The Complexity of Inferences and Explanations in Probabilistic Logic Programming

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. A popular family of probabilistic logic programming languages combines logic programs with independent probabilistic facts. We study the complexity of marginal inference, most probable explanations, and maximum a posteriori calculations for propositional/relational probabilistic logic programs that are acyclic/definite/stratified/normal/disjunctive. We show that complexity classes Σ_k and PP^{Σ_k} (for various values of k) and $\mathsf{NP}^{\mathsf{PP}}$ are all reached by such computations.

1 Introduction

The goal of this paper is to shed light on the computational complexity of inference for probabilistic logic programs interpreted in the spirit of Sato's distribution semantics [25]; that is, we have logic programs where some facts are annotated with probabilities, so as to define probability distributions over models. This framework has been shown to be quite useful in modeling practical problems [15, 24].

The distribution defined by a probabilistic logic program can be used to answer many queries of interest. Two common queries are to compute the probability of some ground atom given evidence (inference), and to find a (partial) interpretation that maximizes probability while being consistent with evidence (MPE/MAP).

We present results on the complexity of *acyclic*, *definite*, *stratified*, *normal* and *disjunctive* probabilistic logic programs; these results are summarized by Table 1. While most semantics agree on stratified programs, there is less consensus on non-stratified programs. Here we examine two semantics: the *credal semantics*, based on stable models, and the *well-founded semantics*.

We start in Sections 2 and 3 by reviewing relevant background on probabilistic logic programs and on complexity theory. Our contributions appear in Section 4. These results are further discussed in the concluding Section 5.

2 Background

The results in this paper depend on an understanding of logic and answer set programming; the topic is dense and cannot be described in detail in the space

	Propositional			Bounded arity		
	Inferential	MPE	MAP	Inferential	MPE	MAP
Acyclic normal	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
Definite (positive query)	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
Stratified normal	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
Normal, credal	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\Sigma_2^P}$	Σ_3^P	$\mathsf{NP}^{\mathsf{PP}}$
Normal, well-founded	PP	NP	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\mathsf{NP}}$	Σ_2^P	$\mathsf{NP}^{\mathsf{PP}}$
Disjunctive, credal	$\mathsf{PP}^{\Sigma_2^P}$	Σ_3^P	$\mathsf{NP}^{\mathsf{PP}}$	$\mathsf{PP}^{\Sigma_3^P}$	Σ_4^P	$\mathsf{NP}^{\mathsf{PP}}$

Table 1. Summary of complexity results presented in this paper (all entries indicate completeness with respect to many-one reductions). Entries containing previously known results have orange background (grey background if printed in black-and-white).

we have here. We just mention the main concepts, and refer the reader to any in-depth presentation in the literature [9, 14].

We have a vocabulary consisting of *predicates*, *constants*, and *logical variables*; a *term* is a constant or logical variable, and an *atom* is a predicate of *arity* k associated with k terms; a *ground atom* is an atom without logical variables. We often resort to *grounding* to produce ground atoms. A *disjunctive logic program* (DLP) consists of a set of rules written as

$$A_1 \vee \ldots \vee A_h \coloneq B_1, \ldots, B_{b'}, \mathbf{not}\ B_{b'+1}, \ldots, \mathbf{not}\ B_b\ \mathbf{.}$$

where each A_i and B_i is an atom. The lefthand side is the *head* of the rule; the remainder is its *body*. A rule without disjunction (i.e., $h = 1$) and with empty body, written simply as $A_1\mathbf{.}$, is a *fact*. A program without disjunction is a *normal logic program*; a normal logic program without negation is a *definite program*; finally, a program without variables is a *propositional program*. The *dependency graph* of a program is the graph where each atom is a vertex and there are arcs from atoms in the bodies to atoms in the heads of a same rule; a normal logic program is *acyclic* when the dependency graph of its grounding is acyclic. A normal logic program is (locally) *stratified* when the dependency graph of its grounding has no cycles containing an arc involving a negated literal.

The *Herbrand base* of a program is the set of all ground atoms built from constants and predicates in the program. An *interpretation* is a set of ground literals mentioning exactly once each atom in the Herbrand base. A *model* is an interpretation that *satisfies* every grounding of a rule (a rule is satisfied iff the interpretation contains all of $B_1, \ldots, B_{b'}$, none of $B_{b'+1}, \ldots, B_b$, and some of $A_1, \ldots, A_h$). A *minimal model* minimizes the number of non-negated literals.

The most common semantics for DLPs is the *stable model* semantics. Given program and interpretation, define their *reduct* to be program obtained by removing every rule whose body is not satisfied. An interpretation is a *stable model* if it is a minimal model of its reduct. A normal program may have zero, one or several stable models. *Brave reasoning* asks whether there is a stable model containing a specific literal (possibly returning one). *Cautious reasoning* asks whether a specific literal appears in all stable models (possibly listing all).

An alternative semantics for normal logic programs is the *well-founded semantics*; a model under this semantics might fix only a truth-value for some of the atoms (levaing the remaining atoms $\mathsf{undefined}$) [31]. One way to define the well-founded semantics is as follows [4]. Write $\mathbb{LFT}_{\mathbf{P}}(\mathcal{I})$ to mean the least fixpoint of $\mathbb{T}_{\mathbf{P}^{\mathcal{I}}}$, where $\mathbb{T}_{\mathbf{P}}$ is a transformation such that: atom A is in $\mathbb{T}_{\mathbf{P}}(\mathcal{I})$ iff there is grounded rule with head A with the whole body true in interpretation $\mathcal{I}$. Then the well-founded semantics of $\mathbf{P}$ consists of those atoms A that are in the least fixpoint of $\mathbb{LFT}_{\mathbf{P}}(\mathbb{LFT}_{\mathbf{P}}(\cdot))$ plus the literals $\neg A$ for those atoms A that are *not* in the greatest fixpoint of $\mathbb{LFT}_{\mathbf{P}}(\mathbb{LFT}_{\mathbf{P}}(\cdot))$.

Here is an example: Several robots can perform, each, one of three operations, called "red", "green", "yellow". A robot placed in a site also covers adjacent sites, so there is no need to place same-color robots on adjacent sites. There is a list of robots and a list of (one-way) connections between sites; the goal is to distribute the robots and verify whether the sites are connected. This disguised 3-coloring problem can be encoded as [14]:

$$\mathsf{color}(X, \mathsf{red}) \vee \mathsf{color}(X, \mathsf{green}) \vee \mathsf{color}(X, \mathsf{yellow}) \coloneq \mathsf{site}(X).$$
$$\mathsf{clash} \coloneq \mathbf{not}\ \mathsf{clash}, \mathsf{edge}(X, Y), \mathsf{color}(X, C), \mathsf{color}(Y, C).$$
$$\mathsf{path}(X, Y) \coloneq \mathsf{edge}(X, Y). \qquad \mathsf{path}(X, Y) \coloneq \mathsf{edge}(X, Z), \mathsf{path}(Z, Y).$$

We might have a database of facts, consisting of a list of sites and their connections, say $\mathsf{site}(\mathsf{s}_1), \mathsf{site}(\mathsf{s}_2), \ldots, \mathsf{edge}(\mathsf{s}_1, \mathsf{s}_4)$, and so on. Each stable model of this program is a possible placement (a 3-coloring) and a list of paths between sites.

We also need standard concepts from complexity theory: languages (sets of strings), decision problems (deciding whether input is in language), complexity classes (sets of languages), many-one reductions [22]. We use well-known complexity classes such as P, NP, PP. We also consider oracle machines and corresponding complexity classes such as Π_i^{P} and Σ_i^{P} (the so-called *polynomial hierarchy*). We also use Wagner's *polynomial counting hierarchy* defined as the smallest set of classes containing P and, recursively, for any class C in the polynomial counting hierarchy, the classes PP^{C}, NP^{C}, and $\mathsf{coNP}^{\mathsf{C}}$ [29, 32].

3 Probabilistic logic programming

In this paper we focus on a particularly simple combination of logic programming and probabilities [23, 25]. A *probabilistic disjunctive logic program*, abbreviated PDLP, is a pair $\langle \mathbf{P}, \mathbf{PF} \rangle$ consisting of a disjunctive logic program $\mathbf{P}$ and a set of *probabilistic facts* $\mathbf{PF}$. A probabilistic fact is a pair consisting of an atom A and a probability value α, written as $\alpha :: A.$ [15]. Note that we allow a probabilistic fact to contain logical variables. As an instance of PDLP, take our running example on robots and sites: to generate a random graph over a set of five sites add the rules: $0.5 :: \mathsf{edge}(X, Y).$, $\mathsf{site}(\mathsf{s}_1).$, $\mathsf{site}(\mathsf{s}_2).$, $\mathsf{site}(\mathsf{s}_3).$, $\mathsf{site}(\mathsf{s}_4).$, $\mathsf{site}(\mathsf{s}_5).$. If $\mathbf{P}$ is a normal logic program, we just write *probabilistic logic program*, abbreviated PLP. If $\mathbf{P}$ is normal and acyclic/definite/stratified, we say the PLP is acyclic/definite/stratified.

To build the semantics of a PDLP, we first take its grounding. Suppose we have a PDLP with n ground probabilistic facts. From this we can generate 2^n DLPs: for each probabilistic fact $\alpha :: A.$, either keep fact $A.$ with probability α, or erase $A.$ with probability $1-\alpha$. A *total choice* is a subset of the set of ground probabilistic facts that is selected to be kept (other grounded probabilistic facts are discarded). For any total choice θ we obtain a DLP $\mathbf{P} \cup \mathbf{PF}^{\downarrow\theta}$ with probability $\prod_{A_i \in \theta} \alpha_i \prod_{A_i \notin \theta}(1-\alpha_i)$. Hence the distribution over total choices induces a distribution over DLPs.

We first define a semantics proposed by Lukasiewicz [18, 19]. A *probability model* for a PDLP $\langle \mathbf{P}, \mathbf{PF} \rangle$ is a probability measure $\mathbb{P}$ over interpretations, such that (i) every interpretation $\mathcal{I}$ with $\mathbb{P}(\mathcal{I}) > 0$ is a stable model of $\mathbf{P} \cup \mathbf{PF}^{\downarrow\theta}$ for the total choice θ that agrees with $\mathcal{I}$ on the probabilistic facts; and (ii) the probability of a total choice θ is $\mathbb{P}(\theta) = \prod_{A_i \in \theta} \alpha_i \prod_{A_i \notin \theta}(1-\alpha_i)$. The set of all probability models for a PDLP is the semantics of the program; note that if a PDLP does not have stable models for some total choice, there is no semantics for it (the program is *inconsistent*). Because a set of probability measures is often called a *credal set* [2]; we adopt the term *credal semantics*.

If $\mathbf{P}$ is definite, then $\mathbf{P} \cup \mathbf{PF}^{\downarrow\theta}$ is definite for any θ, and $\mathbf{P} \cup \mathbf{PF}^{\downarrow\theta}$ has a unique minimal model that is also its unique stable/well-founded model. Thus the distribution over total choices induces a single probability model. This is Sato's *distribution semantics* [25]. Similarly, suppose that $\mathbf{P}$ is acyclic or stratified; then $\mathbf{P} \cup \mathbf{PF}^{\downarrow\theta}$ is respectively acyclic or stratified for any θ, and $\mathbf{P} \cup \mathbf{PF}^{\downarrow\theta}$ has a unique stable model that is also its unique well-founded model [1].

Given a consistent PDLP whose credal semantics is the credal set $\mathbb{K}$, we may be interested in computing *lower conditional probabilities*, defined as $\underline{\mathbb{P}}(\mathbf{Q}|\mathbf{E}) = \inf_{\mathbb{P} \in \mathbb{K} : \mathbb{P}(\mathbf{E})>0} \mathbb{P}(\mathbf{Q}|\mathbf{E})$ or *upper conditional probabilities*, defined as $\overline{\mathbb{P}}(\mathbf{Q}|\mathbf{E}) = \sup_{\mathbb{P} \in \mathbb{K} : \mathbb{P}(\mathbf{E})>0} \mathbb{P}(\mathbf{Q}|\mathbf{E})$, where $\mathbf{Q}$ and $\mathbf{E}$ are consistent set of literals. Note that we leave conditional lower/upper probabilities undefined when $\overline{\mathbb{P}}(\mathbf{E}) = 0$ (that is, when $\mathbb{P}(\mathbf{E}) = 0$ for every probability model).

Consider again our running example. Suppose we have a graph over five sites, with edges (to save space, e means edge):
$0.5 :: \mathsf{e}(\mathsf{s}_4, \mathsf{s}_5).\ \ \mathsf{e}(\mathsf{s}_1, \mathsf{s}_3).\ \ \mathsf{e}(\mathsf{s}_1, \mathsf{s}_4).\ \ \mathsf{e}(\mathsf{s}_2, \mathsf{s}_1).\ \ \mathsf{e}(\mathsf{s}_2, \mathsf{s}_4).\ \ \mathsf{e}(\mathsf{s}_3, \mathsf{s}_5).\ \ \mathsf{e}(\mathsf{s}_4, \mathsf{s}_3)..$
That is, we have an edge $\mathsf{e}(\mathsf{s}_4, \mathsf{s}_5)$ which appears with probability 0.5. If this edge is kept, there are 6 stable models; if it is discarded, there are 12 stable models. If additional facts $\mathsf{color}(\mathsf{s}_2, \mathsf{red}).$ and $\mathsf{color}(\mathsf{s}_5, \mathsf{green}).$ are given, then there is a single stable model if $\mathsf{edge}(\mathsf{s}_4, \mathsf{s}_5)$ is kept, and 2 stable models if it is discarded. Then, $\underline{\mathbb{P}}(\mathsf{color}(\mathsf{s}_4, \mathsf{green})) = 0$ and $\underline{\mathbb{P}}(\mathsf{color}(\mathsf{s}_4, \mathsf{green})) = 1/2$.

A different semantics is defined by Hadjichristodoulou and Warren [17] for PLPs (note the restriction to normal logic programs!): they allow probabilities directly over well-founded models, thus allowing probabilities over atoms that are $\mathsf{undefined}$. That is, given a PLP $\langle \mathbf{P}, \mathbf{PF} \rangle$, associate to each total choice θ the unique well-founded model of $\mathbf{P} \cup \mathbf{PF}^{\downarrow\theta}$ to θ; the unique distribution over total choices induces a unique distribution over well-founded models. One can find other semantics that deserve future study, by Sato et al. [26], by Lukasiewicz [18, 19], by Baral et al. [3], by Michels et al. [21], and by Ceylan et al. [5].

4 Complexity results

We consider three different problems in this paper: (marginal) inference, most probable explanation (MPE), and maximum a posteriori (MAP).

In the following problem definitions a PDLP $\langle \mathbf{P}, \mathbf{PF} \rangle$ is always specified using rational numbers as probability values, and with a bound on the arity of predicates (so the Herbrand base is always polynomial in the input size). A query $(\mathbf{Q}, \mathbf{E})$ is always a pair of sets of consistent literals (consistent here means that the set does not contain both a literal and its negation). The set $\mathbf{E}$ is called *evidence*. The symbol $\mathbf{M}$ denotes a set of atoms in the Herbrand base of the union of the program $\mathbf{P}$ and all the facts in $\mathbf{PF}$. The symbol γ is always a rational number in $[0, 1]$.

The *inferential complexity* of a class of PDLPs is the complexity of the following decision problem: with **input** equal to a PDLP $\langle \mathbf{P}, \mathbf{PF} \rangle$, a query $(\mathbf{Q}, \mathbf{E})$, and a number γ, the **output** is whether or not $\underline{\mathbb{P}}(\mathbf{Q}|\mathbf{E}) > \gamma$; by convention, the input is rejected if $\underline{\mathbb{P}}(\mathbf{E}) = 0$.

The *MPE complexity* of a class of PDLPs is the complexity of the following decision problem: with **input** equal to a PDLP $\langle \mathbf{P}, \mathbf{PF} \rangle$, evidence $\mathbf{E}$, and a number γ, the **output** is whether or not there is an interpretation $\mathcal{I}$ that agrees with $\mathbf{E}$ and satisfies $\underline{\mathbb{P}}(\mathcal{I}) > \gamma$.

The *MAP complexity* of a class of PDLPs is the complexity of the following decision problem: with **input** equal to a PDLP $\langle \mathbf{P}, \mathbf{PF} \rangle$, a set $\mathbf{M}$, and a number γ, the **output** is whether or not there is a consistent set of literals $\mathbf{Q}$ mentioning all atoms in $\mathbf{M}$ such that $\underline{\mathbb{P}}(\mathbf{Q}|\mathbf{E}) > \gamma$; by convention, the input is rejected if $\underline{\mathbb{P}}(\mathbf{E}) = 0$.

The contributions of this paper are summarized by Table 1. Darker entries are already known [6–8], and the some entries on acyclic PLPs can be found in work by Ceylan et al. [5]. All entries in this table indicate completeness with respect to many-one reductions. In this section we prove these facts through a series of results. In all proofs the argument for membership depends on the complexity of logical reasoning on logic programs that are obtained by fixing all total choices; this suffices to even make decisions concerning conditional probabilities (using for instance techniques by Park [10, Theorem 11.5]).

Theorem 1 *The MPE complexity of acyclic propositional* PLP*s is* NP*-hard, and the MPE complexity of stratified propositional* PLP*s is in* NP*. The MPE complexity of acyclic* PLP*s is* Σ_2^P*-hard, and the MPE complexity of stratified* PLP*s is in* Σ_2^P*.*

Proof. Membership in NP for stratified propositional PLPs is shown by "guessing" a polynomial-sized interpretation consistent with evidence, and then deciding if its probability exceeds a given threshold in polynomial time. This involves producing the reduct, obtaining the stable model, and verifying whether it matches the fixed interpretation. Each of these steps takes polynomial effort [12, Table 4]. To obtain NP-hardness, note that acyclic propositional PLPs can encode Bayesian networks with binary variables (where MPE is NP-complete [11]).

To prove membership for stratified PLPs, note that deciding whether a given interpretation is a stable model of a stratified logic program can be reduced to an instance of cautious reasoning as follows: fix the total choice according to the interpretation, and include a rule with a fresh atom in the head and with the literals in the interpretation as the body. The interpretation is then a stable model iff there is (exactly) one stable model, and it contains the fresh atom. Hence, we can "guess" an interpretation consistent with evidence, then decide whether it is a stable model using a P^{NP} oracle in constant time P^{NP}-complete [12, Table 5], and finally compute its probability in polynomial time. To obtain Σ_2^P-hardness, we use an encoding employed by Eiter et al. [12]. Suppose we have formula $\phi = \exists \mathbf{X} : \neg\exists \mathbf{Y} : \varphi(\mathbf{X}, \mathbf{Y})$, where $\varphi(\mathbf{X}, \mathbf{Y})$ is a propositional formula in 3CNF with sets $\mathbf{X}$ and $\mathbf{Y}$ of propositional variables. Deciding satisfiability of such formulas is a Σ_2^P-complete problem [27]. Introduce a probabilistic fact $0.5 :: \mathsf{x}.$ with predicate x for each propositional variable x in $\mathbf{X}$. A clause c in φ contains $k \in \{0, \ldots, 3\}$ propositional variables from $\mathbf{Y}$. Introduce a predicate c of arity k, and for each predicate c introduce a set of rules as follows. For each one of the 2^k groundings $\mathbf{y}'$ of the logical variables $\mathbf{Y}'$ in c, if $\mathbf{y}'$ satisfies c (for all assignments of $\mathbf{X}$), introduce a fact $\mathsf{c}(\mathbf{y}').$; if $\mathbf{y}'$ does not satisfy c (for some assignment of $\mathbf{X}$), introduce $3-k$ rules of the form $\mathsf{c}(\mathbf{y}') :- [\mathbf{not}]\ \mathsf{x}.$, where **not** appears in the rule depending on whether x is preceded by negation or not in the clause c. The formula φ is then encoded by the rule $\mathsf{cnf} :- \mathsf{c}_1, \mathsf{c}_2, \ldots.$, where the conjunction extends over all clauses. Then the MPE with evidence $\{\neg\mathsf{cnf}\}$ and threshold $\gamma = 0$ decides whether ϕ is satisfiable. □

As definite programs are stratified, they are already covered by previous results. However, it makes sense to assume that any query with respect to such a program will also be *positive* in the sense that it only contains non-negated literals. Even then we have the same complexity as stratified programs:

Theorem 2 *Assume all queries are positive. The inferential complexity of definite propositional* PLP*s is* PP*-complete, and the inferential complexity of definite* PLP*s is* $\mathsf{PP}^{\mathsf{NP}}$*-complete. The MPE complexity of definite propositional* PLP*s is* NP*-complete, and the MPE complexity of definite* PLP*s is* Σ_2^P*-complete.*

Proof. Membership follows from results for stratified PLPs [7].

To show hardness for propositional programs, consider a 3CNF formula φ over variables $x_1, \ldots, x_n$, and obtain a new monotone formula $\tilde{\varphi}$ by replacing every literal $\neg x_i$ by a fresh variable y_i. Now the formula φ has M satisfying assigments iff the formula $\tilde{\varphi} \wedge (\bigwedge_i x_i \vee y_i) \vee \bigvee_i (x_i \wedge y_i)$ has $M + 2^{2n} - 3^n$ satisfying assignments [16, Proposition 4]. The latter formula is monotone (i.e., contains no negated variables), so we can encode it as a definite program using probabilistic facts $0.5 :: \mathsf{x}$ to represent each logical variable, $\mathsf{c_i}$ to represent clauses, and cnf to represent the value of the formula. To decide whether the number of solutions of

φ exceeds M, verify whether $\mathbb{P}(\mathsf{cnf}) > (2^{2n} - 3^n + M)/2^{2n}$. To decide whether there is a solution, decide the MPE with evidence $\{\mathsf{cnf}\}$ and threshold $2^{2n} - 3^n$.

The same reasoning applies to definite PLPs, by building existential quantification over part of the variables as in the proof of Theorem 1. □

Non-stratified programs climb one step up in the polynomial hierarchy:

Theorem 3 *Assume the credal semantics for* PLP*s. The MPE complexity of propositional* PLP*s is Σ_2^P-complete. The MPE complexity of* PLP*s is Σ_3^P-complete.*

Proof. Membership for propositional PLPs follows as we can "guess" an interpretation consistent with evidence, and then verify whether it is a stable model (cautious reasoning) with an oracle coNP [12, Table 2]. To pose the latter problem as cautious reasoning, include a rule with a fresh atom in the head and with the body encoding the interpretation, and ask whether all stable models include the head. To obtain Σ_2^P-hardness, consider a formula $\phi = \exists \mathbf{X} : \forall \mathbf{Y} : \varphi(\mathbf{X}, \mathbf{Y})$, where $\varphi(\mathbf{X}, \mathbf{Y})$ is a propositional formula in 3DNF with conjuncts d_j and sets of propositional variables $\mathbf{X}$ and $\mathbf{Y}$. Introduce a predicate $\mathsf{x_i}$ for each propositional variable x_i in $\mathbf{X}$, associated with probabilistic fact $0.5 :: \mathsf{x_i}$**.**, and introduce predicates $\mathsf{y_i}$ and $\mathsf{ny_i}$ for each propositional variable y_i in $\mathbf{Y}$, together with rules $\mathsf{y_i}$:− **not** $\mathsf{ny_i}$**.** and $\mathsf{ny_i}$:− **not** $\mathsf{y_i}$**.** (the intuition behind these rules is simple: they produce a stable model for each truth assignment of propositional variables in $\mathbf{Y}$). Introduce a predicate $\mathsf{d_j}$ to represent a conjunct d_j, with associated rule $\mathsf{d_j}$:− L_j^1, L_j^2, L_j^3**.**, where L_j^r is x or **not** x or y or **not** y depending on whether the rth literal of d_j is x, $\neg x$, y, $\neg y$ (where x indicates a propositional variable in $\mathbf{X}$ and y indicates a propositional variable in $\mathbf{Y}$). Then build φ by introducing dnf :− $\mathsf{d_j}$**.** for each $\mathsf{d_j}$. Then the MPE of this program with evidence $\{\mathsf{dnf}\}$ and threshold 0 decides whether ϕ is satisfiable (the "inner" universal quantifier is "produced" by going over all stable models when doing cautious reasoning).

Membership for PLPs follows since once an interpretation consistent with evidence is guessed, the cost of checking whether it holds in all stable models is in Π_2^P [12, Table 5]. To obtain Σ_3^P-hardness, we use a combination of strategies used in the proof of Theorem 1 and in the proof for propositional PLPs (previous paragraph). So, consider a formula $\phi = \exists \mathbf{X} : \forall \mathbf{Y} : \exists \mathbf{Z} : \varphi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where $\varphi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is a propositional formula in 3CNF with clauses c_j and sets of propositional variables $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. Deciding satisfiability of such formulas is a Σ_3^P-complete problem [27]. Again, introduce a predicate $\mathsf{x_i}$ for each propositional variable x_i in $\mathbf{X}$, associated with probabilistic fact $0.5 :: \mathsf{x_i}$**.**, and introduce predicates $\mathsf{y_i}$ and $\mathsf{ny_i}$ for each propositional variable y_i in $\mathbf{Y}$, together with rules $\mathsf{y_i}$:− **not** $\mathsf{ny_i}$**.** and $\mathsf{ny_i}$:− **not** $\mathsf{y_i}$**.** (thus generating a stable model per configuration of $\mathbf{Y}$). A clause c contains k propositional variables from $\mathbf{Z}$; as in the proof of Theorem 1, introduce a predicate c for each clause c; the arity of c is k and for each one of the 2^k configurations of logical variables from $\mathbf{Z}$ in c, introduce $3 - k$ rules $\mathsf{c}(\mathbf{z}')$:− L**.** so as to encode the clause, where L depends on the corresponding literal L_j in the clause c (L might be empty if z' satisfies the formula). Finally build the formula φ by introducing cnf :− $\mathsf{c_1}, \mathsf{c_2}, \ldots$**.**, where the conjunction extends over all clauses. Then the MPE of this program with evidence $\{\mathsf{cnf} = \mathsf{true}\}$ and threshold 0 decides whether ϕ is satisfiable. $\square$

Now consider inferential complexity under the credal semantics, for PDLPs. The credal semantics of a PDLP is a credal set that dominates an infinite monotone Choquet capacity [6]. This result is important because it implies that $\underline{\mathbb{P}}(\mathcal{M}) = \sum_{\theta \in \Theta : \Gamma(\theta) \subseteq \mathcal{M}} \mathbb{P}(\theta)$ and $\overline{\mathbb{P}}(\mathcal{M}) = \sum_{\theta \in \Theta : \Gamma(\theta) \cap \mathcal{M} \neq \emptyset} \mathbb{P}(\theta)$, where Θ is the set of total choices and Γ maps a total choice to the set of resulting stable models. Also, we have that $\underline{\mathbb{P}}(A|B) = \underline{\mathbb{P}}(A \cap B) / (\underline{\mathbb{P}}(A \cap B) + \overline{\mathbb{P}}(A^c \cap B))$ whenever $\underline{\mathbb{P}}(A \cap B) + \overline{\mathbb{P}}(A^c \cap B) > 0$; otherwise, either $\underline{\mathbb{P}}(A|B) = 1$ when $\underline{\mathbb{P}}(A \cap B) + \overline{\mathbb{P}}(A^c \cap B) = 0$ and $\overline{\mathbb{P}}(A \cap B) > 0$, or $\underline{\mathbb{P}}(A|B)$ is undefined. Similarly, $\overline{\mathbb{P}}(A|B) = \overline{\mathbb{P}}(A \cap B) / (\overline{\mathbb{P}}(A \cap B) + \underline{\mathbb{P}}(A^c \cap B))$, when $\overline{\mathbb{P}}(A \cap B) + \underline{\mathbb{P}}(A^c \cap B) > 0$, with similar special cases. Using these results we see that computing lower and upper probabilities can be reduced to going through the total choices and running brave/cautious inference for each total choice [6].

Theorem 4 *Assume the credal semantics for* PDLP*s. The inferential complexity of propositional* PDLP*s is* $\mathsf{PP}^{\Sigma_2^P}$*-complete. The inferential complexity of* PDLP*s is* $\mathsf{PP}^{\Sigma_3^P}$*-complete.*

Proof. To prove membership for propositional PDLPs, note that once a total choice is guessed, the cost of checking whether a set of literals holds under cautious reasoning is in Π_2^P [12, Table 2]; inference is obtained by going through all these total choices, running cautious inference for each one of them. To obtain $\mathsf{PP}^{\Sigma_2^P}$-hardness, consider a formula $\phi(\mathbf{X}) = \forall \mathbf{Y} : \neg \forall \mathbf{Z} : \varphi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where $\varphi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is a propositional formula in 3DNF with conjuncts d_j and sets of propositional variables $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. Deciding whether the number of truth assignments to $\mathbf{X}$ that satisfy the formula is strictly larger than an integer M is a $\mathsf{PP}^{\Sigma_2^P}$-complete problem [32, Theorem 7]. To emulate counting, introduce a predicate $\mathsf{x_i}$ for each propositional variable x_i in $\mathbf{X}$, associated with a probabilistic fact $0.5 :: \mathsf{x_i}$.. And to encode $\phi(\mathbf{X})$, we combine the use of stable sets as in the proof of Theorem 3 with an adapted version of a proof by Eiter and Gottlob [13] on disjunctive programming. So, to go over truth assignments of $\mathbf{Y}$, introduce predicates $\mathsf{y_i}$ and $\mathsf{ny_i}$ for each propositional variable y_i in $\mathbf{Y}$, together with rules $\mathsf{y_i} :- \textbf{not } \mathsf{ny_i}$. and $\mathsf{ny_i} :- \textbf{not } \mathsf{y_i}$.. And to encode the "innermost" universal quantifier, introduce predicates $\mathsf{z_i}$ and $\mathsf{nz_i}$ for each propositional variable z_i in $\mathbf{Z}$, and auxiliary predicate w, together with the rules

$$\mathsf{z_i} \vee \mathsf{nz_i}. \qquad \mathsf{z_i} :- \mathsf{w}. \qquad \mathsf{nz_i} :- \mathsf{w}.$$

for each z_i in $\mathbf{Z}$, plus the rule $\mathsf{w} :- L_j^1, L_j^2, L_j^3$. for each conjunct d_j, where L_j^r is obtained from L, the rth literal of d_j, as follows: (1) if $L = z_i$, then $L_j^r = \mathsf{z_i}$; (2) if $L = \neg z_i$, then $L_j^r = \mathsf{nz_i}$; (3) if $L = x_i$, then $L_j^r = \mathsf{x_i}$; (4) if $L = \neg x_i$, then $L_j^r = \textbf{not } \mathsf{x_i}$; (5) if $L = y_i$, then $L_j^r = \mathsf{y_i}$; (6) if $L = \neg y_i$, then $L_j^r = \textbf{not } \mathsf{y_i}$. Finally, introduce $\mathsf{nw} :- \textbf{not } \mathsf{w}$.. Now reason as follows. To decide whether $\underline{\mathbb{P}}(\mathsf{nw} = \mathsf{true}) > \gamma$, we must go through all possible total choices;

each one of them has probability 2^{-n} where n is the length of $\mathbf{X}$. For each total choice, we must run cautious inference; this is done by going through all stable models, and verifying whether nw is true in all of them. For each truth assignment of $\mathbf{Y}$, the program has a stable model where w is true iff for all truth assignments of $\mathbf{Z}$ we have that φ holds [13, Theorem 3.2]. That is, for fixed $\mathbf{y}$, all resulting stable models have nw as true iff $\forall \mathbf{Z} : \varphi(\mathbf{x}, \mathbf{y}, \mathbf{Z})$ is false (note that $\mathbf{x}$ is fixed by the selected total choice). Thus if we take $\gamma = M/2^{-n}$, we obtain that $\underline{\mathbb{P}}(\mathsf{nw} = \mathsf{true}) > \gamma$ decides whether $\phi(\mathbf{X})$ has a number of satisfying truth assignments of $\mathbf{X}$ that is strictly larger than M.

To prove membership for PDLPs, note that once a total choice is guessed, the cost of checking whether a set of literals holds under cautious reasoning is in Π_3^P [12, Table 5]; inference is obtained by going through all these total choices, running cautious inference for each one of them. To obtain $\mathsf{PP}^{\Sigma_3^P}$-hardness, we use a combination of strategies used in the proof of Theorem 1 and in the proof for propositional PLPs (previous paragraph). So, consider a formula $\phi(\mathbf{X}) = \forall \mathbf{Y} : \neg \forall \mathbf{Z} : \exists \mathbf{V} : \varphi(\mathbf{V}, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where $\varphi(\mathbf{V}, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is a propositional formula in 3CNF with clauses c_j and sets of propositional variables $\mathbf{V}$, $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. Deciding whether the number of truth assignments to $\mathbf{X}$ that satisfy the formula is strictly larger than an integer M is a $\mathsf{PP}^{\Sigma_3^P}$-complete problem [32, Theorem 7]. Again, introduce a predicate $\mathsf{x_i}$ for each propositional variable x_i in $\mathbf{X}$, associated with probabilistic fact $0.5 :: \mathsf{x_i}.$, and introduce predicates $\mathsf{y_i}$ and $\mathsf{ny_i}$ for each propositional variable y_i in $\mathbf{Y}$, together with rules $\mathsf{y_i} :- \textbf{not } \mathsf{ny_i}.$, $\mathsf{ny_i} :- \textbf{not } \mathsf{y_i}.$ and $\mathsf{z_i} \vee \mathsf{nz_i}.$. We encode the clauses by adapting the construction in the proof of Theorem 1. A clause c contains k propositional variables from $\mathbf{V}$; introduce a predicate $\mathsf{c_j}$ for each clause c_j; the arity of $\mathsf{c_j}$ is k and for each one of the 2^k configurations of logical variables from $\mathbf{V}$ in c_j, introduce $3-k$ rules $\mathsf{c_j}(\mathbf{v}') :- L_j^r.$ so as to encode the clause, where L_j^r is obtained from the corresponding literal L from c_j as follows: (1) if $L = v_i$, then $L_j^r = \mathsf{v_i}$; (2) if $L = \neg v_i$, then $L_j^r = \mathsf{nv_i}$; (3) if $L = z_i$, then $L_j^r = \mathsf{z_i}$; (4) if $L = \neg z_i$, then $L_j^r = \mathsf{nz_i}$; (5) if $L = x_i$, then $L_j^r = \mathsf{x_i}$; (6) if $L = \neg x_i$, then $L_j^r = \textbf{not } \mathsf{x_i}$; (7) if $L = y_i$, then $L_j^r = \mathsf{y_i}$; (8) if $L = \neg y_i$, then $L_j^r = \textbf{not } \mathsf{y_i}$. Finally build the formula φ by introducing $\mathsf{w} :- \mathsf{c_1}, \mathsf{c_2}, \ldots.$, where the conjunction extends over all clauses. So we have the "innermost" existential quantifier; to create the "middle" universal quantifier, introduce

$$\mathsf{y_k} :- \mathsf{w}. \qquad \mathsf{ny_k} :- \mathsf{w}. \qquad \mathsf{z_i} :- \mathsf{w}. \qquad \mathsf{nz_i} :- \mathsf{w}.$$

for each y_k in $\mathbf{Y}$ and each z_i in $\mathbf{Z}$. Finally add the rule $\mathsf{nw} :- \textbf{not } \mathsf{w}.$. If we take $\gamma = M/2^{-n}$, we obtain that $\underline{\mathbb{P}}(\mathsf{nw} = \mathsf{true}) > \gamma$ decides whether $\phi(\mathbf{X})$ has a number of satisfying truth assignments of $\mathbf{X}$ that is strictly larger than M. □

Theorem 5 *Assume the credal semantics for* PDLP*s. The MPE complexity of propositional* PDLP*s is Σ_3^P-complete. The MPE complexity of* PDLP*s is Σ_4^P-complete.*

Proof. To prove membership for propositional PDLPs, note that once an interpretation is guessed, the cost of checking whether it holds under cautious reasoning is in Π_2^P [12, Table 2]. To obtain Σ_3^P-hardness, consider a formula $\phi = \exists \mathbf{X} : \forall \mathbf{Y} : \neg\forall \mathbf{Z} : \varphi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where $\varphi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is a propositional formula in 3DNF with conjuncts d_j and sets of propositional variables $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. Deciding whether ϕ is satisfiable is a Σ_3^P-complete problem [27]. So, build the program described in the propositional part of the proof of Theorem 4. Then the MPE of the resulting program with evidence $\{\mathsf{nw} = \mathsf{true}\}$ and threshold 0 decides whether ϕ is satisfiable.

To prove membership for PDLPs, note that once an interpretation is guessed, the cost of checking whether it holds under cautious reasoning is in Π_3^P [12, Table 5]. To obtain Σ_4^P-hardness, consider a formula $\phi = \exists \mathbf{X} : \forall \mathbf{Y} : \neg\forall \mathbf{Z} : \exists \mathbf{V} : \varphi(\mathbf{V}, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where $\varphi(\mathbf{V}, \mathbf{X}, \mathbf{Y}, \mathbf{Z})$ is a propositional formula in 3CNF with clauses c_j and sets of propositional variables $\mathbf{V}$, $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. Deciding whether ϕ is satisfiable is a Σ_4^P-complete problem [27]. So, build the program described in the bounded-arity part of the proof of Theorem 4. Then the MPE of the resulting program with evidence $\{\mathsf{nw} = \mathsf{true}\}$ and threshold 0 decides whether ϕ is satisfiable. □

Theorem 6 *Assume the well-founded semantics for* PLP*s. The MPE complexity of propositional* PLP*s is* NP*-complete. The MPE complexity of* PLP*s is* Σ_2^P*-complete.*

Proof. Hardness follows from Theorem 1 (in both cases). Membership for the propositional case can be proved using the corresponding argument in the proof of Theorem 1, only noting that logical inference with propositional PLPs under the well-founded semantics is in P [9]. Similarly, membership for the relational bounded-arity case uses the corresponding argument in the proof of Theorem 1, noting that logical inference with bounded-arity PLPs is P^{NP}-complete [8]. □

Theorem 7 *Assume the credal semantics both for* PLP*s and for* PDLP*s. The MAP complexity of propositional* PLP*s is* $\mathsf{NP}^{\mathsf{PP}}$*-hard. The MAP complexity of* PDLP*s is in* $\mathsf{NP}^{\mathsf{PP}}$*.*

Proof. Hardness is simple: a propositional PLP can encode a Bayesian network with binary variables, and MAP in such networks is $\mathsf{NP}^{\mathsf{PP}}$-complete [11]. To prove membership, reason in two steps. First note that one can solve MAP by first guessing literals for the MAP-predicates that are not fixed by evidence, and then running inference in an $\mathsf{PP}^{\Sigma_3^P}$ oracle. That is, the decision problem is in $\mathsf{NP}^{\mathsf{PP}^{\Sigma_3^P}}$. Now resort to a theorem by Toda and Watanabe [28] that shows that, for any k, $\mathsf{P}^{\mathsf{PP}^{\Sigma_k^P}}$ collapses to P^{PP}, to note that the decision problem of interest is in $\mathsf{NP}^{\mathsf{PP}}$. □

Theorem 8 *Assume the credal semantics for* PLP*s and positive queries The MAP complexity of definite propositional* PLP*s is* $\mathsf{NP}^{\mathsf{PP}}$*-hard, and of* PLP*s is in* $\mathsf{NP}^{\mathsf{PP}}$*.*

Proof. Membership follows from Theorem 7. Hardness follows from the fact that MAP for Bayesian networks described without negation is already $\mathsf{NP}^{\mathsf{PP}}$-hard [20, Theorem 5]. □

5 Conclusion

As conveyed by Table 1, we have presented a number of novel results concerning the complexity of probabilistic logic programming — the complexity of computing conditional probabilities (inferences) and two kinds of explanations (MPE and MAP). These computations go up several layers within the counting hierarchy, reaching some interesting complexity classes that are rarely visited. Note in particular that MAP is *always* $\mathsf{NP}^{\mathsf{PP}}$-complete, an interesting result.

A future step is to obtain the complexity of relational programs without bounds on arity (exponential complexity is sure to appear), and perhaps the complexity of programs with functions (with suitable restrictions to guarantee decidability). The complexity of other constructs, such as aggregates, should also be explored. Future work should look at "query" complexity; that is, the complexity of computing inferences when the program is fixed and the query varies — this is akin to data complexity as studied in database theory [7].

Acknowledgements

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

References

1. K. R. Apt and M. Bezem. Acyclic programs. *New Generation Computing*, 9:335–363, 1991.
2. T. Augustin, F. P. A. Coolen, G. de Cooman, and M. C. M. Troffaes. *Introduction to Imprecise Probabilities*. Wiley, 2014.
3. C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.
4. C. Baral and V. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *J. of Automated Reasoning*, 10(3):399–420, 1993.
5. Í. Í. Ceylan, T. Lukasiewicz, and R. Peñaloza. Complexity results for probabilistic Danalog$^{\pm}$. In *European Conf. on Artificial Intelligence*, pages 1414–1422, 2016.
6. F. G. Cozman and D. D. Mauá. The structure and complexity of credal semantics. In *Workshop on Probabilistic Logic Programming*, pages 3–14, 2016.
7. F. G. Cozman and D. D. Mauá. Probabilistic graphical models specified by probabilistic logic programs: Semantics and complexity. In *Conf. on Probabilistic Graphical Models — JMLR Proceedings*, volume 52, pages 110–121, 2016.
8. F. G. Cozman and D. D. Mauá. The well-founded semantics of cyclic probabilistic logic programs: meaning and complexity. In *Encontro Nacional de Inteligência Artificial e Computacional*, pages 1–12, 2016.
9. E. Dantsin, T. Eiter, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
10. A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

11. C. Polpo de Campos and F. G. Cozman. The inferential complexity of Bayesian and credal networks. In *IJCAI*, pages 1313–1318, 2005.
12. T. Eiter, W. Faber, M. Fink, and S. Woltran. Complexity results for answer set programming with bounded predicate arities and implications. *Annals of Mathematics and Artificial Intelligence*, 5:123–165, 2007.
13. T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15:289–323, 1995.
14. T. Eiter, G. Ianni, and T. Krennwalner. Answer set programming: a primer. In *Reasoning Web*, pages 40–110. Springer-Verlag, 2009.
15. D. Fierens, G. Van den Broeck, J. Renkens, D. Shrerionov, B. Gutmann, G. Janssens, and L. de Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2014.
16. J. Goldsmith, M. Hagen, and M. Mundhenk. Complexity of DNF minimization and isomorphism testing for monotone formulas. *Information and Computation*, 206(6):760–775, 2008.
17. S. Hadjichristodoulou and D. S. Warren. Probabilistic logic programming with well-founded negation. *Int. Symp. on Multiple-Valued Logic*, pages 232–237, 2012.
18. T. Lukasiewicz. Probabilistic description logic programs. In *Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 737–749, 2005.
19. T. Lukasiewicz. Probabilistic description logic programs. *In. J. of Approximate Reasoning*, 45(2):288–307, 2007.
20. D. D. Mauá, C. Polpo de Campos, F. G. Cozman. The complexity of MAP inference in Bayesian networks specified through logical languages. *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 889–895, 2015.
21. S. Michels, A. Hommersom, P. J. F. Lucas, and M. Velikova. A new probabilistic constraint logic programming language based on a generalised distribution semantics. *Artificial Intelligence J.*, 228:1–44, 2015.
22. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
23. D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.
24. D. Poole. The Independent Choice Logic and beyond. In L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors, *Probabilistic Inductive Logic Programming*, volume 4911 of *Lecture Notes in Computer Science*, pages 222–243. Springer, 2008.
25. T. Sato. A statistical learning method for logic programs with distribution semantics. In *Int. Conf. on Logic Programming*, pages 715–729, 1995.
26. T. Sato, Y. Kameya, and N.-F. Zhou. Generative modeling with failure in PRISM. In *Int. Joint Conf. on Artificial Intelligence*, pages 847–852, 2005.
27. L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1, 1–22, 1976.
28. S. Toda and O. Watanabe. Polynomial-time 1-Turing reductions from #PH to #P. *Theoretical Computer Science*, 100:205–221, 1992.
29. J. Tóran. Complexity classes defined by counting quantifiers. *J. of the ACM*, 38(3):753–774, 1991.
30. L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.
31. A. van Gelder, J. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *J. Association for Computing Machinery*, 38(3):620–650, 1991.
32. K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.