The Finite Model Theory of Bayesian Networks: Descriptive Complexity

Fabio Gagliardi Cozman[1], **Denis Deratani Mauá**[2]
[1] Escola Politécnica, Universidade de São Paulo
[2] Instituto de Matemática e Estatística, Universidade de São Paulo

Abstract

We adapt the theory of descriptive complexity to encompass Bayesian networks, so as to quantify the expressivity of Bayesian network specifications based on predicates and quantifiers. We show that Bayesian network specifications that employ first-order quantification *capture* the complexity class PP; by allowing quantification over predicates, the resulting Bayesian network specifications *capture* each class in the hierarchy $\mathsf{PP}^{\mathsf{NP}^{\cdots\mathsf{NP}}}$, a result that does not seem to have equivalent in the literature.[1]

1 Introduction

There is a rich literature on "relational" Bayesian networks, where constructs from first-order logic are used to represent populations with repetitive patterns [Getoor and Taskar, 2007; De Raedt, 2008; De Raedt *et al.*, 2010]. A relational specification is often viewed as a template that can be grounded into a finite propositional Bayesian network.

It is only natural to ask what is the *expressivity* of Bayesian network specifications based on predicates and quantifiers. That is, to ask 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* as developed within finite model theory [Ebbinghaus and Flum, 1995; Grädel *et al.*, 2007]. It does not seem that the descriptive complexity of Bayesian networks has been investigated in previous work; as the topic is novel, most of this paper consists of building a framework in which to operate.

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. The power of relational probabilistic specifications has been already noted in connection with *lifted* inference algorithms [Jaeger, 2014; Van den Broeck, 2011], but has not been characterized in terms of descriptive complexity.

[1]This paper is an abridged and corrected version of a paper selected from ECSQARU 2017 [Cozman and Maua, 2017]; some novel results appear here. The original paper was entitled "The Descriptive Complexity of Bayesian Network Specifications".

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. 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 complexity classes in the hierarchy $\mathsf{PP}^{\mathsf{NP}^{\cdots\mathsf{NP}}}$. It does not seem that previous results on descriptive theory have reached this latter complexity classes.

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 output following the same distribution produced by the phenomenon. Our results show that the phenomenon can also 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 probabilities over the output 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 comment on the significance of these results in the concluding Section 6.

2 Specifying Bayesian Networks with Logical Constructs

To make any progress, we must precisely define what are the "relational" Bayesian networks we contemplate. Our strategy, described in this section, is to adopt a proposal by Poole [2010] to mix probabilistic assessments and logical equivalences [Cozman and Mauá, 2015].

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 $\mathbb{G}$, 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 joint probabilities [Koller and Friedman, 2009].

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 use 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 quantification ($\exists$) and universal quantification ($\forall$).

First-order theories are interpreted as usual [Enderton, 1972], using *domains*, that are just sets, and *interpretations* that associate predicates with relations (an interpretation can be viewed as 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 $r(d_1, \dots, d_k)$, for a fixed vocabulary/domain, as a random variable over interpretations. That is, given a domain $\mathcal{D}$, we understand $r(d_1, \dots, d_k)$ as a function over all possible interpretations of the vocabulary, so that $r(d_1, \dots, d_k)(\mathbb{I})$ yields 1 if $r(d_1, \dots, d_k)$ 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 each 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 of arity k is associated with a probabilistic assessment (α is a rational in $[0, 1]$):
$$\mathbb{P}(r(\mathcal{X}_1, \dots, \mathcal{X}_k) = 1) = \alpha, \qquad (1)$$
2. while each *non-root* node s of arity k is associated with a formula (called the *definition* of s)
$$s(\mathcal{X}_1, \dots, \mathcal{X}_k) \Leftrightarrow \phi(\mathcal{X}_1, \dots, \mathcal{X}_k), \qquad (2)$$
where $\phi(\mathcal{X}_1, \dots, \mathcal{X}_k)$ is a formula in FFFO with free variables $\mathcal{X}_1, \dots, \mathcal{X}_k$.

Given a domain, a RELBN can be grounded into a unique Bayesian network, as follows.
• First, produce every grounding of the predicates.
• Second, associate with each grounding $r(d_1, \dots, d_k)$ of a root predicate the grounded assessment
$$\mathbb{P}(r(d_1, \dots, d_k) = 1) = \alpha.$$
• Third, associate with each grounding $s(d_1, \dots, d_k)$ of a non-root predicate the grounded definition
$$s(d_1, \dots, d_k) \Leftrightarrow \phi(d_1, \dots, d_k),$$
and replace universal/existential quantification in ϕ by conjunction/disjunction over the domain. Thus $\phi(d_1, \dots, d_k)$ becomes a (possibly very large!) propositional expression containing groundings.
• Finally draw a graph where each grounded predicate is a node, and where edges are obtained as follows. Add an edge into each grounded non-root predicate $s(d_1, \dots, d_k)$ from each grounding of a predicate that appears in the grounded definition of $s(d_1, \dots, d_k)$ given by $\phi(d_1, \dots, d_k)$.

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:

$$\mathbb{P}(\mathsf{fan}(\mathcal{X}) = 1) = 0.2,$$
$$\mathbb{P}\big(\mathsf{other}(\mathcal{X}, \mathcal{Y}) = 1\big) = 0.1,$$
$$\mathsf{friends}(\mathcal{X}, \mathcal{Y}) \Leftrightarrow (\mathcal{X} = \mathcal{Y}) \vee (\mathsf{fan}(\mathcal{X}) \wedge \mathsf{fan}(\mathcal{Y})) \vee \mathsf{other}(\mathcal{X}, \mathcal{Y}).$$

Figure 1 depicts the Bayesian network induced by domain $\mathcal{D} = \{a, b, c\}$.

For a given RELBN τ and a domain $\mathcal{D}$, denote by $\tau(\mathcal{D})$ the Bayesian network obtained by grounding 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 review some relevant concepts from finite model theory and computational complexity [Grädel, 2007; Papadimitriou, 1994].

Complexity Classes 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 iff there is a polynomial nondeterministic Turing machine such

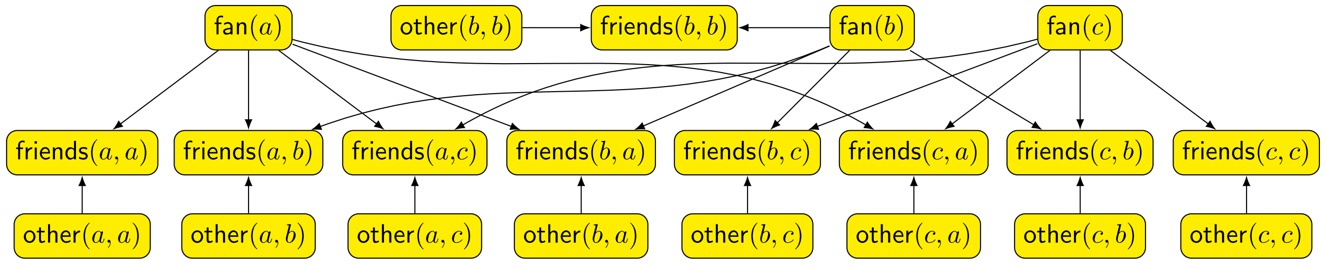

Figure 1: The Bayesian network generated by the specification of asymmetric friendship, with domain $\mathcal{D} = \{a, b, c\}$.

that a string is in the language iff more than half of the computation paths of the machine end in the accepting state when the string is the input.

An oracle Turing machine $M^{\mathcal{L}}$, where $\mathcal{L}$ is a language, is a Turing machine with additional tapes, such that it can write a string ℓ to a tape and obtain from the oracle, in unit time, the decision as to whether $\ell \in \mathcal{L}$ or not. If a set of languages A is defined by a set of Turing machines $\mathcal{M}$ (that is, the languages are decided by these machines), then define $\mathsf{A}^{\mathcal{L}}$ to be the set of languages that are decided by $\{M^{\mathcal{L}} : M \in \mathcal{M}\}$. If A and B are sets of languages, $\mathsf{A}^{\mathsf{B}} = \cup_{x \in \mathsf{B}} \mathsf{A}^x$. The *polynomial hierarchy* contains classes Σ_k^p, where Σ_0^p is P and $\Sigma_k^p = \mathsf{NP}^{\Sigma_{k-1}^p}$ for $k > 0$ (note that $\Sigma_1^p = \mathsf{NP}$).

Capturing a Complexity Class If a formula $\phi(\mathcal{x}_1, \dots, \mathcal{x}_k)$ has free logical variables $\mathcal{x}_1, \dots, \mathcal{x}_k$, then structure $\mathfrak{A}$ is a *model* of $\phi(d_1, \dots, d_k)$ iff $\phi(\mathcal{x}_1, \dots, \mathcal{x}_k)$ is true in structure $\mathfrak{A}$ when the logical variables $\mathcal{x}_1, \dots, \mathcal{x}_k$ are replaced by elements $d_1, \dots, d_k$ of the domain.

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 $r(d_1, \dots, d_k)$ is true in $\mathfrak{A}_1$, then $r(g(d_1), \dots, g(d_k))$ is true in $\mathfrak{A}_2$, and moreover if $r(d_1, \dots, d_k)$ is true in $\mathfrak{A}_2$, then $r(g^{-1}(d_1), \dots, g^{-1}(d_k))$ is true in $\mathfrak{A}_1$ (where g^{-1} denotes the inverse of g). A *isomorphism-closed* set of structures is a set of structures such that if a structure is in the set, all structures that are isomorphic to it are also in the set.

We assume that every structure is written as a string, encoded as follows for a fixed vocabulary where the maximum predicate arity is K [Grädel, 2007, Section 3.1.5]. Suppose the domain contains elements $d_1, \dots, d_N$. To encode interpretations with respect to the domain, we start by ordering the elements of the domain, say $d_1 < d_2 < \dots < d_N$. This linear ordering is assumed for now to be always available; for the logical languages we consider later, the ordering itself can be defined with a formula. The ordering starts by encoding the domain. To do so, the string begins with N symbols 1 followed by one symbol 0. We take some order for the predicates, $r_1, \dots, r_n$. We then append, in this order, the encoding of the interpretation of each predicate. Focus on predicate r_i of arity k. Using the linear ordering on the domain, we can enumerate lexicographically all k-tuples over the domain (all N^k tuples). We then encode the interpretation of r_i by N^k symbols followed by $N^K - N^k$ symbols 0; the jth first symbol (up to the N^kth symbol) is 1 if the jth k-tuple belongs to the interpretation, and 0 otherwise. Thus we have n blocks of N^K symbols and an initial block of $N + 1$ symbols, for a total of $(N + 1) + nN^K$ symbols in the string.

The *data complexity* of logical language $\mathcal{L}$ is the complexity of deciding whether an input structure satisfies a fixed sentence in $\mathcal{L}$.

Logical language $\mathcal{L}$ *captures* complexity class $\mathcal{C}$ if and only if: first, for every sentence ϕ of $\mathcal{L}$, the data complexity of $\mathcal{L}$ is in $\mathcal{C}$ and, second, for each isomorphism-closed set of finite structures $\mathcal{S}$, for a non-empty vocabulary $\mathcal{V}$, such that these structures are strings of a language in $\mathcal{C}$, there is a sentence ϕ in $\mathcal{L}$, with vocabulary $\mathcal{V}$, such that $\mathcal{S}$ is exactly the set of all models of ϕ.

Note that descriptive complexity is distinct from data complexity. For example, the data complexity of FFFO is within polynomial bounds, but it is not the case that FFFO captures polynomial complexity: 2-colorability of a given graph (a graph can be encoded as a structure) can be checked in polynomial time, but 2-colorability cannot be expressed in FFFO.

Existential Second-Order Logic A formula ϕ in existential function-free second-order logic is a formula of the form

$$\exists r_1 \dots \exists r_m \phi',$$

where ϕ' is a sentence of FFFO containing predicates $r_1, \dots, r_m$. The set of such formulas is denoted ESO. Note that again we have equality ($=$) in the language. 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 $r_1, \dots, r_m$ and $s_1, \dots, s_M$, but $r_1, \dots, r_m$ are all existentially quantified, then a model for ϕ contains an intepretation for $s_1, \dots, s_M$).

As an example, consider a formula in ESO that is only satisfied if it is possible to find a domain, to be interpreted as a set of vertices, such that it is possible to partition the set of vertices into two subsets: if two vertices are connected by an edge, they are in distinct subsets. Here is the formula [Grädel, 2007]:

$$\exists \mathsf{pt} : \forall \mathcal{x} : \forall \mathcal{y} : \left(\mathsf{edge}(\mathcal{x}, \mathcal{y}) \Rightarrow (\mathsf{pt}(\mathcal{x}) \Leftrightarrow \neg \mathsf{pt}(\mathcal{y}))\right).$$

Fagin's Theorem The most celebrated result in descriptive complexity is Fagin's theorem: ESO captures NP [Fagin, 1976]. Or, in more detail:

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

Fagin's theorem offers a definition of NP that is not tied to any computational model; rather, it is tied to the expressivity of ESO. Note that it is not surprising that NP contains the problem of deciding whether an input structure is a model of a fixed ESO sentence; 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

Suppose we have an isomorphism-closed set of finite structures $\mathcal{S}$, of some non-empty vocabulary $\mathcal{V}$. Once encoded, they form a language. Fagin's theorem focuses on languages that belong to NP, while here we wish to look at languages that belong to PP.

We show that any language of structures in PP can be "decided" by some RELBN τ on an extended vocabulary $\mathcal{V}'$ consisting of $\mathcal{V}$ plus some additional predicates. Clearly the question is whether it is possible to generate such a RELBN; however, a preliminary question is how exactly to accept/reject an input structure when we already have some RELBN.

Note that if we have a given structure $\mathfrak{A}$, we have a domain $\mathcal{D}$ and consequently we can produce a Bayesian network $\tau(\mathcal{D})$. We denote by $\mathbb{P}_{\tau(\mathcal{D})}$ the probability measure encoded by the Bayesian network $\tau(\mathcal{D})$. Note also that each grounding of a predicate in $\mathcal{V}$ appears in $\tau(\mathcal{D})$ as a random variable; this random variable is set to true or false by the interpretation in $\mathfrak{A}$. The *evidence* $\mathbf{E}$ induced by $\mathfrak{A}$ is just the set of these assignments to random variables corresponding to groundings of predicates in $\mathcal{V}$.

We adopt the following scheme to accept/reject an input structure $\mathfrak{A}$ using a given RELBN τ whose vocabulary contains $\mathcal{V}$:
• We assume that τ contains two distinguished predicates of zero arity, A and B. The predicate A is the *conditioned predicate*, while B is the *conditioning predicate*.
• We assume that if $\mathbb{P}_{\tau(\mathcal{D})}(A = 1|B = 1, \mathbf{E}) > 1/2$, then $\mathfrak{A}$ is accepted; otherwise, $\mathfrak{A}$ is rejected.

With the proper context in place, here is the main result:

Theorem 2. *Let $\mathcal{S}$ be an isomorphism-closed set of finite structures of some non-empty finite vocabulary. Then $\mathcal{S}$ is in complexity class* PP *if and only if $\mathcal{S}$ is the class of finite structures that are accepted by a fixed* RELBN *with fixed conditioned and conditioning predicates.*

This offers a definition of PP that is not tied to any computational model. In short, there is always a polynomial-time probabilistic Turing machine $\mathbb{TM}$ and a triplet (τ, A, B) such that both $\mathbb{TM}$ and (τ, A, B) can decide the same language $\mathcal{S}$ of structures.

5 Moving to Second-Order

Suppose we have a specification that follows the syntax of RELBN, except for the fact that one logical definition contains, in its right hand side, a formula in ESO. We denote this set of specifications by 1ESOBNs; that is, they are similar to relational Bayesian network specifications, except that one logical definition is not in FFFO but rather in ESO. We can actually extend this to 2ESOBNs and any other number of second-order definitions, until we reach *second-order Bayesian network specifications*, where arbitrary second-order formulas can be used.

What kind of complexity class is captured by specifications in 1ESOBNs? We adopt a strategy for acceptance/rejection by 1ESOBNs that mimicks our previous strategy for acceptance/rejection by RELBNs. That is, we have an input structure on a vocabulary $\mathcal{V}$, and we assume that there is a conditioned predicate A and a conditioning predicate B in the 1ESOBN; we also assume that quantified predicates are not in $\mathcal{V}$. The decision as to whether to accept or reject the input structure is based on whether the probability of A given B and $\mathbf{E}$ (the evidence conveyed by the input structure) is larger than $1/2$.

We have:[2]

Theorem 3. *Let $\mathcal{S}$ be an isomorphism-closed set of finite structures of some non-empty finite vocabulary. Then $\mathcal{S}$ is in complexity class* $\mathsf{PP}^{\mathsf{NP}}$ *if and only if $\mathcal{S}$ is the class of finite structures that are accepted by a fixed* 1ESOBN *with fixed conditioned and conditioning predicates.*

By allowing an increasing number of existential second-order logical definitions, we can produce increasingly more complex alternation of quantifiers (by negating an existential definition so as to obtain a universal one whenever necessary). That is, with a kESOBN, where we can use k logical definitions based on existential second-order logic, we can write down formulas in Σ_k^1; that is, formulas that consist of a FFFO formula preceded by k blocks of quantifiers, each block either existential or universal, and the first block existential. Now Σ_k^1 captures the complexity class Σ_k^p in the polynomial hierarchy [Stockmeyer, 1977]. Hence we can extend the previous theorem so as to allow k logical definitions based on ESO, with the result that the oracle machine decides Σ_k^p. Thus we can capture various classes in the counting polynomial hierarchy [Wagner, 1986]:

Theorem 4. *Let $\mathcal{S}$ be an isomorphism-closed set of finite structures of some non-empty finite vocabulary. Then $\mathcal{S}$ is in complexity class* $\mathsf{PP}^{\Sigma_k^p}$ *if and only if $\mathcal{S}$ is the class of finite structures that are accepted by a fixed* kESOBN *with fixed conditioned and conditioning predicates.*

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. 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 complexity classes $\mathsf{PP}^{\Sigma_k^p}$. Our results can be extended in a variety of directions, for instance to various fixpoint logics that are the basis of logic programming [Ebbinghaus and Flum, 1995; Libkin, 2004].

[2]In the original version of this paper we argued that second-order Bayesian network specifications capture $\mathsf{PP}^{\mathsf{NP}}$, failing to state the restriction to a single second-order formula [Cozman and Maua, 2017]. We correct that mistake here.

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 [Abadi and Halpern, 1994; Gaifman, 1964]. However, by dealing with domains of arbitrary cardinality, and with logics that both include too many constructs (for instance, functions) and exclude valuable techniques (for instance, 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 $\#\mathsf{P}$ using counting [Saluja and Subrahmanyam, 1995]. We offer a more concrete modeling language that captures PP, and we move into the counting hierarchy with classes $\mathsf{PP}^{\Sigma_k^p}$ — 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. The second author is partially supported by CNPq grants 303920/2016-5 and 420669/2016-7. This paper was partially funded by FAPESP grants 2016/18841-0, 2016/01055-1, and 2015/21880-4.

References

[Abadi and Halpern, 1994] Martín Abadi and Joseph Y. Halpern. Decidability and expressiveness for first-order logics of probability. *Information and Computation*, 112(1):1–36, 1994.

[Cozman and Mauá, 2015] Fabio Gagliardi Cozman and Denis Deratani Mauá. Bayesian networks specified using propositional and relational constructs: Combined, data, and domain complexity. In *AAAI Conference on Artificial Intelligence*, 2015.

[Cozman and Maua, 2017] Fabio Gagliardi Cozman and Denis Deratani Mauá. The descriptive complexity of Bayesian network specifications. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 93–103. Springer, 2017.

[De Raedt *et al.*, 2010] Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton. *Probabilistic Inductive Logic Programming*. Springer, 2010.

[De Raedt, 2008] Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

[Ebbinghaus and Flum, 1995] Heinz-Dieter Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1995.

[Enderton, 1972] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[Fagin, 1976] Ronald Fagin. Probabilities on finite models. *Journal of Symbolic Logic*, 41(1):50–58, 1976.

[Gaifman, 1964] Haim Gaifman. Concerning measures on first-order calculi. *Israel Journal of Mathematics*, 2:1–18, 1964.

[Getoor and Taskar, 2007] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[Grädel *et al.*, 2007] Erich E. Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and its Applications*. Springer, 2007.

[Grädel, 2007] Erich Grädel. Finite model theory and descriptive complexity. In *Finite Model Theory and its Applications*, pages 125–229. Springer, 2007.

[Jaeger, 2014] Manfred Jaeger. Lower complexity bounds for lifted inference. *Theory and Practice of Logic Programming*, 15(2):246–264, 2014.

[Koller and Friedman, 2009] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[Libkin, 2004] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing, 1994.

[Poole, 2010] David Poole. Probabilistic programming languages: Independent choices and deterministic systems. In Rina Dechter, Hector Geffner, and Joseph Y. Halpern, editors, *Heuristics, Probability and Causality — A Tribute to Judea Pearl*, pages 253–269. College Publications, 2010.

[Saluja and Subrahmanyam, 1995] Sanjeev Saluja and K. V. Subrahmanyam. Descriptive complexity of #P functions. *Journal of Computer and Systems Sciences*, 50:493–505, 1995.

[Stockmeyer, 1977] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1–22), 1977.

[Van den Broeck, 2011] Guy Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Neural Processing Information Systems*, pages 1386–1394, 2011.

[Wagner, 1986] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.