The Descriptive Complexity of Bayesian Network Specifications

Fabio G. Cozman[1] and Denis D. Mauá[2]

[1] Escola Politécnica, Universidade de São Paulo - Brazil
[2] Instituto de Matemática e Estatística, Universidade de São Paulo - Brazil

Abstract. We adapt the theory of descriptive complexity to Bayesian networks, by investigating how expressive can be specifications based on predicates and quantifiers. We show that Bayesian network specifications that employ first-order quantification *capture* the complexity class PP; that is, any phenomenon that can be simulated with a polynomial time probabilistic Turing machine can be also modeled by such a network. We also show that, by allowing quantification over predicates, the resulting Bayesian network specifications *capture* the complexity class $\mathsf{PP}^{\mathsf{NP}}$, a result that does not seem to have equivalent in the literature.

1 Introduction

One can find a variety of "relational" Bayesian networks in the literature, where constructs from first-order logic are used to represent whole populations and repetitive patterns [7, 15, 16]. Such networks may be specified by diagrams or by text; typically they are viewed as templates that can be grounded into finite propositional Bayesian networks whenever needed.

It is only natural to ask what is the *expressivity* of Bayesian network specifications based on predicates and quantifiers. That is, what can and what cannot be modeled by these specifications, and at what computational costs. To address these questions, we are inspired by the well-known theory of *descriptive complexity* for logical languages [4, 9]. It does not seem that the descriptive complexity of Bayesian networks has been investigated in previous work, and to do so we adapt existing insights and results. Because the topic is novel, most of this paper consists of building a framework in which to operate.

In fact, one should expect "relational" Bayesian network specifications to exhibit properties that cannot be matched by propositional networks — much as first-order logic goes beyond propositional logic. This sort of phenomenon has been already noted in connection with *lifted* inference algorithms [18, 10], but has not been characterized in terms of descriptive complexity.

We define precisely what we mean by "Bayesian network specifications" in Section 2, and present some necessary background in Section 3. We then move to our main results in Sections 4 and 5.

We show that Bayesian network specifications that employ first-order quantification *capture* the complexity class PP. That is, a language is in PP *if and* *only if* its strings encode valid inferences in a Bayesian network specified with predicates and first-order quantifiers. Note that this is a much stronger statement than PP-completeness. And then we look at specifications that allow quantification over predicates, and show that such "second-order" Bayesian networks capture the complexity class $\mathsf{PP}^{\mathsf{NP}}$. It does not seem that previous results on descriptive theory have reached this latter complexity class.

Intuitively, these results can be interpreted as follows: suppose we have a (physical, social, economic) phenomenon that can be simulated by a probabilistic Turing machine in polynomial time: given an input, the machine will run for a number of steps that is polynomial in the length of the input, and the machine will stop with the same YES-NO answer as produced by the phenomenon. Our results show that the phenomenon can be modeled by a Bayesian network specification based on predicates and first-order quantification in the sense that, given the input as evidence, an inference with the network will produce the same result as the phenomenon. But what happens if the phenomenon is so complex that it requires even more computation to be simulated? For instance, what happens if the phenomenon requires a polynomial time probabilistic Turing machine with another nondeterministic Turing machine as oracle? This phenomenon cannot be modeled by a "relational" Bayesian network specification, unless widely accepted assumptions about complexity classes collapse. However, our results show that this phenomenon *can* be modeled by a Bayesian network specification that allows quantification over predicates.

We further discuss the intellectual significance of these results in the concluding Section 6.

2 Specifying Bayesian networks with logical constructs

In the Introduction we loosely mentioned "relational" Bayesian networks. To make any progress, we must precisely define what is here allowed in specifying Bayesian networks. Our strategy, described in this section, is to adopt a proposal by Poole [14] to mix probabilistic assessments and logical equivalences [2].

2.1 Preliminaries

First, to recap: a *Bayesian network* is a pair consisting of a directed acyclic graph $\mathbb{G}$ whose nodes are random variables, and a joint probability distribution $\mathbb{P}$ over the variables in the graph, so that $\mathbb{G}$ and $\mathbb{P}$ satisfy the Markov condition (a random variable is independent of its nondescendants given its parents). The Markov condition induces a factorization of probabilities [11].

In this paper every random variable is binary with values 0 and 1, respectively signifying false and true.

We only consider textual specifications, mostly relying on formulas of function-free first-order logic with equality (denoted by FFFO). That is, most formulas we contemplate are well-formed formulas of first-order logic with equality but without functions, containing predicates from a finite relational vocabulary, negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\Rightarrow$), equivalence ($\Leftrightarrow$), existential quatification ($\exists$) and universal quantification ($\forall$). Later we discuss second-order quantification over predicates.

First-order theories are interpreted as usual [5], using *domains*, that are just sets, and *interpretations* that associate predicates with relations, and constants with elements; that is, an interpretation is a truth assignment for every grounding of every predicate. A pair domain/interpretation is a *structure*. We only deal with finite domains in this paper.

Throughout the paper it will be convenient to view each grounded predicate $r(\overrightarrow{a})$, for a fixed vocabulary/domain, as a random variable over interpretations (note: an overline arrow denotes a tuple). That is, given a domain $\mathcal{D}$, we understand $r(\overrightarrow{a})$ as a function over all possible interpretations of the vocabulary, so that $r(\overrightarrow{a})(\mathbb{I})$ yields 1 if $r(\overrightarrow{a})$ is true in interpretation $\mathbb{I}$, and 0 otherwise.

For instance, say we have two unary predicates r and s, and we are given a domain $\mathcal{D} = \{a, b\}$. Then we have groundings $\{r(a), r(b), s(a), s(b)\}$, and there are 2^4 possible interpretations. Each interpretation assigns true or false to $r(a)$, and similarly to every grounding. So $r(a)$ can be viewed as a random variable over the possible interpretations.

2.2 Relational Bayesian network specifications

A *relational Bayesian network specification*, abbreviated relBN, is a directed acyclic graph where each node is a predicate (from a finite relational vocabulary), and where

1. each *root* node r is associated with a probabilistic assessment
$$\mathbb{P}\Big(r(\overrightarrow{\chi}) = 1\Big) = \alpha, \qquad (1)$$
2. while each *non-root* node s is associated with a formula (called the *definition* of s)
$$s(\overrightarrow{\chi}) \Leftrightarrow \phi(\overrightarrow{\chi}), \qquad (2)$$
where $\phi(\overrightarrow{\chi})$ is a formula in FFFO with free variables $\overrightarrow{\chi}$.

Given a domain, a relBN can be grounded into a unique Bayesian network:

1. by producing every grounding of the predicates;
2. by associating with each grounding $r(\overrightarrow{a})$ of a root predicate the grounded assessment $\mathbb{P}\Big(r(\overrightarrow{a}) = 1\Big) = \alpha$;
3. by associating with each grounding $s(\overrightarrow{a})$ of a non-root predicate the grounded definition $s(\overrightarrow{a}) \Leftrightarrow \phi(\overrightarrow{a})$, and by replacing univeral/existential quantification by conjunction/disjunction over the domain;
4. finally, by drawing a graph where each node is a grounded predicate and where there is an edge into each grounded non-root predicate $s(\overrightarrow{a})$ from each grounding of a predicate that appears in the grounded definition of $s(\overrightarrow{a})$.

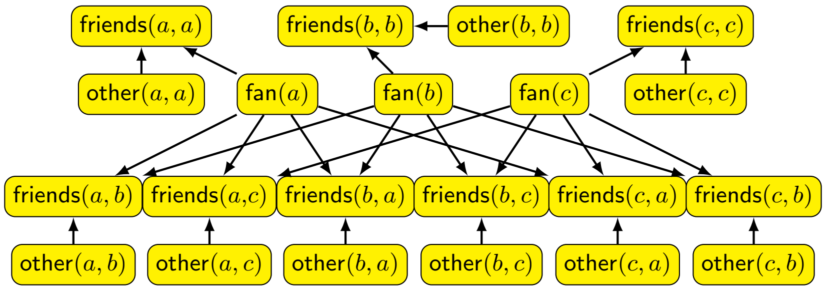

Fig. 1. The Bayesian network generated by Expression (3) and domain $\mathcal{D} = \{a, b, c\}$.

Consider, as an example, the following model of asymmetric friendship, where an individual is always a friend of herself, and where two individuals are friends if they are both fans (of a writer, say) or if there is some "other" reason for it:

$$\begin{gathered}\mathbb{P}(\mathsf{fan}(\chi)) = 0.2, \qquad \mathbb{P}\big(\mathsf{other}(\chi, y)\big) = 0.1, \\ \mathbb{P}\big(\mathsf{friends}(\chi, y)\big) \Leftrightarrow (\chi = y) \vee (\mathsf{fan}(\chi) \wedge \mathsf{fan}(y)) \vee \mathsf{other}(\chi, y),\end{gathered} \tag{3}$$

Suppose we have domain $\mathcal{D} = \{a, b, c\}$. Figure 1 depicts the Bayesian network generated by $\mathcal{D}$ and Expression (3).

For a given RELBN τ and a domain $\mathcal{D}$, denote by $\mathbb{B}(\tau, \mathcal{D})$ the Bayesian network obtained by grouning τ with respect to $\mathcal{D}$. The set of all relational Bayesian network specifications is denoted by $\mathcal{B}(\mathsf{FFFO})$.

3 A bit of descriptive complexity

We employ several concepts from finite model theory [4, 8, 12] and complexity theory [13]. We consider input strings in the alphabet $\{0, 1\}$; that is, a *string* is a sequence of 0s and 1s. A *language* is a set of strings; a *complexity class* is a set of languages. A language is *decided* by a Turing machine if the machine accepts each string in the language, and rejects each string not in the language. The complexity class NP contains each language that can be decided by a nondeterministic Turing machine with a polynomial time bound. If a Turing machine is such that, whenever its transition function maps to a non-singleton set, the transition is selected with uniform probability within that set, then the Turing machine is a *probabilistic Turing machine*. The complexity class PP is the set of languages that are decided by a probabilistic Turing machine in polynomial time, with an error probability strictly less than 1/2 for all input strings. This complexity class can be equivalently defined as follows: a language is in PP if and only if there is a polynomial nondeterministic Turing machine such that a string is in the language if and only if more than half of the computation paths of the machine end in the accepting state when the string is the input.

If a formula $\phi(\vec{\chi})$ has free logical variables $\vec{\chi}$, then structure $\mathfrak{A}$ is a *model* of $\phi(\vec{a})$ iff $\phi(\vec{\chi})$ is true in structure $\mathfrak{A}$ when the logical variables $\vec{\chi}$ are replaced by elements $\vec{a}$ of the domain.

A formula ϕ in existential function-free second-order logic (denoted by ESO) is a formula of the form $\exists r_1 \ldots \exists r_m \phi'$, where ϕ' is a sentence of FFFO containing predicates $r_1, \ldots, r_m$. Such a sentence allows existential quantification over the predicates themselves. Note that again we have equality in the language (that is, the built-in predicate $=$ is always available). Here a structure $\mathfrak{A}$ is a pair domain/interpretation, but the interpretation does not touch predicates that are existentially quantified (that is, if ϕ contains predicates $r_1, \ldots, r_m$ and $s_1, \ldots, s_M$, but $r_1, \ldots, r_m$ are all existentially quantified, then a model for ϕ contains an intepretation for $s_1, \ldots, s_M$).

As an example, consider the following formula of ESO [8]:

$$\exists \mathsf{partition} : \forall \mathcal{X} : \forall \mathcal{Y} : \left(\mathsf{edge}(\mathcal{X}, \mathcal{Y}) \Rightarrow (\mathsf{partition}(\mathcal{X}) \Leftrightarrow \neg \mathsf{partition}(\mathcal{Y})) \right). \quad (4)$$

Here a domain can be viewed as a set of nodes, and an interpretation can be viewed as a set of edges; the formula is satisfied if and only if it is possible to partition the vertices into two subsets such that if a node is in one subset, it is not in the other (that is, the graph is bipartite).

There is an *isomorphism* between structures $\mathfrak{A}_1$ and $\mathfrak{A}_2$ when there is a bijective mapping g between the domains such that if $r(a_1, \ldots, a_k)$ is true in $\mathfrak{A}_1$, then $r(g(a_1), \ldots, g(a_k))$ is true in $\mathfrak{A}_2$, and moreover if $r(a_1, \ldots, a_k)$ is true in $\mathfrak{A}_2$, then $r(g^{-1}(a_1), \ldots, g^{-1}(a_k))$ is true in $\mathfrak{A}_1$ (where g^{-1} denotes the inverse of g). A set of structures is *isomorphism-closed* if whenever a structure is in the set, all structures that are isomorphic to it are also in the set.

We assume that every structure is given as a string, encoded as follows for a fixed vocabulary (encoding from Ref. [12, Section 6.1]). First, if the domain contains elements $a_1, \ldots, a_n$, then the string begins with n symbols 0 followed by 1. The vocabulary is fixed, so we take some order for the predicates, $r_1, \ldots, r_m$. We then append, in this order, the encoding of the interpretation of each predicate. Focus on predicate r_i of arity k. To encode it with respect to a domain, we need to order the elements of the domain, say $a_1 < a_2 < \cdots < a_n$. This total ordering is assumed for now to be always available; it will be important later to check that the ordering itself can be defined. In any case, with a total ordering we can enumerate lexicographically all k-tuples over the domain. Now suppose $\overrightarrow{a}_j$ is the jth tuple in this enumeration; then the $j$the bit of the encoding of r_i is 1 if $r(\overrightarrow{a}_j)$ is true in the given interpretation, and 0 otherwise. Thus the encoding is a string containing $n + 1 + \sum_{i=1}^{m} n^{\mathrm{arity}(r_i)}$ symbols (either 0 or 1).

We can now state the celebrated theorem by Fagin on descriptive complexity:

Theorem 1 *Let $\mathcal{S}$ be an isomorphism-closed set of finite structures of some non-empty finite vocabulary. Then $\mathcal{S}$ is in* NP *if and only if $\mathcal{S}$ is the class of finite models of a sentence in* ESO*.*

The significance of Fagin's theorem is that it offers a definition of NP that is not tied to any computational model; rather, it is tied to the expressivity of the language that is used to specify problems. The surprising part of Fagin's theorem is that every language in NP can be exactly encoded by an ESO sentence.

4 $\mathcal{B}$(FFFO) captures PP

Given a RELBN and a domain, an *evidence piece* $\mathbf{E}$ is a partial interpretation; that is, an evidence piece assigns a truth value to some groundings of predicates.

We encode a pair domain-evidence $(\mathcal{D}, \mathbf{E})$ with the same strategy used in the previous section to encode a structure; however, we must take into account the fact that a particular grounding of a predicate can be either assigned true or false or be left without assignment. So we use a pair of symbols in $\{0, 1\}$ to encode each grounding; we assume that 00 means false and 11 means true, while say 01 means lack of assignment.

Say there is an *isomorphism* between pairs $(\mathcal{D}_1, \mathbf{E}_1)$ and $(\mathcal{D}_2, \mathbf{E}_2)$ when there is a bijective mapping g between the domains such that if $r(a_1, \dots, a_k)$ is true in $\mathbf{E}_1$, then $r(g(a_1), \dots, g(a_k))$ is true in $\mathbf{E}_2$, and moreover if $r(a_1, \dots, a_k)$ is true in $\mathbf{E}_2$, then $r(g^{-1}(a_1), \dots, g^{-1}(a_k))$ is true in $\mathbf{E}_1$ (where again g^{-1} denotes the inverse of g). A set of pairs domain-evidence is *isomorphism-closed* if whenever a pair is in the set, all pairs that are isomorphic to it are also in the set.

Suppose a set of pairs domain-evidence is given with respect to a fixed vocabulary σ. Once encoded, these pairs form a language $\mathcal{L}$ that can for instance belong to NP or to PP. One can imagine building a relational Bayesian network specification τ on an extended vocabulary consisting of σ plus some additional predicates, so as to decide this language $\mathcal{L}$ of domain-evidence pairs. For a given input pair $(\mathcal{D}, \mathbf{E})$, the Bayesian network specification and the domain lead to a Bayesian network $\mathbb{B}(\tau, \mathcal{D})$; this network can be used to compute the probability of some groundings, and that probabiility in turn can be used to accept/reject the input. This is the sort of strategy we pursue.

The point is that we must determine some prescription by which, given a Bayesian network and an evidence piece, one can generate an actual decision so as to accept/reject the input pair domain/evidence. We adopt the following strategy. Assume that in the extended vocabulary of τ there are two sets of distinguished auxiliary predicates $A_1, \dots, A_{m'}$ and $B_1, \dots, B_{m''}$ that are not in σ. We can use the Bayesian network $\mathbb{B}(\tau, \mathcal{D})$ to compute the probability $\mathbb{P}(\mathbf{A}|\mathbf{B}, \mathbf{E})$ where $\mathbf{A}$ and $\mathbf{B}$ are interpretations of $A_1, \dots, A_{m'}$ and $B_1, \dots, B_{m''}$ respectively. And then we might accept/reject the input on the basis of $\mathbb{P}(\mathbf{A}|\mathbf{B}, \mathbf{E})$. However, we cannot specify particular intepretations $\mathbf{A}$ and $\mathbf{B}$ as the related predicates are not in the vocabulary σ. Thus the sensible strategy is to fix attention to some selected, fixed, pair of intepretations for these predicates; we simply take the interpretations that assign true to every grounding.

In short: use the Bayesian network $\mathbb{B}(\tau, \mathcal{D})$ to determine whether or not $\mathbb{P}(\mathbf{A}|\mathbf{B}, \mathbf{E}) > 1/2$, where $\mathbf{A}$ assigns true to every grounding of $A_1, \dots, A_{m'}$, and $\mathbf{B}$ assigns true to every grounding of $B_1, \dots, B_{m''}$. If this inequality is satisfied, the input pair is *accepted*; if not, the input pair is *rejected*.

We refer to $A_1, \dots, A_{m'}$ as the *conditioned predicates*, and to $B_1, \dots, B_{m''}$ as the *conditioning predicates*.

Here is the main result:

Theorem 2 *Let $\mathcal{S}$ be an isomorphism-closed set of pairs domain-evidence of some non-empty finite vocabulary, where all domains are finite. Then $\mathcal{S}$ is in* PP *if and only if $\mathcal{S}$ is the class of domain-evidence pairs that are accepted by a fixed* relBN *with fixed conditioned and conditioning predicates.*

Proof. First, if $\mathcal{S}$ is a class of domain-query pairs that are accepted by a fixed relBN, they can be decided by a polynomial time probability Turing machine. To see that, note that we can build a nondeterministic Turing machine that guesses the truth value of all groundings that do not appear in the query (that is, not in $\mathbf{A} \cup \mathbf{B} \cup \mathbf{E}$), and then verifies whether the resulting complete interpretation is a model of the relBN. Recall that model checking of a fixed first-order sentence is in P [12].

To prove the other direction, we adapt the proof of Fagin's theorem as described by Grädel [8], along the same lines as the proof of Theorem 1 by Saluja et al. [17]. So, suppose that $\mathcal{L}$ is a language decided by some probabilistic Turing machine. Equivalently, there is a nondeterministic Turing machine that determines whether the majority of its computation paths accept an input, and accepts/rejects the input accordingly. By the mentioned proof of Fagin's theorem, there is a first-order sentence ϕ' with vocabulary consisting of the vocabulary of the input plus additional auxiliary predicates, such that each interpretation of this joint vocabulary is a model of the sentence if it is encodes a computation path of the Turing machine, as long as there is an available additional predicate that is guaranteed to be a linear order on the domain. Due to the lack of space, details of the construction are omitted; suffice to say that the same predicates X_q (one per state), Y_σ (one per symbol), and Z, employed by Grädel, are to be used here, with the same associated definitions. Denote by A the zero arity predicate with associated definition $A \Leftrightarrow \phi' \wedge \phi_E$, where ϕ_E is satisfied when an accepting state is reached. Suppose a linear order is indeed available; then by creating a relBN where all groundings are associated with probability 1/2, and where a non-root node is associated with the sentence in the proof of Fagin's theorem, we have that the probability of the query is larger than 1/2 iff the majority of computation paths accept. The challenge is to encode a linear order. To do so, introduce a new predicate $<$ and the first-order sentence ϕ'' that forces $<$ to be a total order, and a zero arity predicate B that is associated with definition $B \Leftrightarrow \phi' \wedge \phi''$. Now an input domain-pair $(\mathcal{D}, \mathbf{E})$ is accepted by the majority of computation paths in the Turing machine if and only if we have $\mathbb{P}(A|B, \mathbf{E}) > 1/2$. Note that there are actually $n!$ linear orders that satisfy B, but for each one of these linear orders we have the same assignments for all other predicates, hence the ratio between accepting computations and all computations is as desired. □

We might picture this as follows. There is always a Turing machine $\mathbb{TM}$ and a corresponding triplet (τ, A, B) such that for any pair $(\mathcal{D}, \mathbf{E})$, we have

$$(\mathcal{D}, \mathbf{E}) \text{ as input to } \mathbb{TM} \text{ with output given by } \mathbb{P}(\mathbb{TM} \text{ accepts } (\mathcal{D}, \mathbf{E})) > 1/2,$$

if and only if

$$(\mathcal{D}, \mathbf{E}) \text{ as ``input'' to } (\tau, A, B) \text{ with ``output'' given by } \mathbb{P}_{\tau,\mathcal{D}}(A|B, \mathbf{E}) > 1/2,$$

where $\mathbb{P}_{\tau,\mathcal{D}}(A|B,\mathbf{E})$ denotes probability with respect to $\mathbb{B}(\tau,\mathcal{D})$. (Of course, there is no even need to be restricted to zero-arity predicates A and B, as Theorem 2 allows for sets of predicates.)

Note that the same result could be proved if every evidence piece was taken to be a complete interpretation for the vocabulary σ. In that case we could directly speak of structures as inputs, and then the result would more closely mirror Fagin's theorem. However it is very appropriate, and entirely in line with practical use, to take the inputs to a Bayesian network as the groundings of a partially observed interpretation. Hence we have preferred to present our main result as stated in Theorem 2.

5 Moving to second-order

Suppose we have a phenomenon whose simulation requires computational powers that go beyond a polynomial time probabilistic Turing machine. There are in fact description languages whose inferences require exponential time probabilistic Turing machines [2, 3], but we need not go to such extremes. We might for instance have a phenomenon that requires a polynomial time probabilistic Turing machine to use as oracle a polynomial time nondeterministic Turing machine. That is, we might have a phenomenon with requirements within $\mathsf{PP}^{\mathsf{NP}}$, a level above PP in the polynomial counting hierarchy [19]. Given current beliefs about complexity classes, Theorem 2 shows that such a phenomenon cannot be modeled with a relBN. Can we find a "reasonable" specification language that allows one to model such a phenomenon?

Indeed we can, by putting together Fagin's theorem and Theorem 2. Consider a specification language as follows: we have a directed acyclic graph where nodes are predicates, and where as before root and non-root nodes are respectively associated with assessments $\mathbb{P}(r(\overrightarrow{\chi}) = 1) = \alpha$ and definitions $s(\overrightarrow{\chi}) \Leftrightarrow \phi(\overrightarrow{\chi})$, but the latter are now enlarged so that ϕ is a formula in ESO. A directed graph associated with such assessments and formulas is referred to as a *existential second-order Bayesian network specification*, abbreviated esoBN.

For instance, consider the model of friendship presented in Expression (3), and suppose we add a variable that indicates whether the graph of friends can be partitioned, using Expression (4) as follows:

$$\mathsf{partitioned} \Leftrightarrow \exists \mathsf{partition} : \forall \chi : \forall y : \left(\mathsf{edge}(\chi, y) \Rightarrow (\mathsf{partition}(\chi) \Leftrightarrow \neg\mathsf{partition}(y))\right).$$

The presence of probabilities and second-order quantification gives us the desired modeling power:

Theorem 3 *Let $\mathcal{S}$ be an isomorphism-closed set of pairs domain-evidence of some non-empty finite vocabulary, where all domains are finite. Then $\mathcal{S}$ is in $\mathsf{PP}^{\mathsf{NP}}$ if and only if $\mathcal{S}$ is the class of domain-evidence pairs that are accepted by a fixed* esoBN *with fixed conditioned and conditioning predicates.*

Proof. To prove that a class of domain-query pairs that are accepted by a fixed esoBN can be decided within $\mathsf{PP}^{\mathsf{NP}}$, put together the argument in the first paragraph of the proof of Theorem 2 and the fact that an ESO sentence can be evaluated within NP by Theorem 1.

To prove the other direction, the corresponding part of the proof of Theorem 2 must be enlarged. We introduce again the same predicates X_q, Y_σ and Z for the base machine, together with the whole machinery in Grädel's proof of Fagin's theorem [8], and we also introduce predicates X^o_q, Y^o_σ, Z^o for the oracle machine (the superscript o refers to the "oracle"). Additionally, in the proof of Theorem 2 it is necessary to introduce logical variables that "mark the steps" of the Turing machine (these steps are ordered by the introduced linear order). Here we need two sets of such logical variables and associated machinery; one set "marks the steps" of the base machine, while the other "marks the steps" of the oracle machine. And of course a linear order must also be built as in the proof of Theorem 2. Again, due to lack of space the details of the construction are omitted. And again, an input domain-query is accepted by the majority of computation paths in the (base) Turing machine if and only if we have $\mathbb{P}(A|B, \mathbf{E}) > 1/2$, where A, B and $\mathbf{E}$ are as in the proof of Theorem 2. $\square$

6 Conclusion: A finite model theory of Bayesian networks?

We have introduced a theory of descriptive complexity for Bayesian networks, a topic that does not seem to have received due attention so far. Our results can be extended in a variety of directions, for instance to various fixpoint logics that are the basis of logic programming [4, 12]. To summarize, we have shown that relational Bayesian network specifications capture PP, and we have indicated how we can go beyond PP in our modeling tools. Specifically, we added existential second-order quantification to capture the complexity class $\mathsf{PP}^{\mathsf{NP}}$.

These results can be better appreciated by taking a broader perspective. For several decades now, there has been significant study of models that arise from combinations of probability and logic [1, 6]. However, by dealing with domains of arbitrary cardinality, and with logics that include too many constructs (for instance, functions) and that exclude valuable techniques (for instance, the modularity introduced by independence relations), these previous investigations arrive at results that are often too weak — for instance, almost always obtaining undecidability or very high computational complexity. By focusing on modular tools such as Bayesian networks, and by focusing on finite domains, we are able to obtain much sharper results, nailing down specific complexity classes such as PP and $\mathsf{PP}^{\mathsf{NP}}$. In fact, the purpose here is to initiate a "finite model theory of Bayesian network specifications" that can pin down the expressivity and complexity of Bayesian networks, not only when they are propositional objects, but particularly when they are specified using logical constructs.

Our results are also interesting from a point of view centered on complexity theory. There has been little work on capturing counting/probabilistic classes; the most significant previous results capture #P using counting [17]. We offer a more concrete modeling language that captures PP, and we move into the counting hierarchy, up to $\mathsf{PP}^{\mathsf{NP}}$ — we are not aware of any similar result in the literature. Our results show that classes in the counting hierarchy can be tied to the expressivity of modeling tools, not to any particular computational model (much as Fagin's theorem does for NP).

Acknowledgements

The first author is partially supported by CNPq, grant 308433/2014-9. This paper was partially funded by FAPESP grant #2015/21880-4 (project Proverbs).

References

1. M. Abadi and J. Y. Halpern. Decidability and expressiveness for first-order logics of probability. *Information and Computation*, 112(1):1–36, 1994.
2. F. G. Cozman and D. D. Mauá. Bayesian networks specified using propositional and relational constructs: Combined, data, and domain complexity. In *AAAI Conference on Artificial Intelligence*, 2015.
3. F. G. Cozman and R. B. Polastro. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In *Conf. on Uncertainty in Artificial Intelligence*, pages 117–125, 2009.
4. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, 1995.
5. H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.
6. H. Gaifman. Concerning measures on first-order calculi. *Israel Journal of Mathematics*, 2:1–18, 1964.
7. L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning.* MIT Press, 2007.
8. E. Grädel. Finite model theory and descriptive complexity. In *Finite Model Theory and its Applications*, pages 125–229. Springer, 2007.
9. E. E. Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, S. Weinstein. *Finite Model Theory and its Applications.* Springer, 2007.
10. M. Jaeger. Lower complexity bounds for lifted inference. *Theory and Practice of Logic Programming*, 15(2):246–264, 2014.
11. D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.
12. L. Libkin. *Elements of Finite Model Theory.* Springer, 2004.
13. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
14. D. Poole. Probabilistic programming languages: Independent choices and deterministic systems. In R. Dechter, H. Geffner, and J. Y. Halpern, editors, *Heuristics, Probability and Causality — A Tribute to Judea Pearl*, pages 253–269. College Publications, 2010.
15. L. De Raedt. *Logical and Relational Learning.* Springer, 2008.
16. L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton. *Probabilistic Inductive Logic Programming.* Springer, 2010.
17. S. Saluja and K. V. Subrahmanyam. Descriptive complexity of #P functions. *Journal of Computer and Systems Sciences*, 50:493–505, 1995.
18. G. Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Neural Processing Information Systems*, pages 1386–1394, 2011.
19. K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.