Probabilistic Logic Programming

Thomas Lukasiewicz[1]

Abstract. We present a new approach to probabilistic logic programs with a possible worlds semantics. Classical program clauses are extended by a subinterval of $[0,1]$ that describes the range for the conditional probability of the head of a clause given its body. We show that deduction in the defined probabilistic logic programs is computationally more complex than deduction in classical logic programs. More precisely, restricted deduction problems that are P-complete for classical logic programs are already NP-hard for probabilistic logic programs. We then elaborate a linear programming approach to probabilistic deduction that is efficient in interesting special cases. In the best case, the generated linear programs have a number of variables that is linear in the number of ground instances of purely probabilistic clauses in a probabilistic logic program.

1 INTRODUCTION

There is already a quite extensive literature on probabilistic propositional logics and their various dialects. The most famous approach should be Nilsson's probabilistic logic [23]. Another important work, which deals with reasoning about probabilities that are expressed by linear inequalities over propositional events, is presented and analyzed in [10]. Probabilistic first-order logics are discussed, for example, in [2] and [14]. They can be distinguished in first-order logics in which probabilities are defined over a set of possible worlds and those in which probabilities are given over the elements of the domain. The first ones are suitable for representing degrees of belief, while the latter are appropriate for describing statistical knowledge. The same classification holds for approaches to probabilistic logic programming (see, for example, [22] for probabilities over possible worlds and, for example, [21] for probabilities over the domain).

This paper concentrates on probabilistic logic programming with a possible worlds semantics. Hence, the work closest in spirit to ours is [22]. There are, however, the following crucial differences between our approach and the work in [22]. First of all, our probabilistic logic programs include purely probabilistic rules, which are interpreted in the sense of conditional probabilities, while [22] just allows classical logical rules in which the atoms are annotated probabilistic facts. Moreover, our aim is an approach that is as efficient as possible, while the more general work in [22] is more of theoretical nature.

The main contributions of this work can be summarized as follows. As a first contribution, we present the syntax and the semantics of probabilistic logic programs. We extend classical program clauses by a subinterval of $[0,1]$ that describes the range for the conditional probability of the head of a clause given its body. Based on the notions of logical and precise logical consequence, we then introduce the notions of correct and tight answers for probabilistic queries.

[1] Institut für Informatik, Universität Gießen, Arndtstr. 2, D-35392 Gießen, Germany, E-mail: lukasiewicz@informatik.uni-giessen.de

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

As a second contribution, we show that deduction in probabilistic logic programs is computationally more complex than deduction in classical logic programs: restricted deduction problems that are P-complete for classical logic programs are already NP-hard for probabilistic logic programs. Hence, any attempt towards efficient deduction in probabilistic logic programs should be guided by looking for efficient special-case, average-case, or approximation techniques.

As a third contribution, by generalizing own work from [18], we elaborate a linear programming approach to deduction in probabilistic logic programs, which is efficient in interesting special cases. In our framework, probabilistic deduction problems can easily be represented by linear programs. However, these initial linear programs have a number of variables that is exponential in the cardinality of the Herbrand base. Moreover, also the Herbrand base of a probabilistic logic program is generally quite large. Motivated by this observation, we elaborate a technique that, in the best case, yields linear programs with a number of variables that is linear in the number of ground instances of purely probabilistic clauses. This result is very promising. The work in [22], in contrast, is based on solving an exponential number of linear programs over an exponential number of variables (both in the cardinality of the Herbrand base) in each fixpoint iteration step and in each compilation step for SLDp-refutation.

The rest of this paper is organized as follows. In Section 2, we present the syntax and the semantics of probabilistic logic programs and of queries addressed to them. Section 3 gives an illustrative example. Section 4 concentrates on the computational complexity of probabilistic deduction in our framework. In Sections 5 and 6, we present and discuss an optimized linear programming approach to probabilistic deduction. Section 7 summarizes the main results.

2 PROBABILISTIC LOGIC PROGRAMS

In this section, we define the syntax and the semantics of probabilistic logic programs. For this purpose, let Φ be a first-order vocabulary that contains a finite and nonempty set of predicate symbols and a finite and nonempty set of constant symbols. Thus, we do not consider function symbols in our framework. Let $\mathcal{X}$ be a set of *object variables* and *bound variables*. Object variables represent elements of a certain domain, while bound variables describe real numbers in the interval $[0,1]$. Object variables may occur in probabilistic logic programs and probabilistic queries, whereas bound variables are just allowed in probabilistic queries.

An *object term* is a constant symbol from Φ or an object variable from $\mathcal{X}$. An *atomic formula* is an expression of the kind $p(t_1,\ldots,t_k)$ with a predicate symbol p of arity $k \geq 0$ from Φ and object terms $t_1,\ldots,t_k$. A *conjunctive formula* is the *false formula* $\perp$, the *true formula* $\top$, or an expression $A_1 \wedge \cdots \wedge A_l$ with atomic formulas $A_1,\ldots,A_l$ and $l > 0$. A *probabilistic clause* is an expression $(H|B)[c_1,c_2]$ with conjunctive formulas H, B different from $\perp$ and real numbers $c_1, c_2 \in [0,1]$. In the probabilistic clause $(H|B)[c_1, c_2]$, we call H the *head* and B the *body*. We implicitly assume that all object variables in a probabilistic clause are universally quantified. A *probabilistic program clause* is a probabilistic clause $(H|B)[c_1, c_2]$ with $c_1 \leq c_2$. A *probabilistic logic program* is a finite set of probabilistic program clauses.

Hence, probabilistic program clauses are an extension of classical program clauses in the following two senses. First, we increase a program clause by a subinterval of $[0,1]$ that describes the range for the conditional probability of the head of a clause given its body. Second, we also allow conjunctions of two or more atomic formulas in the head of a clause. We do this, since for a probabilistic program clause with two or more atomic formulas in the head, there generally does not exist anymore a logically equivalent set of probabilistic program clauses with single atomic formulas in their heads.

Probabilistic program clauses can be classified into the following kinds of *logical program clauses* and *purely probabilistic program clauses*. A probabilistic program clause $(H|\top)[1,1]$ is a *logical fact* and more intuitively written as $H \leftarrow \top$. $(H|B)[1,1]$ with $B \neq \top$ is a *logical rule* and more intuitively written as $H \leftarrow B$. $(H|B)[0,0]$ is a *logical constraint* and more intuitively written as $\bot \leftarrow H \wedge B$. A probabilistic program clause $(H|\top)[c_1, c_2]$ with $c_1 < 1$ and $c_2 > 0$ is a *purely probabilistic fact*. $(H|B)[c_1, c_2]$ with $B \neq \top$, $c_1 < 1$, and $c_2 > 0$ is a *purely probabilistic rule*.

An object term, an atomic formula, a conjunctive formula, a probabilistic clause, or a probabilistic logic program is *ground* iff it does not contain any object variables. A *ground instance* of a conjunctive formula C is a ground conjunctive formula $C\theta$, and a *ground instance* of a probabilistic clause $(H|B)[c_1, c_2]$ is a ground probabilistic clause $(H\theta|B\theta)[c_1, c_2]$, where θ is a ground substitution for exactly all object variables that occur in C and $(H|B)[c_1, c_2]$, respectively. Note that the notions of substitutions and of ground substitutions are defined in the classical way. Note also that, since the vocabulary Φ is finite and without function symbols, the number of ground instances of a probabilistic clause is also finite.

The semantics of probabilistic clauses is defined by a possible worlds semantics in which each possible world is identified with a Herbrand interpretation of the classical first-order language for Φ and $\mathcal{X}$. Hence, the *set of possible worlds* $\mathcal{W}_\Phi$ is the set of all subsets of the Herbrand base HB_Φ of the classical first-order language for Φ and $\mathcal{X}$. Thus, since Φ is a finite first-order vocabulary without function symbols, the set of possible worlds is also finite. A ground conjunctive formula C *is true in* a possible world W, denoted $W \models C$, iff C is true in the Herbrand interpretation belonging to W.

A *probabilistic interpretation* Pr is a mapping $Pr\colon \mathcal{W}_\Phi \rightarrow [0,1]$ such that all $Pr(W)$ with $W \in \mathcal{W}_\Phi$ sum up to 1. Pr is extended to ground conjunctive formulas C by $Pr(C) = \sum_{W \in \mathcal{W}_\Phi, W \models C} Pr(W)$. Pr is extended to ground probabilistic clauses by:

$$(H|B)[c_1, c_2] \text{ is true in } Pr \text{ iff}$$
$$c_1 \cdot Pr(B) \leq Pr(B \wedge H) \leq c_2 \cdot Pr(B)\,.$$

Pr is extended to non-ground probabilistic clauses by: $(H|B)[c_1, c_2]$ *is true in* Pr iff all ground instances of $(H|B)[c_1, c_2]$ are true in Pr. We write $Pr \models (H|B)[c_1, c_2]$ iff $(H|B)[c_1, c_2]$ is true in Pr.

The notions of models, satisfiability, and logical consequence for probabilistic clauses are defined in the classical way. A probabilistic interpretation Pr is a *model* of a probabilistic clause $(H|B)[c_1, c_2]$ iff $Pr \models (H|B)[c_1, c_2]$. Pr is a *model* of a set of probabilistic clauses $\mathcal{P}$, denoted $Pr \models \mathcal{P}$, iff Pr is a model of all probabilistic clauses in $\mathcal{P}$. $\mathcal{P}$ is *satisfiable* iff a model of $\mathcal{P}$ exists. $(H|B)[c_1, c_2]$ is a *logical consequence* of $\mathcal{P}$, denoted $\mathcal{P} \models (H|B)[c_1, c_2]$, iff each model of $\mathcal{P}$ is also a model of $(H|B)[c_1, c_2]$.

For a probabilistic clause $(H|B)[c_1, c_2]$ and a set of probabilistic clauses $\mathcal{P}$, let $\boldsymbol{c}$ denote the set of all real numbers $c \in [0,1]$ for which there exists a model Pr of $\mathcal{P}$ and a ground substitution θ for exactly all object variables in $(H|B)[c_1, c_2]$ such that $Pr \models (H\theta|B\theta)[c, c]$ and $Pr(B\theta) > 0$. Now, we see easily that $(H|B)[c_1, c_2]$ is a logical consequence of $\mathcal{P}$ iff $c_1 \leq \inf \boldsymbol{c}$ and $c_2 \geq \sup \boldsymbol{c}$.

This observation yields a canonic notion of tightness for logical consequences as follows. The probabilistic clause $(H|B)[c_1, c_2]$ is a *tight logical consequence* of $\mathcal{P}$, denoted $\mathcal{P} \models_{\mathit{tight}} (H|B)[c_1, c_2]$, iff $c_1 = \inf \boldsymbol{c}$ and $c_2 = \sup \boldsymbol{c}$.

Note that $\boldsymbol{c}$ is a closed interval in the real numbers (see, for example, [11] and [17]). For $\boldsymbol{c} = \emptyset$, we canonically define $\inf \boldsymbol{c} = 1$ and $\sup \boldsymbol{c} = 0$. Now, $\boldsymbol{c} = \emptyset$ iff $\mathcal{P} \models (B|\top)[0,0]$ iff $\mathcal{P} \models_{\mathit{tight}} (H|B)[1,0]$ iff $\mathcal{P} \models (H|B)[c_1, c_2]$ for all $c_1, c_2 \in [0,1]$.

Based on the notions of logical consequence and of tight logical consequence, we can define probabilistic queries to probabilistic logic programs and their correct and tight answers as follows.

A *bound term* is a real number $c \in [0,1]$ or a bound variable from $\mathcal{X}$. A *probabilistic query* to a probabilistic logic program $\mathcal{P}$ is an expression $\exists(H|B)[t_1, t_2]$ with conjunctive formulas H, B different from $\bot$ and bound terms t_1, t_2. A bound term or a probabilistic query is *ground* iff it does not contain any variables. A probabilistic query is *o-ground* iff it does not contain any object variables.

The *correct answer* for the ground probabilistic query $\exists(H|B)[t_1, t_2]$ is *Yes* if $\mathcal{P} \models (H|B)[t_1, t_2]$ and *No* otherwise. A *correct answer* for the non-ground probabilistic query $\exists(H|B)[t_1, t_2]$ is a pair (θ, η) with $\mathcal{P} \models (H\theta|B\theta)[t_1\eta, t_2\eta]$, where θ is a ground substitution for exactly all object variables in $\exists(H|B)[t_1, t_2]$, and η is a ground substitution for exactly all bound variables in $\exists(H|B)[t_1, t_2]$.

The *tight answer* for the ground probabilistic query $\exists(H|B)[t_1, t_2]$ is *Yes* if $\mathcal{P} \models_{\mathit{tight}} (H|B)[t_1, t_2]$ and *No* otherwise. A *tight answer* for the non-ground probabilistic query $\exists(H|B)[t_1, t_2]$ is a pair (θ, η) with $\mathcal{P} \models_{\mathit{tight}} (H\theta|B\theta)[t_1\eta, t_2\eta]$, where θ is a ground substitution for exactly all object variables in $\exists(H|B)[t_1, t_2]$, and η is a ground substitution for exactly all bound variables in $\exists(H|B)[t_1, t_2]$.

Probabilistic queries of the kind $\exists(H|B)[c_1, c_2]$, $\exists(H|B)[x, c_2]$, $\exists(H|B)[c_1, x]$, and $\exists(H|B)[x, x]$ with real numbers $c_1, c_2 \in [0,1]$ and a bound variable x from $\mathcal{X}$ can computationally be reduced to probabilistic queries of the form $\exists(H|B)[x_1, x_2]$ with bound variables x_1, x_2 from $\mathcal{X}$ such that $x_1 \neq x_2$. Hence, in the sequel, we just consider the last, most general, kind of probabilistic queries.

Given a probabilistic query $\exists(H|B)[x_1, x_2]$ with bound variables x_1, x_2 from $\mathcal{X}$ such that $x_1 \neq x_2$, we consider its tight answers as the desired semantics, for the following reasons: first of all, the tight answers for $\exists(H|B)[x_1, x_2]$ subsume all correct answers. Moreover, the number of tight answers is always finite, while the number of correct answers for $\exists(H|B)[x_1, x_2]$ is generally infinite. Finally, also from the practical point of view, we are interested in the tightest bounds that are entailed by a probabilistic logic program.

3 EXAMPLE

Let us assume that John wants to pick up Mary after she stopped working. To do so, he must drive from his home to her office. However, he left quite late. So, he is wondering if he can still reach her in time. Unfortunately, since it is rush hour, it is very probable that he runs into a traffic jam. Now, John has the following knowledge at hand: given a road from x to y, the probability that he can reach y through x without running into a traffic jam is greater than 70% (a), given a road in the south of the town, he knows that this probability is even greater than 80% (b). A friend just called him and told him about some roads without any significant traffic (c). Furthermore, he clearly knows that if he can reach y through z and z through x, both without running into a traffic jam, then he can also reach y through x without running into a traffic jam (d). This knowledge can be expressed as follows (x, y, and z are object variables):

(a) $(reach(x,y)|road(x,y))[.7,1]$
(b) $(reach(x,y)|road(x,y) \wedge south(x,y))[.8,1]$
(c) $(reach(x,y)|road(x,y) \wedge advice(x,y))[1,1]$
(d) $(reach(x,y)|reach(x,z) \wedge reach(z,y))[1,1]$.

John is wondering if he can reach Mary's office (*of*) from his home (*ho*), such that the probability of him running into a traffic jam is smaller than 1%. This can be expressed by the ground probabilistic query $\exists(reach(ho, of)|\top)[.99,1]$. His wondering for the tight probability range of reaching the office, without running into a traffic jam, can be expressed by $\exists(reach(ho, of)|\top)[x_1, x_2]$. Another query is $\exists(reach(ho, of)|reach(ho, a))[x_1, x_2]$. Of course, the correct answer for the first query and the tight answers for the second and the third query depend on the concrete knowledge of the roads, of the roads in the south of the town, and of the roads that John's friend was talking about. Some self-explaining probabilistic program clauses that describe this knowledge could be given as follows (*ho*, *a*, *b*, *c*, and *of* are constant symbols; the fourth clause describes the fact that the road from b to c is probably closed for construction):

$(road(ho,a)|\top)[1,1]$, $(road(a,b)|\top)[1,1]$, $(road(b,of)|\top)[1,1]$
$(road(b,c)|\top)[.3,.4]$, $(south(a,b)|\top)[1,1]$, $(south(b,of)|\top)[0,0]$
$(advice(ho,a)|\top)[1,1]$, $(advice(b,of)|\top)[1,1]$.

4 COMPUTATIONAL COMPLEXITY

Before focusing on deduction in probabilistic logic programs, we would like to get a rough idea on its inherent computational complexity. In this section, we analyze the computational complexity of two special cases of the problem of computing tight answers for probabilistic queries. The first one deals with computing the tight answer for an o-ground probabilistic query to a ground probabilistic logic program. The second one concerns computing the tight answer for an o-ground probabilistic query to a fixed probabilistic logic program under a varying set of ground logical facts.

These two special problems have their counterparts in classical logic programming (without function symbols). The first one is to decide whether a ground classical program clause is a logical consequence of a ground classical logic program. The second one is to decide whether a ground classical program clause is a logical consequence of a fixed classical logic program under a varying set of ground logical facts. However, while these two classical counterparts are known to be P-complete (see, for example, [6] for a survey), our two special problems of deduction in probabilistic logic programs are now shown to be already NP-hard.

We also show the NP-hardness of two corresponding special problems of deciding whether a probabilistic logic program is satisfiable. While classical logic programs (sets of logical facts and logical rules, without logical constraints) are always satisfiable, probabilistic logic programs may become unsatisfiable, just for logical inconsistencies through logical constraints or, more generally, for probabilistic inconsistencies in the assumed probability ranges.

Theorem 1 *Let x_1, x_2 be bound variables from $\mathcal{X}$ with $x_1 \neq x_2$ and let H be a ground atomic formula.*
a) The problem of deciding whether a ground probabilistic logic program is satisfiable is NP-hard.
b) The problem of computing the tight answer for $\exists(H|\top)[x_1, x_2]$ to a ground probabilistic logic program is NP-hard.

Proof sketch. a) The NP-complete problem of graph 3-colorability [12] can be polynomially-reduced to the described problem of probabilistic logic program satisfiability (see also [20]).

b) The claim follows immediately from a), since a probabilistic logic program $\mathcal{P}$ is satisfiable iff $\eta = \{x_1/1, x_2/1\}$ is the tight answer for $\exists(h|\top)[x_1, x_2]$ to $\mathcal{P} \cup \{(h|\top)[1,1]\}$, where h is a new 0-ary predicate symbol that does not occur in $\mathcal{P}$. $\square$

Note that if we restrict our considerations to probabilistic program clauses $(H|B)[c_1, c_2]$ with rational numbers c_1 and c_2, then the problem of deciding whether a ground probabilistic logic program is satisfiable is immediately in NP, since it can easily be reduced to the problem of deciding whether a *weight formula* is satisfiable, which is introduced and shown to be NP-complete in [10].

Theorem 2 *Let $\mathcal{P}$ be a fixed probabilistic logic program and let $\mathcal{F}$ be a varying set of ground logical facts. Let x_1, x_2 be bound variables from $\mathcal{X}$ with $x_1 \neq x_2$ and let H be a ground atomic formula.*
a) The problem of deciding whether $\mathcal{P} \cup \mathcal{F}$ is satisfiable is NP-hard.
b) The problem of computing the tight answer for $\exists(H|\top)[x_1, x_2]$ to $\mathcal{P} \cup \mathcal{F}$ is NP-hard.

Proof sketch. a) Like in the proof of Theorem 1 a), the NP-complete problem of graph 3-colorability can be polynomially-reduced to the described problem of probabilistic logic program satisfiability. The main idea is that the set of logical facts $\mathcal{F}$ can be used to encode any given undirected graph (V, E) with a finite number of nodes V.

b) Like in the proof of Theorem 1 b), the claim follows immediately from a). $\square$

These results show that deduction in probabilistic logic programs is computationally more complex than deduction in classical logic programs. More precisely, restricted deduction problems that are computationally tractable for classical logic programs are presumably intractable for probabilistic logic programs. Hence, any attempt towards efficient deduction in probabilistic logic programs should be guided by looking for efficient special-case, average-case, or approximation techniques. In the next section, we provide a general technique that is efficient in special cases.

5 PROBABILISTIC DEDUCTION

In this section, we focus on the problem of computing tight answers for o-ground probabilistic queries to probabilistic logic programs. Note that both the problem of computing tight answers for non-o-ground probabilistic queries to probabilistic logic programs and the problem of deciding whether a probabilistic logic program is satisfiable can be reduced to this problem. In the sequel, let $\mathcal{P}$ be a probabilistic logic program and let $\exists(F|E)[x_1, x_2]$ be an o-ground probabilistic query with bound variables x_1, x_2 from $\mathcal{X}$ such that $x_1 \neq x_2$.

The tight answer for $\exists(F|E)[x_1, x_2]$ is computable in a straightforward way by linear programming. A linear fractional objective function in possible worlds must be minimized and maximized subject to a number of linear constraints over possible worlds. Now, by a standard technique from [5], these linear fractional programs can be transformed into linear programs (that is, into the problems of minimizing and maximizing a linear objective function in possible worlds subject to a number of linear constraints over possible worlds).

In our case of computing the tight answer for $\exists(F|E)[x_1,x_2]$, we get the following two linear programs in (1) with $y_W \geq 0$ for all $W \in \mathcal{W}_\Phi$ and $\text{opt} \in \{\min,\max\}$. If $\mathcal{P} \not\models (E|\top)[0,0]$, then the two linear programs are solvable and the requested tightest bounds are given by their two optimal values. If $\mathcal{P} \models (E|\top)[0,0]$, then the two linear programs are not solvable.

$$\text{opt} \sum_{W \in \mathcal{W}_\Phi,\, W \models E \wedge F} y_W \ \text{ subject to } \mathcal{LC}_\Phi, \tag{1}$$

where $\mathcal{LC}_\Phi$ is the least set of linear constraints that contains:

- $\sum_{W \in \mathcal{W}_\Phi,\, W \models E} y_W \ = \ 1$
- $c_1 \cdot \sum_{W \in \mathcal{W}_\Phi,\, W \models B} y_W \ \leq \ \sum_{W \in \mathcal{W}_\Phi,\, W \models H \wedge B} y_W$ and $c_2 \cdot \sum_{W \in \mathcal{W}_\Phi,\, W \models B} y_W \ \geq \ \sum_{W \in \mathcal{W}_\Phi,\, W \models H \wedge B} y_W$ for all ground instances $(H|B)[c_1,c_2]$ of probabilistic clauses in $\mathcal{P}$.

The worst-case time complexity of linear programming is known to be polynomial in the size of the linear programs (see, for example, [26]). However, the crux with our linear programs in (1) is that the number of variables, or equally the number of possible worlds, is exponential in the cardinality of the Herbrand base. Moreover, in general, the Herbrand base is itself quite large. For instance, if we assume that Φ contains only one 2-ary predicate symbol and only ten constant symbols, then the Herbrand base contains already 100 ground atomic formulas, and we get $2^{100} \approx 10^{30}$ possible worlds!

So, any attempt to elaborate an efficient technique for computing tight answers for o-ground probabilistic queries should be guided by searching and elaborating techniques that reduce the number of variables in (1). This is exactly the topic of the rest of this section.

Clearly, our possible worlds do not have to contain predicate symbols from Φ that are not in $\mathcal{P}$. Let $HB_\mathcal{P}$ denote the set of all ground atomic formulas that are constructed with predicate symbols from $\mathcal{P}$ (we assume that $\mathcal{P}$ contains at least one predicate symbol) and with constant symbols from Φ. Now, the set of possible worlds $\mathcal{W}_\mathcal{P}$ is the set of all subsets of $HB_\mathcal{P}$. While this reduction is actually not worth being mentioned, it gives a first idea of a general principle that we apply in the sequel: we reduce the number of possible worlds by exploiting more and more information that is encoded in the structure of the probabilistic logic program and the probabilistic query.

The next reduction is motivated by an example. Let us consider the o-ground probabilistic query $\exists(p(a)|\top)[x_1,x_2]$ to the probabilistic logic program $\mathcal{P} = \{(p(a) \wedge q(b) \wedge r(c)|\top)[.3,.7]\}$ (a, b, and c are constant symbols). Surely, we do not need all the 512 possible worlds in $\mathcal{W}_\mathcal{P}$ to compute the requested tight answer, which is easily given by $\eta = \{x_1/.3, x_2/1\}$. We actually need only three possible worlds! The basic idea is to exploit the structure of all ground instances of conjunctive formulas from $\mathcal{P}$ and $\exists(F|E)[x_1,x_2]$.

We define $\mathcal{C}_\mathcal{P}$ as the least set with $E, E \wedge F \in \mathcal{C}_\mathcal{P}$ and with $B, H \wedge B \in \mathcal{C}_\mathcal{P}$ for all ground instances $(H|B)[c_1,c_2]$ of probabilistic program clauses in $\mathcal{P}$. We define a mapping R that assigns to each ground conjunctive formula C a corresponding subset of $HB_\mathcal{P} \cup \{\bot\}$ as follows. If $C \neq \bot$, then $R(C)$ is the set of all ground atomic formulas that occur in C. If $C = \bot$, then $R(C)$ is $HB_\mathcal{P} \cup \{\bot\}$.

We can now specify a reduced set of possible worlds (this reduction is correct with respect to computing the requested tight answer, as we prove in [18]): the *reduced* set of possible worlds $\mathcal{W}^{red}_\mathcal{P} \subseteq \mathcal{W}_\mathcal{P} \cup \{HB_\mathcal{P} \cup \{\bot\}\}$ is the least set with $R(\mathcal{C}_\mathcal{P}) \subseteq \mathcal{W}^{red}_\mathcal{P}$ and with $W_1 \cup W_2 \in \mathcal{W}^{red}_\mathcal{P}$ if $W_1, W_2 \in \mathcal{W}^{red}_\mathcal{P}$ and $W \in R(\mathcal{C}_\mathcal{P})$ such that $W_1, W_2 \supset W$ and W is minimal in $R(\mathcal{C}_\mathcal{P})$ with respect to set inclusion. In our example, we have $\mathcal{W}^{red}_\mathcal{P} = R(\mathcal{C}_\mathcal{P}) = \{\emptyset, \{p(a)\}, \{p(a), q(b), r(c)\}\}$.

By results from [18], we can now formulate the following two linear programs in (2) with $y_W \geq 0$ for all $W \in \mathcal{W}^{red}_\mathcal{P} \cup \{HB_\mathcal{P} \cup \{\bot\}\}$ and $\text{opt} \in \{\min,\max\}$. Again, if $\mathcal{P} \not\models (E|\top)[0,0]$, then the two linear programs are solvable and the requested tightest bounds are given by their two optimal values. If $\mathcal{P} \models (E|\top)[0,0]$, then the two linear programs are not solvable.

$$\text{opt} \sum_{W \in \mathcal{W}^{red}_\mathcal{P},\, W \models E \wedge F} y_W \ \text{ subject to } \mathcal{LC}^{red}_\mathcal{P}, \tag{2}$$

where $\mathcal{LC}^{red}_\mathcal{P}$ is the least set of linear constraints that contains:

- $\sum_{W \in \mathcal{W}^{red}_\mathcal{P},\, W \models E} y_W \ = \ 1$ and $y_{HB_\mathcal{P} \cup \{\bot\}} \ = \ 0$
- $c_1 \cdot \sum_{W \in \mathcal{W}^{red}_\mathcal{P},\, W \models B} y_W \ \leq \ \sum_{W \in \mathcal{W}^{red}_\mathcal{P},\, W \models H \wedge B} y_W$ and $c_2 \cdot \sum_{W \in \mathcal{W}^{red}_\mathcal{P},\, W \models B} y_W \ \geq \ \sum_{W \in \mathcal{W}^{red}_\mathcal{P},\, W \models H \wedge B} y_W$ for all ground instances $(H|B)[c_1,c_2]$ of probabilistic clauses in $\mathcal{P}$.

Another example motivates the next reduction. Let us consider the o-ground probabilistic query $\exists(p(a) \wedge q(b) \wedge r(c)|\top)[x_1,x_2]$ to the probabilistic logic program $\mathcal{P} = \{(p(a)|\top)[.3,.7], (q(b)|\top)[1,1], (r(c)|\top)[1,1]\}$ (a, b, and c are constant symbols). In this example, $\mathcal{W}^{red}_\mathcal{P}$ still consists of 8 possible worlds, while $\mathcal{W}_\mathcal{P}$ even consists of 512 possible worlds. However, we actually need only two possible worlds to compute the requested tight answer $\eta = \{x_1/.3, x_2/.7\}$.

The main idea is to partition $\mathcal{P}$ into a set of logical program clauses $\mathcal{L}$ and a set of purely probabilistic program clauses $\mathcal{P} \setminus \mathcal{L}$. In the deduction process, we then replace ground conjunctive formulas by equivalence classes of ground conjunctive formulas, since there is a canonic equivalence relation defined by $C \sim D$ iff $\mathcal{L} \models (C|D)[1,1]$ and $\mathcal{L} \models (D|C)[1,1]$ for all ground conjunctive formulas C and D. Now, in general, the number of different equivalence classes is much lower than the number of different ground conjunctive formulas. Moreover, the ground instances of conjunctive formulas from $\mathcal{L}$ do not have to be considered anymore in constructing the reduced set of possible worlds. Finally, the logical program clauses in $\mathcal{L}$ do not have to be represented anymore by linear constraints in (2).

We define the monotonic operator $T_\mathcal{L}\!\uparrow\!\omega$ on the set of all subsets of $HB_\mathcal{P} \cup \{\bot\}$ in the classical way. For this task, we need the monotonic *immediate consequence operator* $T_\mathcal{L}$ on the set of all subsets of $HB_\mathcal{P} \cup \{\bot\}$. For all $I \subseteq HB_\mathcal{P} \cup \{\bot\}$:

$$T_\mathcal{L}(I) \ = \ \bigcup\{R(H) \mid H \leftarrow B \text{ is a ground instance of a logical program clause in } \mathcal{L} \text{ with } R(B) \subseteq I\}.$$

For all $I \subseteq HB_\mathcal{P} \cup \{\bot\}$, we define $T_\mathcal{L}\uparrow\omega(I)$ as the union of all $T_\mathcal{L}\!\uparrow\! n(I)$ with $n < \omega$, where $T_\mathcal{L}\!\uparrow\!0(I) = I$ and $T_\mathcal{L}\!\uparrow\!(n+1)(I) = T_\mathcal{L}(T_\mathcal{L}\!\uparrow\! n(I))$ for all $n < \omega$.

Since for a ground conjunctive formula C, a unique representative of the equivalence class $[C]_\sim$ is given by $T_\mathcal{L}\uparrow\omega(R(C))$, we can now specify a new reduced set of possible worlds (a proof for the correctness of this reduction is given in [18]): the *reduced* set of possible worlds $\mathcal{W}^{red}_{\mathcal{L},\mathcal{P}} \subseteq \mathcal{W}_\mathcal{P} \cup \{HB_\mathcal{P} \cup \{\bot\}\}$ is the least set with $T_\mathcal{L}\!\uparrow\!\omega(R(\mathcal{C}_{\mathcal{P}\setminus\mathcal{L}})) \subseteq \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}$ and with $T_\mathcal{L}\!\uparrow\!\omega(W_1 \cup W_2) \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}$ if $W_1, W_2 \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}$ and $W \in T_\mathcal{L}\!\uparrow\!\omega(R(\mathcal{C}_{\mathcal{P}\setminus\mathcal{L}}))$ such that $W_1, W_2 \supset W$ and W is minimal in $T_\mathcal{L}\uparrow\omega(R(\mathcal{C}_{\mathcal{P}\setminus\mathcal{L}}))$ with respect to set inclusion. In our example, we have $\mathcal{W}^{red}_{\mathcal{L},\mathcal{P}} = T_\mathcal{L}\uparrow\omega(R(\mathcal{C}_{\mathcal{P}\setminus\mathcal{L}})) = \{\{q(b), r(c)\}, \{q(b), r(c), p(a)\}\}$.

By results from [18], we can formulate the following two linear programs in (3) with $y_W \geq 0$ for all $W \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}} \cup \{HB_\mathcal{P} \cup \{\bot\}\}$ and $\text{opt} \in \{\min,\max\}$. Again, if $\mathcal{P} \not\models (E|\top)[0,0]$, then the two linear programs are solvable and the requested tightest bounds are given by their two optimal values. If $\mathcal{P} \models (E|\top)[0,0]$, then the two linear programs are not solvable.

$$\text{opt} \sum_{W \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}},\, W \models E \wedge F} y_W \ \text{ subject to } \mathcal{LC}^{red}_{\mathcal{L},\mathcal{P}}, \tag{3}$$

where $\mathcal{LC}^{red}_{\mathcal{L},\mathcal{P}}$ is the least set of linear constraints that contains:

- $\sum_{W \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}},\, W \models E} y_W = 1$ and $y_{HB_{\mathcal{P}} \cup \{\bot\}} = 0$
- $c_1 \cdot \sum_{W \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}},\, W \models B} y_W \leq \sum_{W \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}},\, W \models H \wedge B} y_W$ and $c_2 \cdot \sum_{W \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}},\, W \models B} y_W \geq \sum_{W \in \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}},\, W \models H \wedge B} y_W$ for all ground instances $(H|B)[c_1, c_2]$ of probabilistic clauses in $\mathcal{P} \setminus \mathcal{L}$.

6 DISCUSSION

In the previous section, we showed that the tight answer for an o-ground probabilistic query to a probabilistic logic program $\mathcal{P}$ can be computed by solving two linear programs. However, the straightforward linear programs have a number of variables that is equal to the number of possible worlds in $\mathcal{W}_{\mathcal{P}}$, which is exponential in the cardinality of the Herbrand base $HB_{\mathcal{P}}$. We then presented a technique to generate linear programs that generally have a much lower number of variables. For this purpose, $\mathcal{P}$ is partitioned into the set $\mathcal{L}$ of logical program clauses and the set $\mathcal{P} \setminus \mathcal{L}$ of purely probabilistic program clauses. The variables of the new linear programs are given by the possible worlds in $\mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}$, which is constructed by a closure operation from the set of equivalence classes $T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))$.

In the best case, the closure operation does not produce any new possible worlds and $|\mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}|$ is equal to $|T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))|$.

The average case is quite hard to analyze. However, for small $T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))$ and large $\mathcal{W}_{\mathcal{P}}$, the closure operation is very unlikely to produce new possible worlds. Consequently, at least for small $T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))$, the average case should be close to the best case (this argumentation just gives an intuitive idea; a more detailed theoretical analysis must follow in future work).

In [18], we identified important deduction problems in probabilistic propositional logics in which the presented technique works very well. So, the problem of deduction along a bidirectional chain of basic events (see, for example, [8]) results in a quadratic number of variables in the length of the chain (the classical linear programs have an exponential number of variables in the length of the chain). Moreover, probabilistic deduction in hierarchies of basic events and of disjoint basic events also produces a very low number of variables.

In the worst case, $|\mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}|$ is exponential in $|T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))|$. This can easily be proved by taking a probabilistic logic program that contains a purely probabilistic fact $(H|\top)[c_1, c_2]$ for each ground atomic formula H. In this case, we even get $\mathcal{W}_{\mathcal{P}} = \mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}$. However, while $|\mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}|$ has always the upper bound $|\mathcal{W}_{\mathcal{P}}|$, $|\mathcal{W}^{red}_{\mathcal{L},\mathcal{P}}|$ is in the worst case exponential in $|T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))|$ and not in $|HB_{\mathcal{P}}|$.

In summary, our technique works very well if $T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))$ is small (that is, $\mathcal{P}$ contains few purely probabilistic program clauses) or if $T_{\mathcal{L}} \uparrow \omega(R(\mathcal{C}_{\mathcal{P} \setminus \mathcal{L}}))$ contains many minimal members with respect to set inclusion.

7 SUMMARY AND OUTLOOK

We presented a new approach to probabilistic logic programs with a possible worlds semantics. We showed that probabilistic deduction in our framework is computationally more complex than deduction in classical logic programs. We then elaborated a linear programming approach to probabilistic deduction that is efficient in interesting special cases. In the best case, the generated linear programs have a number of variables that is linear in the number of ground instances of purely probabilistic clauses in a probabilistic logic program.

An interesting topic of future research is how to extend our approach to probabilistic clauses that also contain negations or even disjunctions of atomic formulas.

ACKNOWLEDGEMENTS

I am very grateful to Thomas Eiter for his useful comments and his support. I also want to thank the referees for their useful comments.

REFERENCES

[1] K. R. Apt, ‘Logic programming’, in *Handbook of Theoretical Computer Science*, ed., J. van Leeuwen, volume B, chapter 10, 493–574, The MIT Press, (1990).

[2] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities*, The MIT Press, 1990.

[3] F. Bacchus, A. Grove, J. Y. Halpern, and D. Koller, ‘From statistical knowledge bases to degrees of beliefs’, *Artificial Intelligence*, **87**, 75–143, (1996).

[4] R. Carnap, *Logical Foundations of Probability*, University of Chicago Press, Chicago, 1950.

[5] A. Charnes and W. W. Cooper, ‘Programming with linear fractional functionals’, *Naval Research Logistics Quaterly*, **9**, 181–186, (1962).

[6] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, ‘Complexity and expressive power of logic programming’, in *Proc. of the 12th Annual IEEE Conference on Computational Complexity*, pp. 82–101, (1997).

[7] B. de Finetti, *Theory of Probability*, Wiley, 1974.

[8] D. Dubois, H. Prade, and J.-M. Touscas, ‘Inference with imprecise numerical quantifiers’, in *Intelligent Systems*, eds., Z. W. Ras and M. Zemankova, chapter 3, 53–72, Ellis Horwood, (1990).

[9] H. B. Enderton, *A Mathematical Introduction to Logic*, Academic Press, Inc., 1972.

[10] R. Fagin, J. Y. Halpern, and N. Megiddo, ‘A logic for reasoning about probabilities’, *Information and Computation*, **87**, 78–128, (1990).

[11] A. M. Frisch and P. Haddawy, ‘Anytime deduction for probabilistic logic’, *Artificial Intelligence*, **69**, 93–122, (1994).

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman & Co., 1979.

[13] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou, ‘Probabilistic satisfiability’, *Journal of Complexity*, **4**(1), 1–11, (1988).

[14] J. Y. Halpern, ‘An analysis of first-order logics of probability’, *Artifical Intelligence*, **46**, 311–350, (1990).

[15] L. V. S. Lakshmanan and F. Sadri, ‘Probabilistic deductive databases’, in *Proc. Internat. Logic Programming Symp.*, pp. 254–268, (1994).

[16] J. W. Lloyd, *Foundations of Logic Programming*, Springer, Berlin, 2nd edn., 1987.

[17] T. Lukasiewicz, *Precision of Probabilistic Deduction under Taxonomic Knowledge*, Doctoral Dissertation, Universität Augsburg, 1996.

[18] T. Lukasiewicz, ‘Efficient global probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events’, in *Proc. of the 6th International Conference on Information and Knowledge Management*, pp. 75–82. ACM Press, (1997).

[19] T. Lukasiewicz, ‘Magic inference rules for probabilistic deduction under taxonomic knowledge’, in *Proc. of the 14th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, (1998).

[20] T. Lukasiewicz, ‘Probabilistic deduction with conditional constraints over basic events’, in *Principles of Knowledge Representation and Reasoning: Proc. of the 6th International Conference*. Morgan Kaufmann Publishers, (1998).

[21] R. T. Ng, ‘Semantics, consistency, and query processing of empirical deductive databases’, *IEEE Transactions on Knowledge and Data Engineering*, **9**(1), 32–49, (1997).

[22] R. T. Ng and V. S. Subrahmanian, ‘A semantical framework for supporting subjective and conditional probabilities in deductive databases’, *Journal of Automated Reasoning*, **10**(2), 191–235, (1993).

[23] N. J. Nilsson, ‘Probabilistic logic’, *Artif. Intell.*, **28**, 71–88, (1986).

[24] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, 1988.

[25] D. Poole, ‘Probabilistic Horn abduction and Bayesian networks’, *Artificial Intelligence*, **64**, 81–129, (1993).

[26] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley, 1986.