* functions computed by a counting Turing machine in polynomial time; a counting Turing machine is a non-deterministic Turing machine that prints in binary notation, on a separate tape, the number of accepting computations induced by the input [35]. The canonical problem for #P is model counting: given a CNF formula, count the number of satisfying assignments. Roth [31] showed that model counting can be reduced to inference in Bayesian networks, and hence the latter is #P-hard. The inference problem however is not in #P, since the latter is defined as integer-valued problems and the former produces rational values. In fact, Roth notes that "strictly speaking the problem of computing the degree of belief is not in #P, but easily seem equivalent to a problem in this class"; but he did not go as far as to define equivalence. Valiant himself used the class #P to study the complexity of the problem of computing the probability that a path between two given nodes will exist in a random graph. To cope with the non-integer character of this problem, he investigated instead a different problem that asked to compute the number of graphs produced by realizations of the random graph that connected the two nodes. The two problems are related by a normalization constant (the total number of graphs). An inference problem with no evidence can be seen from the same perspective, and investigated as a weighted counting problem with integer weights. This is somewhat the approach followed by Kwisthout [25], who defined a function as in #P modulo normalization if it can be written as $f(x)/(k!)^{\mathrm{poly}(|x|)}$, where f is a function in #P, k is some constant, and $\mathrm{poly}(|x|)$ denotes a polynomial in the size of the input.

The case of computing conditional probabilities is even more problematic. One could argue that inference with evidence should be cast as the division of two integer-valued functions (one computing the numerator, and the other computing), thus moving away from issues with rational values. However, it is believed that the class #P is not closed under division. One could then envisage developing a theory of counting with rationals; such a theory, if developed, would require an extensive work of finding complete problems, building hierarchies, and so on.

An alternative is to define a notion of equivalence so that we can use the #P-completeness theory to investigate the complexity of rational-valued functions. This is for instance the approach followed by Grove et al. [19], who coined the concept of #P-easy functions: A function is #P-easy if it is computed by a function in #P followed by polynomial post-processing. And they defined a function as #P-complete if it is #P-easy and can compute any #P-easy function with a 1-Turing reduction (i.e. a Turing reduction with a single call to the oracle). Toda and Watanabe [33] showed that a every function that can be computed by counting Turing machine with a oracle PH can be reduced via 1-Turing reduction to a function computed by a counting Turing machine with an oracle #P. Thus, Grove et al.'s approach cannot differentiate between the cases in Theorems 7 and 15. This same issue appears in the work of de Campos et al. [10]. They defined functions that are equivalent to #P-complete functions as functions that can be computed by a 1-Turing machine with a single call to an oracle #P.

A more stringent approach was adopted by Bulatov et al. [5] to study the complexity of counting solutions to weighted constraint satisfaction problems. In lieu of Turing reductions, they used weighted reductions: a Karp reduction that preserves the number of accepting paths followed by a polynomial-time scaling of the solution (so that rationals can be brought up to bear on integer values). While this guarantees the distinction of classes that uses oracles in the polynomial hierarchy, it requires redefining other concepts, such as the classes in the counting hierarchy. It is worth noting however, that PP and #P offer the same power when used as oracles to a polynomial (non)deterministic machine [3].

In this paper, we avoided the difficulties introduced by rational-valued functions and investigated the complexity of the decision variant of the inference problem (with evidence). We feel that the decision variant captures the main ingredients of the computational power of inferences with relational Bayesian networks, without requiring extra definitions and argumentation (as the functional version). Many of the results we show here could be equally proved for the functional version, as long as we defined an appropriate notion of equivalence (say, using weighted reductions). For example, Theorems 7, 8, 10, 11 could all be restated in terms of its functional variant, and their proofs would be nearly identical. Also, the class of functions computed by probabilistic Turing machines with polynomial space is also FPSPACE (the class of functions computed by polynomial-space deterministic Turing machines) [26]; hence, Theorem 12 could also be restated in its functional version. Theorem 15 could also be "translated" to the function version, with some care. This is because the canonical complete problem for $\#\mathrm{P}^{\Sigma_k^{\mathrm{p}}}$ uses either formulas in CNF or in DNF, depending on the parity of k [13]. So different proofs would have to be given to each case. Regarding Theorem 17, we are not aware of complete problems for the functional equivalent of $\mathrm{C}_k\mathrm{P}$. Functional versions of Theorems 19, 21, 23 and 24 could also be proved with little effort.

7. Conclusion

We have examined the complexity of inference in relational Bayesian networks. We argued that a great deal of complexity is added by the combination functions alone, which led us to analyze the complexity in terms of the type of combination functions allowed. We first considered networks with no combination function, and showed that (from a complexity viewpoint) they are essentially as powerful as Bayesian networks. We then analyzed the complexity when we only allow maximization as combination function. In this case, complexity results were much more interesting, covering classes defined by probabilistic Turing machines with oracles in the polynomial hierarchy, PSPACE and even PEXP. The distinction of complexity classes was produced by limiting the arity of relations, the encoding of the domain and the number of nestings of combination functions. We then discussed the complexity of mean and threshold as combination functions: the former specified probabilities in terms of proportions; the latter allows for piecewise functions. We argued that by combining these functions we obtain at least the same computation power as maximization, so the previous results apply here as well. And we showed that with a suitable constraint on the use of both functions, we obtain complexity in all levels of the counting hierarchy. At last, we investigated the use of arbitrary combination functions. We showed that enforcing polynomial combination functions can still lead to exponential inference problems. On the other hand, constraining probability formulas to be polynomial in their encoding brings inference to the same complexity as Bayesian networks, and simplifies the discussion. We left as future work the development of criteria that allow arbitrary combination functions and yet enables interesting complexity results. We also left open the complexity when only mean is allowed.

We then considered complexity when the relational model is fixed. This can be either because the relational model is concise enough so that its size can be considered constant, or because we are interested in analyzing the amortized complexity of many inferences with the same relational Bayesian network. We show that when the model is fixed and probability formulas are polynomial (e.g., there are no combination functions), inference is polynomial. And when the model is fixed and combination functions are polynomial (e.g., maximization, threshold and mean), then inference is PP-complete.

An interesting direction for the future is to investigate the complexity with respect to graphical features of the model. For instance, inference in bounded-treewidth Bayesian networks is polynomial [22]. Our results suggest that even when the grounding of the relational network produces a polynomial-sized graph, inference can still remain intractable.

Acknowledgements

The first author received financial support from the São Paulo Research Foundation (FAPESP) grant #2016/01055-1 and the CNPq grant #303920/2016-5 (PQ). The second author is partially supported by the CNPq grant #308433/2014-9 (PQ).

References

[1] S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.
[2] D. Bailey, V. Dalmau, P. Kolaitis, Phase transitions of PP-complete satisfiability problems, Discrete Appl. Math. 155 (12) (2007) 1627–1639.
[3] J. Balcázar, R. Book, U. Schöning, The polynomial-time hierarchy and sparse oracles, J. ACM 33 (1986) 603–617.
[4] R. Beigel, N. Reingold, D. Spielman, PP is closed under intersection, J. Comput. Syst. Sci. 50 (1) (1995) 191–202.
[5] A. Bulatov, M. Dyer, L. Goldberg, M. Jalsenius, M. Jerrum, D. Richerby, The complexity of weighted and unweighted #CSP, J. Comput. Syst. Sci. 78 (2012) 681–688.
[6] M. Chavira, A. Darwiche, M. Jaeger, Compiling relational Bayesian networks for exact inference, Int. J. Approx. Reason. 42 (1–2) (2006) 4–20.
[7] F. Cozman, D. Mauá, Bayesian networks specified using propositional and relational constructs: combined, data, and domain complexity, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 3519–3525.
[8] A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge University Press, 2009.
[9] A. Darwiche, G. Provan, Query DAGs: a practical paradigm for implementing belief-network inference, J. Artif. Intell. Res. 6 (1997) 147–176.
[10] C. de Campos, G. Stamoulis, D. Weyland, A Structured View on Weighted Counting with Relations to Quantum Computation and Applications, Tech. rep., Electronic Colloquium on Computational Complexity, 2013.
[11] L. de Raedt, Logical and Relational Learning, Springer-Verlag, 2008.
[12] L. de Raedt, P. Frasconi, K. Kersting, S. Muggleton, Probabilistic Inductive Logic Programming: Theory and Applications, Springer-Verlag, 2008.
[13] A. Durand, M. Hermann, P. Kolaitis, Subtractive reductions and complete problems for counting complexity classes, Theor. Comput. Sci. 340 (3) (2005) 496–513.
[14] H. Enderton, A Mathematical Introduction to Logic, Academic Press, 1972.
[15] D. Geiger, T. Verma, J. Pearl, Identifying independence in Bayesian networks, Networks 20 (5) (1990) 507–534.
[16] L. Getoor, B. Taskar, Introduction to Statistical Relational Learning, The MIT Press, 2007.
[17] W. Gilks, A. Thomas, D. Spiegelhalter, A language and program for complex Bayesian modelling, Statistician 43 (1993) 169–178.
[18] E. Grädel, Finite model theory and descriptive complexity, in: Finite Model Theory and Its Applications, Springer, Berlin, Heidelberg, 2007, pp. 125–229.
[19] A. Grove, J. Halpern, D. Koller, Asymptotic conditional probabilities: the unary case, SIAM J. Comput. 25 (1) (1996) 1–51.
[20] M. Jaeger, Relational Bayesian networks, in: Proc. of the 13th Conf. of Uncertainty in AI (UAI), 1997, pp. 266–273.
[21] M. Jaeger, Complex probabilistic modeling with recursive relational Bayesian networks, Ann. Math. Artif. Intell. 32 (1) (2001) 179–220.
[22] D. Koller, N. Friedman, Probabilistic Graphical Models, The MIT Press, 2009.
[23] D. Koller, A. Pfeffer, Object-oriented Bayesian networks, in: Proc. of the 13th Conf. on Uncertainty in AI (UAI), 1997, pp. 302–313.
[24] D. Koller, A. Pfeffer, Probabilistic frame-based systems, in: Proc. of the 15th National Conf. on AI (AAAI), 1998, pp. 580–587.
[25] J. Kwisthout, The Computational Complexity of Probabilistic Inference, Tech. Rep. ICIS-R11003, Radboud University Nijmegen, 2011.
[26] R. Ladner, Polynomial space counting problems, SIAM J. Comput. 18 (6) (1989) 1087–1097.
[27] D. Lunn, C. Jackson, N. Best, A. Thomas, D. Spiegelhalter, The BUGS Book: A Practical Introduction to Bayesian Analysis, CRC Press/Chapman and Hall, 2012.

[28] A. Meyer, L. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential time, in: Proceedings of the 13th Annual Symposium on Switching and Automata Theory, 1972, pp. 125–129.
[29] C. Papadimitriou, Computational Complexity, Addison-Wesley Publishing, 1994.
[30] D. Poole, Probabilistic Horn abduction and Bayesian networks, Artif. Intell. 64 (1993) 81–129.
[31] D. Roth, On the hardness of approximate reasoning, Artif. Intell. 82 (1–2) (1996) 273–302.
[32] S. Toda, PP is as hard as the polynomial-time hierarchy, SIAM J. Comput. 20 (5) (1989) 865–877.
[33] S. Toda, O. Watanabe, Polynomial-time 1-Turing reductions from #PH to #P, Theor. Comput. Sci. 100 (1992) 205–221.
[34] J. Tóran, Complexity classes defined by counting quantifiers, J. ACM 38 (3) (1991) 753–774.
[35] L. Valiant, The complexity of enumeration and reliability problems, SIAM J. Comput. 8 (3) (1979) 410–421.
[36] K. Wagner, The complexity of combinatorial problems with succinct input representation, Acta Inform. 23 (1986) 325–356.