INFSYS

RESEARCH

REPORT

INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG WISSENSBASIERTE SYSTEME

PROBABILISTIC LOGIC PROGRAMMING WITH CONDITIONAL CONSTRAINTS

Thomas Lukasiewicz

INFSYS RESEARCH REPORT 1843-00-01

FEBRUARY & MAY 2000

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-00-01, FEBRUARY & MAY 2000

PROBABILISTIC LOGIC PROGRAMMING WITH CONDITIONAL CONSTRAINTS

REVISED VERSION, MAY 2000

Thomas Lukasiewicz [1]

Abstract. We introduce a new approach to probabilistic logic programming in which probabilities are defined over a set of possible worlds. More precisely, classical program clauses are extended by a subinterval of $[0, 1]$ that describes a range for the conditional probability of the head of a clause given its body. We then analyze the complexity of selected probabilistic logic programming tasks. It turns out that probabilistic logic programming is computationally more complex than classical logic programming. More precisely, the tractability of special cases of classical logic programming generally does not carry over to the corresponding special cases of probabilistic logic programming. Moreover, we also draw a precise picture of the complexity of deciding and computing tight logical consequences in probabilistic reasoning with conditional constraints in general. We then present linear optimization techniques for deciding satisfiability and computing tight logical consequences of probabilistic logic programs. These techniques are efficient in the special case in which we have few relevant purely probabilistic knowledge. We finally show that probabilistic logic programming under certain syntactic and semantic restrictions is closely related to van Emden's quantitative deduction, and thus has computational properties similar to classical logic programming. Based on this result, we present an efficient approximation technique for probabilistic logic programming.

[1] Institut und Ludwig Wittgenstein Labor für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: `lukasiewicz@kr.tuwien.ac.at`.

Acknowledgements: I am very grateful to Thomas Eiter, Georg Gottlob, Nicola Leone, and Cristinel Mateis for useful discussions. Many thanks also to Gabriele Kern-Isberner for useful comments on an earlier version of this paper. This work was supported by a DFG grant and the Austrian Science Fund Project N Z29-INF.

This paper is a substantially extended and revised version of a paper that appeared in *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 388-392, 1998 [42].

1 Introduction

During the recent decades, the management of uncertainty has come to play an important role in knowledge representation and reasoning. Many different formalisms and methodologies have been proposed for handling uncertain knowledge. Most of them are directly or indirectly based on probability theory.

Probabilistic propositional logics and their various dialects have been thoroughly studied in the literature (cf. especially the work by Nilsson [56], Frisch and Haddawy [20], and Fagin et al. [18]). Their extensions to probabilistic first-order logics can be classified into first-order logics in which probabilities are defined over the domain and those in which probabilities are given over a set of possible worlds (cf. especially the work by Halpern [29] and Bacchus et al. [4]). The first ones are suitable for describing statistical knowledge, while the latter are appropriate for representing subjective knowledge (also called degrees of belief).

One important element of many formal languages for representing probabilistic knowledge are interval restrictions for conditional probabilities, also called *conditional constraints* [47]. In particular, the literature contains extensive work on probabilistic reasoning about propositional conditional constraints (cf. especially the work by Dubois and Prade's group [17, 1, 16], Frisch and Haddawy [20], and the author [43, 47]).

As an important formalism for reasoning with classical knowledge, *logic programming* [39, 3] started in the early 1970's [36] based on earlier work in automated theorem proving, and began to flourish especially with the spreading of PROLOG. Logic programming is now a well-established knowledge representation and reasoning formalism in artificial intelligence and deductive databases.

Many applications in practice make it indispensable for a knowledge-representation and reasoning system to deal with uncertain knowledge. The need for representing uncertainty in the logic programming framework is already reported by a great number of publications on many-valued logic programming [67, 5, 19, 7], logic programming with signed formulas [40, 27], generalized annotated logic programming [35, 38], and probabilistic logic programming (see references below).

In this paper, we elaborate an approach to *probabilistic logic programming with conditional constraints*, which can be understood both as a probabilistic generalization of classical logic programming and as a first-order generalization of probabilistic reasoning with propositional conditional constraints. More precisely, we extend classical program clauses by a subinterval of $[0, 1]$ that describes a range for the conditional probability of the head of a clause given its body. Moreover, our probabilistic logic programs have a subjective semantics in which probabilities are defined over a set of possible worlds.

In this paper, we focus especially on the computational aspects of probabilistic logic programming with conditional constraints. We start by analyzing the computational complexity of some representative decision and optimization problems in probabilistic reasoning with conditional constraints. In particular, we draw a precise picture of the complexity of selected probabilistic logic programming problems. Our complexity results show that probabilistic logic programming is computationally more complex than classical logic programming. In detail, the tractability of special cases of classical logic programming generally does not carry over to the corresponding special cases of probabilistic logic programming.

We then elaborate techniques for deciding the satisfiability and computing tight logical consequences of probabilistic logic programs. More precisely, these problems are reduced to linear optimization problems, which have a small size when we have few relevant purely probabilistic knowledge. That is, rather than generalizing the fixpoint characterization and proof theory of classical logic programming, we use linear optimization techniques in order to perform probabilistic logic programming.

Especially the complexity results show that probabilistic logic programming differs significantly in its computational properties from classical logic programming. Why this big difference?

It turns out that the probabilistic semantics in its full generality is responsible for this computational difference between classical and probabilistic logic programming. That is, we just have to focus on more specific probabilistic interpretations in order to bring probabilistic logic programming closer to classical logic programming. More precisely, we just have to increase the axioms of probability by an additional axiom that brings probabilistic logics closer to truth-functional logics [45, 46, 44].

We will see that under this more restricted probabilistic semantics, *ordinary probabilistic logic programs* (in which probabilistic program clauses have only atoms in their heads and are assigned only probability intervals of the form $[c, 1]$) have similar computational properties as classical logic programs. More precisely, ordinary probabilistic logic programming under this semantics has a special-case tractability similar to classical logic programming. Moreover, it has a model and fixpoint characterization and a proof theory similar to classical logic programming. Interestingly, this fixpoint characterization is well-known in the literature as van Emden's *quantitative deduction* [67], which is an important approach to *truth-functional many-valued logic programming*. That is, this paper reveals a deep relationship between probabilistic logic programming, on the one hand, and van Emden's approach to truth-functional many-valued logic programming, on the other hand. This relationship is surprising, if we look at the following comments on van Emden's work:

- Ng and Subrahmanian [52] conclude that "van Emden [67] develops a quantitative logic programming language in which multiplication is used to assign truth values to conjunctions. Of course, probabilities can be multiplied only if the events are independent, and hence van Emden's framework is also *non-probabilistic*".
- Poole [62] writes that "there have been other attempts to incorporate uncertainty into logic programming. These have typically *not* considered *probability*, but other uncertainty calculi such as certainty factors" and points to van Emden's quantitative deduction [67].

In addition to tying together probabilistic and truth-functional many-valued logic programming, this discovered relationship also leads us to another important result. More precisely, we will see that van Emden's quantitative deduction can be used as an *efficient approximation* of probabilistic logic programming.

The literature contains several related approaches to probabilistic logic programming, which can be roughly divided into those that were developed as an extension of classical and annotated logic programming [52, 53, 54, 37, 51, 13, 14] and those that evolved from Bayesian networks [62, 61, 26, 55, 30, 31].

The work closest in spirit to the present paper is perhaps the one by Ng and Subrahmanian [53], which describes an annotation-based approach to subjective probabilistic logic programming with fixpoint theory and proof procedure. There are, however, several crucial differences. First, we present a full-fledged approach to probabilistic logic programming with uncertainty on the rule level, while the annotation-based approach in [53] is perhaps best described as classical logic programming about probabilities of conjunctions and disjunctions of atoms. Second, our work is directed towards efficient algorithms, while [53] has a more conceptual and theoretical nature, since it is based on solving an exponential number of linear programs over an exponential number of variables (where both numbers are exponential in the cardinality of the Herbrand base) in each fixpoint iteration step and in each compilation step for SLDp-refutation.

Another closely related work is the approach to statistical deductive databases by Ng [51]. More precisely, Ng concentrates especially on the model-theoretic semantics of such databases and on algorithms for consistency checking and query processing. However, as Ng's work is about the statistical setting, it assumes that probabilities are defined over the domain. Our work, in contrast, is a subjective one in which probabilities are defined over a set of possible worlds. That is, it can be considered orthogonal to Ng's work.

The work on probabilistic logic programming that evolved from Bayesian networks [60] is less closely related. In detail, Poole's work [62, 61] describes an approach to Horn clause abduction in which probabilities are associated with hypotheses. It is implemented by a generalization of SLD resolution. In contrast to our work, Poole's approach does not allow for uncertainty on the rule level. Haddawy and his group [26, 55] describe an approach to query processing in first-order probabilistic knowledge bases by Bayesian network construction and inference. Jaeger's work [30, 31] goes in a similar direction.

Since these approaches originated from Bayesian networks, they all assume some strong structural restrictions on probabilistic knowledge bases. In particular, they all require that the grounding of a knowledge base is acyclic. Moreover, conditional probabilities are always given by a precise point value, rather than by an interval. Our work, in contrast, is free from such strong restrictions. Furthermore, we do not assume any implicit conditional independencies or explicit combining rules. Finally, rather than being guided by Bayesian network techniques, our approach is closer to classical logic programming.

The main contributions of this paper can be briefly summarized as follows:

- We present a new approach to probabilistic logic programming in which probabilities are defined over a set of possible worlds and in which classical program clauses are extended by a subinterval of $[0, 1]$ that describes a range for the conditional probability of the head of a clause given its body.
- We analyze the computational complexity of some representative decision and optimization problems in probabilistic reasoning with conditional constraints. In particular, we draw a precise picture of the complexity of selected probabilistic logic programming problems. It turns out that probabilistic logic programming is computationally more complex than classical logic programming. In detail, the tractability of special cases of classical logic programming generally does not carry over to the corresponding special cases of probabilistic logic programming. Moreover, we sharply describe the complexity of deciding and computing tight logical consequences in probabilistic reasoning with conditional constraints in general. To our knowledge, these results are unknown in the literature so far.
- We reduce the problem of deciding whether a probabilistic logic program P is satisfiable to the problem of deciding whether a system of linear constraints is solvable. The generated systems of linear constraints have a small size when P contains few relevant purely probabilistic knowledge.
- Similarly, we reduce the problem of computing tight logical consequences of a probabilistic logic program to the problem of solving two linear programs. Again, the generated linear programs have a small size when P contains few relevant purely probabilistic knowledge.
- We show that ordinary probabilistic logic programming under a more restricted probabilistic semantics as presented in [45, 46, 44] is characterized by van Emden's quantitative deduction. It thus has the same computational properties as classical logic programming.
- We present an efficient approximation technique for probabilistic logic programming, which uses van Emden's quantitative deduction to compute a candidate for a model of a probabilistic logic program. In the ordinary case, this method approximates tight logical consequences through sharper intervals.

The rest of this paper is organized as follows. Sections 2 and 3 provide the technical background and a motivating example, respectively. In Section 4, we analyze the computational complexity of probabilistic reasoning with conditional constraints. Section 5 describes the main ideas behind the reasoning techniques introduced in this paper. In Sections 6–8, we present our techniques for deciding satisfiability and computing tight logical consequences of probabilistic logic programs. Section 9 concentrates on the relationship to van Emden's quantitative deduction and presents our efficient approximation technique for probabilistic logic programming. In Section 10, we give some general comments on the model-theoretic notion of logical consequence. Section 11 finally summarizes the main results and gives some concluding remarks.

Note that some longer proofs and technical details have been moved to Appendices A–D.

2 Preliminaries

In this section, we give some technical preliminaries. In detail, we first describe the probabilistic background of this work. We then introduce probabilistic knowledge bases and their restriction to probabilistic logic programs. We finally give a formal definition of the main problems analyzed in this work.

2.1 Probabilistic Background

We now briefly describe how first-order logics of probability are given a semantics in which probabilities are defined over a set of possible worlds (see especially the work by Carnap [8], Gaifman [21], Scott and Krauss [65], and Halpern [29]). We restrict our considerations to a function-free language of first-order conditional constraints that are implicitly universally quantified and that are interpreted by probabilities over a set of Herbrand interpretations.

Let Φ be a first-order vocabulary that contains a finite set of predicate symbols and a finite set of constant symbols (that is, we do not consider function symbols in this framework). Let $\mathcal{X}$ be a set of *object* and *bound variables*. Object variables represent elements of a certain domain, while bound variables express real numbers in the unit interval $[0,1]$.

An *object term* is a constant symbol from Φ or an object variable from $\mathcal{X}$. We define *classical formulas* by induction as follows. If p is a predicate symbol of arity $k \geq 0$ from Φ and $t_1, \ldots, t_k$ are object terms, then $p(t_1, \ldots, t_k)$ is a classical formula (called *atom*). If F and G are classical formulas, then $\neg F$ and $(F \wedge G)$ are classical formulas. We use $(F \vee G)$ and $(F \leftarrow G)$ to abbreviate $\neg(\neg F \wedge \neg G)$ and $\neg(\neg F \wedge G)$, respectively. We use $\bot$ and $\top$ to abbreviate $(F \wedge \neg F)$ and $\neg(F \wedge \neg F)$, respectively. A *conditional constraint* is an expression $(H|B)[c_1, c_2]$ with real numbers $c_1, c_2 \in [0,1]$ and classical formulas H and B. We call H its *consequent* (or also *head*) and B its *antecedent* (or also *body*).

We adopt the usual conventions to eliminate parentheses. Object terms, classical formulas, and conditional constraints are *ground* iff they do not contain any variables. The notions of substitutions, ground substitutions, and ground instances of conditional constraints are canonically defined.

We distinguish between classical and purely probabilistic conditional constraints. *Classical conditional constraints* are of the kind $(H|B)[1,1]$ or $(H|B)[0,0]$, while *purely probabilistic conditional constraints* are of the form $(H|B)[c_1, c_2]$ with $c_1 < 1$ and $c_2 > 0$. We identify the classical conditional constraints $(H|B)[1,1]$ and $(H|B)[0,0]$ with the classical formulas $H \leftarrow B$ and $\bot \leftarrow H \wedge B$, respectively.

We use HB_Φ (resp., HU_Φ) to denote the Herbrand base (resp., Herbrand universe) over Φ. In the sequel, we assume that HB_Φ is nonempty. A *possible world* I is a subset of HB_Φ. We use $\mathcal{I}_\Phi$ to denote the set of all possible worlds over Φ. A *variable assignment* σ maps each object variable to an element of HU_Φ, and each bound variable to a real number from $[0,1]$. It is extended to object terms by defining $\sigma(c) = c$ for all constant symbols c from Φ. The *truth* of classical formulas F in I under σ, denoted $I \models_\sigma F$, is inductively defined as follows (we write $I \models F$ when F is ground):

- $I \models_\sigma p(t_1, \ldots, t_k)$ iff $p(\sigma(t_1), \ldots, \sigma(t_k)) \in I$.
- $I \models_\sigma \neg F$ iff not $I \models_\sigma F$, and $I \models_\sigma (F \wedge G)$ iff $I \models_\sigma F$ and $I \models_\sigma G$.

A classical formula F *is true* in I, or I is a *model* of F iff $I \models_\sigma F$ for all variable assignments σ. A possible world I is a *model* of a set of classical formulas $\mathcal{F}$ iff I is a model of all $F \in \mathcal{F}$. A set of classical formulas $\mathcal{F}$ is *satisfiable* iff a model of $\mathcal{F}$ exists. A classical formula F is a *logical consequence* of $\mathcal{F}$, denoted $\mathcal{F} \models F$, iff each model of $\mathcal{F}$ is also a model of F.

A *probabilistic interpretation* Pr is a probability function on $\mathcal{I}_\Phi$ (that is, since $\mathcal{I}_\Phi$ is finite, simply a mapping from $\mathcal{I}_\Phi$ to the unit interval $[0,1]$ such that all $Pr(I)$ with $I \in \mathcal{I}_\Phi$ sum up to 1). The *probability* of a classical formula F in the probabilistic interpretation Pr under a variable assignment σ, denoted $Pr_\sigma(F)$, is defined as follows (we write $Pr(F)$ when F is ground):

$$Pr_\sigma(F) \quad = \sum_{I \in \mathcal{I}_\Phi,\, I \models_\sigma F} Pr(I)\,.$$

For classical formulas H and B with $Pr_\sigma(B) > 0$, we use $Pr_\sigma(H|B)$ to denote $Pr_\sigma(H \wedge B) \,/\, Pr_\sigma(B)$. A conditional constraint $(H|B)[c_1,c_2]$ *is true* in Pr under σ, denoted $Pr \models_\sigma (H|B)[c_1,c_2]$, iff

$$Pr_\sigma(B) = 0 \;\text{ or }\; Pr_\sigma(H|B) \in [c_1,c_2]\,.$$

A conditional constraint F *is true* in Pr, or Pr is a *model* of F, denoted $Pr \models F$, iff $Pr \models_\sigma F$ for all variable assignments σ. A probabilistic interpretation Pr is a *model* of a set of conditional constraints $\mathcal{F}$, denoted $Pr \models \mathcal{F}$, iff Pr is a model of all $F \in \mathcal{F}$. A set of conditional constraints $\mathcal{F}$ is *satisfiable* iff a model of $\mathcal{F}$ exists. A conditional constraint F is a *logical consequence* of a set of conditional constraints $\mathcal{F}$, denoted $\mathcal{F} \models F$, iff each model of $\mathcal{F}$ is also a model of F.

A conditional constraint $(H|B)[c_1,c_2]$ is a *tight logical consequence* of $\mathcal{F}$, denoted $\mathcal{F} \models_{\mathit{tight}} (H|B)[c_1,c_2]$, iff c_1 (resp., c_2) is the infimum (resp., supremum) of $Pr_\sigma(H|B)$ subject to all models Pr of $\mathcal{F}$ and all variable assignments σ such that $Pr_\sigma(B) > 0$ (note that we canonically define $c_1 = 1$ and $c_2 = 0$ when $Pr_\sigma(B) = 0$ for all models Pr of $\mathcal{F}$ and all variable assignments σ).

2.2 Probabilistic Knowledge Bases

A *probabilistic knowledge base* KB is a finite set of conditional constraints $(H|B)[c_1,c_2]$ with rational numbers $c_1, c_2 \in [0,1]$ such that $c_1 \leq c_2$. We use $\mathit{ground}(KB)$ to denote the set of all ground instances of conditional constraints in KB. A *probabilistic query* is an expression $\exists(H|B)[t_1,t_2]$, where H and B are two classical formulas, and t_1 and t_2 are either two real numbers from $[0,1]$ or two distinct variables from $\mathcal{X}$. A probabilistic query $\exists(H|B)[t_1,t_2]$ is *object-ground* iff H and B are ground and $t_1, t_2 \in \mathcal{X}$. A *classical knowledge base* is a finite set of classical conditional constraints.

Given a probabilistic query $\exists(H|B)[c_1,c_2]$ with $c_1, c_2 \in [0,1]$ to a probabilistic knowledge base KB, its *correct answer substitutions* are substitutions θ such that $KB \models (H\theta|B\theta)[c_1,c_2]$ and that θ acts only on variables in $\exists(H|B)[c_1,c_2]$. Its *correct answer* is Yes if such a θ exists and No otherwise.

Given a probabilistic query $\exists(H|B)[x_1,x_2]$ with $x_1, x_2 \in \mathcal{X}$ to a probabilistic knowledge base KB, its *tight answer substitutions* are substitutions θ such that $KB \models_{\mathit{tight}} (H\theta|B\theta)[x_1\theta, x_2\theta]$, that θ acts only on variables in $\exists(H|B)[x_1,x_2]$, and that $x_1\theta, x_2\theta \in [0,1]$. Note that for probabilistic queries $\exists(H|B)[x_1,x_2]$ with $x_1, x_2 \in \mathcal{X}$, there always exist tight answer substitutions (in particular, object-ground probabilistic queries always have a unique tight answer substitution).

2.3 Probabilistic Logic Programs

A conditional constraint $(H|B)[c_1,c_2]$ is *conjunctive* (resp., *1-conjunctive*) iff H is a conjunction of atoms (resp., an atom) and B is either $\top$ or a conjunction of atoms. A conditional constraint $(H|B)[c_1,c_2]$ is *atomic* iff H is an atom and B is either $\top$ or an atom. A conditional constraint $(H|B)[c_1,c_2]$ is *ordinary* iff H is an atom, B is either $\top$ or a conjunction of atoms, and $c_2 = 1$.

A probabilistic knowledge base KB is *conjunctive* (resp., *1-conjunctive*, *atomic*, *ordinary*) iff all members of KB are conjunctive (resp., 1-conjunctive, atomic, ordinary). A *probabilistic query* $\exists(H|B)[c_1, c_2]$ with $c_1, c_2 \in [0,1]$ is *conjunctive* (resp., *1-conjunctive*, *atomic*, *ordinary*) iff $(H|B)[c_1, c_2]$ is conjunctive (resp., 1-conjunctive, atomic, ordinary). A *probabilistic query* $\exists(H|B)[x_1, x_2]$ with $x_1, x_2 \in \mathcal{X}$ is *conjunctive* (resp., *1-conjunctive*, *atomic*, *ordinary*) iff $(H|B)[0,1]$ is conjunctive (resp., 1-conjunctive, atomic, ordinary).

A *probabilistic logic program* is a conjunctive probabilistic knowledge base. We assume that probabilistic queries to probabilistic logic programs are always conjunctive. A *1-conjunctive* (resp., *atomic*, *ordinary*) *probabilistic logic program* is a 1-conjunctive (resp., atomic, ordinary) probabilistic knowledge base. We assume that queries to such probabilistic logic program are always 1-conjunctive (resp., atomic, ordinary).

The members of probabilistic logic programs are also called *probabilistic Horn clauses*. We classify probabilistic Horn clauses into *integrity clauses* and *probabilistic program clauses*. The former are of the kind $(H|B)[0,0]$, while the latter have the form $(H|B)[c_1, c_2]$ with $c_2 > 0$. Moreover, probabilistic program clauses are divided into *probabilistic facts* and *probabilistic rules*, which are of the form $(H|\top)[c_1, c_2]$ and $(H|B)[c_1, c_2]$, respectively, where $B \neq \top$.

A *classical Horn clause* is a classical 1-conjunctive conditional constraint of the form $(H|B)[1,1]$ or $(H|B)[0,0]$. A *classical logic program* is a finite set of classical Horn clauses. A *classical program clause* is a classical Horn clause of the kind $(H|B)[1,1]$. A *classical definite logic program* is a finite set of classical program clauses. Finally, *classical facts* and *classical rules* are classical program clauses of the form $(H|\top)[1,1]$ and $(H|B)[1,1]$, respectively, where $B \neq \top$.

Like classical definite logic programs, ordinary probabilistic logic programs are always satisfiable. Probabilistic logic programs, in contrast, may be unsatisfiable, because of logical inconsistencies through integrity clauses or, more generally, for probabilistic inconsistencies in the assumed probability ranges.

2.4 Problem Statements

In this paper, the notion of *probabilistic logic programming* is associated with solving the following decision and optimization problems for conjunctive probabilistic knowledge bases KB and queries Q:

SATISFIABILITY: Given a probabilistic knowledge base KB, decide whether KB is satisfiable.

ENTAILMENT: Given a probabilistic knowledge base KB and a ground query $Q = \exists(G|A)[d_1, d_2]$ with rational numbers $d_1, d_2 \in [0,1]$, decide whether Yes is the correct answer for Q to KB.

TIGHT ENTAILMENT (D): Given a probabilistic knowledge base KB, an object-ground probabilistic query $Q = \exists(G|A)[x_1, x_2]$, and two rational numbers $d_1, d_2 \in [0,1]$, decide whether $\{x_1/d_1,\ x_2/d_2\}$ is the tight answer substitution for Q to KB.

TIGHT ENTAILMENT (OPT): Given a probabilistic knowledge base KB and an object-ground probabilistic query $Q = \exists(G|A)[x_1, x_2]$, compute the tight answer substitution for Q to KB.

As usual, in SATISFIABILITY, we identify Φ with the vocabulary of all predicate and constant symbols that occur in KB. Moreover, in ENTAILMENT, TIGHT ENTAILMENT (D), and TIGHT ENTAILMENT (OPT), we identify Φ with the vocabulary of all predicate and constant symbols that occur in KB and Q.

3 Example

In this section, we give an illustrative example. In the sequel, predicate and constant symbols begin with lower case letters, while object and bound variables start with upper case letters.

Example 3.1 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. But, 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 (*ro*) from R to S, the probability that he can reach (*re*) S from R without running into a traffic jam is at least 0.7. Given a road in the south (*so*) of the town, this probability is at least 0.9. A friend just called him and gave him advice (*ad*) about some roads without any significant traffic. Clearly, if he can reach S from T and T from R, both without running into a traffic jam, then he can also reach S from R without running into a traffic jam. This knowledge can be expressed by the following probabilistic rules:

$$\begin{aligned}
&(\mathit{re}(R,S) \mid \mathit{ro}(R,S))[0.7,1],\\
&(\mathit{re}(R,S) \mid \mathit{ro}(R,S) \wedge \mathit{so}(R,S))[0.9,1],\\
&(\mathit{re}(R,S) \mid \mathit{ro}(R,S) \wedge \mathit{ad}(R,S))[1,1],\\
&(\mathit{re}(R,S) \mid \mathit{re}(R,T) \wedge \mathit{re}(T,S))[1,1].
\end{aligned}$$

Finally, John has some concrete knowledge about the roads, the roads in the south of the town, and the roads that his friend was talking about. For example, he knows that there is a road from his home (h) to the university (u), from the university to the airport (a), and from the airport to Mary's office (o). Moreover, John believes that his friend was talking about the road from the university to the airport with a probability between 0.8 and 0.9 (as he is not completely sure about it anymore). This and some other concrete knowledge is expressed by the following probabilistic facts:

$$\begin{aligned}
&(\mathit{ro}(h,u) \mid \top)[1,1],\\
&(\mathit{ro}(u,a) \mid \top)[1,1],\\
&(\mathit{ro}(a,o) \mid \top)[1,1],\\
&(\mathit{ad}(h,u) \mid \top)[1,1],\\
&(\mathit{ad}(u,a) \mid \top)[0.8,0.9],\\
&(\mathit{so}(a,o) \mid \top)[1,1].
\end{aligned}$$

Let the probabilistic logic program P contain all the rules and facts listed above.

John is now wondering whether he can reach Mary's office from his home, such that the probability of him running into a traffic jam is at most 0.01. This can be expressed by the probabilistic query $Q_1 = \exists(\mathit{re}(h,o) \mid \top)[.99,1]$. His wondering about the probability of reaching the office, without running into a traffic jam, can be expressed by the probabilistic query $Q_2 = \exists(\mathit{re}(h,o) \mid \top)[X_1, X_2]$. Finally, his wondering about this probability, given that his friend was talking about the road from the university to the airport, can be expressed by the probabilistic query $Q_3 = \exists(\mathit{re}(h,o) \mid \mathit{ad}(u,a))[X_1, X_2]$.

The correct answer for Q_1 to P is given by No. Moreover, the tight answer substitutions for Q_2 and Q_3 to P are given by $\{X_1/0.7,\ X_2/1\}$ and $\{X_1/0.875,\ X_2/1\}$, respectively.

4 Computational Complexity

In this section, we analyze the complexity of probabilistic reasoning with conditional constraints. In particular, we analyze the complexity of some representative probabilistic logic programming problems.

4.1 Complexity Classes

We assume some basic knowledge in complexity theory. We now briefly describe the complexity classes that will occur in our results. See especially [23, 33, 57] for further background.

The class P (resp., NP) contains all decision problems that can be solved in deterministic (resp., nondeterministic) polynomial time. The class co-NP is the complementary class of NP, which has Yes- and No-instances interchanged. The class D^{P} contains all decision problems that can be expressed as a logical conjunction of a problem in NP and a problem in co-NP. The class P^{NP} contains all decision problems that can be solved in deterministic polynomial time with an oracle for NP.

The following inclusion hierarchy describes the relationship between these complexity classes (note that all inclusions are believed to be strict):

$$\mathrm{P} \subseteq \mathrm{NP}, \text{co-NP} \subseteq \mathrm{D}^{\mathrm{P}} \subseteq \mathrm{P}^{\mathrm{NP}}.$$

To classify problems that compute an output value, rather than a Yes / No-answer, function classes have been introduced. In particular, the class FP (resp., $\mathrm{FP}^{\mathrm{NP}}$) is the functional analog to P (resp., P^{NP}).

4.2 Complexity Overview

We analyze the computational complexity of SATISFIABILITY, ENTAILMENT, TIGHT ENTAILMENT (D), and TIGHT ENTAILMENT (OPT). We consider the *general case* as well as the restriction to *conjunctive* and *1-conjunctive* probabilistic knowledge bases KB and queries Q (see Figure 1).

More precisely, we analyze the *propositional complexity* as well as the *data complexity*. The former is defined as the complexity under the assumption that KB is ground, while the latter is the complexity under the assumption that KB is of the form $KB^\star \cup \mathcal{F}$, where $KB^\star$ is a fixed probabilistic knowledge base and $\mathcal{F}$ is a varying finite set of probabilistic facts. Informally, the data complexity takes into account that our knowledge is often divided into some fixed generic knowledge and some varying concrete evidence.

An overview of the complexity results is given in Tables 1–3.

Table 1 shows the propositional complexity of the analyzed decision problems. More precisely, propositional SATISFIABILITY, ENTAILMENT, and TIGHT ENTAILMENT (D) are NP-complete, co-NP-complete, and D^{P}-complete, respectively, in the general, conjunctive, and 1-conjunctive case. Here, the NP- and co-NP-completeness results for propositional SATISFIABILITY and ENTAILMENT follow easily from well-known results in the literature by Fagin et al. [18] and Georgakopoulos et al. [24]. More precisely, membership follows from Fagin et al.'s [18] result that the satisfiability problem is in NP even for the rich probabilis-

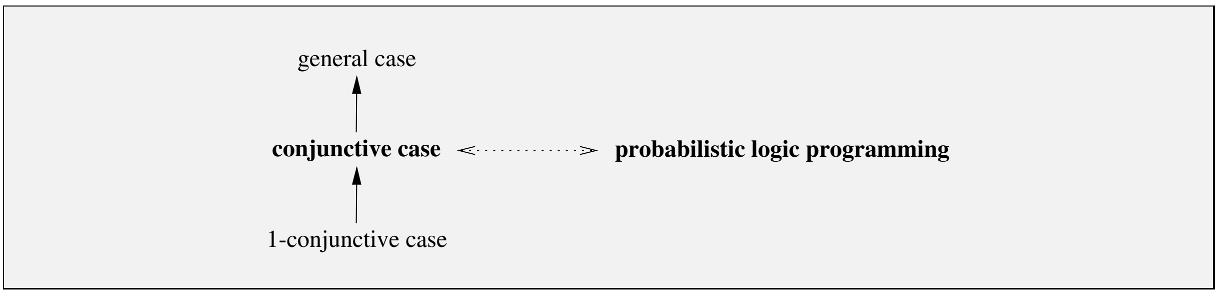

Figure 1: Hierarchy of analyzed cases

tic language of *weight formulas* (which are Boolean combinations of linear inequalities over propositional events). Moreover, hardness is immediate by Georgakopoulos et al.'s [24] proof that 2PSAT is NP-hard.

Table 2 shows the propositional complexity of TIGHT ENTAILMENT (OPT). More precisely, propositional TIGHT ENTAILMENT (OPT) is $\mathrm{FP}^{\mathrm{NP}}$-complete in the general, conjunctive, and 1-conjunctive case.

Table 3 shows the data complexity of SATISFIABILITY, ENTAILMENT, and TIGHT ENTAILMENT (D), which coincides with their propositional complexity shown in Table 1.

We remark that all hardness results shown in Table 1 hold even for the restriction to atomic probabilistic knowledge bases KB and atomic probabilistic queries of the kind $Q = \exists(G|\top)[t_1, t_2]$. Moreover, all hardness results for TIGHT ENTAILMENT (OPT), ENTAILMENT, and TIGHT ENTAILMENT (D) in Tables 2 and 3 hold for the restriction to atomic probabilistic queries of the form $Q = \exists(G|\top)[t_1, t_2]$. Finally, all hardness results shown in Table 3 hold even when $\mathcal{F}$ is a varying finite set of classical facts.

Table 1: Propositional complexity of decision problems

	general case	**conjunctive case**	1-conjunctive case
SATISFIABILITY	NP-complete	**NP-complete**	NP-complete
ENTAILMENT	co-NP-complete	**co-NP-complete**	co-NP-complete
TIGHT ENTAILMENT (D)	D^{P}-complete	**$\mathbf{D}^{\mathbf{P}}$-complete**	D^{P}-complete

Table 2: Propositional complexity of TIGHT ENTAILMENT (OPT)

	general case	**conjunctive case**	1-conjunctive case
TIGHT ENTAILMENT (OPT)	$\mathrm{FP}^{\mathrm{NP}}$-complete	**$\mathbf{FP}^{\mathbf{NP}}$-complete**	$\mathrm{FP}^{\mathrm{NP}}$-complete

Table 3: Data complexity of decision problems

	general case	**conjunctive case**	1-conjunctive case
SATISFIABILITY	NP-complete	**NP-complete**	NP-complete
ENTAILMENT	co-NP-complete	**co-NP-complete**	co-NP-complete
TIGHT ENTAILMENT (D)	D^{P}-complete	**$\mathbf{D}^{\mathbf{P}}$-complete**	D^{P}-complete

4.3 Discussion

It turns out that all analyzed problems are (presumably) intractable. In particular, the propositional and the data versions of the analyzed probabilistic logic programmming problems are (presumably) intractable.

We remark that the corresponding classical logic programming problems are well-known to be tractable. More precisely, the two problems of deciding whether a ground classical logic program with integrity clauses is satisfiable and of deciding whether a ground classical logic program logically entails a ground atom can in fact be solved in linear time [15]. Moreover, in the function-free framework, deciding whether a fixed classical logic program under a varying finite set of ground classical facts logically entails a ground atom is P-complete under logspace reductions (see especially [11] for a survey).

Hence, probabilistic logic programming is computationally more complex than classical logic programming. That is, restricted decision problems that are tractable for classical logic programs are (presumably) intractable for probabilistic logic programs. Thus, any attempt towards efficient probabilistic logic programming should be guided by looking for efficient special-case, average-case, or approximation techniques.

4.4 Membership Results

We now focus on the membership results shown in Tables 1–3. The following theorem shows that propositional SATISFIABILITY and ENTAILMENT are in NP and co-NP, respectively, in the general case.

Theorem 4.1 *a) Deciding whether a given ground probabilistic knowledge base KB is satisfiable is in* NP.

b) Given a ground probabilistic knowledge base KB and a ground probabilistic query $Q = \exists(G|A)[d_1, d_2]$ with rational numbers $d_1, d_2 \in [0, 1]$, deciding whether **Yes** *is the correct answer for Q to KB is in* co-NP.

To prove the other membership results for the general case, we need some preparation.

We now briefly describe a well-known translation of SATISFIABILITY and TIGHT ENTAILMENT (OPT) into equivalent decision and optimization problems with systems of linear constraints.

Assume first that we want to check whether a probabilistic knowledge base is satisfiable. The next theorem states the immediate observation that this problem can easily be reduced to the problem of deciding whether a system of linear constraints is solvable.

Theorem 4.2 *A probabilistic knowledge base KB is satisfiable iff the following system of linear constraints over the variables y_I $(I \in \mathcal{I}_\Phi)$ is solvable:*

$$\begin{array}{rcl} \sum\limits_{I \in \mathcal{I}_\Phi} y_I & = & 1 \\ \sum\limits_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} -c_1\, y_I + \sum\limits_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (1 - c_1)\, y_I & \geq & 0 \quad (\textit{for all } (H|B)[c_1, c_2] \in \textit{ground}(KB),\ c_1 > 0) \\ \sum\limits_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} c_2\, y_I + \sum\limits_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (c_2 - 1)\, y_I & \geq & 0 \quad (\textit{for all } (H|B)[c_1, c_2] \in \textit{ground}(KB),\ c_2 < 1) \\ y_I & \geq & 0 \quad (\textit{for all } I \in \mathcal{I}_\Phi)\,. \end{array} \tag{1}$$

Assume next that we are interested in the tight answer substitution for an object-ground probabilistic query. In this case, we immediately get two optimization problems in which a linear fractional objective function must be minimized and maximized subject to a system of linear constraints.

Theorem 4.3 *Let KB be a probabilistic knowledge base and let $Q = \exists(G|A)[x_1, x_2]$ be an object-ground probabilistic query such that $KB \not\models \bot \leftarrow A$.*

Then, the tight answer substitution for Q to KB is given by $\sigma = \{x_1/d_1,\ x_2/d_2\}$, where d_1 (resp., d_2) is the optimal value of the following linear fractional program over the variables y_I $(I \in \mathcal{I}_\Phi)$:

$$\begin{array}{l} \textit{minimize (resp., maximize)} \;\; \Big(\sum\limits_{I \in \mathcal{I}_\Phi,\, I \models G \wedge A} y_I\Big) \,/\, \Big(\sum\limits_{I \in \mathcal{I}_\Phi,\, I \models A} y_I\Big) \\ \textit{subject to (1) and} \;\; \sum\limits_{I \in \mathcal{I}_\Phi,\, I \models A} y_I \; > \; 0\,. \end{array} \tag{2}$$

By a standard technique going back to Charnes and Cooper [9], the two generated linear fractional programs can now easily be transformed into two equivalent linear programs.

Theorem 4.4 (essentially [28, 66, 1]) *Let KB be a probabilistic knowledge base and let $Q=\exists(G|A)[x_1,x_2]$ be an object-ground probabilistic query such that $KB \not\models \bot \leftarrow A$.*

Then, the tight answer substitution for Q to KB is given by $\sigma = \{x_1/d_1,\ x_2/d_2\}$, where d_1 (resp., d_2) is the optimal value of the following linear program over the variables y_I ($I \in \mathcal{I}_\Phi$):

$$\textit{minimize (resp., maximize)} \quad \sum_{I \in \mathcal{I}_\Phi,\, I \models G \wedge A} y_I$$

$$\textit{subject to}$$

$$\begin{aligned}
\sum_{I \in \mathcal{I}_\Phi,\, I \models A} y_I &= 1 \\
\sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} -c_1\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (1-c_1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1,c_2] \in \textit{ground}(KB),\ c_1 > 0) \\
\sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} c_2\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (c_2-1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1,c_2] \in \textit{ground}(KB),\ c_2 < 1) \\
y_I &\geq 0 \quad (\textit{for all } I \in \mathcal{I}_\Phi)\,.
\end{aligned} \tag{3}$$

After this summary of well-known results from the literature, we are now ready to proceed with deriving the remaining membership results. We next define the size of natural and rational numbers.

The *size* of a nonnegative integer a, denoted $\|a\|$, is defined as $1 + \lfloor \log a \rfloor$, if $a > 0$, otherwise 1. The *size* of a nonnegative rational number r, denoted $\|r\|$, is defined as $\max(\|p\|, \|q\|)$, where $p \geq 0$ and $q > 0$ are relatively prime integers such that $p/q = r$ (we assume $q = 1$ when $r = 0$).

In the sequel, we concentrate on object-ground probabilistic queries to ground probabilistic knowledge bases KB. The following *small model theorem* says that the optimal values of (3) have a polynomial size and that they are attained at some models of KB of polynomial size.

Theorem 4.5 *Let KB be a ground probabilistic knowledge base. Let $(G|A)[d_1,d_2]$ be a ground conditional constraint such that $KB \models_{\mathit{tight}} (G|A)[d_1,d_2]$ and $KB \not\models \bot \leftarrow A$. Let c be the cardinality of KB and let l be the greatest size of a rational number in KB (we assume $l = 1$ when $KB = \emptyset$). Let $m = 2c + 1$.*

a) For each $d \in \{d_1, d_2\}$, there exists a model Pr of KB such that:

(i) $Pr(A) > 0$ and $Pr(G \wedge A) = d \cdot Pr(A)$,

(ii) the number of all $I \in \mathcal{I}_\Phi$ with $Pr(I) > 0$ is at most m,

(iii) the size of each $Pr(I) > 0$ is at most $1 + lm^2 + m + \lfloor (m^2 + m + 1) \log m \rfloor$.

b) Each $d \in \{d_1, d_2\}$ has a size of at most $1 + lm^2 + \lfloor m^2 \log m \rfloor$.

We are now ready to show that propositional TIGHT ENTAILMENT (D) is in D^{P} in the general case.

Theorem 4.6 *Given a ground probabilistic knowledge base KB, an object-ground probabilistic query $Q = \exists(G|A)[x_1, x_2]$, and two rational numbers $d_1, d_2 \in [0, 1]$, deciding whether $\{x_1/d_1,\ x_2/d_2\}$ is the tight answer substitution for Q to KB is in D^{P}.*

Proof (sketch). Algorithm tight-entailment (see Fig. 2) checks whether $\sigma = \{x_1/d_1,\ x_2/d_2\}$ is the tight answer substitution for $Q = \exists(G|A)[x_1, x_2]$ to KB, which is reduced to checking whether $(G|A)[d_1, d_2]$ is a logical consequence of KB (Step 1), and either checking whether $(d_1, d_2) = (1, 0)$, or checking whether KB has two models Pr_1 and Pr_2 such that $Pr_i(A) > 0$ and $Pr_i(G \wedge A) = d_i \cdot Pr_i(A)$ for all $i \in \{1, 2\}$ (Steps 2–4). By Theorem 4.1 b), Step 1 is in co-NP. Moreover, Steps 2–4 are in NP (see Appendix A). □

The next theorem shows that propositional TIGHT ENTAILMENT (OPT) is in $\mathrm{FP}^{\mathrm{NP}}$ in the general case.

Theorem 4.7 *Given a ground probabilistic knowledge base KB and an object-ground probabilistic query $Q = \exists(G|A)[x_1, x_2]$, computing the tight answer substitution for Q to KB is in* $\mathrm{FP}^{\mathrm{NP}}$.

Proof (sketch). Algorithm tight-entailment-opt (see Fig. 3) computes the tight answer substitution $\sigma = \{x_1/d_1, x_2/d_2\}$ for $Q = \exists(G|A)[x_1, x_2]$ to KB. The main idea behind this algorithm can be informally described as follows. By Theorem 4.5 b), the size of d_1 and d_2 is bounded by a polynomial in l and m. By Theorem 4.5 a), given $d_1 \leq d_2$, there are two "small" models Pr_1 and Pr_2 of KB such that $Pr_i(A) > 0$ and $Pr_i(G \wedge A) = d_i \cdot Pr_i(A)$ for all $i \in \{1, 2\}$. This suggests that d_1 and d_2 can be computed by binary search in deterministic polynomial time with an NP oracle. However, standard binary search just produces two intervals $I_1, I_2 \subseteq [0, 1]$ such that $d_1 \in I_1$ and $d_2 \in I_2$. Nonetheless, d_1 and d_2 are uniquely determined through these intervals I_1 and I_2. Thus, we just have to perform another binary search to obtain d_1 and d_2 from I_1 and I_2, respectively. More details and the proof that tight-entailment-opt can be done in deterministic polynomial time with a polynomial number of NP oracle calls are given in Appendix A. □

Finally, the following theorem shows that the data versions of SATISFIABILITY, ENTAILMENT, and TIGHT ENTAILMENT (D) are in NP, co-NP, and D^{P}, respectively, in the general case.

Theorem 4.8 *Let KB be a fixed probabilistic knowledge base.*

a) Given a finite set of ground probabilistic facts $\mathcal{F}$, deciding whether $KB \cup \mathcal{F}$ is satisfiable is in NP.

b) Given a finite set of ground probabilistic facts $\mathcal{F}$ and a ground probabilistic query $Q = \exists(G|A)[d_1, d_2]$, deciding whether Yes *is the correct answer for Q to $KB \cup \mathcal{F}$ is in* co-NP.

c) Given a finite set of ground probabilistic facts $\mathcal{F}$, an object-ground probabilistic query $Q = \exists(G|A)[x_1, x_2]$, and two rational numbers $d_1, d_2 \in [0, 1]$, deciding whether $\{x_1/d_1,\ x_2/d_2\}$ is the tight answer substitution for Q to $KB \cup \mathcal{F}$ is in D^{P}.

Proof. The statements follow immediately from Theorems 4.1 and 4.6, since the number of all ground instances of conditional constraints in $KB \cup \mathcal{F}$, which are constructed with all constant symbols from $KB \cup \mathcal{F}$ (resp., $KB \cup \mathcal{F}$ and Q), is polynomial in the input size of $\mathcal{F}$ (resp., $\mathcal{F}$ and Q). □

4.5 Hardness Results

We now concentrate on the hardness results in Tables 1–3. The following theorem shows that propositional SATISFIABILITY and ENTAILMENT are NP-hard and co-NP-hard, respectively, in the atomic case.

Theorem 4.9 *a) Deciding whether a ground atomic probabilistic logic program P is satisfiable is* NP*-hard.*

b) Given a ground atomic probabilistic logic program P and a ground atomic query $Q = \exists(G|\top)[d_1, d_2]$ with rational numbers $d_1, d_2 \in [0, 1]$, deciding whether Yes *is the correct answer for Q to P is* co-NP*-hard.*

Proof. a) The statement follows immediately from the proof of NP-hardness of 2PSAT by Georgakopoulos et al. [24] (see also the work by Paris [59]).

b) We give a polynomial transformation from the complement of the NP-complete problem described in a). Given a ground atomic probabilistic logic program P, we are asked whether P is unsatisfiable. Let H be any ground atom. Then, P is unsatisfiable iff $P \models (H|\top)[1, 0]$. □

The next theorem shows that propositional TIGHT ENTAILMENT (D) is D^{P}-hard in the atomic case.

Theorem 4.10 *Given a ground atomic probabilistic logic program P, an object-ground atomic probabilistic query $Q = \exists(H|\top)[x_1, x_2]$, and two rational numbers $d_1, d_2 \in [0, 1]$, deciding whether $\{x_1/d_1,\ x_2/d_2\}$ is the tight answer substitution for Q to P is* D^{P}*-hard.*

Algorithm tight-entailment

Input: Ground probabilistic knowledge base KB, object-ground probabilistic query $Q = \exists(G|A)[x_1, x_2]$, and substitution $\sigma = \{x_1/d_1, x_2/d_2\}$.
Output: "Yes", if σ is the tight answer substitution for Q to KB, otherwise "No".

$b_1 :=$ "$(G|A)[d_1, d_2]$ is a logical consequence of KB";
if $d_1 > d_2$ **then** $b_2 := (d_1 = 1) \wedge (d_2 = 0)$
  else $b_2 :=$ "KB has two models Pr_1 and Pr_2 such that $Pr_1(A) > 0$, $Pr_2(A) > 0$,
    $Pr_1(G \wedge A) = d_1 \cdot Pr_1(A)$, and $Pr_2(G \wedge A) = d_2 \cdot Pr_2(A)$";
return $b_1 \wedge b_2$.

Figure 2: Algorithm tight-entailment

Algorithm tight-entailment-opt

Input: Ground probabilistic knowledge base KB and object-ground probabilistic query $Q = \exists(G|A)[x_1, x_2]$.
Output: Tight answer substitution $\sigma = \{x_1/d_1, x_2/d_2\}$ for Q to KB.
Notation: We use c to denote the cardinality of KB and l to denote the greatest size of a rational number in KB (we assume $l = 1$ when $KB = \emptyset$). Let $m = 2c + 1$ and $k = 1 + lm^2 + \lfloor m^2 \log m \rfloor$.

if $KB \models \bot \leftarrow A$ **then return** $\sigma = \{x_1/1, x_2/0\}$;
$[l_1, l_2] := [0, 1]$;
$[u_1, u_2] := [0, 1]$;
for $i := 1$ **to** $2k + 1$ **do begin**
  if KB has a model Pr with $Pr(A) > 0$ and $l_1 \leq Pr(G \wedge A) \,/\, Pr(A) < (l_1 + l_2) \,/\, 2$
  then $l_2 := (l_1 + l_2) \,/\, 2$ **else** $l_1 := (l_1 + l_2) \,/\, 2$;
  if KB has a model Pr with $Pr(A) > 0$ and $(u_1 + u_2) \,/\, 2 < Pr(G \wedge A) \,/\, Pr(A) \leq u_2$
  then $u_1 := (u_1 + u_2) \,/\, 2$ **else** $u_2 := (u_1 + u_2) \,/\, 2$
end;
$[q_1, q_2] := [0, 2^k]$;
$[s_1, s_2] := [0, 2^k]$;
for $i := 1$ **to** $k + 1$ **do begin**
  if $\{q_1 \leq q < (q_1 + q_2) \,/\, 2,\ l_1 \leq p \,/\, q \leq l_2,\ p, q \in \{0, \ldots, 2^k - 1\},\ q > 0\}$ is solvable
  then $q_2 := (q_1 + q_2) \,/\, 2$ **else** $q_1 := (q_1 + q_2) \,/\, 2$;
  if $\{s_1 \leq s < (s_1 + s_2) \,/\, 2,\ u_1 \leq r \,/\, s \leq u_2,\ r, s \in \{0, \ldots, 2^k - 1\},\ s > 0\}$ is solvable
  then $s_2 := (s_1 + s_2) \,/\, 2$ **else** $s_1 := (s_1 + s_2) \,/\, 2$
end;
$d_1 := \lfloor \lfloor q_2 \rfloor \, l_2 \rfloor \,/\, \lfloor q_2 \rfloor$;
$d_2 := \lfloor \lfloor s_2 \rfloor \, u_2 \rfloor \,/\, \lfloor s_2 \rfloor$;
return $\sigma = \{x_1/d_1, x_2/d_2\}$.

Figure 3: Algorithm tight-entailment-opt

Proof. We give a polynomial transformation from the following D^{P}-complete problem [68]. Given a finite undirected graph $G=(V,E)$, decide whether G has the chromatic number 7, that is, whether G is 7-colorable, but not 6-colorable.

We construct P and $Q = \exists(H|\top)[x_1, x_2]$ as in the statement of the theorem such that the chromatic number of G is 7 iff the tight answer substitution for Q to P is given by $\sigma = \{x_1/0,\, x_2/0\}$.

The vocabulary Φ contains only 0-ary predicate symbols and is defined as follows:

$$\Phi \;=\; \{p\} \cup \{p_{i,j} \mid i \in V,\, j \in \{1,\ldots,6\}\} \cup \{q_{i,j} \mid i \in V,\, j \in \{1,\ldots,7\}\}\,.$$

The program P is defined as the union of two programs P_1 and P_2 such that (i) P_1 is satisfiable iff G is 7-colorable, (ii) P_2 is always satisfiable, and (iii) $P_2 \models (p\,|\,\top)[0,0]$ iff G is not 6-colorable.

We first define $P_1 = P_{1,1} \cup P_{1,2} \cup P_{1,3}$ as follows:

$$\begin{aligned}
P_{1,1} &= \{(q_{i,j} \mid q_{i,k})[0,0] \mid i \in V,\, j,k \in \{1,\ldots,7\},\, j<k\}\,,\\
P_{1,2} &= \{(q_{i,j} \mid q_{k,j})[0,0] \mid \{i,k\} \in E,\, j \in \{1,\ldots,7\}\}\,,\\
P_{1,3} &= \{(q_{i,j} \,|\, \top)[1/7, 1/7] \mid i \in V,\, j \in \{1,\ldots,7\}\}\,.
\end{aligned}$$

We next define $P_2 = P_{2,1} \cup P_{2,2} \cup P_{2,3}$ as follows:

$$\begin{aligned}
P_{2,1} &= \{(p_{i,j} \mid p_{i,k})[0,0] \mid i \in V,\, j,k \in \{1,\ldots,6\},\, j<k\}\,,\\
P_{2,2} &= \{(p_{i,j} \mid p_{k,j})[0,0] \mid \{i,k\} \in E,\, j \in \{1,\ldots,6\}\}\,,\\
P_{2,3} &= \{(p_{i,j} \mid p)[1/6, 1/6] \mid i \in V,\, j \in \{1,\ldots,6\}\}\,.
\end{aligned}$$

Obviously, (ii) holds. Moreover, (i) and (iii) follow easily from the properties of a similar construction in the proof of NP-hardness of 2PSAT by Georgakopoulos et al. [24].

Finally, we define $P = P_1 \cup P_2$. Clearly, $P \models_{\mathit{tight}} (p\,|\,\top)[0,0]$ iff P is satisfiable and $P \models (p\,|\,\top)[0,0]$. Obviously, the latter is equivalent to P_1 being satisfiable and $P_2 \models (p\,|\,\top)[0,0]$, which in turn is equivalent to G being 7-colorable, but not 6-colorable. In summary, the chromatic number of G is 7 iff the tight answer substitution for $Q = \exists(p\,|\,\top)[x_1, x_2]$ to P is given by $\sigma = \{x_1/0,\, x_2/0\}$. □

We next show that propositional TIGHT ENTAILMENT (OPT) is $\mathrm{FP}^{\mathrm{NP}}$-hard in the 1-conjunctive case.

Theorem 4.11 *Given a ground 1-conjunctive probabilistic logic program P and an object-ground atomic probabilistic query $Q = \exists(G|\top)[x_1, x_2]$, computing the tight answer substitution for Q to P is $\mathrm{FP}^{\mathrm{NP}}$-hard.*

Proof (sketch). We give a polynomial transformation from the $\mathrm{FP}^{\mathrm{NP}}$-complete *traveling salesman cost* problem [57]. We now sketch the main ideas. The rest of the proof is given in Appendix A.

Given a set of $n \geq 1$ cities $V = \{1, 2, \ldots, n\}$ and a nonnegative integer distance $d_{i,j} = d_{j,i}$ between any two cities i and j, we have to compute the smallest length of a tour through all the cities, that is, the minimum of $\sum_{i=1}^{n} d_{\pi(i),\pi(\sigma(i))}$ subject to all permutations π, where $\sigma(n) = 1$, and $\sigma(i) = i+1$ for all $i < n$. Without affecting the complexity of the problem, we can assume $n \geq 3$.

Let d denote the smallest length of a tour, and let s be the sum of all $d_{i,j}$ with $i,j \in V$ and $i<j$. We now construct P and $Q = \exists(G|\top)[x_1, x_2]$ as in the statement of the theorem such that d is given by ls, where $\sigma = \{x_1/l,\, x_2/u\}$ is the tight answer substitution for Q to P.

Let $E = \{\{i,j\} \subseteq V \mid i \neq j\}$, $w = d/s$, and $w_{\{i,j\}} = d_{i,j}/s$ for all $\{i,j\} \in E$. The vocabulary Φ contains only 0-ary predicate symbols and is defined as $\Phi_1 \cup \Phi_2$, where Φ_1 and Φ_2 are given as follows:

$$\begin{aligned}\Phi_1 &= \{p_{i,j} \mid i,j \in V\}\,, \\ \Phi_2 &= \{p\} \cup \{p_e \mid e \in E\}\,.\end{aligned}$$

We next define a set of 1-conjunctive conditional constraints P_1 that describes the set of all permutations of the members in V. More precisely, we define $P_1 = P_{1,1} \cup P_{1,2} \cup P_{1,3}$ as follows:

$$\begin{aligned}P_{1,1} &= \{(p_{i,j} \mid p_{i,k})[0,0] \mid i,j,k \in V,\, j < k\}\,, \\ P_{1,2} &= \{(p_{i,j} \mid p_{k,j})[0,0] \mid i,j,k \in V,\, i < k\}\,, \\ P_{1,3} &= \{(p_{i,j} \mid \top)[1/n, 1/n] \mid i,j \in V\}\,.\end{aligned}$$

Roughly speaking, each possible world I with $Pr_1(I) > 0$ for some model Pr_1 of P_1 corresponds to a permutation of the members in V, and vice versa. We next define a set of 1-conjunctive conditional constraints P_2 that associates each such permutation with its tour length, and the predicate symbol p with the sum of all such tour lengths. In detail, we define $P_2 = P_{2,1} \cup P_{2,2} \cup P_{2,3}$ as follows:

$$\begin{aligned}P_{2,1} &= \{(p_{e_1} \mid p_{e_2})[0,0] \mid e_1, e_2 \in E,\, e_1 \neq e_2\}\,, \\ P_{2,2} &= \{(p_{\{i,j\}} \mid p_{u,i} \wedge p_{\sigma(u),j})[w_{\{i,j\}}, w_{\{i,j\}}] \mid u \in V,\, \{i,j\} \in E\}\,, \\ P_{2,3} &= \{(p \mid p_e)[1,1] \mid e \in E\}\,.\end{aligned}$$

Finally, we define $P = P_1 \cup P_2$ and $Q = \exists(p \mid \top)[x_1, x_2]$. Let the tight answer substitution for Q to P be given by $\sigma = \{x_1/l,\, x_2/u\}$. It can now be shown that $l = w$ (see Appendix A). □

The following result shows that the data versions of SATISFIABILITY, ENTAILMENT, and TIGHT ENTAILMENT (D) are NP-hard, co-NP-hard, and $\mathrm{D^P}$-hard, respectively, in the 1-conjunctive case.

Theorem 4.12 *Let P be a fixed 1-conjunctive probabilistic logic program.*

a) Given a finite set of ground classical facts $\mathcal{F}$, deciding whether $P \cup \mathcal{F}$ is satisfiable is NP*-hard.*

b) Given a finite set of ground classical facts $\mathcal{F}$ and a ground atomic probabilistic query $Q = \exists(H|\top)[d_1, d_2]$, deciding whether Yes *is the correct answer for Q to $P \cup \mathcal{F}$ is* co-NP*-hard.*

c) Given a finite set of ground classical facts $\mathcal{F}$, an object-ground atomic query $Q = \exists(H|\top)[x_1, x_2]$, and two rational numbers $d_1, d_2 \in [0,1]$, deciding whether $\{x_1/d_1,\, x_2/d_2\}$ is the tight answer substitution for Q to $P \cup \mathcal{F}$ is $\mathrm{D^P}$*-hard.*

5 Towards Efficient Algorithms

How do we obtain efficient algorithms for probabilistic logic programming? One idea could be to syntactically restrict probabilistic logic programs such that probabilistic reasoning becomes tractable.

Two syntactically quite restricted cases are atomic and ordinary probabilistic logic programs (see Fig. 4, left side). However, by Theorems 4.9 and 4.10, propositional SATISFIABILITY, ENTAILMENT, and TIGHT ENTAILMENT (D) are (presumably) intractable even in the atomic case. Moreover, even though all ordinary probabilistic logic programs are satisfiable, propositional ENTAILMENT and TIGHT ENTAILMENT (D) are also (presumably) intractable in the ordinary case. This is shown by the following theorem.

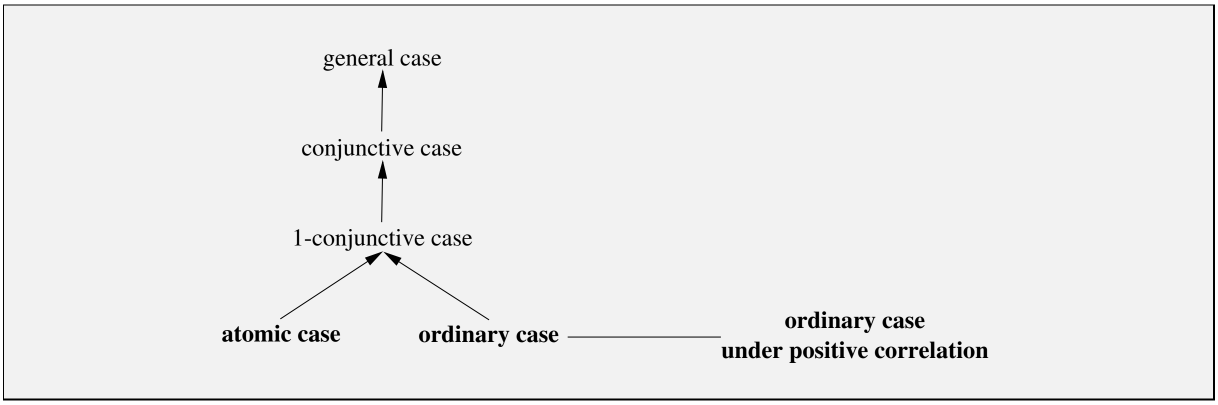

Figure 4: Further syntactic and semantic restrictions

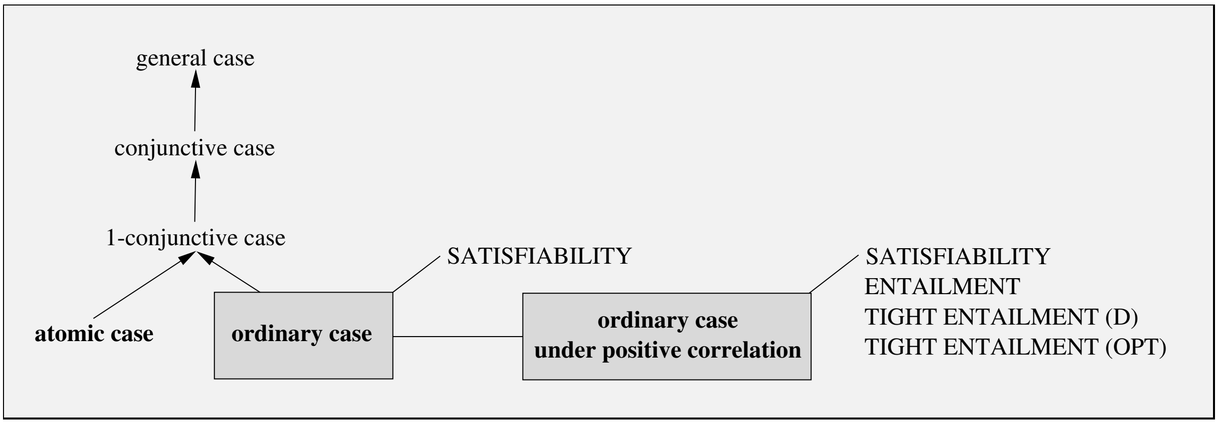

Figure 5: Propositional and data tractability

Theorem 5.1 *a) Every ordinary probabilistic logic program P is satisfiable.*

b) Given a ground ordinary probabilistic logic program P and a ground ordinary query $Q = \exists(H|\top)[d_1, d_2]$ with rational numbers $d_1, d_2 \in [0, 1]$, deciding whether Yes *is the correct answer for Q to P is* co-NP*-hard.*

c) Given a ground ordinary probabilistic logic program P, an object-ground ordinary probabilistic query $Q = \exists(H|\top)[x_1, x_2]$, and two rational numbers $d_1, d_2 \in [0, 1]$, deciding whether $\{x_1/d_1,\ x_2/d_2\}$ is the tight answer substitution for Q to P is NP*-hard.*

We thus have to look for other restrictions that may carry us to efficient algorithms. The main source of complexity of probabilistic logic programming is the exponential number of possible worlds in $\mathcal{I}_\Phi$. Do we need all these possible worlds in order to describe a given probabilistic logic programming task?

Indeed, it turns out that many possible worlds in $\mathcal{I}_\Phi$ are not necessary. Some of them can be excluded in advance by the available classical knowledge, while others are simply not relevant for a given probabilistic logic programming task. Finally, we often do not need the fine granularity of the set of all relevant possible worlds. More precisely, it is sufficient to consider only equivalence classes of relevant possible worlds.

In the following Sections 6 and 7, we follow these ideas and elaborate techniques for solving SATISFIA-

BILITY and TIGHT ENTAILMENT (OPT) in the conjunctive case, which are efficient in the special case in which we have few relevant purely probabilistic knowledge. Roughly speaking, the main idea is to exploit the tractability of classical inference in the conjunctive case in order to reduce the number of possible worlds that must be taken into consideration.

Another idea towards efficient algorithms is to give up the probabilistic semantics in its full generality, that is, to semantically restrict probabilistic interpretations. We could, for example, consider only *positively correlated probabilistic interpretations*, which satisfy the axioms of probability as well as an additional axiom that brings probabilistic logics closer to truth-functional logics.

Indeed, it turns out that the propositional and data versions of all probabilistic logic programming tasks are tractable for ordinary probabilistic logic programs under positive correlation (see Figure 5, right side)

In Section 9, we thus concentrate on ordinary probabilistic logic programs under positive correlation, and present an efficient approximation technique for solving SATISFIABILITY and TIGHT ENTAILMENT (OPT) under general probabilistic interpretations in the conjunctive case.

6 Satisfiability

We now focus on the problem of deciding whether a probabilistic knowledge base is satisfiable. We reduce this problem to the problem of deciding whether a system of linear constraints is solvable. These systems of linear constraints have a small size when we have few relevant purely probabilistic knowledge. After describing the main ideas for the general case, we restrict our attention to probabilistic logic programs.

6.1 Naive Systems of Linear Constraints

By Theorem 4.2, the problem of deciding whether a probabilistic knowledge base KB is satisfiable can easily be reduced to the problem of deciding whether a system of linear constraints is solvable.

The crux with the *naive system of linear constraints* (1) is that the number of variables is given by the number of possible worlds over Φ and that the number of linear constraints is linear in the number of ground instances of conditional constraints in KB. Hence, especially the number of variables is generally quite large, as the following example immediately shows.

Example 6.1 Let KB be given by the probabilistic logic program P of Example 3.1. The naive system of linear constraints (1) has $2^{64} \approx 18 \cdot 10^{18}$ (!) variables and 64 linear constraints.

6.2 Reduced Systems of Linear Constraints

We now introduce a more sophisticated system of linear constraints to characterize the satisfiability of a probabilistic knowledge base. This new system of linear constraints generally has a much lower size than the naive one. In the sequel, let KB be a probabilistic knowledge base. Let CKB (resp., PKB) denote the set of all classical (resp., purely probabilistic) members of KB.

6.2.1 Key Ideas

The key ideas behind the reduced system of linear constraints can be summarized as follows:

R1 We exploit all classical members of KB. That is, rather than having a variable for each possible world $I \in \mathcal{I}_\Phi$, we just have a variable for each model $I \in \mathcal{I}_\Phi$ of CKB. Moreover, we introduce a pair of linear inequalities only for each member of $ground(PKB)$, rather than for each member of $ground(KB)$.

R2 We assume the existence of some appropriate classical knowledge base Rel such that KB is satisfiable iff $KB \cup Rel$ is satisfiable. We then have a variable for each model $I \in \mathcal{I}_\Phi$ of $CKB \cup Rel$, and not for each model $I \in \mathcal{I}_\Phi$ of CKB anymore. Roughly speaking, all models $I \in \mathcal{I}_\Phi$ of $CKB \cup Rel$ are relevant for deciding satisfiability, while the other models $I \in \mathcal{I}_\Phi$ of CKB are irrelevant. That is, Rel cuts down the set of all models $I \in \mathcal{I}_\Phi$ of CKB to a *relevant context*.

R3 We obtain $S^\star_{PKB}(CKB \cup Rel)$ from $CKB \cup Rel$ by adding classical conditional constraints that trivially follow from $KB \cup Rel$. We then have a variable only for each model $I \in \mathcal{I}_\Phi$ of $S^\star_{PKB}(CKB \cup Rel)$, rather than for each model $I \in \mathcal{I}_\Phi$ of $CKB \cup Rel$.

R4 We divide $ground(PKB)$ into active and inactive ground instances w.r.t. $S^\star_{PKB}(CKB \cup Rel)$. We then have a pair of linear inequalities only for each active member of $ground(PKB)$.

R5 We exploit the structure of all active members of $ground(PKB)$, which imposes an equivalence relation on the set of all models $I \in \mathcal{I}_\Phi$ of $S^\star_{PKB}(CKB \cup Rel)$. We then have a variable only for each equivalence class of possible worlds.

Informally, **R1** is very useful for probabilistic knowledge bases KB with a lot of classical knowledge. Reduction **R2** can be motivated from the linear programming background as follows. If KB is satisfiable, then there exists a *small model* of KB in which all except for a polynomial number of possible worlds are assigned a zero probability. The classical knowledge Rel now fixes a set of possible worlds that can be assigned a zero probability in some model of KB. Note that a trivial Rel is always given by $\emptyset$. Reduction **R3** identifies incompatibilities between conditional constraints and is very useful especially for detecting an unsatisfiability of KB. By **R4**, we can ignore all those members of $ground(PKB)$ that do not apply to the context given by the available classical knowledge. Finally, the intuition behind **R5** is that we actually do not need the fine granularity of the set of relevant possible worlds. It is sufficient to work with equivalence classes of possible worlds. Note that these equivalence classes are a generalization of the equivalence classes presented in [41] for conjunctive events. They are based on the concept of conditional events (see especially [12] and [6]).

6.2.2 Definitions

We start with some preparative definitions.

Let $\mathcal{L}$ be a classical knowledge base and let $\mathcal{F}$ be a probabilistic knowledge base. We first define the operator $S^\star_\mathcal{F}$, which increases $\mathcal{L}$ by classical knowledge that trivially follows from $\mathcal{L} \cup \mathcal{F}$. For this purpose, we need the operator $S_\mathcal{F}$ on classical knowledge bases $\mathcal{L}$, which is defined as follows:

$$\begin{aligned} S_\mathcal{F}(\mathcal{L}) \;\; = \;\; & \mathcal{L} \cup \{\bot \leftarrow B \mid (H|B)[c_1, c_2] \in ground(\mathcal{F}),\ \mathcal{L} \models \bot \leftarrow H \wedge B,\ c_1 > 0\} \\ & \cup \{\bot \leftarrow B \mid (H|B)[c_1, c_2] \in ground(\mathcal{F}),\ \mathcal{L} \models H \leftarrow B,\ c_2 < 1\}\,. \end{aligned}$$

For classical knowledge bases $\mathcal{L}$, let $S^0_\mathcal{F}(\mathcal{L}) = \mathcal{L}$ and $S^{n+1}_\mathcal{F}(\mathcal{L}) = S_\mathcal{F}(S^n_\mathcal{F}(\mathcal{L}))$ for all $n > 0$. We now define $S^\star_\mathcal{F} = S^n_\mathcal{F}$, where n is the least $n \geq 0$ such that $S^n_\mathcal{F} = S^{n+1}_\mathcal{F}$.

For classical and probabilistic knowledge bases $\mathcal{L}$ and $\mathcal{F}$, respectively, the set of all *active* (resp., *inactive*) members of $ground(\mathcal{F})$ w.r.t. $\mathcal{L}$, denoted $active_{\mathcal{L}}(\mathcal{F})$ (resp., $inactive_{\mathcal{L}}(\mathcal{F})$), is defined as follows:

$$\begin{aligned} active_{\mathcal{L}}(\mathcal{F}) &= \{(H|B)[c_1, c_2] \in ground(\mathcal{F}) \mid \mathcal{L} \not\models \bot \leftarrow B,\ \mathcal{L} \not\models \bot \leftarrow H \wedge B \text{ or } c_1 > 0, \\ &\qquad \mathcal{L} \not\models H \leftarrow B \text{ or } c_2 < 1,\ c_1 > 0 \text{ or } c_2 < 1\} \\ inactive_{\mathcal{L}}(\mathcal{F}) &= ground(\mathcal{F}) - active_{\mathcal{L}}(\mathcal{F})\,. \end{aligned}$$

Given a ground probabilistic knowledge base $\mathcal{F}$, we use $\mathcal{M}(\mathcal{F})$ to denote the set of all mappings M that associate each $(H|B)[c_1, c_2] \in \mathcal{F}$ with an element of $\{\neg B, \neg H \wedge B, H \wedge B\}$. For such a mapping M, we use $C(M)$ to denote the conjunction of all $M(F)$ with $F \in \mathcal{F}$. For classical knowledge bases $\mathcal{L}$ and ground probabilistic knowledge bases $\mathcal{F}$, we define $\mathcal{C}_{\mathcal{L}}(\mathcal{F})$ as follows:

$$\mathcal{C}_{\mathcal{L}}(\mathcal{F}) = \begin{cases} \emptyset & \text{if } \mathcal{L} \models \bot \leftarrow \top \\ \{\top\} & \text{if } \mathcal{L} \not\models \bot \leftarrow \top,\ \mathcal{F} = \emptyset \\ \{C(M) \mid M \in \mathcal{M}(\mathcal{F}), \mathcal{L} \not\models \bot \leftarrow C(M)\} & \text{if } \mathcal{L} \not\models \bot \leftarrow \top,\ \mathcal{F} \neq \emptyset\,. \end{cases}$$

A classical knowledge base Rel is *stable w.r.t.* $\mathcal{L}$ and $\mathcal{F}$ iff the following condition holds:

ST For all $C \in \mathcal{C}_{\mathcal{L}}(\mathcal{F})$, it holds $\mathcal{L} \cup Rel \not\models \bot \leftarrow C$.

6.2.3 Main Results

In the sequel, let Rel be a classical knowledge base. Let $\mathcal{L}_1 = S^{\star}_{PKB}(CKB)$, $\mathcal{F}_1 = active_{\mathcal{L}_1 \cup Rel}(PKB)$, $\mathcal{L}_2 = S^{\star}_{PKB}(CKB \cup Rel)$, and $\mathcal{F}_2 = active_{\mathcal{L}_2}(PKB)$. Let the index set $\mathcal{C}_{KB}$ be defined as $\mathcal{C}_{\mathcal{L}_2}(\mathcal{F}_2)$. For classical formulas C and F, we use $C \preceq F$ to abbreviate $\mathcal{L}_2 \cup \{C\} \models F$.

We are now ready to formulate our first important result. The following theorem shows that, whenever Rel has a certain stability property, KB is satisfiable iff $KB \cup Rel$ is satisfiable.

Theorem 6.2 *Let Rel be stable w.r.t. $\mathcal{L}_1$ and $\mathcal{F}_1$. Then, KB is satisfiable iff $KB \cup Rel$ is satisfiable.*

Finally, the next important result characterizes the satisfiability of a probabilistic knowledge base by a more sophisticated system of linear constraints.

Theorem 6.3 *Let Rel be stable w.r.t. $\mathcal{L}_1$ and $\mathcal{F}_1$. Then, KB is satisfiable iff the following system of linear constraints over the variables y_C ($C \in \mathcal{C}_{KB}$) is solvable:*

$$\begin{aligned} \sum_{C \in \mathcal{C}_{KB}} y_C &= 1 \\ \sum_{C \in \mathcal{C}_{KB},\, C \preceq \neg H \wedge B} -c_1\, y_C + \sum_{C \in \mathcal{C}_{KB},\, C \preceq H \wedge B} (1 - c_1)\, y_C &\geq 0 \quad (\textit{for all } (H|B)[c_1, c_2] \in \mathcal{F}_2,\ c_1 > 0) \\ \sum_{C \in \mathcal{C}_{KB},\, C \preceq \neg H \wedge B} c_2\, y_C + \sum_{C \in \mathcal{C}_{KB},\, C \preceq H \wedge B} (c_2 - 1)\, y_C &\geq 0 \quad (\textit{for all } (H|B)[c_1, c_2] \in \mathcal{F}_2,\ c_2 < 1) \\ y_C &\geq 0 \quad (\textit{for all } C \in \mathcal{C}_{KB})\,. \end{aligned} \tag{4}$$

6.3 Satisfiability of Probabilistic Logic Programs

We now focus on deciding whether a probabilistic logic program P is satisfiable. It turns out that probabilistic logic programs have a nontrivial natural stable Rel. In the sequel, let P be a probabilistic logic program.

Let CP (resp., PP) denote the set of all classical (resp., purely probabilistic) members in P.

6.3.1 Definitions

We will replace the index set $\mathcal{C}_P$ in (4) by a new index set $\mathcal{I}_P \subseteq \mathcal{I}_\Phi$. More precisely, each $C \in \mathcal{C}_P$ will then represented by a possible world $I \in \mathcal{I}_\Phi$ with $I \models C$. We need the following preparation.

We first define a *classical approximation* of P as follows. Let $\overline{P}$ be the set of all classical conjunctive conditional constraints $H \leftarrow B$ such that $(H|B)[c_1, c_2] \in P$ for some $c_1, c_2 \in [0,1]$ with $c_1 > 0$.

Let R associate $\perp$ and each conjunction of ground atoms C with a subset of $HB_\Phi \cup \{\perp\}$ as follows. We define $R(\perp) = HB_\Phi \cup \{\perp\}$, and $R(C)$ as the set of all ground atoms that occur in C.

For classical logic programs $\mathcal{L}$, we define the operator $T_\mathcal{L}\!\uparrow\omega$ on the set of all subsets of $HB_\Phi \cup \{\perp\}$ as usual. For this task, we need the *immediate consequence operator* $T_\mathcal{L}$. For all $I \subseteq HB_\Phi \cup \{\perp\}$:

$$T_\mathcal{L}(I) \;=\; \bigcup \{R(H) \mid H \leftarrow B \in \mathit{ground}(\mathcal{L}) \text{ with } R(B) \subseteq I\}\,.$$

For all $I \subseteq HB_\Phi \cup \{\perp\}$, 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 $n < \omega$. We abbreviate $T_\mathcal{L}\!\uparrow\alpha(\emptyset)$ by $T_\mathcal{L}\!\uparrow\alpha$.

For all sets $\mathcal{I}$ of subsets of $HB_\Phi \cup \{\perp\}$, the *closure* of $\mathcal{I}$ w.r.t. $\mathcal{L}$, denoted $\mathit{closure}_\mathcal{L}(\mathcal{I})$, is the least set of subsets of $HB_\Phi \cup \{\perp\}$ that satisfies the following conditions:

C1 $T_\mathcal{L}\!\uparrow\omega \in \mathit{closure}_\mathcal{L}(\mathcal{I})$,

C2 $T_\mathcal{L}\!\uparrow\omega(I) \in \mathit{closure}_\mathcal{L}(\mathcal{I})$ for all $I \in \mathcal{I}$,

C3 $T_\mathcal{L}\!\uparrow\omega(I_1 \cup I_2) \in \mathit{closure}_\mathcal{L}(\mathcal{I})$ for all $I_1, I_2 \in \mathit{closure}_\mathcal{L}(\mathcal{I})$.

For all ground probabilistic logic programs $\mathcal{F}$, we define $\mathcal{I}_\mathcal{L}(\mathcal{F})$ as follows:

$$\mathcal{I}_\mathcal{L}(\mathcal{F}) \;=\; \mathit{closure}_\mathcal{L}(\bigcup\{\{R(B), R(H \wedge B)\} \mid (H|B)[c_1, c_2] \in \mathcal{F}\}) \cap \mathcal{I}_\Phi\,.$$

6.3.2 Main Results

As shown in Appendix C, an appropriate classical knowledge base Rel is given by the set of all $\perp \leftarrow A$ such that $A \in HB_\Phi$ and that A is not logically entailed by the logical approximation $\overline{P}$ of P. That is, we define:

$$\mathit{Rel} \;=\; \{\perp \leftarrow A \mid A \in HB_\Phi,\, \overline{P} \not\models A\}\,.$$

In the sequel, we define the classical knowledge base $\underline{P}$ and the index set $\mathcal{I}_P$ by $\underline{P} = S^\star_{PP}(CP \cup \mathit{Rel})$ and $\mathcal{I}_P = \mathcal{I}_{\underline{P}}(\mathit{active}_{\underline{P}}(PP))$, respectively.

The following theorem characterizes the satisfiability of P by a reduced system of linear constraints, which is obtained from (4) by replacing the index set $\mathcal{C}_P$ by the new index set $\mathcal{I}_P$.

Theorem 6.4 *The probabilistic logic program P is satisfiable iff the following system of linear constraints* *over the variables* y_I $(I \in \mathcal{I}_P)$ *is solvable:*

$$\begin{array}{rcl} \sum_{I \in \mathcal{I}_P} y_I & = 1 & \\ \sum_{I \in \mathcal{I}_P,\, I \models \neg H \wedge B} -c_1\, y_I + \sum_{I \in \mathcal{I}_P,\, I \models H \wedge B} (1-c_1)\, y_I & \geq 0 & (\textit{for all } (H|B)[c_1,c_2] \in \mathit{active}_{\underline{P}}(PP),\ c_1 > 0) \\ \sum_{I \in \mathcal{I}_P,\, I \models \neg H \wedge B} c_2\, y_I + \sum_{I \in \mathcal{I}_P,\, I \models H \wedge B} (c_2-1)\, y_I & \geq 0 & (\textit{for all } (H|B)[c_1,c_2] \in \mathit{active}_{\underline{P}}(PP),\ c_2 < 1) \\ y_I & \geq 0 & (\textit{for all } I \in \mathcal{I}_P)\,. \end{array} \tag{5}$$

The next result characterizes the ground instances of PP that are active w.r.t. $\underline{P}$. In particular, it shows that all ground atoms in $\mathit{active}_{\underline{P}}(PP)$ belong to the least model $I \in \mathcal{I}_\Phi$ of $\overline{P}$.

Lemma 6.5 *For all* $(H|B)[c_1,c_2] \in \mathit{ground}(PP)$*:*

(i) $(H|B)[c_1,c_2] \in \mathit{active}_{\underline{P}}(PP)$ *iff* $\underline{P} \not\models \bot \leftarrow H \wedge B$, $\underline{P} \not\models H \leftarrow B$, *and* $c_1 > 0$ *or* $c_2 < 1$.

(ii) $(H|B)[c_1,c_2] \in \mathit{active}_{\underline{P}}(PP)$ *implies* $\overline{P} \models H \wedge B$.

The next result shows that all ground atoms in $\mathcal{I}_P$ belong to the least model $I \in \mathcal{I}_\Phi$ of $\overline{P}$.

Lemma 6.6 *It holds* $I \subseteq T_{\overline{P}}\!\uparrow\omega$ *for all* $I \in \mathcal{I}_P$.

Informally, Lemmata 6.5 (ii) and 6.6 show the important result that the system of linear constraints (5) is defined over all ground atoms that are logically entailed by the logical approximation $\overline{P}$ of P.

6.4 Example

We now give an illustrative example.

Example 6.7 Consider again the probabilistic logic program P of Examples 3.1 and 6.1. By Theorem 6.4, the problem of deciding whether P is satisfiable can be reduced to the problem of deciding whether the system of linear constraints (5) is solvable. Interestingly, this new system of linear constraints (5) just consists of *six* linear constraints over *six* variables, as the following construction shows.

The set CP (resp., PP) of all classical (resp. purely probabilistic) members of P are given by:

$$\begin{aligned} CP = \{ & (\mathit{ro}(h,u) \mid \top)[1,1], (\mathit{ro}(u,a) \mid \top)[1,1], (\mathit{ro}(a,o) \mid \top)[1,1], (\mathit{ad}(h,u) \mid \top)[1,1], \\ & (\mathit{so}(a,o) \mid \top)[1,1], (\mathit{re}(R,S) \mid \mathit{ro}(R,S) \wedge \mathit{ad}(R,S))[1,1], \\ & (\mathit{re}(R,S) \mid \mathit{re}(R,T) \wedge \mathit{re}(T,S))[1,1]\}\,, \\ PP = \{ & (\mathit{ad}(u,a) \mid \top)[0.8,0.9], (\mathit{re}(R,S) \mid \mathit{ro}(R,S))[0.7,1], \\ & (\mathit{re}(R,S) \mid \mathit{ro}(R,S) \wedge \mathit{so}(R,S))[0.9,1]\}\,. \end{aligned}$$

The classical approximation $\overline{P}$ of P is given as follows:

$$\begin{aligned} \overline{P} = \{ & (\mathit{ro}(h,u) \mid \top)[1,1], (\mathit{ro}(u,a) \mid \top)[1,1], (\mathit{ro}(a,o) \mid \top)[1,1], (\mathit{ad}(h,u) \mid \top)[1,1], \\ & (\mathit{ad}(u,a) \mid \top)[1,1], (\mathit{so}(a,o) \mid \top)[1,1], (\mathit{re}(R,S) \mid \mathit{ro}(R,S))[1,1], \\ & (\mathit{re}(R,S) \mid \mathit{ro}(R,S) \wedge \mathit{so}(R,S))[1,1], (\mathit{re}(R,S) \mid \mathit{ro}(R,S) \wedge \mathit{ad}(R,S))[1,1], \\ & (\mathit{re}(R,S) \mid \mathit{re}(R,T) \wedge \mathit{re}(T,S))[1,1]\}\,. \end{aligned}$$

The sets $T_{\overline{P}}\!\uparrow\omega = \{A \in HB_\Phi \mid \overline{P} \models A\}$, Rel, and $\underline{P}$ are given as follows:

$$\begin{aligned}
T_{\overline{P}}\!\uparrow\omega &= \{ro(h,u), ro(u,a), ro(a,o), ad(h,u), ad(u,a), so(a,o),\\
&\qquad re(h,u), re(u,a), re(a,o), re(h,a), re(u,o), re(h,o)\}\,,\\
Rel &= \{\bot \leftarrow A \mid A \in HB_\Phi - T_{\overline{P}}\!\uparrow\omega\}\,,\\
\underline{P} &= S^\star_{PP}(CP \cup Rel) \;=\; CP \cup Rel\,.
\end{aligned}$$

The set $active_{\underline{P}}(PP)$ is given by the following ground instances:

$$(re(u,a) \mid ro(u,a))[0.7,1], (re(a,o) \mid ro(a,o))[0.7,1],$$
$$(ad(u,a) \mid \top)[0.8,0.9], (re(a,o) \mid ro(a,o) \wedge so(a,o))[0.9,1]\,.$$

The index set $\mathcal{C}_P$ is given by the following ground classical formulas C_i ($i \in \{0,\ldots,5\}$):

$$\begin{aligned}
C_0 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge \neg re(u,a) \wedge \neg re(a,o)\\
C_1 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge re(u,a) \wedge \neg re(a,o)\\
C_2 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge \neg re(u,a) \wedge re(a,o)\\
C_3 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge re(u,a) \wedge re(a,o)\\
C_4 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge ad(u,a) \wedge re(u,a) \wedge \neg re(a,o)\\
C_5 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge ad(u,a) \wedge re(u,a) \wedge re(a,o)\,.
\end{aligned}$$

Moreover, the index set $\mathcal{I}_P$ is given by the following possible worlds I_i ($i \in \{0,\ldots,5\}$), written in binary as subsets of $T_{\overline{P}}\!\uparrow\omega$ (note that $I_i \models C_i$ for all $i \in \{0,\ldots,5\}$):

$$I_0 \mathrel{\hat=} \mathtt{111\ 101\ 100\ 000},\ I_1 \mathrel{\hat=} \mathtt{111\ 101\ 110\ 100},\ I_2 \mathrel{\hat=} \mathtt{111\ 101\ 101\ 000},$$
$$I_3 \mathrel{\hat=} \mathtt{111\ 101\ 111\ 111},\ I_4 \mathrel{\hat=} \mathtt{111\ 111\ 110\ 100},\ I_5 \mathrel{\hat=} \mathtt{111\ 111\ 111\ 111}\,.$$

Finally, the system of linear constraints (5) is given as follows (note that each variable y_i corresponds to the ground classical formula C_i and the possible world I_i):

$$\begin{aligned}
y_0 + y_1 + y_2 + y_3 + y_4 + y_5 &= 1\\
-0.7 \cdot (y_0 + y_2) + 0.3 \cdot (y_1 + y_3 + y_4 + y_5) &\geq 0\\
-0.7 \cdot (y_0 + y_1 + y_4) + 0.3 \cdot (y_2 + y_3 + y_5) &\geq 0\\
-0.8 \cdot (y_0 + y_1 + y_2 + y_3) + 0.2 \cdot (y_4 + y_5) &\geq 0\\
0.9 \cdot (y_0 + y_1 + y_2 + y_3) - 0.1 \cdot (y_4 + y_5) &\geq 0\\
-0.9 \cdot (y_0 + y_1 + y_4) + 0.1 \cdot (y_2 + y_3 + y_5) &\geq 0\\
y_i &\geq 0 \quad \text{(for all } i \in \{0,\ldots,5\}).
\end{aligned}$$

7 Tight Entailment

In this section, we generalize the approach introduced in the previous section to the problem of computing tight answer substitutions for object-ground probabilistic queries. Again, after describing the main ideas for the general case, we restrict our attention to probabilistic logic programs.

7.1 Naive Linear Programs

Let KB be a probabilistic knowledge base and let $Q = \exists(G|A)[x_1, x_2]$ be an object-ground probabilistic query. To compute the tight answer substitution for Q to KB, we first have to decide whether KB logically entails $\bot \leftarrow A$. This can easily be done by deciding whether a system of linear constraints is unsolvable:

Theorem 7.1 *A probabilistic knowledge base KB logically entails $\bot \leftarrow A$ iff the following system of linear constraints over the variables y_I $(I \in \mathcal{I}_\Phi)$ is unsolvable:*

$$\begin{aligned}
\sum_{I \in \mathcal{I}_\Phi,\, I \models A} y_I &= 1 \\
\sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} -c_1\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (1-c_1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1, c_2] \in \textit{ground}(KB),\ c_1 > 0) \\
\sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} c_2\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (c_2-1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1, c_2] \in \textit{ground}(KB),\ c_2 < 1) \\
y_I &\geq 0 \quad (\textit{for all } I \in \mathcal{I}_\Phi)\,.
\end{aligned} \tag{6}$$

If $\bot \leftarrow A$ is indeed a logical consequence of KB, then the requested tight answer substitution is immediately given by $\{x_1/1,\ x_2/0\}$. Otherwise, we additionally have to solve two linear programs. In detail, by Theorem 4.4, the requested tight answer substitution is then given by $\{x_1/d_1,\ x_2/d_2\}$, where d_1 (resp., d_2) is the optimal value of the linear program (3), which contains (6) as system of linear constraints.

Again, the crux with the *naive system of linear constraints* (6) and thus also the *naive linear program* (3) is that especially the number of variables is generally quite large:

Example 7.2 Let KB be given by the probabilistic logic program P of Example 3.1. Consider the object-ground probabilistic query $\exists(\textit{re}(h,o) \mid \textit{ad}(u,a))[X_1, X_2]$. The naive system of linear constraints (6) then has the same size as (1). That is, it has $2^{64} \approx 18 \cdot 10^{18}$ variables and 64 linear constraints.

It is thus natural to wonder whether the reduction technique of Sections 6.2 and 6.3 can also be applied to the problem of computing tight answer substitutions to object-ground probabilistic queries. This is indeed the case, as we will show in the following two subsections.

7.2 Reduced Linear Programs

We now present a reduced system of linear constraints and reduced linear programs for the computation of tight answer substitutions for object-ground queries to probabilistic knowledge bases.

In the sequel, let KB be a probabilistic knowledge base and let $Q = \exists(G|A)[x_1, x_2]$ be an object-ground query. Let CKB (resp., PKB) denote the set of all classical (resp., purely probabilistic) members of KB.

The main idea of this subsection is to slightly adapt the reduction technique of Section 6.2 by additionally including the query Q inside the stability preconditions and the index sets.

In the sequel, let Rel be a classical knowledge base. We use the same abbreviations as in Section 6.2. That is, $\mathcal{L}_1 = S^\star_{PKB}(CKB)$, $\mathcal{F}_1 = \textit{active}_{\mathcal{L}_1 \cup Rel}(PKB)$, $\mathcal{L}_2 = S^\star_{PKB}(CKB \cup Rel)$, and $\mathcal{F}_2 = \textit{active}_{\mathcal{L}_2}(PKB)$. For classical formulas C and F, we use $C \preceq F$ to abbreviate $\mathcal{L}_2 \cup \{C\} \models F$.

Generalizing Theorem 6.2, the following result shows that, when Rel has a certain stability property, KB logically entails $\bot \leftarrow A$ iff $KB \cup Rel$ logically entails $\bot \leftarrow A$.

Theorem 7.3 *Let Rel be stable w.r.t. $\mathcal{L}_1$ and $\mathcal{F}_1 \cup \{(A|\top)[0,1]\}$.*

Then, $KB \models \bot \leftarrow A$ iff $KB \cup Rel \models \bot \leftarrow A$.

Generalizing Theorem 6.3, the following theorem provides a more sophisticated system of linear constraints for deciding whether KB logically entails $\bot \leftarrow A$.

Theorem 7.4 *Let Rel be stable w.r.t. $\mathcal{L}_1$ and $\mathcal{F}_1 \cup \{(A|\top)[0,1]\}$. Let the index set $\mathcal{C}'_{KB}$ be defined as $\mathcal{C}_{\mathcal{L}_2}(\mathcal{F}_2 \cup \{(A|\top)[0,1]\})$. Then, KB logically entails $\bot \leftarrow A$ iff the following system of linear constraints over the variables y_C ($C \in \mathcal{C}'_{KB}$) is unsolvable:*

$$\begin{aligned}
\sum_{C \in \mathcal{C}'_{KB},\, C \preceq A} y_C &= 1 \\
\sum_{C \in \mathcal{C}'_{KB},\, C \preceq \neg H \wedge B} -c_1\, y_C + \sum_{C \in \mathcal{C}'_{KB},\, C \preceq H \wedge B} (1-c_1)\, y_C &\geq 0 \quad \textit{(for all } (H|B)[c_1,c_2] \in \mathcal{F}_2,\, c_1 > 0) \\
\sum_{C \in \mathcal{C}'_{KB},\, C \preceq \neg H \wedge B} c_2\, y_C + \sum_{C \in \mathcal{C}'_{KB},\, C \preceq H \wedge B} (c_2-1)\, y_C &\geq 0 \quad \textit{(for all } (H|B)[c_1,c_2] \in \mathcal{F}_2,\, c_2 < 1) \\
y_C &\geq 0 \quad \textit{(for all } C \in \mathcal{C}'_{KB})\,.
\end{aligned} \tag{7}$$

Roughly speaking, the next important theorem shows the following. Assume that KB does not logically entail $\bot \leftarrow A$ and that Rel has a certain stability property. Then, $(G|A)[d_1, d_2]$ is a tight logical consequence of KB iff $(G|A)[d_1, d_2]$ is a tight logical consequence of $KB \cup Rel$. Moreover, KB logically entails $(G|A)[d_1, d_2]$ iff $KB \cup Rel$ logically entails $(G|A)[d_1, d_2]$.

Theorem 7.5 *Let Rel be stable w.r.t. $\mathcal{L}_1$ and $\mathcal{F}_1 \cup \{(G|A)[0,1]\}$. Let $KB \not\models \bot \leftarrow A$.*
Then, $KB \models_{tight} (G|A)[d_1, d_2]$ iff $KB \cup Rel \models_{tight} (G|A)[d_1, d_2]$.

Finally, the following important theorem provides more sophisticated linear programs for computing the requested tight answer substitution in the case $KB \not\models \bot \leftarrow A$.

Theorem 7.6 *Let Rel be stable w.r.t. $\mathcal{L}_1$ and $\mathcal{F}_1 \cup \{(G|A)[0,1]\}$. Let the index set $\mathcal{C}''_{KB}$ be defined as $\mathcal{C}_{\mathcal{L}_2}(\mathcal{F}_2 \cup \{(G|A)[0,1]\})$. Let $KB \not\models \bot \leftarrow A$.*
Then, the tight answer substitution for Q to KB is given by $\{x_1/d_1,\, x_2/d_2\}$, where d_1 (resp., d_2) is the optimal value of the following linear program over the variables y_C ($C \in \mathcal{C}''_{KB}$):

$$\textit{minimize (resp., maximize)} \sum_{C \in \mathcal{C}''_{KB},\, C \preceq G \wedge A} y_C$$

subject to

$$\begin{aligned}
\sum_{C \in \mathcal{C}''_{KB},\, C \preceq A} y_C &= 1 \\
\sum_{C \in \mathcal{C}''_{KB},\, C \preceq \neg H \wedge B} -c_1\, y_C + \sum_{C \in \mathcal{C}''_{KB},\, C \preceq H \wedge B} (1-c_1)\, y_C &\geq 0 \quad \textit{(for all } (H|B)[c_1,c_2] \in \mathcal{F}_2,\, c_1 > 0) \\
\sum_{C \in \mathcal{C}''_{KB},\, C \preceq \neg H \wedge B} c_2\, y_C + \sum_{C \in \mathcal{C}''_{KB},\, C \preceq H \wedge B} (c_2-1)\, y_C &\geq 0 \quad \textit{(for all } (H|B)[c_1,c_2] \in \mathcal{F}_2,\, c_2 < 1) \\
y_C &\geq 0 \quad \textit{(for all } C \in \mathcal{C}''_{KB})\,.
\end{aligned} \tag{8}$$

7.3 Tight Probabilistic Logic Programming

We now present a reduced system of linear constraints and reduced linear programs for computing tight answer substitutions for object-ground probabilistic queries to probabilistic logic programs.

In the sequel, let P be a probabilistic logic program and let $Q = \exists (G|A)[x_1, x_2]$ be an object-ground query. Let CP (resp., PP) denote the set of all classical (resp., purely probabilistic) members in P.

We will apply the results of Section 7.2. We thus have to find some classical knowledge bases Rel' and

Rel'' that satisfy the stability preconditions of Theorems 7.4 and 7.6, respectively.

We first concentrate on the problem of deciding whether P logically entails $\bot \leftarrow A$.

As shown in Appendix D, the classical knowledge base $Rel' = \{\bot \leftarrow E \mid E \in HB_\Phi,\, \overline{P} \cup \{A\} \not\models E\}$ has the desired stability properties. Let $\underline{P}' = S^\star_{PP}(CP \cup Rel')$ and $\mathcal{I}'_P = \mathcal{I}_{\underline{P}'}(active_{\underline{P}'}(PP) \cup \{(A|\top)[0,1]\})$.

We are now ready to provide a reduced system of linear constraints for deciding whether P logically entails $\bot \leftarrow A$. It is is obtained from (7) by replacing the index set $\mathcal{C}'_P$ by the new index set $\mathcal{I}'_P$.

Theorem 7.7 *The probabilistic logic program P logically entails $\bot \leftarrow A$ iff the following system of linear constraints over the variables y_I $(I \in \mathcal{I}'_P)$ is unsolvable:*

$$\begin{aligned}
\sum_{I \in \mathcal{I}'_P,\, I \models A} y_I &= 1 \\
\sum_{I \in \mathcal{I}'_P,\, I \models \neg H \wedge B} -c_1\, y_I + \sum_{I \in \mathcal{I}'_P,\, I \models H \wedge B} (1-c_1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1,c_2] \in active_{\underline{P}'}(PP),\ c_1 > 0) \\
\sum_{I \in \mathcal{I}'_P,\, I \models \neg H \wedge B} c_2\, y_I + \sum_{I \in \mathcal{I}'_P,\, I \models H \wedge B} (c_2-1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1,c_2] \in active_{\underline{P}'}(PP),\ c_2 < 1) \\
y_I &\geq 0 \quad (\textit{for all } I \in \mathcal{I}'_P).
\end{aligned} \tag{9}$$

The following results are immediate extensions of Lemmata 6.5 and 6.6. In particular, they show that the system of linear constraints (9) is defined over all ground atoms that are logically entailed by $\overline{P} \cup \{A\}$.

Lemma 7.8 *For all $(H|B)[c_1,c_2] \in ground(PP)$:*

(i) $(H|B)[c_1,c_2] \in active_{\underline{P}'}(PP)$ *iff* $\underline{P}' \not\models \bot \leftarrow H \wedge B$, $\underline{P}' \not\models H \leftarrow B$, *and* $c_1 > 0$ *or* $c_2 < 1$.

(ii) $(H|B)[c_1,c_2] \in active_{\underline{P}'}(PP)$ *implies* $\overline{P} \cup \{A\} \models H \wedge B$.

Lemma 7.9 *It holds $I \subseteq T_{\overline{P}}\!\uparrow\omega(R(A))$ for all $I \in \mathcal{I}'_P$.*

We next focus on computing the requested tight answer substitution in the case $P \not\models \bot \leftarrow A$.

In this case, the classical knowledge base $Rel'' = \{\bot \leftarrow E \mid E \in HB_\Phi,\, \overline{P} \cup \{G \wedge A\} \not\models E\}$ has the desired stability properties. Let $\underline{P}'' = S^\star_{PP}(CP \cup Rel'')$ and $\mathcal{I}''_P = \mathcal{I}_{\underline{P}''}(active_{\underline{P}''}(PP) \cup \{(G|A)[0,1]\})$.

The following theorem now provides reduced linear programs for computing the requested tight answer substitution in the case $P \not\models \bot \leftarrow A$. These linear programs are obtained from (8) by replacing the index set $\mathcal{C}''_{KB}$ by the new index set $\mathcal{I}''_P$.

Theorem 7.10 *If the probabilistic logic program P does not logically entail $\bot \leftarrow A$, then the tight answer substitution for Q to P is given by $\sigma = \{x_1/d_1,\, x_2/d_2\}$, where d_1 (resp., d_2) is the optimal value of the following linear program over the variables y_I $(I \in \mathcal{I}''_P)$:*

$$\textit{minimize (resp., maximize)} \sum_{I \in \mathcal{I}''_P,\, I \models G \wedge A} y_I$$

subject to

$$\begin{aligned}
\sum_{I \in \mathcal{I}''_P,\, I \models A} y_I &= 1 \\
\sum_{I \in \mathcal{I}''_P,\, I \models \neg H \wedge B} -c_1\, y_I + \sum_{I \in \mathcal{I}''_P,\, I \models H \wedge B} (1-c_1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1,c_2] \in active_{\underline{P}''}(PP),\ c_1 > 0) \\
\sum_{I \in \mathcal{I}''_P,\, I \models \neg H \wedge B} c_2\, y_I + \sum_{I \in \mathcal{I}''_P,\, I \models H \wedge B} (c_2-1)\, y_I &\geq 0 \quad (\textit{for all } (H|B)[c_1,c_2] \in active_{\underline{P}''}(PP),\ c_2 < 1) \\
y_I &\geq 0 \quad (\textit{for all } I \in \mathcal{I}''_P).
\end{aligned} \tag{10}$$

The following results are immediate extensions of Lemmata 6.5 and 7.9. In particular, they show that the linear programs (10) are defined over all ground atoms that are logically entailed by $\overline{P} \cup \{G \wedge A\}$.

Lemma 7.11 *For all* $(H|B)[c_1, c_2] \in ground(PP)$*:*

(i) $(H|B)[c_1, c_2] \in active_{\underline{P}''}(PP)$ *iff* $\underline{P}'' \not\models \bot \leftarrow H \wedge B$*,* $\underline{P}'' \not\models H \leftarrow B$*, and* $c_1 > 0$ *or* $c_2 < 1$.

(ii) $(H|B)[c_1, c_2] \in active_{\underline{P}''}(PP)$ *implies* $\overline{P} \cup \{G \wedge A\} \models H \wedge B$.

Lemma 7.12 *It holds* $I \subseteq T_{\overline{P}}\!\uparrow\omega(R(G \wedge A))$ *for all* $I \in \mathcal{I}''_P$.

7.4 Example

We give an illustrative example.

Example 7.13 As in Example 7.2, let us concentrate again on the probabilistic logic program P of Example 3.1 and the object-ground probabilistic query $\exists(re(h,o) \mid ad(u,a))[X_1, X_2]$.

We now first have to decide whether P logically entails $\bot \leftarrow ad(u,a)$. By Theorem 7.7, this problem can be reduced to the problem of deciding whether the system of linear constraints (9) is unsolvable.

Interestingly, the new system of linear constraints (9) just consists of *six* linear constraints over *six* variables (and thus has the same size as (5)). In detail, the sets CP, PP, and $\overline{P}$ are the same as in Example 6.7. Moreover, as $T_{\overline{P}}\!\uparrow\omega(R(A)) = T_{\overline{P}}\!\uparrow\omega$, the set $\underline{P}'$ (resp., $active_{\underline{P}'}(PP)$) coincides with $\underline{P}$ (resp., $active_{\underline{P}}(PP)$) of Example 6.7. Finally, it turns out that also the index set $\mathcal{C}'_P$ (resp., $\mathcal{I}'_P$) coincides with the index set $\mathcal{C}_P$ (resp., $\mathcal{I}_P$) of Example 6.7. The system of linear constraints (9) is now given as follows:

$$\begin{aligned}
y_4 + y_5 &= 1 \\
-0.7 \cdot (y_0 + y_2) + 0.3 \cdot (y_1 + y_3 + y_4 + y_5) &\geq 0 \\
-0.7 \cdot (y_0 + y_1 + y_4) + 0.3 \cdot (y_2 + y_3 + y_5) &\geq 0 \\
-0.8 \cdot (y_0 + y_1 + y_2 + y_3) + 0.2 \cdot (y_4 + y_5) &\geq 0 \\
0.9 \cdot (y_0 + y_1 + y_2 + y_3) - 0.1 \cdot (y_4 + y_5) &\geq 0 \\
-0.9 \cdot (y_0 + y_1 + y_4) + 0.1 \cdot (y_2 + y_3 + y_5) &\geq 0 \\
y_i &\geq 0 \quad \text{(for all } i \in \{0, \ldots, 5\}\text{)}.
\end{aligned}$$

Since this system of linear constraints is solvable, P does not logically entail $\bot \leftarrow ad(u,a)$. By Theorem 7.10, the requested tight answer substitution is given by $\{X_1/d_1,\, X_2/d_2\}$, where d_1 (resp., d_2) is the optimal value of the linear program (10), which consists of only *six* linear constraints over *seven* variables:

The sets CP, PP, and $\overline{P}$ are the same as above. Moreover, since $T_{\overline{P}}\!\uparrow\omega(R(G \wedge A))$ is equal to $T_{\overline{P}}\!\uparrow\omega$, the set $\underline{P}''$ (resp., $active_{\underline{P}''}(PP)$) coincides with $\underline{P}$ (resp., $active_{\underline{P}}(PP)$) of Example 6.7.

The index set $\mathcal{C}''_P$ is given by the following ground classical formulas C''_i $(i \in \{0, \ldots, 6\})$:

$$\begin{aligned}
C''_0 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge \neg re(u,a) \wedge \neg re(a,o) \\
C''_1 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge re(u,a) \wedge \neg re(a,o) \\
C''_2 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge \neg re(u,a) \wedge re(a,o) \\
C''_3 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge \neg ad(u,a) \wedge re(u,a) \wedge re(a,o) \\
C''_4 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge ad(u,a) \wedge re(u,a) \wedge \neg re(a,o) \wedge \neg re(h,o) \\
C''_5 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge ad(u,a) \wedge re(u,a) \wedge \neg re(a,o) \wedge re(h,o) \\
C''_6 &= ro(u,a) \wedge ro(a,o) \wedge so(a,o) \wedge ad(u,a) \wedge re(u,a) \wedge re(a,o) \wedge re(h,o)\,.
\end{aligned}$$

Moreover, the index set $\mathcal{I}''_P$ is given by the following possible worlds I''_i $(i \in \{0, \ldots, 6\})$, written in binary as subsets of $T_{\overline{P}}\!\uparrow \omega$ (note that $I''_i \models C''_i$ for all $i \in \{0, \ldots, 6\}$):

$$I''_0 \hat{=} 111\ 101\ 100\ 000,\ I''_1 \hat{=} 111\ 101\ 110\ 100,\ I''_2 \hat{=} 111\ 101\ 101\ 000,\ I''_3 \hat{=} 111\ 101\ 111\ 111,$$
$$I''_4 \hat{=} 111\ 111\ 110\ 100,\ I''_5 \hat{=} 111\ 111\ 110\ 101,\ I''_6 \hat{=} 111\ 111\ 111\ 111\,.$$

The linear program (10) is now given as follows (note that each variable y_i corresponds to the ground classical formula C''_i and the possible world I''_i):

$$\text{minimize (resp., maximize)} \quad y_5 + y_6$$

subject to

$$\begin{aligned}
y_4 + y_5 + y_6 &= 1 \\
-0.7 \cdot (y_0 + y_2) + 0.3 \cdot (y_1 + y_3 + y_4 + y_5 + y_6) &\geq 0 \\
-0.7 \cdot (y_0 + y_1 + y_4 + y_5) + 0.3 \cdot (y_2 + y_3 + y_6) &\geq 0 \\
-0.8 \cdot (y_0 + y_1 + y_2 + y_3) + 0.2 \cdot (y_4 + y_5 + y_6) &\geq 0 \\
0.9 \cdot (y_0 + y_1 + y_2 + y_3) - 0.1 \cdot (y_4 + y_5 + y_6) &\geq 0 \\
-0.9 \cdot (y_0 + y_1 + y_4 + y_5) + 0.1 \cdot (y_2 + y_3 + y_6) &\geq 0 \\
y_i &\geq 0 \quad \text{(for all } i \in \{0, \ldots, 6\}).
\end{aligned}$$

Finally, as the optimal value of this linear program is given by 0.875 (resp., 1), the tight answer substitution for $\exists(re(h, o) \mid ad(u, a))[X_1, X_2]$ to P is given by $\{X_1/0.875,\ X_2/1\}$.

8 Algorithms and Complexity

We now briefly concentrate on the computational details of the techniques presented in Sections 6.3 and 7.3 for deciding whether a probabilistic logic program is satisfiable and for computing tight answer substitutions for object-ground queries to probabilistic logic programs.

In detail, to decide whether a probabilistic logic program is satisfiable, we generate the system of linear constraints (5) and check its solvability. Moreover, to compute tight answer substitutions for object-ground queries, we generate the system of linear constraints (9) and check whether it has a feasible solution. If this is the case, we also generate the two linear programs (10) and compute their optimal values.

The problems of deciding whether a system of linear constraints has a feasible solution and of computing the optimal value of a linear program can be solved with standard linear optimization techniques and are well-known to be tractable (see especially [58] and [64]). Recently, column generation techniques have been successfully applied to the context of probabilistic reasoning (see especially [34], [32], and [2]).

In the sequel, we thus concentrate only on the problems of generating our reduced systems of linear constraints and our reduced linear programs. Clearly, even though these linear optimization problems have a small size in the special case of few relevant purely probabilistic knowledge, we must always be aware of the fact they have an exponential size in the worst case (since the propositional and the data versions of SATISFIABILITY and TIGHT ENTAILMENT (OPT) are intractable in the conjunctive case).

That is, roughly speaking, the best that we can hope for is that the introduced systems of linear constraints and linear programs can be generated with small effort whenever they have a small size.

In the following, we restrict our considerations to the problem of deciding whether a probabilistic logic program is satisfiable. Note that all subsequent results can easily be extended to the problem of computing tight answer substitutions for object-ground queries.

In the sequel, let P be a probabilistic logic program. Let CP, PP, $\overline{P}$, Rel, and $\underline{P}$ be defined as in Section 6.3. Let $|PP|$ and $\|P\|$ denote the cardinality of PP and the input size of P, respectively.

The system of linear constraints (5) is given by the set $active_{\underline{P}}(PP)$ of all active purely probabilistic ground instances and the index set $\mathcal{I}_P$. Clearly, $active_{\underline{P}}(PP)$ can easily be computed along its definition. The following theorem shows that this can be done in polynomial time in the propositional case.

Theorem 8.1 *Given a ground probabilistic logic program P, computing the sets $\underline{P}$ and $active_{\underline{P}}(PP)$ can be done in time $O(|PP|^2\|P\|)$.*

Proof. We start by computing the sets CP, PP, and $\overline{P}$ in time $O(\|P\|)$. We then compute the set $T_{\overline{P}}\!\uparrow\omega$ of all ground atoms that are logically entailed by $\overline{P}$, which can also be done in time $O(\|P\|)$. Moreover, we compute $\underline{P}$ in time $O(|PP|^2\,\|P\|)$. Finally, we compute $active_{\underline{P}}(PP)$ in time $O(|PP|\,\|P\|)$. Thus, in summary, $\underline{P}$ and $active_{\underline{P}}(PP)$ can be computed in time $O(|PP|^2\|P\|)$. □

Computing the index set $\mathcal{I}_P$ of (5) is a more tricky task. Clearly, computing $\mathcal{I}_P$ as a whole takes exponential time in the input size of P in the worst case. Nonetheless, the next theorem shows that each member of $\mathcal{I}_P$ can be computed in polynomial time in the propositional case. That is, roughly speaking, when $\mathcal{I}_P$ is small, then it can also be computed with small efforts.

Theorem 8.2 *Let P be a ground probabilistic logic program. Given $\underline{P}$ and $active_{\underline{P}}(PP)$, each member of $\mathcal{I}_P$ can be computed in time $O(|PP|\,\|P\|)$.*

For the proof of Theorem 8.2, we need some preparation as follows. In the sequel, let $\mathcal{L}$ be a classical logic program and let $\mathcal{F}$ be a ground probabilistic logic program. We now show that $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$ can be expressed as the set of all fixpoints of a closure operator. Let $\mathcal{C}(\mathcal{F}) = \bigcup\{\{H, H \wedge B\} \mid (H|B)[c_1, c_2] \in \mathcal{F}\}$.

We define the operator H on the set of all subsets of $\mathcal{C}(\mathcal{F})$ as follows. For all $\mathcal{D} \subseteq \mathcal{C}(\mathcal{F})$:

$$H(\mathcal{D}) \;=\; \{C \in \mathcal{C}(\mathcal{F}) \mid \mathcal{L} \cup \mathcal{D} \models C\}\,.$$

The following lemma shows that H is a closure operator.

Lemma 8.3 *The operator H is a closure operator.*

Proof. It is easy to see that for all $\mathcal{C}, \mathcal{D} \subseteq \mathcal{C}(\mathcal{F})$, it holds (1) $\mathcal{C} \subseteq H(\mathcal{C})$, (2) $H(H(\mathcal{C})) \subseteq H(\mathcal{C})$, and (3) $\mathcal{C} \subseteq \mathcal{D}$ implies $H(\mathcal{C}) \subseteq H(\mathcal{D})$. □

The next result shows that each element of $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$ corresponds to a fixpoint of H.

Theorem 8.4 $\mathcal{I}_{\mathcal{L}}(\mathcal{F}) \;=\; \{\{A \in HB_\Phi \mid \mathcal{L} \cup \mathcal{C} \models A\} \mid \mathcal{C} \subseteq \mathcal{C}(\mathcal{F}),\ H(\mathcal{C}) = \mathcal{C},\ \mathcal{L} \cup \mathcal{C} \not\models \bot\}\,.$

Proof. Recall that $\mathcal{I}_{\mathcal{L}}(\mathcal{F}) = \{T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(\mathcal{C})) \mid \mathcal{C} \subseteq \mathcal{C}(\mathcal{F}),\ \mathcal{L} \cup \mathcal{C} \not\models \bot\}$. It is thus sufficient to show that for all $\mathcal{C} \subseteq \mathcal{C}(\mathcal{F})$, it holds $T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(\mathcal{C})) = T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(H(\mathcal{C})))$.

Let $\mathcal{C} \subseteq \mathcal{C}(\mathcal{F})$. By Lemma 8.3, it holds $\mathcal{C} \subseteq H(\mathcal{C})$. Hence, it follows $\bigcup R(\mathcal{C}) \subseteq \bigcup R(H(\mathcal{C}))$, and thus we get $T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(\mathcal{C})) \subseteq T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(H(\mathcal{C})))$. Conversely, by the definition of the operator H, it holds $\bigcup R(H(\mathcal{C})) \subseteq T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(\mathcal{C}))$. Hence, it follows $T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(H(\mathcal{C}))) \subseteq T_{\mathcal{L}}\!\uparrow\omega(\bigcup R(\mathcal{C}))$. □

Hence, the computation of $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$ can be reduced to the computation of the set of all fixpoints of the closure operator H. The latter can be done by an algorithm from [22]. Adopting this algorithm, we can compute $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$ by Algorithm index_set_1. The main idea behind this algorithm is that the members of $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$ are computed in a lexicographic order. We are now ready to prove Theorem 8.2.

Algorithm index_set_1

Input: Classical logic program $\mathcal{L}$ and ground probabilistic logic program $\mathcal{F}$.
Output: $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$.
Notation: We use $C_1, C_2, \ldots, C_n$ with $n \geq 0$ to denote the members of $\mathcal{C}(\mathcal{F})$.

if $\bot \in T_{\mathcal{L}}\!\uparrow\!\omega$ **then return** $\emptyset$;
$\mathcal{C} := H(\emptyset)$;
$\mathcal{I} := \{T_{\mathcal{L}}\!\uparrow\!\omega\}$;
while $\mathcal{C} \neq \mathcal{C}(\mathcal{F})$ **do begin**
  $k := n$;
  repeat
    $k := k - 1$;
    $\mathcal{D} := H((\mathcal{C} \cap \{C_1, C_2, \ldots, C_k\}) \cup \{C_{k+1}\})$
  until $C_{k+1} \not\in \mathcal{C}$ **and** $\mathcal{C} \cap \{C_1, C_2, \ldots, C_k\} = \mathcal{D} \cap \{C_1, C_2, \ldots, C_k\}$;
  if $\mathcal{L} \cup \mathcal{D}$ is satisfiable **then** $\mathcal{I} := \mathcal{I} \cup \{\{A \in HB_\Phi \mid \mathcal{L} \cup \mathcal{D} \models A\}\}$;
  $\mathcal{C} := \mathcal{D}$
end;
return $\mathcal{I}$.

Figure 6: Algorithm index_set_1

Algorithm index_set_2

Input: Classical knowledge base $\mathcal{L}$ and ground probabilistic knowledge base $\mathcal{F}$.
Output: $\mathcal{C}_{\mathcal{L}}(\mathcal{F})$.

if $\mathcal{L}$ is unsatisfiable **then return** $\emptyset$;
$\mathcal{C} := \{\top\}$;
for each $(H|B)[c_1, c_2] \in \mathcal{F}$ **do begin**
  $\mathcal{D} := \emptyset$;
  for each $C \in \mathcal{C}$ **do begin**
    if $\mathcal{L} \cup \{C \wedge \neg B\}$ is satisfiable **then** $\mathcal{D} := \mathcal{D} \cup \{C \wedge \neg B\}$;
    if $\mathcal{L} \cup \{C \wedge B \wedge \neg H\}$ is satisfiable **then** $\mathcal{D} := \mathcal{D} \cup \{C \wedge B \wedge \neg H\}$;
    if $\mathcal{L} \cup \{C \wedge B \wedge H\}$ is satisfiable **then** $\mathcal{D} := \mathcal{D} \cup \{C \wedge B \wedge H\}$
  end;
  $\mathcal{C} := \mathcal{D}$
end;
return $\mathcal{C}$.

Figure 7: Algorithm index_set_2

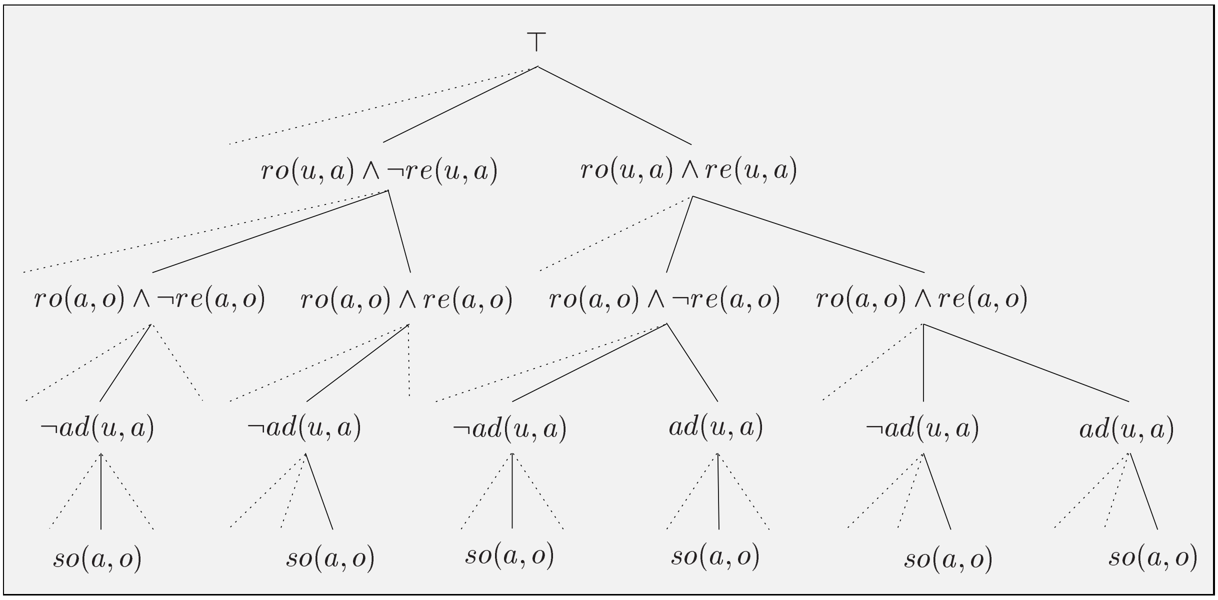

Figure 8: Semantic tree

Proof of Theorem 8.2. Given $\mathcal{L} = \underline{P}$ and $\mathcal{F} = active_{\underline{P}}(PP)$, the set $\mathcal{I}_P = \mathcal{I}_{\mathcal{L}}(\mathcal{F})$ can be computed by Algorithm index_set_1. In detail, the members of $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$ are computed in a strict order. That is, Steps 5–11 are performed exactly $|\mathcal{I}_{\mathcal{L}}(\mathcal{F})| - 1$ times [22]. Assuming standard data structures, all set-operations and all applications of the operator H can be done in time $O(\|P\|)$. Moreover, the satisfiability of $\mathcal{L} \cup \mathcal{D}$ in Step 10 can be checked in time $O(\|P\|)$. Finally, the set $T_{\mathcal{L}}\!\uparrow\omega$ in Steps 1 and 3, and the set $\{A \in HB_{\Phi} \mid \mathcal{L} \cup \mathcal{D} \models A\}$ in Step 10 can also be computed in time $O(\|P\|)$. Thus, in summary, the computation of each member of $\mathcal{I}_{\mathcal{L}}(\mathcal{F})$ in Steps 5–11 can be done in time $O(|PP|\,\|P\|)$. □

Finally, Algorithm index_set_2 shows an alternative way of computing $\mathcal{I}_P$. More precisely, it can be used to compute the set $\mathcal{C}_P = \mathcal{C}_{\underline{P}}(active_{\underline{P}}(PP))$. Then, for every $C \in \mathcal{C}_P$, the corresponding $I \in \mathcal{I}_P$ with $I \models C$ is given by the least model I of $\underline{P} \cup \{C\}$.

Example 8.5 Consider again the probabilistic logic program P of Example 6.7. The computation of $\mathcal{C}_P$ by index_set_2 is illustrated by the *semantic tree* shown in Fig. 8. Each $C \in \mathcal{C}_P$ corresponds to a path from the root $\top$ to some leaf, and each dashed line stands for a detected unsatisfiability in Steps 6–8 of index_set_2.

9 Relationship to van Emden's Quantitative Deduction

As shown in Section 4, the tractability of special cases of classical logic programming generally does not carry over to the corresponding special cases of probabilistic logic programming. A closely related observation is that the model and fixpoint characterization and the proof theory of classical logic programming generally do not directly carry over to probabilistic logic programming. It is in general more natural to give a linear optimization characterization of probabilistic logic programming.

In this section, we will see that the probabilistic semantics in its full generality is responsible for this computational difference between classical and probabilistic logic programming. That is, we just have to focus on more specific probabilistic interpretations in order to bring probabilistic logic programming closer to classical logic programming. In detail, we just have to increase the axioms of probability by an additional axiom that brings probabilistic logics closer to truth-functional logics.

Under this more restricted probabilistic semantics, the conditional probability associated with each probabilistic program clause depends truth-functionally on the probabilities of the contained atoms. As a consequence, ordinary probabilistic logic programs under this new semantics have similar computational properties as classical logic programs. In detail, ordinary probabilistic logic programming under this semantics has a special-case tractability similar to classical logic programming. Moreover, it has a model and fixpoint characterization and a proof theory similar to classical logic programming.

Interestingly, this fixpoint characterization is well-known in the literature as *van Emden's quantitative deduction* [67], which is an important approach to *truth-functional many-valued logic programming*. We thus show that van Emden's quantitative logic programs can be given a semantics in which probabilities are defined over a set of possible worlds. To our knowledge, such a probabilistic semantics of van Emden's quantitative logic programs is unknown in the literature so far.

This fundamental relationship to van Emden's quantitative deduction is important in its own right. Moreover, it provides us with another important result: We will see that van Emden's quantitative deduction can be used as an *efficient approximation* of probabilistic logic programming.

9.1 PCP-Interpretations

We define a special kind of probabilistic interpretations as follows. A *positively correlated probabilistic interpretation (pcp-interpretation)* is a probabilistic interpretation Pr such that:

$$Pr(A \wedge B) = \min(Pr(A), Pr(B)) \text{ for all ground atoms } A, B \in HB_\Phi \,. \tag{11}$$

Note that (11) just refers to ground atoms $A, B \in HB_\Phi$. We next introduce the notions of satisfiability, logical consequence, and tight logical consequence for positively correlated probabilistic interpretations.

A set of conditional constraints $\mathcal{F}$ is *satisfiable under positive correlation* (*pc-satisfiable*) iff a positively correlated model of $\mathcal{F}$ exists. A conditional constraint F is a *consequence under positive correlation* (*pc-consequence*) of $\mathcal{F}$, denoted $\mathcal{F} \models^{pc} F$, iff each positively correlated model of $\mathcal{F}$ is also a model of F. A conditional constraint $(H|B)[c_1, c_2]$ is a *tight consequence under positive correlation* (*tight pc-consequence*) of $\mathcal{F}$, denoted $\mathcal{F} \models^{pc}_{tight} (H|B)[c_1, c_2]$, iff c_1 (resp., c_2) is the infimum (resp., supremum) of $Pr_\sigma(H|B)$ subject to all positively correlated models Pr of $\mathcal{F}$ and all variable assignments σ such that $Pr_\sigma(B) > 0$.

Clearly, pc-satisfiability is a sufficient condition for general satisfiability.

Theorem 9.1 *Let KB be a probabilistic knowledge base.*
If KB is pc-satisfiable, then KB is satisfiable.

The next immediate result shows that tight pc-consequences approximate tight logical consequences. More precisely, each tight interval that is entailed under positive correlation is contained in the corresponding tight interval that is logically entailed.

Theorem 9.2 *Let KB be a probabilistic knowledge base and let $(H|B)[c_1, c_2]$ be a conditional constraint.*
If $KB \models^{pc}_{tight} (H|B)[c_1, c_2]$, then $KB \models_{tight} (H|B)[d_1, d_2]$ with some $[d_1, d_2] \supseteq [c_1, c_2]$.

In particular, if 0 (resp., 1) is the entailed tight lower (resp., upper) bound under positive correlation, then 0 (resp., 1) is also the logically entailed tight lower (resp., upper) bound.

Corollary 9.3 *Let KB be a probabilistic knowledge base and let $(H|B)[c_1, c_2]$ be a conditional constraint.*

a) If $KB \models^{pc}_{tight} (H|B)[0, c_2]$, then $KB \models_{tight} (H|B)[0, d_2]$ with some $d_2 \geq c_2$.

b) If $KB \models^{pc}_{tight} (H|B)[c_1, 1]$, then $KB \models_{tight} (H|B)[d_1, 1]$ with some $d_1 \leq c_1$.

9.2 Properties of PCP-Interpretations

We now analyze some properties of pcp-interpretations.

The following theorem shows that condition (11) is equivalent to the assumption of a subset relationship between possible worlds.

Theorem 9.4 *Let Pr be a pcp-interpretation. Let $\mathcal{I}^+_\Phi$ be defined as $\{I \in \mathcal{I}_\Phi \mid Pr(I) > 0\}$, and for each ground atom $A \in HB_\Phi$, let $\mathcal{I}^+_\Phi(A)$ be defined as $\{I \in \mathcal{I}^+_\Phi \mid I \models A\}$. Then, condition (11) is equivalent to each of the following conditions (12) and (13):*

$$\mathcal{I}^+_\Phi(A) \subseteq \mathcal{I}^+_\Phi(B) \ \textit{or}\ \mathcal{I}^+_\Phi(A) \supseteq \mathcal{I}^+_\Phi(B) \qquad \textit{for all } A, B \in HB_\Phi \tag{12}$$

$$I_1 \subseteq I_2 \ \textit{or}\ I_1 \supseteq I_2 \qquad \textit{for all } I_1, I_2 \in \mathcal{I}^+_\Phi\,. \tag{13}$$

Proof. The claim follows immediately from the observation that each condition among (11), (12), and (13) is equivalent to the following condition: for all $A, B \in HB_\Phi$ and all $I_1, I_2 \in \mathcal{I}^+_\Phi$, it is not the case that $I_1 \models A$, $I_1 \not\models B$, $I_2 \not\models A$, and $I_2 \models B$. □

The next theorem shows that the probability of certain classical formulas under pcp-interpretations is truth-functionally defined on the probabilities of their components. Note that the truth functions are the same as in the nondenumerable infinite-valued Łukasiewicz logic $Ł_{\aleph_1}$ (see [63] for a survey).

Theorem 9.5 *Let Pr be a pcp-interpretation and let σ be a variable assignment. For all classical formulas F, and all classical formulas G and H that are built without the logical connectives $\neg$ and $\leftarrow$:*

$$Pr_\sigma(\neg F) = 1 - Pr_\sigma(F) \tag{14}$$

$$Pr_\sigma(G \wedge H) = \min(Pr_\sigma(G), Pr_\sigma(H)) \tag{15}$$

$$Pr_\sigma(G \vee H) = \max(Pr_\sigma(G), Pr_\sigma(H)) \tag{16}$$

$$Pr_\sigma(G \leftarrow H) = \min(1, Pr_\sigma(G) - Pr_\sigma(H) + 1)\,. \tag{17}$$

Proof. The claim (14) is immediate by the definition of probabilistic interpretations. The other claims are proved as follows. By induction, using (12), it can easily be shown that for all ground classical formulas G built without $\neg$ and $\leftarrow$, there exists some $A \in HB_\Phi$ with $\mathcal{I}^+_\Phi(G) = \mathcal{I}^+_\Phi(A)$. The claims then follow from (12) and the definition of probabilistic interpretations. □

Finally, the following result shows that pcp-interpretations are already uniquely determined by the probabilities they assign to all ground atoms.

Theorem 9.6 *Every mapping $f\colon HB_\Phi \rightarrow [0, 1]$ determines a unique pcp-interpretation Pr such that $Pr(A) = f(A)$ for all $A \in HB_\Phi$.*

Proof. Let $V = \{0, 1\} \cup \{f(A) \mid A \in HB_\Phi\}$. Let $\langle l_1, u_1\rangle, \ldots, \langle l_k, u_k\rangle$ be an enumeration of all pairs $\langle l_i, u_i\rangle \in V \times V$ such that $l_i < u_i$ and $(l_i, u_i) \cap V = \emptyset$ for all $i \in [1:k]$. For each $i \in [1:k]$, we define the possible world I_i as $\{A \in HB_\Phi \mid f(A) \geq u_i\}$. The desired pcp-interpretation Pr is now given by $Pr(I_i) = u_i - l_i$ for all $I_i \in \mathcal{I}_\Phi$, and $Pr(I) = 0$ for all the other $I \in \mathcal{I}_\Phi$. $\Box$

That is, we can identify pcp-interpretations with mappings $f\colon HB_\Phi \to [0,1]$. Since such mappings can be viewed as fuzzy sets [69], we get a natural subset relation on pcp-interpretations. For pcp-interpretations Pr_1 and Pr_2, we say Pr_1 is a *subset* of Pr_2, denoted $Pr_1 \subseteq Pr_2$, iff $Pr_1(A) \leq Pr_2(A)$ for all $A \in HB_\Phi$. We use $Pr_1 \subset Pr_2$ as an abbreviation of $Pr_1 \subseteq Pr_2$ and $Pr_1 \neq Pr_2$. Finally, the *intersection* (resp., *union*) of Pr_1 and Pr_2, denoted $Pr_1 \cap Pr_2$ (resp., $Pr_1 \cup Pr_2$), is defined as $\min(Pr_1, Pr_2)$ (resp., $\max(Pr_1, Pr_2)$).

9.3 Conjunctive Conditional Constraints under PCP-Interpretations

The following result shows that the truth of conjunctive conditional constraints in pcp-interpretations depends truth-functionally on the probabilities of the contained atoms.

Theorem 9.7 *Let Pr be a pcp-interpretation and let σ be a variable assignment. For all conditional constraints $(H_1 \wedge \cdots \wedge H_k | B_1 \wedge \cdots \wedge B_l)[c_1, c_2]$ with atoms $H_1, \ldots, H_k, B_1, \ldots, B_l$ and $k, l \geq 0$:*

$$Pr \models_\sigma (H_1 \wedge \cdots \wedge H_k | B_1 \wedge \cdots \wedge B_l)[c_1, c_2] \text{ iff}$$
$$c_1 \cdot \min_{1 \leq i \leq l} Pr_\sigma(B_i) \;\leq\; \min(\min_{1 \leq i \leq k} Pr_\sigma(H_i), \min_{1 \leq i \leq l} Pr_\sigma(B_i)) \;\leq\; c_2 \cdot \min_{1 \leq i \leq l} Pr_\sigma(B_i)\,.$$

Note that for $k = 0$ (resp., $l = 0$), we naturally define $H_1 \wedge \cdots \wedge H_k$ (resp., $B_1 \wedge \cdots \wedge B_l$) as $\top$. Note also that we canonically define the minimum of an empty list of arguments as 1.

Proof. The claim follows immediately from Theorem 9.5. $\Box$

The next result shows that, under pcp-interpretations, positive conjunctive conditional constraints can be translated into ordinary conditional constraints.

Lemma 9.8 *Let Pr be a pcp-interpretation and let σ be a variable assignment. For all conditional constraints $(H_1 \wedge \cdots \wedge H_k | B_1 \wedge \cdots \wedge B_l)[c_1, 1]$ with atoms $H_1, \ldots, H_k, B_1, \ldots, B_l$, $k \geq 1$, and $l \geq 0$:*

$$Pr \models_\sigma (H_1 \wedge \cdots \wedge H_k | B_1 \wedge \cdots \wedge B_l)[c_1, 1] \text{ iff}$$
$$Pr \models_\sigma (H_i | B_1 \wedge \cdots \wedge B_l)[c_1, 1] \text{ for all } i \in [1 : k]\,.$$

Proof. The claim follows immediately from Theorem 9.7. $\Box$

The following result concentrates on the truth of ordinary conditional constraints.

Lemma 9.9 *Let Pr be a pcp-interpretation and let σ be a variable assignment. For all conditional constraints $(H | B_1 \wedge \cdots \wedge B_k)[c_1, 1]$ with atoms $H, B_1, \ldots, B_k$ and $k \geq 0$:*

$$Pr \models_\sigma (H | B_1 \wedge \cdots \wedge B_k)[c_1, 1] \text{ iff } c_1 \cdot \min_{1 \leq i \leq k} Pr_\sigma(B_i) \;\leq\; Pr_\sigma(H)\,.$$

Proof. Since $c_1 \cdot \min(Pr_\sigma(B_1), \ldots, Pr_\sigma(B_k)) \leq \min(Pr_\sigma(B_1), \ldots, Pr_\sigma(B_k))$, the claim follows immediately from Theorem 9.7. $\Box$

9.4 Ordinary Probabilistic Logic Programs under PCP-Interpretations

We now show that van Emden's quantitative deduction [67] provides a model and fixpoint semantics for ordinary probabilistic logic programs under pcp-interpretations. Moreover, we show that the special-case tractability of classical logic programming in the propositional and the data case carries over to ordinary probabilistic logic programming under pcp-interpretations.

9.4.1 Model and Fixpoint Semantics

In the sequel, let P be an ordinary probabilistic logic program.

We first recall the definition of van Emden's fixpoint operator $L_P\!\uparrow\omega$ [67]. Let the monotonic immediate consequence operator L_P on pcp-interpretations Pr be defined as follows. For all $H \in HB_\Phi$:

$$L_P(Pr)[H] \;=\; \max(\{c_1 \cdot \min_{1\leq i\leq k} Pr(B_i) \mid (H|B_1 \wedge \cdots \wedge B_k)[c_1, 1] \in \mathit{ground}(P)\} \cup \{0\})\,.$$

For all pcp-interpretations Pr, we define $L_P\!\uparrow\omega(Pr)$ as the union of all $L_P\!\uparrow n(Pr)$ with $n < \omega$, where $L_P\!\uparrow 0(Pr) = Pr$ and $L_P\!\uparrow(n+1)(Pr) = L_P(L_P\!\uparrow n(Pr))$ for all $n < \omega$. We use $L_P\!\uparrow\omega$ to abbreviate $L_P\!\uparrow\omega(\mathbf{0})$, where $\mathbf{0}$ is the pcp-interpretation that maps each $A \in HB_\Phi$ to 0.

We next show that L_P makes sense for ordinary probabilistic logic programs under pcp-interpretations, that is, that each pcp-interpretation that satisfies P corresponds to a pre-fixpoint of L_P.

Lemma 9.10 *A pcp-interpretation Pr is a model of P iff $L_P(Pr) \subseteq Pr$.*

Proof. The claim follows immediately from Lemma 9.9. □

The following theorem is the first part of the model and fixpoint semantics of ordinary probabilistic logic programs under pcp-interpretations. It says that ordinary probabilistic logic programs P have a least positively correlated model, which coincides with the least fixpoint of L_P.

Theorem 9.11 $\bigcap\{Pr \mid Pr \text{ is a positively correlated model of } P\} \;=\; \mathit{lfp}(L_P)\,.$

Proof. The monotonic operator L_P has a least fixpoint $\mathit{lfp}(L_P)$, which is equal to its least pre-fixpoint. By Lemma 9.10, this least pre-fixpoint is equal to $\bigcap\{Pr \mid Pr$ is a positively correlated model of $P\}$. □

The second part of the model and fixpoint semantics of ordinary probabilistic logic programs under pcp-interpretations goes directly back to van Emden [67]. It follows from the continuity of L_P and says that the least fixpoint of L_P is attained at the first limit ordinal.

Theorem 9.12 (Van Emden [67]) $\mathit{lfp}(L_P) \;=\; L_P\!\uparrow\omega\,.$

We finally get the following characterization of tight pc-consequences.

Theorem 9.13 *Let $H = H_1 \wedge \cdots \wedge H_k$ be a classical formula with ground atoms $H_1, \ldots, H_k$ and $k \geq 1$. Then, $P \models^{pc}_{\mathit{tight}} (H|\top)[d_1, 1]$, where $d_1 = \min_{1\leq i\leq k} L_P\!\uparrow\omega[H_i]$.*

Proof. The claim follows immediately from Theorems 9.5, 9.11, and 9.12. □

9.4.2 Computational Complexity

The following theorem shows the propositional and the data tractability of computing entailed tight intervals for ordinary probabilistic logic programs under pcp-interpretations. Trivially, this shows also the propositional and the data tractability of deciding entailment and of deciding tight entailment for ordinary probabilistic logic programs under pcp-interpretations.

Theorem 9.14 *a) Given a ground ordinary probabilistic logic program P and an object-ground ordinary probabilistic query $Q = \exists(H|\top)[x_1, x_2]$, computing the tight answer substitution for Q to P under positive correlation can be done in polynomial time.*

b) Let P be a fixed ordinary probabilistic logic program. Given a finite set of ground probabilistic facts $\mathcal{F}$ and an object-ground ordinary probabilistic query $Q = \exists(H|\top)[x_1, x_2]$, computing the tight answer substitution for Q to $P \cup \mathcal{F}$ under positive correlation can be done in polynomial time.

Proof. a) By Theorem 9.13, it is sufficient to show that $L_P\!\uparrow\omega$ can be computed in polynomial time. But this is indeed the case, since each application of L_P runs in polynomial time in the size of P, and since the number of applications of L_P to reach $L_P\!\uparrow\omega$ is bounded by the number of ground atoms in P. To see the latter, assume that $L_P\!\uparrow\omega$ contains a pair (A, p) that is not contained in $L_P\!\uparrow m$, where m is the number of ground atoms in P. Hence, the derivation tree of (A, p) must contain at least one path from a leaf to the root (A, p) with first a node (B, p_1) and thereafter another node (B, p_2), where B is a ground atom and $p_1, p_2 \in [0, 1]$ with $p_1 < p_2$. However, the immediate consequence operator L_P shows that the probabilities are decreasing along every path from a leaf to the root. That is, we also get $p_1 \geq p_2$.

b) The statement follows immediately from a), since the number of all ground instances of conditional constraints in $P \cup \mathcal{F}$, which are constructed with all constant symbols from $P \cup \mathcal{F}$ and Q, is polynomial in the input size of $\mathcal{F}$ and Q. □

9.5 Approximative Solutions to Satisfiability and Tight Entailment

We now show that the fixpoint semantics of ordinary probabilistic logic programs under positive correlation yields approximative solutions to SATISFIABILITY and TIGHT ENTAILMENT in the conjunctive case.

In the sequel, let P be a probabilistic logic program. We define its *ordinary approximation* $P^\star$ as the set of all $(A|B)[c_1, 1]$ for which there exists some $(H|B)[c_1, c_2] \in P$ such that A is an atom contained in H.

Clearly, P is pc-satisfiable, if the least positively correlated model of $P^\star$ is also a model of P.

Theorem 9.15 *If $L_{P^\star}\!\uparrow\omega \models P$, then P is pc-satisfiable.*

Hence, we also get a sufficient condition for the satisfiability of P.

Corollary 9.16 *If $L_{P^\star}\!\uparrow\omega \models P$, then P is satisfiable.*

The next immediate result shows that ordinary probabilistic logic programming under positive correlation with $P^\star$ is an approximation of probabilistic logic programming with P. More precisely, it provides an upper (resp., lower) bound for the tight lower (resp., upper) bound that is logically entailed by P. Note that the precondition that $L_{P^\star}\!\uparrow\omega$ is a model of P can be removed for ordinary P.

Theorem 9.17 *Let $H = H_1 \wedge \cdots \wedge H_k$ with atoms $H_1, \ldots, H_k$ and $k \geq 1$. Let $d = \min_{1 \leq i \leq k} L_{P^\star}\!\uparrow\omega[H_i]$.*

If $L_{P^\star}\!\uparrow\omega \models P$, then $P \models_{tight} (H|\top)[d_1, d_2]$ with some $d_1 \in [0, d]$ and $d_2 \in [d, 1]$.

In particular, if 0 (resp., 1) is the tight lower (resp., upper) bound entailed by $P^\star$ under positive correlation, then 0 (resp., 1) is the tight lower (resp., upper) bound entailed by P.

Corollary 9.18 *Let $H = H_1 \wedge \cdots \wedge H_k$ with atoms $H_1, \ldots, H_k$ and $k \geq 1$.*

a) If $L_{P^\star}\!\uparrow\omega \models P$ and $\min_{1 \leq i \leq k} L_{P^\star}\!\uparrow\omega[H_i] = 0$, then $P \models_{tight} (H|\top)[0, d_2]$ with some $d_2 \in [0, 1]$.

b) If $L_{P^\star}\!\uparrow\omega \models P$ and $\min_{1 \leq i \leq k} L_{P^\star}\!\uparrow\omega[H_i] = 1$, then $P \models_{tight} (H|\top)[d_1, 1]$ with some $d_1 \in [0, 1]$.

9.6 Example

We now give an illustrative example.

Example 9.19 Consider again the probabilistic logic program P of Example 3.1.

The ordinary approximation $P^\star$ of P is given as follows:

$$\begin{aligned} P^\star \;=\; &\{(ro(h,u)\,|\,\top)[1,1], (ro(u,a)\,|\,\top)[1,1], (ro(a,o)\,|\,\top)[1,1], (ad(h,u)\,|\,\top)[1,1],\\ &(ad(u,a)\,|\,\top)[0.8,1], (so(a,o)\,|\,\top)[1,1], (re(R,S)\,|\,ro(R,S))[0.7,1],\\ &(re(R,S)\,|\,ro(R,S)\wedge so(R,S))[0.9,1], (re(R,S)\,|\,ro(R,S)\wedge ad(R,S))[1,1],\\ &(re(R,S)\,|\,re(R,T)\wedge re(T,S))[1,1]\}\,. \end{aligned}$$

The fixpoint iteration and the resulting fixpoint $L_{P^\star}\!\uparrow\omega$ is shown in Table 4 (we assume $L_{P^\star}\!\uparrow\alpha[A] = 0$ for all ground atoms A that are not mentioned).

Table 4: Fixpoint iteration

	$ro(h,u)$	$ro(u,a)$	$ro(a,o)$	$ad(h,u)$	$ad(u,a)$	$so(a,o)$	$re(h,u)$	$re(u,a)$	$re(a,o)$	$re(h,a)$	$re(u,o)$	$re(h,o)$
$L_{P^\star}\!\uparrow 0$	0	0	0	0	0	0	0	0	0	0	0	0
$L_{P^\star}\!\uparrow 1$	1	1	1	1	0.8	1	0	0	0	0	0	0
$L_{P^\star}\!\uparrow 2$	1	1	1	1	0.8	1	1	0.8	0.9	0	0	0
$L_{P^\star}\!\uparrow 3$	1	1	1	1	0.8	1	1	0.8	0.9	0.8	0.8	0
$L_{P^\star}\!\uparrow 4$	1	1	1	1	0.8	1	1	0.8	0.9	0.8	0.8	0.8
$L_{P^\star}\!\uparrow\omega$	1	1	1	1	0.8	1	1	0.8	0.9	0.8	0.8	0.8

It is now easy to verify that $L_{P^\star}\!\uparrow\omega$ is a model of P. By Theorem 9.15, it thus follows that P is satisfiable under positive correlation. Hence, P is satisfiable.

It holds $L_{P^\star}\!\uparrow\omega[re(h,o)] = 0.8$. Hence, by Theorem 9.17, it follows that $P \models_{tight} (re(h,o)|\top)[d_1, d_2]$ with some $d_1 \in [0, 0.8]$ and $d_2 \in [0.8, 1]$. Recall that, in fact, $P \models_{tight} (re(h,o)|\top)[0.7, 1]$.

10 Logical Consequence in Probabilistic Logics

Even though a great number of publications around logics of probability are based on the model-theoretic notion of logical consequence, it has been argued in the literature that this notion of entailment is actually too weak. For this reason, already Nilsson [56] suggested to apply the principle of maximum entropy to obtain a stronger notion of entailment. This criticism also seems to apply to our probabilistic logic programs, as they are based only on the model-theoretic notion of entailment.

Indeed, this criticism is justified if we adhere completely to the classical notion of conditioning. For example, given the knowledge that "all penguins are birds" and that "every bird has legs with probability of at least 0.95", we can only conclude that "the probability of penguins having legs is between 0 and 1".

However, our approach to probabilistic logic programming is not restricted to the modeling of some concrete evidence as conditioning event. More precisely, it also allows to express concrete evidence in the form of *probabilistic facts*, which are naturally imported from classical logic programming. That is, for example, given again the knowledge that "all penguins are birds" and that "every bird has legs with probability of at least 0.95", and a concrete scenario in which Tweety is a penguin, our approach allows to conclude in a correct way that "the probability of Tweety having legs is at least 0.95".

That is, we can circumvent the weakness of the model-theoretic notion of logical consequence by modeling concrete evidence as probabilistic facts. This is easily possible for probabilistic logic programs that contain only probabilistic Horn clauses of the form $(H|B)[c, 1]$. In general, however, a satisfiable set of probabilistic Horn clauses might be inconsistent under a collection of probabilistic facts. This is due to the fact that several incompatible probabilistic rules might be applicable to the same concrete scenario. Then, we have to remove all those probabilistic rules that should not be applied anymore, which can be done by a probabilistic generalization of classical default reasoning from conditional knowledge bases [49].

Finally, depending on our application, we can also use the stronger notion of consequence under positive correlation or the stronger notion of consequence under maximum entropy. The former carries us immediately to nice computational properties (as shown in Section 9), while the latter can be realized by a generalization of the technique introduced in Section 7.3 [50]. Note also that there is a refinement of the principle of maximum entropy that encodes Bayesian-network-oriented conditional independencies [48].

11 Summary and Conclusion

We presented a new approach to probabilistic logic programming with a possible worlds semantics in which classical program clauses are extended by a subinterval of $[0, 1]$ that describes a range for the conditional probability of the head of a clause given its body. We then analyzed the computational complexity of selected probabilistic logic programming tasks. It turned out that probabilistic logic programming is computationally more complex than classical logic programming. Moreover, we also drew a precise picture of the complexity of deciding and computing tight logical consequences in probabilistic reasoning with conditional constraints in general. We then presented linear optimization techniques for deciding satisfiability and computing tight logical consequences of probabilistic logic programs. These techniques are efficient in the special case in which we have few relevant purely probabilistic knowledge. We finally showed that ordinary probabilistic logic programming under positive correlation is characterized by van Emden's quantitative deduction, and thus has computational properties similar to classical logic programming. Based on this result, we presented an efficient approximation technique for probabilistic logic programming.

This paper sheds a positive light on the efficient feasibility of probabilistic logic programming. In particular, the introduced linear optimization techniques for deciding satisfiability and computing tight logical consequences show an efficient feasibility of probabilistic logic programming in special cases. Moreover, the presented approximation technique based on van Emden's quantitative deduction shows that we can obtain the computational features of classical logic programming by giving up some semantic precision.

This paper is a step towards understanding the computational properties of probabilistic reasoning with conditional constraints in general and of probabilistic logic programming in particular. In particular, we discovered that probabilistic logic programming and truth-functional many-valued logic programming are closely related through an axiom that brings probabilistic logic closer to truth-functional logics. In detail, this axiom separates intractability results and linear optimization techniques, on the one hand, from tractability results and classical logic programming techniques, on the other hand.

Finally, we saw that positively correlated probabilistic interpretations combine a probabilistic semantics with nice computational features (see also [45, 46, 44]). For this reason, they are well worth being studied more deeply. This is a very interesting topic of future research.

A Appendix: Proofs for Section 4

Proof of Theorem 4.1. We show that the two problems of deciding whether KB is satisfiable and of deciding whether No is the correct answer for Q to KB polynomially reduce to the NP-complete problem of deciding whether a weight formula is satisfiable in a measurable probability structure [18].

More precisely, *weight formulas* are defined as Boolean combinations of *basic weight formulas*, which are expressions $a_1 \cdot w(\phi_1) + \cdots + a_k \cdot w(\phi_k) \geq a$ with integers $a_1, \ldots, a_k, a$ and propositional formulas $\phi_1, \ldots, \phi_k$. A *measurable probability structure* can be identified with a probability function on the finite set of all truth assignments to the primitive propositions, which is extended in a natural way to propositional formulas, basic weight formulas, and weight formulas.

Each conditional constraint is now translated into a conjunction of two basic weight formulas. In detail, for each $F = (H|B)[p_1/q_1,\, p_2/q_2]$, we define $bwf_1(F)$ and $bwf_2(F)$ as follows:

$$\begin{aligned} bwf_1(F) &= ((-p_1) \cdot w(\neg H \wedge B) + (q_1 - p_1) \cdot w(H \wedge B) \;\geq\; 0)\,, \\ bwf_2(F) &= (p_2 \cdot w(\neg H \wedge B) + (p_2 - q_2) \cdot w(H \wedge B) \;\geq\; 0)\,. \end{aligned}$$

It is then easy to see that KB is satisfiable iff the following weight formula is satisfiable:

$$\bigwedge_{F \in KB} (bwf_1(F) \wedge bwf_2(F))\,.$$

Moreover, KB does not logically entail $(G|A)[c_1, c_2]$ iff the following weight formula is satisfiable:

$$(\bigwedge_{F \in KB} (bwf_1(F) \wedge bwf_2(F))) \wedge \neg(bwf_1((G|A)[c_1, c_2]) \wedge bwf_2((G|A)[c_1, c_2]))\,.$$

The former proves a), while the latter shows b). $\Box$

Proof of Theorem 4.5. We apply some basic results from linear programming (cf. especially [58, 10, 64, 25]) to the characterization of tight logical consequence given in Theorem 4.4. Without loss of generality, we prove the claim for $d = d_1$. Let LP denote the minimization instance of (3).

By Theorem 4.4, the optimal value of LP is given by d_1. We now transform LP into an equivalent linear program LP', which contains only linear equations with integer coefficients, as follows. For each conditional constraint $F = (H|B)[p_1/q_1,\, p_2/q_2]$ in KB, the pair of linear inequalities

$$\begin{aligned} \sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} -(p_1/q_1)\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (1 - p_1/q_1)\, y_I &\;\geq\; 0 \\ \sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} (p_2/q_2)\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (p_2/q_2 - 1)\, y_I &\;\geq\; 0 \end{aligned}$$

is replaced by the following pair of linear equations:

$$\begin{aligned} \sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} -p_1\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (q_1 - p_1)\, y_I + y_{F'} &= 0 \\ \sum_{I \in \mathcal{I}_\Phi,\, I \models \neg H \wedge B} p_2\, y_I + \sum_{I \in \mathcal{I}_\Phi,\, I \models H \wedge B} (p_2 - q_2)\, y_I + y_{F''} &= 0\,, \end{aligned}$$

where $y_{F'} \geq 0$ and $y_{F''} \geq 0$ are two new variables. Let n denote the number of variables in LP'. Obviously,

$m = 2c + 1$ is the number of linear equations in LP'. Moreover, it is easy to see that the absolute value of each coefficient and constant in LP' has a size of at most l (note that we have 0, 1, and all integers of the form $-p_1$, $(q_1 - p_1)$, p_2, and $(p_2 - q_2)$ from $(H|B)[p_1/q_1,\, p_2/q_2] \in KB$ as coefficients and constants).

The set of all feasible solutions of LP' is nonempty and bounded. By a fundamental theorem of linear programming, the optimal value d_1 of LP' is attained at a feasible solution with at most m entries positive (this is the main idea behind the simplex method). That is, LP' has a feasible solution $\boldsymbol{r} = (r_K)$, where $K \in \mathcal{I}_\Phi \cup \bigcup\{\{F', F''\} \mid F \in KB\}$, with at most m entries positive such that:

$$d_1 \;=\; \sum_{I \in \mathcal{I}_\Phi,\, I \models G \wedge A} r_I\,. \tag{18}$$

By Cramer's rule, each positive entry r_K is of the form $|B| \,/\, |C|$, where $|B|, |C| > 0$ are the determinants of two $h \times h$ matrices B and C with $h \leq m$ and some coefficients and constants from LP' as entries. Recall that the determinant $|A|$ of an $h \times h$ matrix $A = (a_{i,j})$ is defined by

$$|A| \;=\; \sum_{\pi \in \mathcal{S}} sgn(\pi)\, a_{1,\pi(1)}\, a_{2,\pi(2)} \cdots a_{h,\pi(h)}\,,$$

where $\mathcal{S}$ denotes the set of all permutations of the members in $\{1, 2, \ldots, h\}$ and sgn associates each $\pi \in \mathcal{S}$ with an element of $\{-1, +1\}$. Given that all entries in A have a size of at most l, we get:

$$|A| \;\leq\; \sum_{\pi \in \mathcal{S}} |a_{1,\pi(1)}|\, |a_{2,\pi(2)}| \cdots |a_{h,\pi(h)}| \;\leq\; \sum_{\pi \in \mathcal{S}} (2^l - 1)^h \;\leq\; h^h\, 2^{lh}\,.$$

Hence, in each positive entry r_K, both the nominator and the denominator are at most $h^h\, 2^{lh} \leq m^m\, 2^{lm}$. Thus, by (18), both the nominator and the denominator of $d_1 \leq 1$ are at most $(m^m\, 2^{lm})^m$. Hence, d_1 has a size of at most $1 + lm^2 + \lfloor m^2 \log m \rfloor$. This already shows b).

For the proof of a), define r as the sum of all r_I with $I \in \mathcal{I}_\Phi$. It follows that both the nominator and the denominator of $r \geq 1$ are at most $m\,(m^m\, 2^{lm})^m$.

Define Pr_1 by $Pr_1(I) = r_I/r$ for all $I \in \mathcal{I}_\Phi$. It is now easy to see that Pr_1 is a model of KB with $Pr_1(A) > 0$ and $Pr_1(G \wedge A) = d_1 \cdot Pr_1(A)$. Moreover, the number of all $I \in \mathcal{I}_\Phi$ with $Pr_1(I) > 0$ is at most m, and the size of each $Pr_1(I) > 0$ is at most $1 + lm^2 + m + \lfloor (m^2 + m + 1) \log m \rfloor$. This shows a). □

Proof of Theorem 4.6 (continued). It remains to show that the computation in Steps 2–4 is in NP. Assume first that KB has two models Pr_1 and Pr_2 with $Pr_i(A) > 0$ and $Pr_i(G \wedge A) = d_i \cdot Pr_i(A)$ for all $i \in \{1, 2\}$. Thus, $KB \not\models \bot \leftarrow A$ and $KB \models_{tight} (G|A)[d'_1, d'_2]$ with some $d'_1 \in [0, d_1]$ and $d'_2 \in [d_2, 1]$. Hence, by Theorem 4.5, there are two "small" models Pr'_1 and Pr'_2 of KB such that $Pr'_i(A) > 0$ and $Pr'_i(G \wedge A) = d'_i \cdot Pr'_i(A)$ for all $i \in \{1, 2\}$. Conversely, if KB has two "small" models Pr'_1 and Pr'_2 of KB such that $Pr'_1(A) > 0$, $Pr'_2(A) > 0$, $Pr'_1(G \wedge A) \,/\, Pr'_1(A) \in [0, d_1]$, and $Pr'_2(G \wedge A) \,/\, Pr'_2(A) \in [d_2, 1]$, then KB has two models Pr_1 and Pr_2 with $Pr_i(A) > 0$ and $Pr_i(G \wedge A) = d_i \cdot Pr_i(A)$ for all $i \in \{1, 2\}$ (since the set of all feasible solutions of (3) is closed under convex combinations).

Hence, we just have to guess and verify two "small" models Pr'_1 and Pr'_2 of KB such that $Pr'_1(A) > 0$, $Pr'_2(A) > 0$, $Pr'_1(G \wedge A) \,/\, Pr'_1(A) \in [0, d_1]$, and $Pr'_2(G \wedge A) \,/\, Pr'_2(A) \in [d_2, 1]$. This can be done in nondeterministic polynomial time. That is, Steps 2–4 are in NP. □

Proof of Theorem 4.7 (continued). Step 1 of Algorithm tight-entailment-opt checks whether $KB \models \bot \leftarrow A$. If this is the case, then we return the tight answer substitution $\sigma = \{x_1/1,\, x_2/0\}$. Otherwise, we go through Steps 2–20. In Steps 2–9, we use binary search to compute two intervals $[l_1, l_2], [u_1, u_2] \subseteq [0, 1]$

such that $d_1 \in [l_1, l_2]$ and $d_2 \in [u_1, u_2]$. Note that the length of the computed intervals is $1/2^{2k+1}$. It is now easy to see that $|r_1 - r_2| \geq 1/2^{2k}$ for any two distinct rational numbers r_1 and r_2 with $\|r_1\|, \|r_2\| \leq k$. By Theorem 4.5 b), we thus know that $d_1 = p/q$ and $d_2 = r/s$ are uniquely determined through $[l_1, l_2]$ and $[u_1, u_2]$, respectively. In Steps 10–17, we then compute q and s by binary search. In Steps 18–20, we finally compute d_1 and d_2, and return the tight answer substitution $\sigma = \{x_1/d_1,\, x_2/d_2\}$.

It finally remains to show that Algorithm tight-entailment-opt can be done in deterministic polynomial time with a polynomial number of NP oracle calls. By Theorem 4.1 b), deciding whether $KB \models \bot \leftarrow A$ can be done with one call to an NP oracle. Consider now the iteration in Steps 4–9. By induction on the number of iteration steps, it can be shown that the conditions in Steps 5 and 7 are equivalent to $d_1 \in [l_1, (l_1 + l_2)\,/\,2)$ and $d_2 \in ((u_1 + u_2)\,/\,2, u_2]$, respectively. Hence, by Theorem 4.5 a), the conditions in 5 and 7 can be checked in nondeterministic polynomial time by guessing and verifying two "small" models Pr_1 and Pr_2 of KB such that $Pr_1(A) > 0$, $Pr_2(A) > 0$, $Pr_1(G \wedge A)\,/\,Pr_1(A) \in [l_1, (l_1 + l_2)\,/\,2)$, and $Pr_2(G \wedge A)\,/\,Pr_2(A) \in ((u_1 + u_2)\,/\,2, u_2]$. Thus, the iteration in 4–9 can be done in deterministic polynomial time with a polynomial number of NP oracle calls. Finally, it is easy to see that the same applies to the iteration in 12–17. □

Proof of Theorem 4.11 (continued). It remains to show that $l = w$.

We first show $l \leq w$ by providing a model Pr of P such that $Pr(p) = w$. This model Pr is constructed as a refinement of a model Pr_1 of P_1 as follows.

Let π be the permutation of a tour with the smallest length d. For each $k \in V$, let $\pi^k = \pi \circ \sigma^{k-1}$, where $\sigma^0 = \mathrm{id}$ and $\sigma^i = \sigma \circ \sigma^{i-1}$ for all $i > 0$. For each $k \in V$, define $I^k \in \mathcal{I}_{\Phi_1}$ as follows:

$$I^k \;=\; \{p_{v,i} \in \Phi_1 \mid \pi^k(v) = i\}\,.$$

We now define Pr_1 by $Pr_1(I) = 1/n$ if $I \in \{I^1, \ldots, I^n\}$ and $Pr_1(I) = 0$ otherwise. Clearly, Pr_1 is a probabilistic interpretation on $\mathcal{I}_{\Phi_1}$. Moreover, $Pr_1 \models P_{1,1}$, $Pr_1 \models P_{1,2}$, and $Pr_1 \models P_{1,3}$, since each π^k is a mapping, each π^k is injective, and each $p_{i,j}$ is contained in some possible world among $I^1, \ldots, I^n$. Hence, Pr_1 is a model of P_1.

We next refine Pr_1 to the model Pr of P. For all $J \in \mathcal{I}_\Phi$, define $E(J)$ as the set of all $\{i,j\} \in E$ for which there exists some $u \in V$ such that $J \models p_{u,i} \wedge p_{\sigma(u),j}$. Define Pr on $\mathcal{I}_\Phi$ as follows:

$$Pr(J) \;=\; \begin{cases} w_e \cdot Pr_1(I^k) & \text{if } J = I^k \cup \{p, p_e\} \text{ for some } k \in V \text{ and } e \in E(I^k) \\ (1 - \sum\limits_{e \in E(I^k)} w_e) \cdot Pr_1(I^k) & \text{if } J = I^k \text{ for some } k \in V \\ 0 & \text{otherwise.} \end{cases}$$

Since Pr refines Pr_1, it is a probabilistic interpretation on $\mathcal{I}_\Phi$ and in particular a model of P_1. Moreover, Pr is clearly a model of $P_{2,1}$ and $P_{2,3}$. We next show that Pr is also a model of $P_{2,2}$. Let $\{i,j\} \in E$. Assume first that $\{\pi(v), \pi(\sigma(v))\} = \{i,j\}$ for some $v \in V$. Then, for each $u \in V$, there exists exactly one $k \in V$ with $I^k \models p_{u,i} \wedge p_{\sigma(u),j}$. It thus follows:

$$Pr(p_{\{i,j\}} \wedge p_{u,i} \wedge p_{\sigma(u),j}) \;=\; w_{\{i,j\}} \cdot Pr_1(I^k) \;=\; w_{\{i,j\}} \cdot Pr(p_{u,i} \wedge p_{\sigma(u),j})\,.$$

Assume next that $\{\pi(v), \pi(\sigma(v))\} \neq \{i,j\}$ for all $v \in V$. It then follows $I^k \not\models p_{u,i} \wedge p_{\sigma(u),j}$ for all $u, k \in V$. Hence, we get:

$$Pr(p_{\{i,j\}} \wedge p_{u,i} \wedge p_{\sigma(u),j}) \;=\; 0 \;=\; w_{\{i,j\}} \cdot 0 \;=\; w_{\{i,j\}} \cdot Pr(p_{u,i} \wedge p_{\sigma(u),j})\,.$$

Hence, Pr is a model of $P_{2,2}$, and thus also of P. Let us finally define $E(\pi)$ as the set of all $\{\pi(u), \pi(\sigma(u))\}$ with $u \in V$ (that is, $E(\pi)$ describes the tour given by π). It then follows:

$$Pr(p) \;=\; \sum_{e\in E} Pr(p_e) \;=\; \sum_{e\in E(\pi)} \sum_{k\in V} w_e \cdot Pr_1(I^k) \;=\; \sum_{e\in E(\pi)} w_e \cdot \sum_{k\in V} Pr_1(I^k) \;=\; w\,.$$

This shows $l \leq w$.

We next prove $l \geq w$ by showing that $Pr(p) \geq w$ for all models Pr of P with smallest $Pr(p)$. We first show that all such models Pr of P are also models of $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \mathcal{H}_3 \cup \mathcal{H}_4$, where

$$\begin{aligned}
\mathcal{H}_1 &= \{(\bigvee_{e\in E} p_e \,|\, p)[1,1]\}\,,\\
\mathcal{H}_2 &= \{(\bigvee_{u\in V} ((p_{u,i} \wedge p_{\sigma(u),j}) \vee (p_{u,j} \wedge p_{\sigma(u),i})) \,|\, p_{\{i,j\}})[1,1] \mid \{i,j\} \in E,\, i<j\}\,,\\
\mathcal{H}_3 &= \{(p_{u,i} \wedge p_{\sigma(u),j} \,|\, p_{v,i} \wedge p_{\sigma(v),j})[0,0] \mid u,v \in V,\, u<v,\, \{i,j\} \in E\}\,,\\
\mathcal{H}_4 &= \{(p_{u,i} \wedge p_{\sigma(u),j} \,|\, p_{v,j} \wedge p_{\sigma(v),i})[0,0] \mid u,v \in V,\, \{i,j\} \in E,\, i<j\}\,.
\end{aligned}$$

Let Pr be a model of P with smallest $Pr(p)$. Assume first that Pr is not a model of $\mathcal{H}_1$. That is, there exists some possible world $J \in \mathcal{I}_\Phi$ such that $Pr(J) > 0$, $J \models p$, and $J \not\models p_e$ for all $e \in E$. Define now the probabilistic interpretation Pr' on $\mathcal{I}_\Phi$ as follows:

$$Pr'(I) \;=\; \begin{cases} Pr(I) & \text{if } I \neq J \text{ and } I \neq J - \{p\}\\ Pr(J) + Pr(J - \{p\}) & \text{if } I = J - \{p\}\\ 0 & \text{if } I = J\,. \end{cases}$$

We now show that Pr' is a model of P. Clearly, Pr' is a model of all elements in $P_1 \cup P_{2,1} \cup P_{2,2}$, since they do not contain the predicate symbol p. Moreover, Pr' is a model of $P_{2,3}$, since $J \not\models p_e$ for all $e \in E$. However, it clearly holds $Pr'(p) < Pr(p)$, and this contradicts Pr being a model of P with smallest $Pr(p)$.

Assume next that Pr is not a model of $\mathcal{H}_2$. That is, there exists a possible world $J \in \mathcal{I}_\Phi$ such that $Pr(J) > 0$, $J \models p_{\{i,j\}}$ for some $\{i,j\} \in E$, and $J \not\models p_{u,i} \wedge p_{\sigma(u),j}$ for all $u \in V$. Define now the probabilistic interpretation Pr' on $\mathcal{I}_\Phi$ as follows:

$$Pr'(I) \;=\; \begin{cases} Pr(I) & \text{if } I \neq J \text{ and } I \neq J - \{p, p_{\{i,j\}}\}\\ Pr(J) + Pr(J - \{p, p_{\{i,j\}}\}) & \text{if } I = J - \{p, p_{\{i,j\}}\}\\ 0 & \text{if } I = J\,. \end{cases}$$

We now show that that Pr is a model of P. Clearly, Pr is a model of all elements in P that do not contain p and $p_{\{i,j\}}$. It thus remains to show that Pr is a model of $P'_{2,1} \cup P'_{2,2} \cup P'_{2,3}$, where

$$\begin{aligned}
P'_{2,1} &= \bigcup\{\{(p_{\{i,j\}} \,|\, p_e)[0,0], (p_e \,|\, p_{\{i,j\}})[0,0]\} \mid e \in E,\, e \neq \{i,j\}\}\,,\\
P'_{2,2} &= \{(p_{\{i,j\}} \,|\, p_{u,i} \wedge p_{\sigma(u),j})[w_{\{i,j\}}, w_{\{i,j\}}] \mid u \in V\}\,,\\
P'_{2,3} &= \{(p \,|\, p_e)[1,1] \mid e \in E\}\,.
\end{aligned}$$

Obviously, Pr' is a model of $P'_{2,1} \cup P'_{2,3}$, since $J \not\models p_e$ for all $e \in E - \{\{i,j\}\}$. Moreover, Pr' is a model of $P'_{2,2}$, since $J \not\models p_{u,i} \wedge p_{\sigma(u),j}$ for all $u \in V$. However, it holds $Pr'(p) < Pr(p)$, and this contradicts again the assumption that Pr is a model of P with smallest $Pr(p)$.

Suppose now that Pr is not a model of $\mathcal{H}_3$. That is, there exists a possible world $J \in \mathcal{I}_\Phi$ such that $Pr(J) > 0$ and $J \models p_{u,i} \wedge p_{\sigma(u),j} \wedge p_{v,i} \wedge p_{\sigma(v),j}$ for some $\{i,j\} \in E$ and $u, v \in V$ with $u < v$. But this contradicts the assumption that Pr is a model of $P_{1,2}$.

Suppose finally that Pr is not a model of $\mathcal{H}_4$. That is, there exists a possible world $J \in \mathcal{I}_\Phi$ such that $Pr(J) > 0$ and $J \models p_{u,i} \wedge p_{\sigma(u),j} \wedge p_{v,j} \wedge p_{\sigma(v),i}$ for some $u, v \in V$ and $\{i,j\} \in E$ with $i < j$. Since Pr is a model of $P_{1,2}$, it then follows $u = \sigma(v)$ and $\sigma(u) = v$. But this contradicts $n \geq 3$.

Hence, Pr is a model of $\mathcal{H}$. It thus follows:

$$Pr(p) \;=\; \sum_{\{i,j\} \in E} Pr(p_{\{i,j\}}) \;=\; \sum_{\{i,j\} \in E} \sum_{v \in V} Pr(p_{v,i} \wedge p_{\sigma(v),j}) + Pr(p_{v,j} \wedge p_{\sigma(v),i})\,. \tag{19}$$

We next define a model Pr' of P with $Pr'(p) = Pr(p)$. Define the probabilistic interpretation Pr_1 on $\mathcal{I}_{\Phi_1}$ by $Pr_1(I) = \sum_{J \in \mathcal{I}_\Phi, J \supseteq I} Pr(J)$ for all $I \in \mathcal{I}_{\Phi_1}$. Define $\mathcal{I}^+_{\Phi_1} = \{I \in \mathcal{I}_{\Phi_1} \mid Pr_1(I) > 0\}$. Define now the probabilistic interpretation Pr' on $\mathcal{I}_\Phi$ by refining Pr_1 as follows:

$$Pr'(J) \;=\; \begin{cases} w_e \cdot Pr_1(I) & \text{if } J = I \cup \{p, p_e\} \text{ for some } I \in \mathcal{I}^+_{\Phi_1} \text{ and } e \in E(I) \\ (1 - \sum\limits_{e \in E(I)} w_e) \cdot Pr_1(I) & \text{if } J = I \text{ for some } I \in \mathcal{I}^+_{\Phi_1} \\ 0 & \text{otherwise.} \end{cases}$$

We now show that Pr' is a model of $P \cup \mathcal{H}$. Since Pr' is a refinement of Pr_1, it is immediately a model of $P_1 \cup \mathcal{H}_3 \cup \mathcal{H}_4$. Clearly, Pr' is also a model of $P_{2,1} \cup P_{2,3} \cup \mathcal{H}_1 \cup \mathcal{H}_2$. Finally, to show that Pr' is a model of $P_{2,2}$, let $u \in V$ and $\{i,j\} \in E$. Assume first that there is some $J \in \mathcal{I}_\Phi$ such that $Pr'(J) > 0$ and $J \models p_{\{i,j\}} \wedge p_{u,i} \wedge p_{\sigma(u),j}$. It then follows:

$$Pr'(p_{\{i,j\}} \wedge p_{u,i} \wedge p_{\sigma(u),j}) \;=\; \sum_{I \in \mathcal{I}^+_{\Phi_1},\; I \models p_{u,i} \wedge p_{\sigma(u),j}} w_{\{i,j\}} \cdot Pr_1(I) \;=\; w_{\{i,j\}} \cdot Pr'(p_{u,i} \wedge p_{\sigma(u),j})\,.$$

Assume next that $J \not\models p_{\{i,j\}} \wedge p_{u,i} \wedge p_{\sigma(u),j}$ for all $J \in \mathcal{I}_\Phi$ with $Pr'(J) > 0$. It is then immediate that $J \not\models p_{u,i} \wedge p_{\sigma(u),j}$ for all $J \in \mathcal{I}_\Phi$ with $Pr'(J) > 0$. Thus, we get:

$$Pr'(p_{\{i,j\}} \wedge p_{u,i} \wedge p_{\sigma(u),j}) \;=\; 0 \;=\; w_{\{i,j\}} \cdot 0 \;=\; w_{\{i,j\}} \cdot Pr'(p_{u,i} \wedge p_{\sigma(u),j})\,.$$

Hence, Pr' is a model of $P_{2,2}$, and thus also of $P \cup \mathcal{H}$. Moreover, we get:

$$Pr'(p) \;=\; \sum_{\{i,j\} \in E} Pr'(p_{\{i,j\}}) \;=\; \sum_{\{i,j\} \in E} \sum_{v \in V} Pr'(p_{v,i} \wedge p_{\sigma(v),j}) + Pr'(p_{v,j} \wedge p_{\sigma(v),i})\,. \tag{20}$$

By (19) and (20), since $Pr(p_{v,i} \wedge p_{\sigma(v),j}) = Pr'(p_{v,i} \wedge p_{\sigma(v),j})$ for all $v \in V$ and $\{i,j\} \in E$, we thus get $Pr(p) = Pr'(p)$.

We finally show $Pr'(p) \geq w$. Each $J \in \mathcal{I}_\Phi$ with $Pr'(J) > 0$ defines a permutation π_J of the members in V by $\pi_J(v) = i$ iff $p_{v,i} \in J$, for all $v \in V$. For each such J, define $d(J)$ as the length of the tour associated with π_J (that is, $d(J)/s$ is the sum of all w_e with $e \in E(J)$). It then follows:

$$\begin{aligned} Pr'(p) &= \sum_{I \in \mathcal{I}^+_{\Phi_1}} \Big(\sum_{e \in E(I)} w_e \cdot Pr_1(I) \Big) = \sum_{I \in \mathcal{I}^+_{\Phi_1}} \Big(\sum_{e \in E(I)} w_e \Big) \cdot Pr_1(I) \\ &= \sum_{I \in \mathcal{I}^+_{\Phi_1}} d(I)/s \cdot Pr_1(I) \geq \sum_{I \in \mathcal{I}^+_{\Phi_1}} w \cdot Pr_1(I) = w \cdot \sum_{I \in \mathcal{I}^+_{\Phi_1}} Pr_1(I) = w\,. \end{aligned}$$

Summarizing the results, given a model Pr of P such that $Pr(p) = l$, we have constructed a model Pr' of P such that $Pr(p) = Pr'(p)$ and $Pr'(p) \geq w$. This shows $l \geq w$. $\Box$

Proof of Theorem 4.12. a) We give a polynomial transformation from the NP-complete *graph 3-colorability* problem [23]: Given a finite undirected graph $G = (V, E)$, decide whether G is 3-colorable.

Roughly speaking, the main idea is to use $\mathcal{F}$ in order to encode G. We now construct P and $\mathcal{F}$ as in the statement of the theorem such that G is 3-colorable iff $P \cup \mathcal{F}$ is satisfiable.

The vocabulary Φ is defined as follows:

$$\Phi = V \cup \{q_1, q_2, q_3, v, e\}\,,$$

where the elements of V are constant symbols, q_1, q_2, q_3, and v are 1-ary predicate symbols, and e is a 2-ary predicate symbol. We define the fixed program $P = P_1 \cup P_2 \cup P_3$ as follows:

$$\begin{aligned} P_1 &= \{(q_j(R) \,|\, q_k(R) \wedge v(R))[0,0] \mid j,k \in \{1,2,3\},\ j < k\}\,, \\ P_2 &= \{(q_j(R) \,|\, q_j(S) \wedge e(R,S))[0,0] \mid j \in \{1,2,3\}\}\,, \\ P_3 &= \{(q_j(R) \,|\, v(R))[1/3,1/3] \mid j \in \{1,2,3\}\}\,. \end{aligned}$$

We define the varying set of facts $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ as follows:

$$\begin{aligned} \mathcal{F}_1 &= \{(v(i) \,|\, \top)[1,1] \mid i \in V\}\,, \\ \mathcal{F}_2 &= \{(e(i,j) \,|\, \top)[1,1] \mid \{i,j\} \in E\}\,. \end{aligned}$$

We now show that $P \cup \mathcal{F}$ is satisfiable iff $P_1 \cup P_2'' \cup P_3 \cup \mathcal{F}$ is satisfiable, where P_2'' is given by:

$$P_2'' = \{(q_j(i) \,|\, q_j(k) \wedge e(i,k))[0,0] \mid \{i,k\} \in E,\ j \in \{1,2,3\}\}\,.$$

Suppose first that $P \cup \mathcal{F}$ is satisfiable. As $ground(P_2) \supseteq P_2''$, it follows that $P_1 \cup P_2'' \cup P_3 \cup \mathcal{F}$ is satisfiable. Conversely, assume that $P_1 \cup P_2'' \cup P_3 \cup \mathcal{F}$ is satisfiable. Let Pr be any model of $P_1 \cup P_2'' \cup P_3 \cup \mathcal{F}$. Let $\Phi_1 = V \cup \{q_1, q_2, q_3\}$ and let the probabilistic interpretation Pr_1 on $\mathcal{I}_{\Phi_1}$ be defined by $Pr_1(I) = \sum_{J \in \mathcal{I}_\Phi,\, J \supseteq I} Pr(I)$ for all $I \in \mathcal{I}_{\Phi_1}$. We next define the probabilistic interpretation Pr' on $\mathcal{I}_\Phi$ as follows. If $I \in \mathcal{I}_\Phi$ is of the form $I_1 \cup I_2$ with $I_1 \in \mathcal{I}_{\Phi_1}$ and $I_2 = \{v(i) \,|\, i \in V\} \cup \{e(i,k) \mid \{i,k\} \in E\}$, then $Pr'(I) = Pr_1(I_1)$; otherwise, $Pr'(I) = 0$. It is now easy to verify that Pr' is a model of $P \cup \mathcal{F}$.

Clearly, $P_1 \cup P_2'' \cup P_3 \cup \mathcal{F}$ is satisfiable iff $P_1' \cup P_2' \cup P_3'$ is satisfiable, where P_1', P_2', and P_3' are given by:

$$\begin{aligned} P_1' &= \{(q_j(i) \,|\, q_k(i))[0,0] \mid i \in V,\ j,k \in \{1,2,3\},\ j < k\}\,, \\ P_2' &= \{(q_j(i) \,|\, q_j(k))[0,0] \mid \{i,k\} \in E,\ j \in \{1,2,3\}\}\,, \\ P_3' &= \{(q_j(i) \,|\, \top)[1/3,1/3] \mid i \in V,\ j \in \{1,2,3\}\}\,. \end{aligned}$$

Finally, it is easy to see that $P_1' \cup P_2' \cup P_3'$ is satisfiable iff G is 3-colorable (see [24] and [59]). In summary, it follows that $P \cup \mathcal{F}$ is satisfiable iff G is 3-colorable.

b) Immediate by a), since $P \cup \mathcal{F}$ is unsatisfiable iff $P \cup \mathcal{F} \models (H|\top)[1,0]$, where H is any ground atom.

c) We give a polynomial transformation from the following D^{P}-complete problem [68]. Given a finite undirected graph $G = (V, E)$, decide whether G has the chromatic number 7. The line of argumentation is a combination of the proof of Theorem 4.10 with the proof of b).

We construct P, $\mathcal{F}$, and $Q = \exists(H|\top)[x_1, x_2]$ as in the statement of the theorem such that the chromatic number of G is 7 iff the tight answer substitution for Q to $P \cup \mathcal{F}$ is given by $\sigma = \{x_1/0,\ x_2/0\}$.

The vocabulary Φ is defined as follows:

$$\Phi \;=\; V \cup \{p\} \cup \{p_j \mid j \in \{1, \ldots, 6\}\} \cup \{q_j \mid j \in \{1, \ldots, 7\}\} \cup \{v, e\}\,,$$

where the members of V act as constant symbols, p is a 0-ary predicate symbol, p_j, $\boldsymbol{q}_j$, and v are 1-ary predicate symbols, and e is a 2-ary predicate symbol. We define the fixed program P as the union of two programs P_1 and P_2. In detail, $P_1 = P_{1,1} \cup P_{1,2} \cup P_{1,3}$ is defined as follows:

$$\begin{aligned}
P_{1,1} &= \{(q_j(R) \mid q_k(R) \wedge v(R))[0,0] \mid j,k \in \{1,\ldots,7\},\ j<k\}\,,\\
P_{1,2} &= \{(q_j(R) \mid q_j(S) \wedge e(R,S))[0,0] \mid j \in \{1,\ldots,7\}\}\,,\\
P_{1,3} &= \{(q_j(R) \mid v(R))[1/7,1/7] \mid j \in \{1,\ldots,7\}\}\,.
\end{aligned}$$

Moreover, $P_2 = P_{2,1} \cup P_{2,2} \cup P_{2,3}$ is defined as follows:

$$\begin{aligned}
P_{2,1} &= \{(p_j(R) \mid p_k(R) \wedge v(R))[0,0] \mid j,k \in \{1,\ldots,6\},\ j<k\}\,,\\
P_{2,2} &= \{(p_j(R) \mid p_j(S) \wedge e(R,S))[0,0] \mid j \in \{1,\ldots,6\}\}\,,\\
P_{2,3} &= \{(p_j(R) \mid p \wedge v(R))[1/6,1/6] \mid j \in \{1,\ldots,6\}\}\,.
\end{aligned}$$

Finally, the varying set of facts $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ is defined as follows:

$$\begin{aligned}
\mathcal{F}_1 &= \{(v(i) \mid \top)[1,1] \mid i \in V\}\,,\\
\mathcal{F}_2 &= \{(e(i,j) \mid \top)[1,1] \mid \{i,j\} \in E\}\,.
\end{aligned}$$

As in the proofs of Theorem 4.10 and b), it can be shown that (i) $P_1 \cup \mathcal{F}$ is satisfiable iff G is 7-colorable, (ii) $P_2 \cup \mathcal{F}$ is always satisfiable, and (iii) $P_2 \cup \mathcal{F} \models (p \mid \top)[0,0]$ iff G is not 6-colorable.

Moreover, $P_1 \cup P_2 \cup \mathcal{F} \models_{\mathit{tight}} (p \mid \top)[0,0]$ iff $P_1 \cup P_2 \cup \mathcal{F}$ is satisfiable and $P_1 \cup P_2 \cup \mathcal{F} \models (p \mid \top)[0,0]$. It is easy to see that the latter is equivalent to $P_1 \cup \mathcal{F}$ being satisfiable and $P_2 \cup \mathcal{F} \models (p \mid \top)[0,0]$. By (i) and (iii), this is in turn equivalent to G being 7-colorable, but not 6-colorable. In summary, the chromatic number of G is 7 iff the tight answer substitution for $Q = \exists(p \mid \top)[x_1, x_2]$ to $P_1 \cup P_2$ is given by $\sigma = \{x_1/0,\ x_2/0\}$. $\Box$

B Appendix: Proofs for Section 5

Proof of Theorem 5.1. a) Let the probabilistic interpretation Pr be defined by $Pr(I) = 1$ for $I = HB_\Phi$ and by $Pr(I) = 0$ for all the other $I \in \mathcal{I}_\Phi$. It is now easy to verify that Pr is a model of P.

b) By polynomial transformation from the complement of the NP-complete *graph 3-colorability* problem [23]: Given a finite undirected graph $G = (V, E)$, decide whether G is 3-colorable. We can assume $V \neq \emptyset$.

We now construct P and $Q = \exists(H|\top)[d_1, d_2]$ as in the statement of the theorem such that G is not 3-

colorable iff $P \models (H|\top)[d_1, d_2]$. In detail, the vocabulary Φ contains only 0-ary predicate symbols and is defined as $\{p\} \cup \{p_{i,j} \mid i \in V,\ j \in \{1,2,3\}\}$. The program $P = P_1 \cup P_2 \cup P_3$ is defined as follows:

$$\begin{aligned}
P_1 &= \{(p \mid p_{i,j} \wedge p_{i,k})[1,1] \mid i \in V,\ j,k \in \{1,2,3\},\ j < k\}\,,\\
P_2 &= \{(p \mid p_{i,j} \wedge p_{k,j})[1,1] \mid \{i,k\} \in E,\ j \in \{1,2,3\}\}\,,\\
P_3 &= \{(p_{i,j} \mid \top)[1/3,1] \mid i \in V,\ j \in \{1,2,3\}\}\,.
\end{aligned}$$

Let c denote the cardinality of P and let l be the greatest size of a rational number in P. Let $m = 2c + 1$, $a = 1 \,/\, 2^{1+lm^2+\lfloor m^2 \log m\rfloor}$, and $Q = \exists(p|\top)[a,1]$. By Theorem 4.5, all models Pr of P satisfy $Pr(p) > 0$ iff all models Pr of P satisfy $Pr(p) \geq a$. The former is equivalent to G not being 3-colorable (see [24] and [59]), while the latter is equivalent to $P \models (p \mid \top)[a,1]$. That is, G is not 3-colorable iff $P \models (p \mid \top)[a,1]$.

c) We give a polynomial transformation from the NP-complete *graph 3-colorability* problem [23]: Given a finite undirected graph $G = (V,E)$, decide whether G is 3-colorable. We can assume $V \neq \emptyset$.

Let the vocabulary Φ and the program P be defined as in the proof of b). Then, G is 3-colorable iff there is a model Pr of P with $Pr(p) = 0$ (see [24] and [59]). The latter is equivalent to $P \models_{\mathit{tight}} (p \mid \top)[0,1]$. $\Box$

C Appendix: Proofs for Section 6

For the proofs of Theorems 6.2 and 6.3, we need some preparative results.

In the sequel, let $\mathcal{L}$ be a classical knowledge base, let $\mathcal{F}$ be a ground probabilistic knowledge base, and let $\mathcal{C} = \mathcal{C}_{\mathcal{L}}(\mathcal{F})$. We first show that $\mathcal{F}$ can be interpreted by a probability function over a partition of $\{I \in \mathcal{I}_\Phi \mid I \models \mathcal{L}\}$. That is, we generally do not need the fine granulation of $\{I \in \mathcal{I}_\Phi \mid I \models \mathcal{L}\}$.

For each $C \in \mathcal{C}$, let $S_C = \{I \in \mathcal{I}_\Phi \mid I \models \mathcal{L} \cup \{C\}\}$. Let $\mathcal{S}_{\mathcal{C}}$ be defined as $\{S_C \mid C \in \mathcal{C}\}$. The following result is immediate by the definition of $\mathcal{C}_{\mathcal{L}}(\mathcal{F})$.

Lemma C.1 *If $\mathcal{C} \neq \emptyset$, then the family $\mathcal{S}_{\mathcal{C}}$ is a partition of $\{I \in \mathcal{I}_\Phi \mid I \models \mathcal{L}\}$.*

In the sequel, we implicitly assume $\mathcal{C} \neq \emptyset$. We next introduce the notion of measurability in $\mathcal{C}$. We say that a ground classical formula F is *measurable in* $\mathcal{C}$ iff for all $C \in \mathcal{C}$:

$$J \models F \text{ for } \textit{some } J \in \mathcal{S}_C \ \text{ iff } \ J \models F \text{ for } \textit{all } J \in \mathcal{S}_C\,.$$

We remark that F is measurable in $\mathcal{C}$ iff it can be interpreted by a probability function over $\mathcal{S}_{\mathcal{C}}$. In particular, $\top$ and all $\neg B$, $\neg H \wedge B$, and $H \wedge B$ with $(H|B)[c_1,c_2] \in \mathcal{F}$ can be interpreted by a probability function over $\mathcal{S}_{\mathcal{C}}$, as the following two lemmata show.

Lemma C.2 *$\top$ is measurable in $\mathcal{C}$.*

Proof. The claim is immediate, since $J \models \top$ for all $J \in \mathcal{S}_C$ and all $C \in \mathcal{C}$. $\Box$

Lemma C.3 *All $\neg B$, $\neg H \wedge B$, and $H \wedge B$ with $(H|B)[c_1,c_2] \in \mathcal{F}$ are measurable in $\mathcal{C}$.*

Proof. Let $F \in \{\neg B, \neg H \wedge B, H \wedge B\}$ for some $(H|B)[c_1,c_2] \in \mathcal{F}$, and let $C \in \mathcal{C}$. Suppose $J \models F$ for some $J \in \mathcal{S}_C$. It then follows $\{C\} \models F$. Hence, we get $J \models F$ for all $J \in \mathcal{S}_C$. Conversely, $J \models F$ for all $J \in \mathcal{S}_C$ trivially implies $J \models F$ for some $J \in \mathcal{S}_C$. $\Box$

In the sequel, let $\mathcal{L}$ be a classical knowledge base and let $\mathcal{F}$ be a probabilistic knowledge base. The following lemma shows some properties of the operator $S^\star_{\mathcal{F}}$.

Lemma C.4 *a) For all* $F \in S^{\star}_{\mathcal{F}}(\mathcal{L})$*:* $\mathcal{L} \cup \mathcal{F} \models F$.

b) For all $(H|B)[c_1, c_2] \in \mathit{ground}(\mathcal{F})$*: If* $S^{\star}_{\mathcal{F}}(\mathcal{L}) \models \bot \leftarrow H \wedge B$ *and* $c_1 > 0$*, then* $S^{\star}_{\mathcal{F}}(\mathcal{L}) \models \bot \leftarrow B$.

c) For all $(H|B)[c_1, c_2] \in \mathit{ground}(\mathcal{F})$*: If* $S^{\star}_{\mathcal{F}}(\mathcal{L}) \models H \leftarrow B$ *and* $c_2 < 1$*, then* $S^{\star}_{\mathcal{F}}(\mathcal{L}) \models \bot \leftarrow B$.

Proof. The claims are immediate by the definition of $S^{\star}_{\mathcal{F}}$. □

We next characterize the ground instances of $\mathcal{F}$ that are active w.r.t. $S^{\star}_{\mathcal{F}}(\mathcal{L})$.

Lemma C.5 *A ground instance* $(H|B)[c_1, c_2] \in \mathit{ground}(\mathcal{F})$ *is active w.r.t.* $S^{\star}_{\mathcal{F}}(\mathcal{L})$ *iff*

$$S^{\star}_{\mathcal{F}}(\mathcal{L}) \not\models \bot \leftarrow H \wedge B, \;\; S^{\star}_{\mathcal{F}}(\mathcal{L}) \not\models H \leftarrow B, \;\; \textit{and } c_1 > 0 \textit{ or } c_2 < 1.$$

Proof. The claim is immediate by the definition of $\mathit{active}_{S^{\star}_{\mathcal{F}}(\mathcal{L})}(\mathcal{F})$ and Lemma C.4 b) and c). □

We are now ready to prove Theorem 6.2.

Proof of Theorem 6.2. ($\Leftarrow$) If $KB \cup \mathit{Rel}$ is satisfiable, then trivially KB is satisfiable.

($\Rightarrow$) We first consider the case $\mathcal{L}_1 \models \bot \leftarrow \top$. By Lemma C.4 a), both KB and $KB \cup \mathit{Rel}$ are unsatisfiable. We next assume $\mathcal{L}_1 \not\models \bot \leftarrow \top$. Let $\mathcal{C}$ be defined as $\mathcal{C}_{\mathcal{L}_1}(\mathcal{F}_1)$. Given a model Pr of KB, we now construct a model Pr'' of $KB \cup \mathit{Rel}$:

Let Pr be a model of KB. By Lemma C.4 a), Pr is also a model of $\mathcal{L}_1$. Let $Pr'(C)$ $(C \in \mathcal{C})$ be defined as the sum of all $Pr(J)$ with $J \in \mathcal{I}_{\Phi}$ and $J \models C$. By Lemmata C.1–C.3, it is easy to see that Pr' is a probability function on $\mathcal{C}$ and in particular a model of $\mathcal{F}_1$.

By **ST**, for each $C \in \mathcal{C}$, there exists some $I_C \in \mathcal{I}_{\Phi}$ such that $I_C \models \mathcal{L}_1 \cup \mathit{Rel} \cup \{C\}$. Let Pr'' be defined by $Pr''(I_C) = Pr'(C)$ for all $C \in \mathcal{C}$ and by $Pr''(I) = 0$ for all the other $I \in \mathcal{I}_{\Phi}$. We now show that Pr'' is a model of $KB \cup \mathit{Rel}$. By construction, each I_C is a model of $\mathit{CKB} \cup \mathit{Rel}$. That is, each $I \in \mathcal{I}_{\Phi}$ with $Pr''(I) > 0$ is a model of $\mathit{CKB} \cup \mathit{Rel}$. Hence, Pr'' is a model of $\mathit{CKB} \cup \mathit{Rel}$. Since Pr' is a model of $\mathcal{F}_1$, it is immediate that Pr'' is also a model of $\mathcal{F}_1$.

We next prove that Pr'' is a model of each $(H|B)[c_1, c_2] \in \mathit{ground}(\mathit{PKB}) - \mathcal{F}_1$. We first assume $\mathcal{L}_1 \cup \mathit{Rel} \models \bot \leftarrow B$. Thus, by the construction of I_C, we get $I_C \models \bot \leftarrow B$ for all $C \in \mathcal{C}$. Hence, it follows $Pr''(B) = 0$ and thus $Pr'' \models (H|B)[c_1, c_2]$. We next assume $\mathcal{L}_1 \cup \mathit{Rel} \models \bot \leftarrow H \wedge B$ and $c_1 = 0$. Thus, by the construction of I_C, we get $I_C \models \bot \leftarrow H \wedge B$ for all $C \in \mathcal{C}$. Hence, it follows $Pr''(H \wedge B) = 0$ and thus $Pr'' \models (H|B)[c_1, c_2]$. We finally assume $\mathcal{L}_1 \cup \mathit{Rel} \models H \leftarrow B$ and $c_2 = 1$. Thus, by the construction of I_C, we get $I_C \models H \leftarrow B$ for all $C \in \mathcal{C}$. Hence, $Pr''(H \wedge B) = Pr(B)$ and thus $Pr'' \models (H|B)[c_1, c_2]$. □

We finally prove Theorem 6.3.

Proof of Theorem 6.3. By Theorem 6.2, it is sufficient to show that $KB \cup \mathit{Rel}$ is satisfiable iff (4) is solvable. Assume first $\mathcal{L}_2 \models \bot \leftarrow \top$. By Lemma C.4 a), $KB \cup \mathit{Rel}$ is unsatisfiable. It holds $\mathcal{C}_{KB} = \emptyset$, and thus (4) is unsolvable. Assume next $\mathcal{L}_2 \not\models \bot \leftarrow \top$:

($\Rightarrow$) Let Pr be a model of $KB \cup \mathit{Rel}$. By Lemma C.4 a), Pr is also a model of $\mathcal{L}_2$. Let y_C $(C \in \mathcal{C}_{KB})$ be defined as the sum of all $Pr(J)$ with $J \in \mathcal{I}_{\Phi}$ and $J \models C$. By Lemmata C.1–C.3, it is now easy to see that y_C $(C \in \mathcal{C}_{KB})$ is a solution of (4).

($\Leftarrow$) Let y_C $(C \in \mathcal{C}_{KB})$ be a solution of (4). For each $C \in \mathcal{C}_{KB}$, there exists some $I_C \in \mathcal{I}_{\Phi}$ such that $I_C \models \mathcal{L}_2 \cup \{C\}$. Let Pr be defined by $Pr(I_C) = y_C$ for all $C \in \mathcal{C}_{KB}$ and by $Pr(I) = 0$ for all the other $I \in \mathcal{I}_{\Phi}$. We now show that Pr is a model of $KB \cup \mathit{Rel}$. By construction, each I_C is a model of $\mathit{CKB} \cup \mathit{Rel}$. Hence, Pr is a model of $\mathit{CKB} \cup \mathit{Rel}$. Since y_C $(C \in \mathcal{C}_{KB})$ is a solution of (4), it is immediate that Pr is also a model of $\mathcal{F}_2$.

We next prove that Pr is a model of each $(H|B)[c_1, c_2] \in ground(PKB) - \mathcal{F}_2$. By Lemma C.5, we have to consider the two cases $\mathcal{L}_2 \models \bot \leftarrow H \wedge B$ and $\mathcal{L}_2 \models H \leftarrow B$. We first assume that $\mathcal{L}_2 \models \bot \leftarrow H \wedge B$. By Lemma C.4 b), we then get $\mathcal{L}_2 \models \bot \leftarrow B$ or $c_1 = 0$. Since $Pr \models \mathcal{L}_2$, it thus follows $Pr \models (H|B)[c_1, c_2]$. Assume next that $\mathcal{L}_2 \models H \leftarrow B$. By Lemma C.4 c), we then get $\mathcal{L}_2 \models \bot \leftarrow B$ or $c_2 = 1$. Since $Pr \models \mathcal{L}_2$, it thus follows $Pr \models (H|B)[c_1, c_2]$. □

We next concentrate on the proofs of Theorem 6.4 and Lemmata 6.5 and 6.6. We first derive some preparative results. The following result shows that, in fact, Rel has the desired stability property.

Lemma C.6 *Rel is stable w.r.t. $\mathcal{L} = S^{\star}_{PP}(CP)$ and $\mathcal{F} = active_{\mathcal{L} \cup Rel}(PP)$.*

Proof. We first show that $(H|B)[c_1, c_2] \in \mathcal{F}$ implies $\overline{P} \models H \wedge B$. We give a proof by contradiction. Assume first that B contains some $A \in HB_\Phi$ with $\overline{P} \not\models A$. But this contradicts $\mathcal{L} \cup Rel \not\models \bot \leftarrow B$. Suppose next that $\overline{P} \models B$ and that H contains some $A \in HB_\Phi$ with $\overline{P} \not\models A$. Hence, we get $\mathcal{L} \cup Rel \models \bot \leftarrow H \wedge B$ and thus $c_1 > 0$. By the definition of $\overline{P}$, it thus follows $\overline{P} \models H$. But this is again a contradiction. Hence, $(H|B)[c_1, c_2] \in \mathcal{F}$ implies $\overline{P} \models H \wedge B$.

We next show that $\mathcal{L} \cup Rel \not\models \bot \leftarrow C$ for all $C \in \mathcal{C}_\mathcal{L}(\mathcal{F})$. By the definition of $\mathcal{C}_\mathcal{L}(\mathcal{F})$, it holds $\mathcal{L} \not\models \bot \leftarrow C$. Let I be the least possible world that satisfies $\mathcal{L}$ and C. Let J be the least possible world that satisfies $\overline{P}$. Hence, J is a model of D. By the first part of the proof, $J \models H \wedge B$ for all $(H|B)[c_1, c_2] \in \mathcal{F}$. That is, J is a model of all classical facts and rules in $\mathcal{L}$, of all B with $(H|B)[c_1, c_2] \in \mathcal{F}$ and $\{C\} \models B$, and of all $H \wedge B$ with $(H|B)[c_1, c_2] \in \mathcal{F}$ and $\{C\} \models H \wedge B$. Thus, $I \subseteq J$. Hence, I is a model of $\mathcal{L}$, C, and D. That is, $\mathcal{L} \cup Rel \not\models \bot \leftarrow C$. □

In the sequel, let $\mathcal{F}$ be a ground probabilistic logic program. Let $\mathcal{C} = \mathcal{C}_{\underline{P}}(\mathcal{F})$ and $\mathcal{I} = \mathcal{I}_{\underline{P}}(\mathcal{F})$.

We will now show that $\mathcal{I}$ defines a partition of the set of all $I \in \mathcal{I}_\Phi$ with $I \models \underline{P}$, which coincides with the partition $\mathcal{S}_\mathcal{C}$ induced by $\mathcal{C}$.

For every $I \in \mathcal{I}$, let $\mathcal{S}_I$ denote the set of all $J \in \mathcal{I}_\Phi$ with $J \models \underline{P}$ that are a superset of I and that are not a superset of any $K \in \mathcal{I}$ that properly includes I. Let $\mathcal{S}_\mathcal{I} = \{\mathcal{S}_I \mid I \in \mathcal{I}\}$.

The following result is immediate.

Lemma C.7 *It holds $I \in \mathcal{S}_I$ for all $I \in \mathcal{I}$.*

We next show that $\mathcal{S}_\mathcal{I}$ is a partition of the set of all $I \in \mathcal{I}_\Phi$ with $I \models \underline{P}$.

Lemma C.8 *If $\mathcal{I} \neq \emptyset$, then the family $\mathcal{S}_\mathcal{I}$ is a partition of $\{I \in \mathcal{I}_\Phi \mid I \models \underline{P}\}$.*

Proof. We must show that the elements of $\mathcal{S}_\mathcal{I}$ are pairwise disjoint and that their union is equal to $\{I \in \mathcal{I}_\Phi \mid I \models \underline{P}\}$. These claims are proved by contradiction as follows.

Assume that there are two distinct $I_1, I_2 \in \mathcal{I}$ and some $J \in \mathcal{I}_\Phi$ with $J \models \underline{P}$ and $J \in \mathcal{S}_{I_1} \cap \mathcal{S}_{I_2}$. Then $J \supseteq I_1 \cup I_2$ and thus $J \supseteq T_{\underline{P}}\!\uparrow\omega(I_1 \cup I_2)$. Moreover, $T_{\underline{P}}\!\uparrow\omega(I_1 \cup I_2) \in \mathcal{I}$ by $(C3)$ and $T_{\underline{P}}\!\uparrow\omega(I_1 \cup I_2) \supset I_1$ or $T_{\underline{P}}\!\uparrow\omega(I_1 \cup I_2) \supset I_2$. But this contradicts the assumption $J \in \mathcal{S}_{I_1} \cap \mathcal{S}_{I_2}$.

Assume next that there are some $J \in \mathcal{I}_\Phi$ with $J \models \underline{P}$ that do not belong to $\bigcup \mathcal{S}_\mathcal{I}$. We now construct an infinite chain $I_0 \subset I_1 \subset \cdots$ of elements of $\mathcal{I}$ as follows. Let us define $I_0 = T_{\underline{P}}\!\uparrow\omega$. It then holds $I_0 \in \mathcal{I}$ by $(C1)$ and also $J \supseteq I_0$. But, since $J \notin \mathcal{S}_{I_0}$, there must be some $I_1 \in \mathcal{I}$ with $J \supseteq I_1$ and $I_1 \supset I_0$. This argumentation can now be continued in an infinite way. But this contradicts the fact that $\mathcal{I}$ is finite. □

In the sequel, we implicitly assume $\mathcal{I} \neq \emptyset$. We next introduce the notion of measurability in $\mathcal{I}$. We say that a ground classical formula F is *measurable in* $\mathcal{I}$ iff for all $I \in \mathcal{I}$:

$$J \models F \text{ for } \textit{some } J \in \mathcal{S}_I \text{ iff } J \models F \text{ for } \textit{all } J \in \mathcal{S}_I\,.$$

Note that F is measurable in $\mathcal{I}$ iff it can be interpreted by a probability function over $\mathcal{S}_{\mathcal{I}}$. The following two lemmata provide some results about measurability in $\mathcal{I}$.

Lemma C.9 *$\top$ and all B and $H \wedge B$ with $(H|B)[c_1, c_2] \in \mathcal{F}$ are measurable in $\mathcal{I}$.*

Proof. Let $C = \top$ or $C \in \{B, H \wedge B\}$ for some $(H|B)[c_1, c_2] \in \mathcal{F}$.

Assume that $J \models C$ for some $J \in \mathcal{S}_I$. It then follows $J \supseteq R(C)$ and $J \supseteq I$. Hence, $J \supseteq T_{\underline{P}}\!\uparrow\omega(R(C))$. We now show by contradiction that $I \supseteq T_{\underline{P}}\!\uparrow\omega(R(C))$. Let us first suppose that $I \subset T_{\underline{P}}\!\uparrow\omega(R(C))$. But this contradicts $J \in \mathcal{S}_I$. Let us next assume that $I \not\subseteq T_{\underline{P}}\!\uparrow\omega(R(C))$ and $I \not\supseteq T_{\underline{P}}\!\uparrow\omega(R(C))$. Since $J \supseteq I \cup T_{\underline{P}}\!\uparrow\omega(R(C))$, we get $J \supseteq T_{\underline{P}}\!\uparrow\omega(I \cup T_{\underline{P}}\!\uparrow\omega(R(C)))$. Moreover, $T_{\underline{P}}\!\uparrow\omega(I \cup T_{\underline{P}}\!\uparrow\omega(R(C))) \in \mathcal{I}$ by $(C3)$ and $T_{\underline{P}}\!\uparrow\omega(I \cup T_{\underline{P}}\!\uparrow\omega(R(C))) \supset I$. But this contradicts $J \in \mathcal{S}_I$. Hence, we get $I \supseteq T_{\underline{P}}\!\uparrow\omega(R(C))$. Since $J \supseteq I$ for all $J \in \mathcal{S}_I$, we thus get $J \supseteq R(C)$ for all $J \in \mathcal{S}_I$. That is, $J \models C$ for all $J \in \mathcal{S}_I$. The converse trivially holds. □

Lemma C.10 *Let F and G be two ground classical formulas such that F and $F \wedge G$ are measurable in $\mathcal{I}$. Then $F \wedge \neg G$ is measurable in $\mathcal{I}$.*

Proof. Let us assume that $J \models F \wedge \neg G$ for some $J \in \mathcal{S}_I$. It follows $J \models F$ for all $J \in \mathcal{S}_I$, since F is measurable in $\mathcal{I}$. Suppose now that $J \models F \wedge G$ for some $J \in \mathcal{S}_I$. Thus, $J \models F \wedge G$ for all $J \in \mathcal{S}_I$, since $F \wedge G$ is measurable in $\mathcal{I}$. But this contradicts the assumption that $J \models F \wedge \neg G$ for some $J \in \mathcal{S}_I$. Hence, it follows $J \models F \wedge \neg G$ for all $J \in \mathcal{S}_I$. The converse trivially holds. □

We are now ready to show that $\mathcal{S}_{\mathcal{C}}$ coincides with $\mathcal{S}_{\mathcal{I}}$.

Theorem C.11 *For all $J_1, J_2 \in \mathcal{I}_\Phi$ with $J_1, J_2 \models \underline{P}$:*

$$J_1, J_2 \in \mathcal{S}_C \textit{ for some } C \in \mathcal{C} \textit{ iff } J_1, J_2 \in \mathcal{S}_I \textit{ for some } I \in \mathcal{I}.$$

Proof. In the case $\underline{P} \models \bot \leftarrow \top$, we get $\mathcal{C} = \emptyset$ and $\mathcal{I} = \emptyset$. Thus, the claim trivially holds. In the case $\underline{P} \not\models \bot \leftarrow \top$ and $\mathcal{F} = \emptyset$, we get $\mathcal{C} = \{\top\}$ and $\mathcal{I} = \{T_{\underline{P}}\!\uparrow\omega\}$. Hence, $\mathcal{S}_{\mathcal{C}} = \{\{I \in \mathcal{I}_\Phi \mid I \models \underline{P}\}\}$ and $\mathcal{S}_{\mathcal{I}} = \{\{I \in \mathcal{I}_\Phi \mid I \models \underline{P}\}\}$. This shows the claim. We finally consider the case $\underline{P} \not\models \bot \leftarrow \top$ and $\mathcal{F} \neq \emptyset$:

($\Leftarrow$) Let $J_1, J_2 \in \mathcal{S}_I$ for some $I \in \mathcal{I}$. By Lemmata C.9 and C.10, we get for all $(H|B)[c_1, c_2] \in \mathcal{F}$: $J_1 \models H \wedge B$ iff $J_2 \models H \wedge B$, $J_1 \models \neg H \wedge B$ iff $J_2 \models \neg H \wedge B$, and $J_1 \models \neg B$ iff $J_2 \models \neg B$. That is, $J_1, J_2 \in \mathcal{S}_C$ for some $C \in \mathcal{C}$.

($\Rightarrow$) Let $J_1, J_2 \in \mathcal{S}_C$ for some $C \in \mathcal{C}$. By Lemma C.8, $\mathcal{S}_{\mathcal{I}}$ is a partition of $\{I \in \mathcal{I}_\Phi \mid I \models \underline{P}\}$. Hence, there are some $I_1, I_2 \in \mathcal{I}$ such that $J_1 \in \mathcal{S}_{I_1}$ and $J_2 \in \mathcal{S}_{I_2}$. Hence, by Lemma C.7 and the first part of this proof, it follows $J_1, I_1 \in \mathcal{S}_{C_1}$ and $J_2, I_2 \in \mathcal{S}_{C_2}$ for some $C_1, C_2 \in \mathcal{C}$. As the members of $\mathcal{S}_{\mathcal{C}}$ are pairwise disjoint, we get $C = C_1 = C_2$. Thus, it follows $I_1, I_2 \in \mathcal{S}_C$. That is, for all $(H|B)[c_1, c_2] \in \mathcal{F}$, it holds $R(B) \subseteq I_1$ iff $R(B) \subseteq I_2$ and $R(H) \cup R(B) \subseteq I_1$ iff $R(H) \cup R(B) \subseteq I_2$. Hence, we get $I_1 = I_2$. Thus, $J_1, J_2 \in \mathcal{S}_I$ for some $I \in \mathcal{I}$. □

We are now ready to prove Theorem 6.4.

Proof of Theorem 6.4. In the case $\underline{P} \models \bot \leftarrow \top$, we get $\mathcal{C}_P = \emptyset$ and $\mathcal{I}_P = \emptyset$. Hence, both systems of linear constraints (4) and (5) are unsolvable. The claim is then immediate by Theorem 6.3. Let us next consider the case $\underline{P} \not\models \bot \leftarrow \top$. By Lemma C.7 and Theorem C.11, each $C \in \mathcal{C}_P$ in the system of linear constraints (4) can be replaced by the unique $I \in \mathcal{I}_P$ with $I \models C$. By definition, each $I \in \mathcal{I}_P$ is a model of $\underline{P}$. Thus, the condition $C \preceq F$ (that is, $\underline{P} \cup \{C\} \models F$) is equivalent to $I \models F$. The claim is then immediate by Theorem 6.3. □

We finally prove Lemmata 6.5 and 6.6.

Proof of Lemma 6.5. (i) The claim is immediate by Lemma C.5.

(ii) Let $(H|B)[c_1, c_2] \in active_{\underline{P}}(PP)$. Suppose that $H \wedge B$ contains some $A \in HB_\Phi$ with $\overline{P} \not\models A$. But this contradicts $\underline{P} \not\models \bot \leftarrow H \wedge B$. Hence, we get $\overline{P} \models H \wedge B$. □

Proof of Lemma 6.6. By the definition of $\mathcal{I}_P$, for all $I \in \mathcal{I}_P$, it holds $I \models \underline{P}$. Thus, in particular, $I \models \bot \leftarrow E$ for all $E \in HB_\Phi - T_{\overline{P}}\!\uparrow\omega$. That is, $I \subseteq T_{\overline{P}}\!\uparrow\omega$. □

D Appendix: Proofs for Section 7

We now concentrate on the proofs of Theorems 7.3–7.6.

Proof of Theorem 7.3. The proof is similar to the proof of Theorem 6.2:

($\Rightarrow$) $KB \models \bot \leftarrow A$ trivially implies $KB \cup Rel \models \bot \leftarrow A$.

($\Leftarrow$) We first consider the case $\mathcal{L}_1 \models \bot \leftarrow \top$. By Lemma C.4 a), both KB and $KB \cup Rel$ are unsatisfiable. Hence, it follows $KB \models \bot \leftarrow A$ and $KB \cup Rel \models \bot \leftarrow A$. Let us next assume $\mathcal{L}_1 \not\models \bot \leftarrow \top$. Let $\mathcal{C} = \mathcal{C}_{\mathcal{L}_1}(\mathcal{F}_1 \cup \{(A|\top)[0,1]\})$. Given a model Pr of KB with $Pr(A) > 0$, we now construct a model Pr'' of $KB \cup Rel$ with $Pr''(A) > 0$:

Let Pr be a model of KB with $Pr(A) > 0$. By Lemma C.4 a), Pr is also a model of $\mathcal{L}_1$. Let $Pr'(C)$ ($C \in \mathcal{C}$) be defined as the sum of all $Pr(J)$ with $J \in \mathcal{I}_\Phi$ with $J \models C$. By Lemmata C.1–C.3, Pr' is a probability function on $\mathcal{C}$ and a model of $\mathcal{F}_1$. Moreover, $Pr'(A) = Pr(A)$.

By **ST**, for each $C \in \mathcal{C}$, there is some $I_C \in \mathcal{I}_\Phi$ such that $I_C \models \mathcal{L}_1 \cup Rel \cup \{C\}$. Let Pr'' be defined by $Pr''(I_C) = Pr'(C)$ for all $C \in \mathcal{C}$ and by $Pr''(I) = 0$ for all the other $I \in \mathcal{I}_\Phi$. We now show that Pr'' is a model of $KB \cup Rel$. By construction, each I_C is a model of $CKB \cup Rel$. Hence, Pr'' is a model of $CKB \cup Rel$. As Pr' is a model of $\mathcal{F}_1$, it follows that Pr'' is also a model of $\mathcal{F}_1$. Moreover, $Pr''(A) = Pr'(A)$.

We next prove that Pr'' is a model of each $(H|B)[c_1, c_2] \in ground(PKB) - \mathcal{F}_1$. We first assume $\mathcal{L}_1 \cup Rel \models \bot \leftarrow B$. Thus, by the construction of I_C, we get $I_C \models \bot \leftarrow B$ for all $C \in \mathcal{C}$. Hence, it follows $Pr''(B) = 0$ and thus $Pr'' \models (H|B)[c_1, c_2]$. We next assume $\mathcal{L}_1 \cup Rel \models \bot \leftarrow H \wedge B$ and $c_1 = 0$. Thus, by the construction of I_C, we get $I_C \models \bot \leftarrow H \wedge B$ for all $C \in \mathcal{C}$. Hence, it follows $Pr''(H \wedge B) = 0$ and thus $Pr'' \models (H|B)[c_1, c_2]$. We finally assume $\mathcal{L}_1 \cup Rel \models H \leftarrow B$ and $c_2 = 1$. Thus, by the construction of I_C, we get $I_C \models H \leftarrow B$ for all $C \in \mathcal{C}$. Hence, $Pr''(H \wedge B) = Pr(B)$ and thus $Pr'' \models (H|B)[c_1, c_2]$. □

Proof of Theorem 7.4. The proof is similar to the proof of Theorem 6.4. By Theorem 7.4, it is sufficient to show that $KB \cup Rel \not\models \bot \leftarrow A$ iff (7) is solvable. Let us first assume $\mathcal{L}_2 \models \bot \leftarrow \top$. By Lemma C.4 a), $KB \cup Rel$ is unsatisfiable, and thus $KB \cup Rel \models \bot \leftarrow A$. It holds $\mathcal{C}'_{KB} = \emptyset$, and thus (7) is unsolvable. Let us next assume $\mathcal{L}_2 \not\models \bot \leftarrow \top$.

($\Rightarrow$) Let Pr be a model of $KB \cup Rel$ with $Pr(A) > 0$. By Lemma C.4 a), Pr is also a model of $\mathcal{L}_2$. Let y_C ($C \in \mathcal{C}'_{KB}$) be defined as the sum of all $Pr(J) \,/\, Pr(A)$ with $J \in \mathcal{I}_\Phi$ and $J \models C$. By Lemmata C.1–C.3, it is now easy to see that y_C ($C \in \mathcal{C}'_{KB}$) is a solution of (7).

($\Leftarrow$) Let y_C ($C \in \mathcal{C}'_{KB}$) be a solution of (7). Let s denote the sum of all y_C with $C \in \mathcal{C}'_{KB}$. For each $C \in \mathcal{C}'_{KB}$, there exists some $I_C \in \mathcal{I}_\Phi$ such that $I_C \models \mathcal{L}_2 \cup \{C\}$. Let Pr be defined by $Pr(I_C) = y_C \,/\, s$ for all $C \in \mathcal{C}'_{KB}$ and by $Pr(I) = 0$ for all the other $I \in \mathcal{I}_\Phi$. We now show that Pr is a model of $KB \cup Rel$ with $Pr(A) > 0$. By construction, each I_C is a model of $CKB \cup Rel$. Hence, Pr is a model of $CKB \cup Rel$. As y_C ($C \in \mathcal{C}'_{KB}$) is a solution of (7), it follows that Pr is a model of $\mathcal{F}_2$. Moreover, it clearly holds $Pr(A) > 0$.

We next prove that Pr is a model of each $(H|B)[c_1, c_2] \in ground(PKB) - \mathcal{F}_2$. By Lemma C.5, we have to consider the two cases $\mathcal{L}_2 \models \bot \leftarrow H \wedge B$ and $\mathcal{L}_2 \models H \leftarrow B$. We first assume that $\mathcal{L}_2 \models \bot \leftarrow H \wedge B$. By Lemma C.4 b), we then get $\mathcal{L}_2 \models \bot \leftarrow B$ or $c_1 = 0$. Since $Pr \models \mathcal{L}_2$, it thus follows $Pr \models (H|B)[c_1, c_2]$. Assume next that $\mathcal{L}_2 \models H \leftarrow B$. By Lemma C.4 c), we then get $\mathcal{L}_2 \models \bot \leftarrow B$ or $c_2 = 1$. Since $Pr \models \mathcal{L}_2$, it thus follows $Pr \models (H|B)[c_1, c_2]$. □

Proof of Theorem 7.5. The proof is similar to the proof of Theorem 7.3. By Theorem 7.3, we know that $KB \not\models \bot \leftarrow A$ implies $KB \cup Rel \not\models \bot \leftarrow A$. That is, there are models Pr of KB with $Pr(A) > 0$. Moreover, there are models Pr of $KB \cup Rel$ with $Pr(A) > 0$. We now show that for each model Pr of KB with $Pr(A) > 0$, there is a model Pr'' of $KB \cup Rel$ with $Pr''(A) = Pr(A)$ and $Pr''(G \wedge A) = Pr(G \wedge A)$. Let $\mathcal{C} = \mathcal{C}_{\mathcal{L}_1}(\mathcal{F}_1 \cup \{(G|A)[0, 1]\})$.

Let Pr be a model of KB with $Pr(A) > 0$. By Lemma C.4 a), Pr is also a model of $\mathcal{L}_1$. Let $Pr'(C)$ ($C \in \mathcal{C}$) be defined as the sum of all $Pr(J)$ with $J \in \mathcal{I}_\Phi$ with $J \models C$. By Lemmata C.1–C.3, Pr' is a probability function on $\mathcal{C}$ and in particular a model of $\mathcal{F}_1$. Moreover, it holds $Pr'(A) = Pr(A)$ and $Pr'(G \wedge A) = Pr(G \wedge A)$.

By **ST**, for each $C \in \mathcal{C}$, there is some $I_C \in \mathcal{I}_\Phi$ such that $I_C \models \mathcal{L}_1 \cup Rel \cup \{C\}$. Let Pr'' be defined by $Pr''(I_C) = Pr'(C)$ for all $C \in \mathcal{C}$ and by $Pr''(I) = 0$ for all the other $I \in \mathcal{I}_\Phi$. We now show that Pr'' is a model of $KB \cup Rel$. By construction, each I_C is a model of $CKB \cup Rel$. Hence, Pr'' is a model of $CKB \cup Rel$. As Pr' is a model of $\mathcal{F}_1$, it is immediate that Pr'' is also a model of $\mathcal{F}_1$. Moreover, $Pr''(A) = Pr'(A)$ and $Pr''(G \wedge A) = Pr'(G \wedge A)$.

We next prove that Pr'' is a model of each $(H|B)[c_1, c_2] \in ground(PKB) - \mathcal{F}_1$. We first assume $\mathcal{L}_1 \cup Rel \models \bot \leftarrow B$. Thus, by the construction of I_C, we get $I_C \models \bot \leftarrow B$ for all $C \in \mathcal{C}$. Hence, it follows $Pr''(B) = 0$ and thus $Pr'' \models (H|B)[c_1, c_2]$. We next assume $\mathcal{L}_1 \cup Rel \models \bot \leftarrow H \wedge B$ and $c_1 = 0$. Thus, by the construction of I_C, we get $I_C \models \bot \leftarrow H \wedge B$ for all $C \in \mathcal{C}$. Hence, it follows $Pr''(H \wedge B) = 0$ and thus $Pr'' \models (H|B)[c_1, c_2]$. We finally assume $\mathcal{L}_1 \cup Rel \models H \leftarrow B$ and $c_2 = 1$. Thus, by the construction of I_C, we get $I_C \models H \leftarrow B$ for all $C \in \mathcal{C}$. Hence, $Pr''(H \wedge B) = Pr(B)$ and thus $Pr'' \models (H|B)[c_1, c_2]$. □

Proof of Theorem 7.6. The proof is similar to the proof of Theorem 7.4. By Theorem 7.5, we can compute the tight answer substitution for Q to $KB \cup Rel$. By Theorem 7.3, $KB \not\models \bot \leftarrow A$ implies $KB \cup Rel \not\models \bot \leftarrow A$. By Theorem 7.4, $KB \not\models \bot \leftarrow A$ implies that (8) has a feasible solution.

We now show that for each model Pr of $KB \cup Rel$ with $Pr(A) > 0$, there is a feasible solution y_C ($C \in \mathcal{C}''_{KB}$) of (8) that satisfies (21), and that for each feasible solution y_C ($C \in \mathcal{C}''_{KB}$) of (8), there exists a model Pr of $KB \cup Rel$ with $Pr(A) > 0$ that satisfies (21):

$$Pr(G \wedge A) \,/\, Pr(A) \quad = \quad \textstyle\sum_{C \in \mathcal{C}''_{KB},\, C \preceq G \wedge A} y_C \,. \tag{21}$$

Let Pr be a model of $KB \cup Rel$ with $Pr(A) > 0$. By Lemma C.4 a), Pr is also a model of $\mathcal{L}_2$. Let y_C ($C \in \mathcal{C}''_{KB}$) be defined as the sum of all $Pr(J) \,/\, Pr(A)$ with $J \in \mathcal{I}_\Phi$ and $J \models C$. By Lemmata C.1–C.3, it is easy to see that y_C ($C \in \mathcal{C}''_{KB}$) is a solution of (8) with (21).

Conversely, let y_C ($C \in \mathcal{C}''_{KB}$) be a solution of (7). Let s denote the sum of all y_C with $C \in \mathcal{C}''_{KB}$. For each $C \in \mathcal{C}''_{KB}$, there exists some $I_C \in \mathcal{I}_\Phi$ such that $I_C \models \mathcal{L}_2 \cup \{C\}$. Let Pr be defined by $Pr(I_C) = y_C \,/\, s$ for all $C \in \mathcal{C}''_{KB}$ and by $Pr(I) = 0$ for all the other $I \in \mathcal{I}_\Phi$. We now show that Pr is a model of $KB \cup Rel$ with $Pr(A) > 0$ and (21). By construction, each I_C is a model of $CKB \cup Rel$. Hence, Pr is a model of $CKB \cup Rel$. As y_C ($C \in \mathcal{C}''_{KB}$) is a solution of (7), it is immediate that Pr is a model of $\mathcal{F}_2$. Moreover, it clearly holds $Pr(A) > 0$ and (21).

We next prove that Pr is a model of each $(H|B)[c_1, c_2] \in ground(PKB) - \mathcal{F}_2$. By Lemma C.5, we have to consider the two cases $\mathcal{L}_2 \models \bot \leftarrow H \wedge B$ and $\mathcal{L}_2 \models H \leftarrow B$. We first assume that $\mathcal{L}_2 \models \bot \leftarrow H \wedge B$. By Lemma C.4 b), we then get $\mathcal{L}_2 \models \bot \leftarrow B$ or $c_1 = 0$. Since $Pr \models \mathcal{L}_2$, it thus follows $Pr \models (H|B)[c_1, c_2]$. Assume next that $\mathcal{L}_2 \models H \leftarrow B$. By Lemma C.4 c), we then get $\mathcal{L}_2 \models \bot \leftarrow B$ or $c_2 = 1$. Since $Pr \models \mathcal{L}_2$, it thus follows $Pr \models (H|B)[c_1, c_2]$. □

We next concentrate on the proofs of Theorems 7.7 and 7.10. The following two lemmata show that Rel' and Rel'' have the desired stability property.

Lemma D.1 *Let $\mathcal{L} = S^{\star}_{PP}(CP)$ and $\mathcal{F} = active_{\mathcal{L} \cup Rel'}(PP)$. Then:*

Rel' is stable w.r.t. $\mathcal{L}$ and $\mathcal{F} \cup \{(A|\top)[0,1]\}$.

Proof. The proof is similar to the proof of Lemma C.6. We start by showing that $(H|B)[c_1, c_2] \in \mathcal{F}$ implies $\overline{P} \cup \{A\} \models H \wedge B$. Assume first that B contains some $E \in HB_\Phi$ with $\overline{P} \cup \{A\} \not\models E$. But this contradicts $\mathcal{L} \cup Rel' \not\models \bot \leftarrow B$. Suppose next that $\overline{P} \cup \{A\} \models B$ and that H contains some $E \in HB_\Phi$ with $\overline{P} \cup \{A\} \not\models E$. Hence, we get $\mathcal{L} \cup Rel' \models \bot \leftarrow H \wedge B$ and thus $c_1 > 0$. By the definition of $\overline{P}$, it thus follows $\overline{P} \cup \{A\} \models H$. But this is again a contradiction. Hence, $(H|B)[c_1, c_2] \in \mathcal{F}$ implies $\overline{P} \cup \{A\} \models H \wedge B$.

Let $\mathcal{C} = \mathcal{C}_{\mathcal{L}}(\mathcal{F} \cup \{(A|\top)[0,1]\})$. We next show that $\mathcal{L} \cup Rel' \not\models \bot \leftarrow C$ for all $C \in \mathcal{C}$. By the definition of $\mathcal{C}$, it holds $\mathcal{L} \not\models \bot \leftarrow C$. Let I be the least possible world that satisfies $\mathcal{L}$ and C. Let J be the least possible world that satisfies $\overline{P} \cup \{A\}$. Hence, J is a model of Rel'. By the first part of the proof, $J \models H \wedge B$ for all $(H|B)[c_1, c_2] \in \mathcal{F}$. Thus, J is a model of all classical facts and rules in $\mathcal{L}$, of all B with $(H|B)[c_1, c_2] \in \mathcal{F} \cup \{(A|\top)[0,1]\}$ and $\{C\} \models B$, and of all $H \wedge B$ with $(H|B)[c_1, c_2] \in \mathcal{F} \cup \{(A|\top)[0,1]\}$ and $\{C\} \models H \wedge B$. Thus, $I \subseteq J$. Hence, I is a model of $\mathcal{L}$, C, and Rel'. That is, $\mathcal{L} \cup Rel' \not\models \bot \leftarrow C$. □

Lemma D.2 *Let $\mathcal{L} = S^{\star}_{PP}(CP)$ and $\mathcal{F} = active_{\mathcal{L} \cup Rel''}(PP)$. Then:*

Rel'' is stable w.r.t. $\mathcal{L}$ and $\mathcal{F} \cup \{(G|A)[0,1]\}$.

Proof. The proof can be done analogously to the proof of Lemma D.1. □

We are now read to prove Theorem 7.7.

Proof of Theorem 7.7. The proof is similar to the proof of Theorem 6.4. In case of $\underline{P}' \models \bot \leftarrow \top$, we get $\mathcal{C}'_P = \emptyset$ and $\mathcal{I}'_P = \emptyset$. Hence, both systems of linear constraints (7) and (9) are unsolvable. The claim is then immediate by Theorem 7.4 and Lemma D.1. Let us next consider the case $\underline{P}' \not\models \bot \leftarrow \top$. By Lemma C.7 and Theorem C.11, each $C \in \mathcal{C}'_P$ in the system of linear constraints (7) can be replaced by the unique $I \in \mathcal{I}'_P$ with $I \models C$. By definition, each $I \in \mathcal{I}'_P$ is a model of $\underline{P}'$. Thus, the condition $C \preceq F$ (that is, $\underline{P}' \cup \{C\} \models F$) is equivalent to $I \models F$. The claim then follows immediately from Theorem 7.4 and Lemma D.1. □

We finally prove Theorem 7.10.

Proof of Theorem 7.10. The proof can be done analogously to the proof of Theorem 7.7. □

References

[1] S. Amarger, D. Dubois, and H. Prade. Constraint propagation with imprecise conditional probabilities. In *Proc. of the 7th Conference on Uncertainty in Artificial Intelligence*, pages 26–34. Morgan Kaufmann, 1991.

[2] K. A. Andersen and J. N. Hooker. Bayesian logic. *Decision Support Systems*, 11:191–210, 1994.

[3] K. R. Apt. Logic programming. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 10, pages 493–574. MIT Press, 1990.

[4] F. Bacchus, A. Grove, J. Y. Halpern, and D. Koller. From statistical knowledge bases to degrees of beliefs. *Artif. Intell.*, 87:75–143, 1996.

[5] J. F. Baldwin. Evidential support logic programming. *Fuzzy Sets Syst.*, 24:1–26, 1987.

[6] V. Biazzo and A. Gilio. A generalization of the fundamental theorem of de Finetti for imprecise conditional probability assessments. *Int. J. Approx. Reasoning*, 24:251–272, 2000.

[7] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theor. Comput. Sci.*, 68:135–154, 1989.

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

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

[10] V. Chvátal. *Linear Programming*. Freeman, San Francisco, 1983.

[11] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 82–101, 1997.

[12] B. de Finetti. *Theory of Probability*. J. Wiley, New York, 1974.

[13] A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. In *Proceedings of the 14th International Conference on Logic Programming*, pages 391–405. MIT Press, 1997.

[14] M. I. Dekhtyar, A. Dekhtyar, and V. S. Subrahmanian. Hybrid probabilistic programs: Algorithms and complexity. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 160–169. Morgan Kaufmann, 1999.

[15] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[16] D. Dubois, H. Prade, L. Godo, and R. López de Màntaras. Qualitative reasoning with imprecise probabilities. *Journal of Intelligent Information Systems*, 2:319–363, 1993.

[17] D. Dubois, H. Prade, and J.-M. Touscas. Inference with imprecise numerical quantifiers. In Z. W. Ras and M. Zemankova, editors, *Intelligent Systems*, chapter 3, pages 53–72. Ellis Horwood, 1990.

[18] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Inf. Comput.*, 87:78–128, 1990.

[19] M. Fitting. Bilattices and the semantics of logic programming. *J. Logic Program.*, 11(1–2):91–116, 1991.

[20] A. M. Frisch and P. Haddawy. Anytime deduction for probabilistic logic. *Artif. Intell.*, 69:93–122, 1994.

[21] H. Gaifman. Concerning measures in first order calculi. *Israel J. Math.*, 2:1–18, 1964.

[22] B. Ganter. Algorithmen zur Formalen Begriffsanalyse. In B. Ganter, R. Wille, and K.E. Wolff, editors, *Beiträge zur Begriffsanalyse*, pages 241–254. B.I.-Wissenschaftsverlag, Mannheim, 1987.

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

[24] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *J. Complexity*, 4(1):1–11, 1988.

[25] D. Goldfarb and M. J. Todd. Linear programming. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, chapter 2, pages 73–170. Elsevier Science, 1989.

[26] P. Haddawy. Generating Bayesian networks from probability logic knowledge bases. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 262–269. Morgan Kaufmann, 1994.

[27] R. Hähnle. Exploiting data dependencies in many-valued logics. *J. Appl. Non-Class. Log.*, 6(1):49–69, 1996.

[28] T. Hailperin. *Boole's logic and probability*. North-Holland, Amsterdam, 2nd ed., 1986.

[29] J. Y. Halpern. An analysis of first-order logics of probability. *Artif. Intell.*, 46:311–350, 1990.

[30] M. Jaeger. Relational Bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 266–273. Morgan Kaufmann, 1997.

[31] M. Jaeger. On the complexity of inference about probabilistic relational models. *Artif. Intell.*, 117:297–308, 1999.

[32] B. Jaumard, P. Hansen, and M. Poggi de Aragão. Column generation methods for probabilistic logic. *ORSA J. Comput.*, 3:135–147, 1991.

[33] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2, pages 67–161. MIT Press, Cambridge, MA, 1990.

[34] D. Kavvadias and C. H. Papadimitriou. A linear programming approach to reasoning about probabilities. *Ann. Math. Artif. Intell.*, 1:189–205, 1990.

[35] M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Logic Program.*, 12(3–4):335–367, 1992.

[36] R. A. Kowalski. Predicate logic as a programming language. In *Information Processing '74*, pages 569–574. North-Holland, 1974.

[37] L. V. S. Lakshmanan and F. Sadri. Probabilistic deductive databases. In *Proceedings of the International Logic Programming Symposium*, pages 254–268. MIT Press, 1994.

[38] S. M. Leach and J. J. Lu. Query processing in annotated logic programming: Theory and implementation. *J. of Intell. Information Systems*, 6(1):33–58, 1996.

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

[40] J. J. Lu. Logic programming with signs and annotations. *J. Log. Comput.*, 6(6):755–778, 1996.

[41] T. Lukasiewicz. Efficient global probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. In *Proceedings of the 6th International Conference on Information and Knowledge Management*, pages 75–82. ACM Press, 1997.

[42] T. Lukasiewicz. Probabilistic logic programming. In *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 388–392. J. Wiley & Sons, 1998.

[43] T. Lukasiewicz. Local probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. *Int. J. Approx. Reasoning*, 21(1):23–61, 1999.

[44] T. Lukasiewicz. Many-valued disjunctive logic programs with probabilistic semantics. In *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 1730 of *LNCS*, pages 277–289. Springer, 1999.

[45] T. Lukasiewicz. Many-valued first-order logics with probabilistic semantics. In *Proceedings of the Annual Conference of the European Association for Computer Science Logic*, volume 1584 of *LNCS*, pages 415–429. Springer, 1999.

[46] T. Lukasiewicz. Probabilistic and truth-functional many-valued logic programming. In *Proceedings of the 29th IEEE International Symposium on Multiple-Valued Logic*, pages 236–241, 1999.

[47] T. Lukasiewicz. Probabilistic deduction with conditional constraints over basic events. *J. Artif. Intell. Res.*, 10:199–241, 1999.

[48] T. Lukasiewicz. Credal networks under maximum entropy. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 2000. To appear.

[49] T. Lukasiewicz. Probabilistic default reasoning with conditional constraints. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning, Special Session on Uncertainty Frameworks in NMR*, 2000.

[50] T. Lukasiewicz and G. Kern-Isberner. Probabilistic logic programming under maximum entropy. In *Proceedings of the 5th European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 1638 of *LNAI*, pages 279–292. Springer, 1999.

[51] R. T. Ng. Semantics, consistency, and query processing of empirical deductive databases. *IEEE Trans. Knowl. Data Eng.*, 9(1):32–49, 1997.

[52] R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Inf. Comput.*, 101:150–201, 1992.

[53] R. T. Ng and V. S. Subrahmanian. A semantical framework for supporting subjective and conditional probabilities in deductive databases. *J. Autom. Reasoning*, 10(2):191–235, 1993.

[54] R. T. Ng and V. S. Subrahmanian. Stable semantics for probabilistic deductive databases. *Inf. Comput.*, 110:42–83, 1994.

[55] L. Ngo and P. Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theor. Comput. Sci.*, 171:147–177, 1997.

[56] N. J. Nilsson. Probabilistic logic. *Artif. Intell.*, 28:71–88, 1986.

[57] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[58] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[59] J. B. Paris. *The Uncertain Reasoner's Companion: A Mathematical Perspective*. Cambridge University Press, Cambridge, UK, 1995.

[60] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[61] D. Poole. Logic programming, abduction and probability. *New Gener. Comput.*, 11:377–400, 1993.

[62] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artif. Intell.*, 64:81–129, 1993.

[63] N. Rescher. *Many-valued Logic*. McGraw-Hill, New York, 1969.

[64] A. Schrijver. *Theory of Linear and Integer Programming*. J. Wiley, New York, 1986.

[65] D. Scott and P. Krauss. Assigning probabilities to logical formulas. In J Hintikka and P. Suppes, editors, *Aspects of Inductive Logic*, pages 219–264. North-Holland, Amsterdam, 1966.

[66] L. van der Gaag. Computing probability intervals under independency constraints. In *Uncertainty in Artificial Intelligence 6*, pages 457–466. North-Holland, Amsterdam, 1991.

[67] M. H. van Emden. Quantitative deduction and its fixpoint theory. *J. Logic Program.*, 3(1):37–53, 1986.

[68] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theor. Comput. Sci.*, 51:53–80, 1987.

[69] L. A. Zadeh. Fuzzy sets. *Inform. and Control*, 8:338–353, 1965.