I N F S Y S
RESEARCH
R E P O R T

INSTITUT FÜR INFORMATIONSSYSTEME
ABTEILUNG WISSENSBASIERTE SYSTEME

PROBABILISTIC LOGIC PROGRAMMING UNDER INHERITANCE WITH OVERRIDING

Thomas Lukasiewicz

INFSYS RESEARCH REPORT 1843-01-05
MAY 2001

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 1843-01-05, MAY 2001

PROBABILISTIC LOGIC PROGRAMMING UNDER INHERITANCE WITH OVERRIDING

(PRELIMINARY REPORT, MAY 2001)

Thomas Lukasiewicz[1]

Abstract. We present probabilistic logic programming under inheritance with overriding. This approach is based on new notions of entailment for reasoning with conditional constraints, which are obtained from the classical notion of logical entailment by adding inheritance with overriding. This is done by using recent approaches to probabilistic default reasoning with conditional constraints. We analyze the semantic properties of the new entailment relations. We also present algorithms for probabilistic logic programming under inheritance with overriding, and we analyze its complexity in the propositional case.

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

Acknowledgements: I am very grateful to Gabriele Kern-Isberner for valuable comments on an earlier version of this paper. This work has been supported by a DFG grant and the Austrian Science Fund under project N Z29-INF.

1 Introduction

A number of recent research efforts are directed towards integrating logic-oriented and probability-based representation and reasoning formalisms. In particular, there are approaches to *probabilistic logic programming* that combine logic programming techniques with probabilities over possible worlds [25, 26, 4, 5, 20]. They are based on the model-theoretic notion of logical entailment, which is well-known from probabilistic propositional logics [28, 7, 6].

The notion of logical entailment, however, has often been criticized in the literature for its inferential weakness. For this reason, many recent approaches towards integrating logic and probabilities combine logic-based formalisms with Bayesian networks [33, 32, 13, 27, 15].

Another way to overcome the inferential weakness of logical entailment is to use the principle of maximum entropy [24] or the principle of sequential maximum entropy [21], where the latter is closely related to Bayesian networks. The maximum entropy approach, however, has the drawback that it does not properly model imprecision in our knowledge base. That is, maximum entropy always produces a single joint distribution, for example, also in the extreme case when our knowledge base is empty.

In this paper, we investigate a very promising new approach of strengthening the notion of logical entailment in probabilistic logic, which does not have the above-mentioned drawback of the maximum entropy approach. This approach also has advantages over the above combinations of logic-based formalisms with Bayesian networks, as it does not assume acyclic Bayesian network structures with complete and precise conditional probabilities.

This new approach is inspired by *reference-class reasoning*, which goes back to Reichenbach [34] and was further refined especially by Kyburg [17, 18] and Pollock [31].

Reichenbach [34] describes reference-class reasoning as follows: "If we are asked to find the probability holding for an individual future event, we must first incorporate the case in a suitable reference class. An individual thing or event may be incorporated in many reference classes .... We then proceed by considering the narrowest reference class for which suitable statistics can be compiled."

That is, Reichenbach suggests to equate our knowledge about a particular individual with the statistics from a reference class, which is informally defined as a set of individuals to which our particular individual belongs and about which we have "suitable statistics". Moreover, if there are several reference classes with conflicting statistics, then we should prefer the smallest one and its associated statistics.

Interestingly, Reichenbach's guidelines may be interpreted as *inheritance with overriding* as it is known from object-oriented programming languages. The aspect of *inheritance* is expressed by the fact that any class containing the particular individual can be considered as reference class, while the aspect of *overriding* is expressed by the fact that smaller reference classes are preferred to larger ones.

It turns out that the classical notion of logical entailment in probabilistic logic does not follow the principle of inheritance with overriding. It is thus a natural idea to strengthen logical entailment by adding inheritance with overriding. In this paper, we realize this idea by using recent approaches to *probabilistic default reasoning* from [22, 23].

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

- We present probabilistic logic programming under inheritance with overriding, which is based on recent approaches to probabilistic default reasoning from [22, 23].
- We describe general nonmontonic properties of entailment under inheritance with overriding.
- We present algorithms for probabilistic logic programming under inheritance with overriding.
- We analyze the computational complexity of probabilistic logic programming under inheritance with overriding in the propositional case.

Note that all proofs are given in the appendix.

2 Preliminaries

In this section, we describe the probabilistic background of this work. We then define probabilistic logic programs and give some illustrative examples.

2.1 Probabilistic Background

We now briefly recall how first-order logics of probability are given a semantics in which probabilities are defined over a set of possible worlds (cf. especially [3, 8, 35, 14]). We restrict our considerations to a language of first-order Boolean combinations of 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 describe 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. The propositional constants *false* and *true*, denoted $\bot$ and $\top$, respectively, are classical formulas. 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 ϕ and ψ are classical formulas, then also $\neg\phi$ and $(\phi \wedge \psi)$. A *conditional constraint* is an expression of the form $(\psi|\phi)[l, u]$ with real numbers $l, u \in [0, 1]$ and classical formulas ϕ and ψ. We define *probabilistic formulas* inductively as follows. Every conditional constraint is a probabilistic formula. If F and G are probabilistic formulas, then also $\neg F$ and $(F \wedge G)$. We use $(F \vee G)$, $(F \Leftarrow G)$, and $(F \Leftrightarrow G)$ to abbreviate $\neg(\neg F \wedge \neg G)$, $\neg(\neg F \wedge G)$, and $(\neg(\neg F \wedge G) \wedge \neg(F \wedge \neg G))$, respectively, and adopt the usual conventions to eliminate parentheses. Object terms and formulas are *ground* iff they do not contain any variables. The notions of substitutions and of ground instances of probabilistic formulas are canonically defined.

We divide conditional constraints into *classical conditional constraints*, which have the form $(\psi|\phi)[1, 1]$ or $(\psi|\phi)[0, 0]$, and *purely probabilistic conditional constraints*, which are of the form $(\psi|\phi)[l, u]$ with $l < 1$ and $u > 0$.

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 $\sigma(c) = c$ for all constant symbols c from Φ. The *truth* of classical formulas ϕ in I under σ, denoted $I \models_\sigma \phi$, is inductively defined as follows (we write $I \models \phi$ when ϕ is ground):

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

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 ϕ in the probabilistic interpretation Pr under a variable assignment σ, denoted $Pr_\sigma(\phi)$ (or simply $Pr(\phi)$ when ϕ is ground), is defined as the sum of all $Pr(I)$ such that $I \in \mathcal{I}_\Phi$ and $I \models_\sigma \phi$. For classical formulas ϕ and ψ with $Pr_\sigma(\phi) > 0$, we use $Pr_\sigma(\psi|\phi)$ to abbreviate $Pr_\sigma(\psi \wedge \phi) \,/\, Pr_\sigma(\phi)$. The *truth* of a probabilistic formula F in a probabilistic interpretation Pr under a variable assignment σ, denoted $Pr \models_\sigma F$, is inductively defined as follows:

- $Pr \models_\sigma (\psi|\phi)[l,u]$ iff $Pr_\sigma(\phi) = 0$ or $Pr_\sigma(\psi|\phi) \in [l,u]$.
- $Pr \models_\sigma \neg F$ iff not $Pr \models_\sigma F$.
- $Pr \models_\sigma (F \wedge G)$ iff $Pr \models_\sigma F$ and $Pr \models_\sigma G$.

A probabilistic formula 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. We say Pr is a *model* of a set of probabilistic formulas $\mathcal{F}$, denoted $Pr \models \mathcal{F}$, iff Pr is a model of all $F \in \mathcal{F}$. We say $\mathcal{F}$ is *satisfiable* iff a model of $\mathcal{F}$ exists.

We next define the notion of *logical entailment* as follows. A probabilistic formula F is a *logical consequence* of a set of probabilistic formulas $\mathcal{F}$, denoted $\mathcal{F} \models F$, iff each model of $\mathcal{F}$ is also a model of F. A conditional constraint $(\psi|\phi)[l,u]$ is a *tight logical consequence* of $\mathcal{F}$, denoted $\mathcal{F} \models_{\mathit{tight}} (\psi|\phi)[l,u]$, iff l (resp., u) is the infimum (resp., supremum) of $Pr_\sigma(\psi|\phi)$ subject to all models Pr of $\mathcal{F}$ and all variable assignments σ with $Pr_\sigma(\phi) > 0$. Note that we assume $l = 1$ and $u = 0$, when $\mathcal{F} \models (\phi|\top)[0,0]$ (that is, $Pr_\sigma(\phi) = 0$ for all models Pr of $\mathcal{F}$ and all σ).

2.2 Probabilistic Logic Programs

A *(general) probabilistic logic program* P is a finite set of conditional constraints $(\psi|\phi)[l,u]$ with $l \leq u$. The *grounding of* P, denoted $ground(P)$, is the set of all ground instances of members of P. A *probabilistic query* is an expression of the form $\exists(\beta|\alpha)[s,t]$, where α and β are two classical formulas, and s and t are either two real numbers from $[0,1]$ or two distinct bound variables from $\mathcal{X}$. It is *object-ground* iff α and β are ground and $s, t \in \mathcal{X}$.

We say $(\psi|\phi)[l,u]$ is *conjunctive* (resp., *1-conjunctive*) iff ψ is a conjunction of atoms (resp., ψ is an atom) and ϕ is either $\top$ or a conjunction of atoms. A probabilistic logic program P is *conjunctive* (resp., *1-conjunctive*) iff all $C \in P$ are conjunctive (resp., 1-conjunctive). A *probabilistic query* $\exists(\beta|\alpha)[s,t]$ is *conjunctive* (resp., *1-conjunctive*) iff $(\beta|\alpha)[0,1]$ is conjunctive (resp., 1-conjunctive).

The meaning of probabilistic queries to probabilistic logic programs is defined by entailment semantics for probabilistic logic programs. Every semantics s is associated with an *s-consequence* relation $\Vdash\!\sim^{s}$ and a *tight s-consequence* relation $\Vdash\!\sim^{s}_{tight}$, which are subsets of $2^{\mathcal{L}_\Phi} \times \mathcal{L}_\Phi$, where $\mathcal{L}_\Phi$ denotes the set of all conditional constraints over Φ.

Two entailment semantics based on logical entailment, called *0-* and *1-entailment*, are defined as follows. The *0-consequence* (resp., *tight 0-consequence*) relation is given by $\models$ (resp., $\models_{tight}$). Note that reasoning in probabilistic logics is in general done with 0-entailment. A conditional constraint $(\psi|\phi)[l,u]$ is a *1-consequence* of a set of conditional constraints $\mathcal{C}$ iff $Pr_\sigma(\psi) \in [l,u]$ for all models Pr of $\mathcal{C}$ and all variable assignments σ such that $Pr_\sigma(\phi) = 1$. We say $(\psi|\phi)[l,u]$ is a *tight 1-consequence* of $\mathcal{C}$ iff l (resp., u) is the infimum (resp., supremum) of $Pr_\sigma(\psi)$ subject to all models Pr of $\mathcal{C}$ and all σ such that $Pr_\sigma(\phi) = 1$.

The main difference between 0- and 1-entailment is that 0-entailment is based on *conditioning*, while 1-entailment realizes some *constraining*. For example, a ground conditional constraint $(\psi|\phi)[l,u]$ is a 0-consequence of $\mathcal{C}$ iff $Pr[\phi](\psi) \in [l,u]$ for every model Pr of $\mathcal{C}$ with $Pr(\phi) > 0$, where $Pr[\phi]$ denotes the conditioning of Pr on ϕ. Whereas $(\psi|\phi)[l,u]$ is a 1-consequence of $\mathcal{C}$ iff $Pr(\psi) \in [l,u]$ for every model Pr of $\mathcal{C}$ with $Pr(\phi) = 1$. Note that under 0- and 1-entailment, probabilistic logic programs P are equivalent to their groundings $ground(P)$.

Given a probabilistic query $\exists(\beta|\alpha)[l,u]$ with $l,u \in [0,1]$ to a probabilistic logic program P, its *correct answer substitutions under* a semantics s are substitutions θ such that $P \Vdash\!\sim^{s} (\beta\theta|\alpha\theta)[l,u]$ and that θ acts only on variables in $\exists(\beta|\alpha)[l,u]$. Its *correct answer under* s is Yes if such a θ exists and No otherwise. Given a probabilistic query $\exists(\beta|\alpha)[x,y]$ with $x,y \in \mathcal{X}$ to a probabilistic logic program P, its *tight answer substitutions under* s are substitutions θ such that $P \Vdash\!\sim^{s}_{tight} (\beta\theta|\alpha\theta)[x\theta, y\theta]$, that θ acts only on variables in $\exists(\beta|\alpha)[x,y]$, and that $x\theta, y\theta \in [0,1]$.

2.3 Examples

We now give some illustrative examples. In the first example, 1-entailment shows the property of inheritance of probabilistic knowledge, while 0-entailment does not.

Example 2.1 The knowledge "all penguins are birds" and "birds have legs with a probability of at least 0.95" can be expressed by the following probabilistic logic program P:

$$P = \{(\mathit{bird}(X) \mid \mathit{penguin}(X))[1,1],\ (\mathit{have\_legs}(X) \mid \mathit{bird}(X))[.95,1]\}.$$

Our wondering about the tight interval for the probability that Tweety has legs given that Tweety is a penguin can be expressed by the object-ground probabilistic query

$$Q = \exists(\mathit{have\_legs}(tweety)|\mathit{penguin}(tweety))[R,S]\,.$$

Its tight answer substitutions under 0- and 1-entailment are given by $\{R/0, S/1\}$ and $\{R/.95, S/1\}$, respectively, as

$$P \models_{\mathit{tight}} (\mathit{have\_legs}(tweety)|\mathit{penguin}(tweety))[0, 1] \text{ and}$$
$$P \cup \{(\mathit{penguin}(tweety)|\top)[1, 1]\} \models_{\mathit{tight}} (\mathit{have\_legs}(tweety)|\top)[.95, 1] .$$

The next example shows that inheritance in 1-entailment may often result in incompatible probabilistic knowledge, as 1-entailment does not have overriding mechanisms.

Example 2.2 The knowledge "all penguins are birds", "birds have legs with a probability of at least 0.95", "birds fly with a probability between 0.9 and 0.95", and "penguins fly with a probability of at most 0.05" can be expressed by the following probabilistic logic program P:

$$P = \{(\mathit{bird}(X) \,|\, \mathit{penguin}(X))[1, 1], (\mathit{have\_legs}(X) \,|\, \mathit{bird}(X))[.95, 1], \\ (\mathit{fly}(X) \,|\, \mathit{bird}(X))[.9, .95], (\mathit{fly}(X) \,|\, \mathit{penguin}(X))[0, .05]\}.$$

Our wondering about the tight interval for the probability that Tweety has legs given that Tweety is a penguin can be expressed by the object-ground probabilistic query

$$Q = \exists(\mathit{have\_legs}(tweety)|\mathit{penguin}(tweety))[R, S] .$$

Its tight answer substitutions under 0- and 1-entailment are given by $\{R/0,\ S/1\}$ and $\{R/1,\ S/0\}$, respectively, as

$$P \models_{\mathit{tight}} (\mathit{have\_legs}(tweety)|\mathit{penguin}(tweety))[0, 1] \text{ and}$$
$$P \cup \{(\mathit{penguin}(tweety)|\top)[1, 1]\} \models_{\mathit{tight}} (\mathit{have\_legs}(tweety)|\top)[1, 0] .$$

Note that we obtain $[1, 0]$ under 1-entailment as $ground(P)$ does not have a model Pr such that $Pr(\mathit{penguin}(tweety)) = 1$.

Example 2.3 The knowledge "all magpies are birds", "birds chirp with a probability between 0.7 and 0.8", and "magpies chirp with a probability of at most 0.99" can be expressed by the following probabilistic logic program P:

$$P = \{(\mathit{bird}(X) \,|\, \mathit{magpie}(X))[1, 1], (\mathit{chirp}(X) \,|\, \mathit{bird}(X))[.7, .8], \\ (\mathit{chirp}(X) \,|\, \mathit{magpie}(X))[0, .99]\}.$$

Our wondering about the tight interval for the probability that Sam chirps given that Sam is a magpie can be expressed by the object-ground probabilistic query

$$Q = \exists(\mathit{chirp}(sam)|\mathit{magpie}(sam))[R, S] .$$

Its tight answer substitutions under 0- and 1-entailment are given by $\{R/0,\ S/.99\}$ and $\{R/.7,\ S/.8\}$, respectively.

3 Motivation and Key Ideas

Under 0- and 1-entailment, classical conditional constraints $(\psi|\phi)[1,1]$ and $(\psi|\phi)[0,0]$ are interpreted as "ϕ implies ψ" and "ϕ implies $\neg\psi$", respectively. That is, both 0- and 1-entailment satisfy the following property of *inheritance of classical knowledge along subclass relationships*:

IC If $\mathcal{C} \Vdash\!\sim (\psi|\phi)[c,c]$ and $\phi \Leftarrow \phi^\star$ is logically valid, then $\mathcal{C} \Vdash\!\sim (\psi|\phi^\star)[c,c]$,

for all ground classical formulas ψ, ϕ, and $\phi^\star$, all sets of ground conditional constraints $\mathcal{C}$, and all $c \in \{0,1\}$.

More generally, however, 0-entailment interprets conditional constraints $(\psi|\phi)[l,u]$ as "the conditional probability of ψ given ϕ lies between l and u". That is, 0-entailment does *not* have the following property of *inheritance of probabilistic knowledge along subclass relationships*:

IP If $\mathcal{C} \Vdash\!\sim (\psi|\phi)[l,u]$ and $\phi \Leftarrow \phi^\star$ is logically valid, then $\mathcal{C} \Vdash\!\sim (\psi|\phi^\star)[l,u]$.

for all ground classical formulas ψ, ϕ, and $\phi^\star$, all sets of ground conditional constraints $\mathcal{C}$, and all $l,u \in [0,1]$.

Moreover, 1-entailment interprets $(\psi|\phi)[l,u]$ as "ϕ implies that ψ holds with a probability between l and u". That is, 1-entailment satisfies **IP**, but it does not realize overriding. As the inherited knowledge is often incompatible, we thus often conclude the empty interval (see Example 2.2).

In summary, 0-entailment does not have the property **IP**, while 1-entailment satisfies **IP**, but does not realize overriding. Inheritance with overriding, however, is a desirable feature of probabilistic entailment relations, which is well-known from *reference class reasoning* [34, 17, 18, 31].

A natural way to obtain inheritance with overriding is to weaken 1-entailment by interpreting purely probabilistic conditional constraints $(\psi|\phi)[l,u]$ as "*generally*, ϕ implies that ψ holds with a probability between l and u". We formalize this idea by using recent notions of entailment for probabilistic default theories [22, 23], which are based on default reasoning with conditional knowledge bases.

4 Inheritance with Overriding

We briefly recall approaches to probabilistic default reasoning from [22, 23], which we then use to define entailment under inheritance with overriding. We consider only z- and lex-entailment here. Another approach to probabilistic default reasoning is c-entailment [22, 23], which can also be used for entailment under inheritance with overriding.

4.1 Preliminaries

A *probabilistic default theory* $T = (S, D)$ is a pair of finite sets S and D of ground conditional constraints. The elements in S and D are called *strict conditional constraints* and *defeasible conditional constraints* (or also *defaults*), respectively. Intuitively, every strict conditional constraint $(\psi|\phi)[l,u] \in S$ is interpreted as "the conditional probability of ψ given ϕ is between l and u", while every default $(\psi|\phi)[l,u] \in D$ is interpreted as "generally, ϕ implies that ψ holds with a probability between l and u".

A probabilistic interpretation Pr *verifies* a ground conditional constraints $(\psi|\phi)[l,u]$ iff $Pr(\phi) = 1$ and $Pr \models (\psi|\phi)[l,u]$. It *falsifies* $(\psi|\phi)[l,u]$ iff $Pr(\phi) = 1$ and $Pr \not\models (\psi|\phi)[l,u]$. A set of ground conditional constraints D *tolerates* a ground conditional constraint C *under* a set of ground conditional constraints S iff $S \cup D$ has a model that verifies C. We say D is *under* S *in conflict* with C iff no model of $S \cup D$ verifies C.

Given a probabilistic default theory $T = (S, D)$, a *default ranking* κ on D maps each $C \in D$ to a nonnegative integer. We say κ is *admissible* with $T = (S, D)$ iff each $D' \subseteq D$ that is under S in conflict with some $C \in D$ contains a conditional constraint C' such that $\kappa(C') < \kappa(C)$. A probabilistic default theory $T = (S, D)$ is *consistent* iff there exists a default ranking on D that is admissible with T. It is *inconsistent* iff no such default ranking exists.

A *probability ranking* κ assigns to each probabilistic interpretation on $\mathcal{I}_\Phi$ a member of $\{0, 1, \ldots\} \cup \{\infty\}$ such that $\kappa(Pr) = 0$ for at least one interpretation Pr.

4.2 Entailment is System *Z*

We now recall z-entailment for probabilistic default theories from [22, 23], which is a proper generalization of Pearl's entailment in system Z [30, 12]. It is defined with respect to a consistent probabilistic default theory $T = (S, D)$.

The notion of z-entailment for probabilistic default theories is linked to an ordered partition of D, a default ranking z on D, and a probability ranking κ^z.

Let $(D_0, \ldots, D_k)$ be the unique ordered partition of D such that each D_i is the set of all $d \in D - \bigcup\{D_j \mid 0 \leq j < i\}$ tolerated under S by $D - \bigcup\{D_j \mid 0 \leq j < i\}$. We call this $(D_0, \ldots, D_k)$ the *z-partition* of D.

For every $j \in \{0, \ldots, k\}$, each $d \in D_j$ is assigned the value j under the default ranking z. The probability ranking κ^z on all probabilistic interpretations Pr is then defined by

$$\kappa^z(Pr) \;=\; \begin{cases} \infty & \text{if } Pr \not\models S \\ 0 & \text{if } Pr \models S \cup D \\ 1 + \max\limits_{d \in D:\ Pr \not\models d} z(d) & \text{otherwise.} \end{cases}$$

Note that z is a default ranking admissible with T [23].

The probability ranking κ^z defines a preference relation on probabilistic interpretations as follows. For probabilistic interpretations Pr and Pr', we say Pr is *z-preferable* to Pr' iff $\kappa^z(Pr) < \kappa^z(Pr')$. A model Pr of a set of ground probabilistic formulas $\mathcal{F}$ is a *z-minimal model* of $\mathcal{F}$ iff no model of $\mathcal{F}$ is z-preferable to Pr.

For ground probabilistic formulas E and F, we say F is a *z-consequence* of E, denoted $E \mathrel{\|\!\!\sim}^z F$, iff every z-minimal model of $S \cup \{E\}$ satisfies F. A ground conditional constraint $(\psi|\phi)[l,u]$ is a *tight z-consequence* of E, denoted $E \mathrel{\|\!\!\sim}^z_{\mathit{tight}} (\psi|\phi)[l,u]$, iff l (resp., u) is the infimum (resp., supremum) of $Pr(\psi|\phi)$ subject to all z-minimal models Pr of $S \cup \{E\}$ with $Pr(\phi) > 0$.

4.3 Lexicographic Entailment

We next recall lex-entailment for probabilistic default theories from [22, 23], which is a proper generalization of Lehmann's lexicographic entailment [19]. In the sequel, consider a consistent probabilistic default theory $T = (S, D)$.

We use the z-partition $(D_0, \ldots, D_k)$ of D to define a lexicographic preference relation on probabilistic interpretations as follows. For probabilistic interpretations Pr and Pr', we say Pr is *lex-preferable* to Pr' iff there is some $i \in \{0, \ldots, k\}$ such that $|\{d \in D_i \mid Pr \models d\}| > |\{d \in D_i \mid Pr' \models d\}|$ and $|\{d \in D_j \mid Pr \models d\}| = |\{d \in D_j \mid Pr' \models d\}|$ for all $i < j \leq k$. A model Pr of a set of probabilistic formulas $\mathcal{F}$ is a *lex-minimal model* of $\mathcal{F}$ iff no model of $\mathcal{F}$ is lex-preferable to Pr.

For ground probabilistic formulas E and F, we say F is a *lex-consequence* of E, denoted $E \,|\!\!\sim^{lex} F$, iff every lex-minimal model of $S \cup \{E\}$ satisfies F. A ground conditional constraint $(\psi|\phi)[l, u]$ is a *tight lex-consequence* of E, denoted $E \,|\!\!\sim^{lex}_{tight} (\psi|\phi)[l, u]$, iff l (resp., u) is the infimum (resp., supremum) of $Pr(\psi|\phi)$ subject to all lex-minimal models Pr of $S \cup \{E\}$ with $Pr(\phi) > 0$.

4.4 Inheritance with Overriding

We now introduce z- and lex-entailment for probabilistic logic programs, which are based on z- and lex-entailment for probabilistic default theories.

Each probabilistic logic program P is associated with the probabilistic default theory $T(P) = (S(P), D(P))$, where $S(P)$ (resp., $D(P)$) is the set of all classical (resp., purely probabilistic) members of $ground(P)$. A probabilistic logic program P is *consistent* iff $T(P)$ is consistent. The *z-partition* of P is the z-partition of $T(P)$.

We now define the notion of s-entailment, $s \in \{z, lex\}$, for probabilistic logic programs P. A ground conditional constraint $C = (\psi|\phi)[l, u]$ is an *s-consequence* of P, denoted $P \,|\!\!\sim^{s} C$, iff $(\phi|\top)[1, 1] \,|\!\!\sim^{s} (\psi|\top)[l, u]$ under $T(P)$. It is a *tight s-consequence* of P, denoted $P \,|\!\!\sim^{s}_{tight} C$, iff $(\phi|\top)[1, 1] \,|\!\!\sim^{s}_{tight} (\psi|\top)[l, u]$ under $T(P)$. A conditional constraint $C = (\psi|\phi)[l, u]$ is an *s-consequence* of P, denoted $P \,|\!\!\sim^{s} C$, iff all ground instances of C are s-consequences of P. It is a *tight s-consequence* of P, denoted $P \,|\!\!\sim^{s}_{tight} C$, iff l (resp., u) is the infimum (resp., supremum) of a (resp., b) subject to $P \,|\!\!\sim_{tight} (\psi'|\phi')[a, b]$ and all ground instances $(\psi'|\phi')$ of $(\psi|\phi)$.

4.5 Examples

We now give some illustrative examples.

Example 4.1 Consider again the probabilistic logic program P and the object-ground probabilistic query $Q = \exists(\mathit{have\_legs}(\mathit{tweety})|\mathit{penguin}(\mathit{tweety}))[R, S]$ in Example 2.1. The tight answer substitution for Q to P under both z- and lex-entailment is given by $\{R/.95,\ S/1\}$.

Example 4.2 Consider the probabilistic logic program P and the object-ground probabilistic query

$Q = \exists(\textit{have\_legs}(tweety)|\textit{penguin}(tweety))[R, S]$ given in Example 2.2. The tight answer substitution for Q to P under both z- and lex-entailment is given by $\{R/.95,\ S/1\}$.

Example 4.3 Consider again the probabilistic logic program P and the object-ground probabilistic query $Q = \exists(\textit{chirp}(sam)|\textit{magpie}(sam))[R, S]$ given in Example 2.3. The tight answer substitution for Q to P under both z- and lex-entailment is given by $\{R/.7,\ S/.8\}$.

5 Semantic Properties

We now analyze some general nonmonotonic properties of z- and lex-entailment for probabilistic logic programs.

In the sequel, we write $P \mathrel{\|\!\sim} (\phi|\varepsilon \veebar \varepsilon')[l, u]$ to denote that $(\varepsilon|\top)[1, 1] \vee (\varepsilon'|\top)[1, 1] \mathrel{\|\!\sim} (\phi|\top)[l, u]$ under $T(P)$. We use $P \mathrel{\|\!\not\sim} C$ to denote that it is not the case that $P \mathrel{\|\!\sim} C$. We implicitly assume that all notions of entailment are naturally extended to negations of conditional constraints of the form $\neg(\beta|\alpha)[r, s]$, which are true in a probabilistic interpretation Pr iff $Pr(\alpha) > 0$ and $Pr(\beta|\alpha) \notin [r, s]$.

We first consider the postulates *Right Weakening (RW)*, *Reflexivity (Ref)*, *Left Logical Equivalence (LLE)*, *Cut*, *Cautious Monotonicity (CM)*, and *Or* proposed by Kraus, Lehmann, and Magidor [16], which are commonly regarded as being particularly desirable for any reasonable notion of nonmonotonic entailment. The following result shows that z- and lex-entailment satisfy these postulates.

Theorem 5.1 *$\mathrel{\|\!\sim}^{z}$ and $\mathrel{\|\!\sim}^{lex}$ satisfy the following properties for all probabilistic logic programs P, all ground classical formulas ε, ε', ϕ, and ψ, and all $l, l', u, u' \in [0, 1]$:*

RW. If $(\phi|\top)[l, u] \Rightarrow (\psi|\top)[l', u']$ is logically valid and $P \mathrel{\|\!\sim} (\phi|\varepsilon)[l, u]$, then $P \mathrel{\|\!\sim} (\psi|\varepsilon)[l', u']$.

Ref. $P \mathrel{\|\!\sim} (\varepsilon|\varepsilon)[1, 1]$.

LLE. If $\varepsilon \Leftrightarrow \varepsilon'$ is logically valid, then $P \mathrel{\|\!\sim} (\phi|\varepsilon)[l, u]$ iff $P \mathrel{\|\!\sim} (\phi|\varepsilon')[l, u]$.

Cut. If $P \mathrel{\|\!\sim} (\varepsilon|\varepsilon')[1, 1]$ and $P \mathrel{\|\!\sim} (\phi|\varepsilon \wedge \varepsilon')[l, u]$, then $P \mathrel{\|\!\sim} (\phi|\varepsilon')[l, u]$.

CM. If $P \mathrel{\|\!\sim} (\varepsilon|\varepsilon')[1, 1]$ and $P \mathrel{\|\!\sim} (\phi|\varepsilon')[l, u]$, then $P \mathrel{\|\!\sim} (\phi|\varepsilon \wedge \varepsilon')[l, u]$.

Or. If $P \mathrel{\|\!\sim} (\phi|\varepsilon)[l, u]$ and $P \mathrel{\|\!\sim} (\phi|\varepsilon')[l, u]$, then $P \mathrel{\|\!\sim} (\phi|\varepsilon \veebar \varepsilon')[l, u]$.

Another desirable property is *Rational Monotonicity (RM)* [16], which describes a restricted form of monotony, and allows to ignore certain kinds of irrelevant knowledge. The next result shows that z- and lex-entailment satisfy *RM*.

Theorem 5.2 *$\mathrel{\|\!\sim}^{z}$ and $\mathrel{\|\!\sim}^{lex}$ satisfy the following property for all probabilistic logic programs P, all ground classical formulas ε, ε', and ψ, and all $l, u \in [0, 1]$:*

RM. If $P \mathrel{\|\!\sim} (\psi|\varepsilon)[l, u]$ and $P \mathrel{\|\!\not\sim} \neg(\varepsilon'|\varepsilon)[1, 1]$, then $P \mathrel{\|\!\sim} (\psi|\varepsilon \wedge \varepsilon')[l, u]$.

We finally consider the properties *Irrelevance (Irr)* and *Direct Inference (DI)*, which are adapted from [2] and [1], respectively. Informally, *Irr* says that ε' is irrelevant to "$P \Vert\!\sim (\psi|\varepsilon)[l,u]$" when they are defined over disjoint sets of atoms. While *DI* expresses that P should entail all its own conditional constraints. The following result shows that z- and lex-entailment satisfy *Irr* and *DI*.

Theorem 5.3 *$\Vert\!\sim^{z}$ and $\Vert\!\sim^{lex}$ satisfy the following properties for all probabilistic logic programs P, all ground classical formulas ε, ε', ϕ, and ψ, and all $l, u \in [0,1]$:*

Irr. If $P \Vert\!\sim (\psi|\varepsilon)[l,u]$, and no atom of $ground(P)$ and $(\psi|\varepsilon)[l,u]$ occurs in ε', then $P \Vert\!\sim (\psi|\varepsilon \wedge \varepsilon')[l,u]$.

DI. If $(\psi|\phi)[l,u] \in ground(P)$ and $\varepsilon \Leftrightarrow \phi$ is logically valid, then $P \Vert\!\sim (\psi|\varepsilon)[l,u]$.

Note that entailment under inheritance with overriding based on c-entailment [22, 23] satisfies all the above properties except for Rational Monotonicity.

6 Algorithms

In this section, we give algorithms for probabilistic logic programming under inheritance with overriding.

6.1 Overview

We consider the following problems:

CONSISTENCY: Given a probabilistic logic program P, decide whether P is consistent.

TIGHT S-CONSEQUENCE: Given a consistent probabilistic logic program P and an object-ground probabilistic query $Q = \exists(\beta|\alpha)[x,y]$, compute the tight answer substitution for Q to P under some fixed $s \in \{z, lex\}$.

The main idea behind our algorithms is to reduce these problems to the problem of deciding whether a probabilistic logic program is satisfiable and the problem of computing the tight answer substitution for an object-ground probabilistic query to a probabilistic logic program under 0-entailment, denoted SATISFIABILITY and TIGHT LOGICAL CONSEQUENCE, respectively.

6.2 Consistency

Algorithm z-partition (see Fig. 1) decides whether a probabilistic logic program P is consistent. If this is the case, then z-partition returns the z-partition of P, otherwise *nil*. Note that in Step 5 of z-partition, a number of instances of SATISFIABILITY must be solved. Algorithm z-partition is essentially a reformulation of an algorithm for deciding ε-consistency in default reasoning from conditional knowledge bases by Goldszmidt and Pearl [11].

Algorithm z-partition (essentially [11])

Input: Probabilistic logic program P with $D(P) \neq \emptyset$.
Output: z-partition of P, if P is consistent, otherwise *nil*.

$R := D(P)$;
$i := -1$;
repeat
  $i := i + 1$
  $D[i] := \{C \in R \mid C$ is tolerated under $S(P)$ by $R\}$;
  $R := R - D[i]$
until $R = \emptyset$ **or** $D[i] = \emptyset$;
if $R = \emptyset$ **then return** $(D[0], \ldots, D[i])$
  else return *nil*.

Figure 1: Algorithm z-partition

6.3 Tight S-Consequence

In the sequel, let P be a consistent probabilistic logic program, and let $(D_0, \ldots, D_k)$ be its z-partition.

For $G, H \subseteq D(P)$, we say G is *z-preferable* to H iff some $i \in \{0, \ldots, k\}$ exists such that $D_i \subseteq G$, $D_i \not\subseteq H$, and $D_j \subseteq G$ and $D_j \subseteq H$ for all $i < j \leq k$. We say G is *lex-preferable* to H iff some $i \in \{0, \ldots, k\}$ exists such that $|G \cap D_i| > |H \cap D_i|$ and $|G \cap D_j| = |H \cap D_j|$ for all $i < j \leq k$. For $\mathcal{D} \subseteq 2^{D(P)}$ and $s \in \{z, lex\}$, we say G is *s-minimal* in $\mathcal{D}$ iff $G \in \mathcal{D}$ and no $H \in \mathcal{D}$ is s-preferable to G.

We now reduce TIGHT S-CONSEQUENCE to SATISFIABILITY and TIGHT LOGICAL CONSEQUENCE. The key idea behind this reduction is that there exists a set $\mathcal{D}^s_\alpha(P)$ of subsets of $D(P)$ such that $P \Vdash\!\sim^s (\beta|\alpha)[l, u]$ iff $S(P) \cup H \cup \{(\alpha|\top)[1, 1]\} \models (\beta|\top)[l, u]$ for all $H \in \mathcal{D}^s_\alpha(P)$.

Theorem 6.1 *Let P be a consistent probabilistic logic program, and let $\exists(\beta|\alpha)[x, y]$ be an object-ground probabilistic query. Let $s \in \{z, lex\}$. Let $R = S(P) \cup \{(\alpha|\top)[1, 1]\}$, and let $\mathcal{D}^s_\alpha(P)$ be the set of all s-minimal elements in $\{H \subseteq D(P) \mid R \cup H$ is satisfiable$\}$. Then, l (resp., u) such that $P \Vdash\!\sim^s_{tight} (\beta|\alpha)[l, u]$ is given as follows:*

(a) If R is unsatisfiable, then $l = 1$ (resp., $u = 0$).

(b) Otherwise, $l = \min\, c$ (resp., $u = \max\, d$) subject to $R \cup H \models_{tight} (\beta|\top)[c, d]$ and $H \in \mathcal{D}^s_\alpha(P)$.

Based on Theorem 6.1, Algorithm tight-z-consequence (resp., tight-lex-consequence) computes tight answer substitutions under z-entailment (resp., lex-entailment). Step 2 checks whether R is unsatisfiable. If this is the case, then $\theta = \{x/1,\ y/0\}$ is returned by Theorem 6.1 (a). Otherwise, we compute $\mathcal{D}^s_\alpha(P)$ along the z-partition of P in steps 3–7 (resp., steps 3–15), and the tight answer substitution using Theorem 6.1 (b) in step 8 (resp., steps 16–20).

Algorithm tight-z-consequence

Input: Consistent probabilistic logic program P and object-ground probabilistic query $Q = \exists(\beta|\alpha)[x, y]$.
Output: Tight answer substitution $\theta = \{x/l,\ y/u\}$ for Q to P under z-entailment.
Notation: $(D_0, \ldots, D_k)$ denotes the z-partition of $T(P)$.

$R := S(P) \cup \{(\alpha|\top)[1, 1]\}$;
if R is unsatisfiable **then return** $\theta = \{x/1,\ y/0\}$;
$j := k$;
while $j \geq 0$ **and** $R \cup D_j$ is satisfiable **do begin**
  $R := R \cup D_j$;
  $j := j - 1$
end;
compute $l, u \in [0, 1]$ such that $R \models_{tight} (\beta|\top)[l, u]$;
return $\theta = \{x/l,\ y/u\}$.

Figure 2: Algorithm tight-z-consequence

Algorithm tight-lex-consequence

Input: Consistent probabilistic logic program P and object-ground probabilistic query $Q = \exists(\beta|\alpha)[x, y]$.
Output: Tight answer substitution $\theta = \{x/l,\ y/u\}$ for Q to P under lex-entailment.
Notation: $(D_0, \ldots, D_k)$ denotes the z-partition of $T(P)$.

$R := S(P) \cup \{(\alpha|\top)[1, 1]\}$;
if R is unsatisfiable **then return** $\theta = \{x/1,\ y/0\}$;
$\mathcal{H} := \{\emptyset\}$;
for $j := k$ **downto** 0 **do begin**
  $n := 0$;
  $\mathcal{H}' := \emptyset$;
  for each $G \subseteq D_j$ and $H \in \mathcal{H}$ **do**
    if $R \cup G \cup H$ is satisfiable **then**
      if $n = |G|$ **then** $\mathcal{H}' := \mathcal{H}' \cup \{G \cup H\}$
        else if $n < |G|$ **then begin**
          $\mathcal{H}' := \{G \cup H\}$;
          $n := |G|$
        end;
  $\mathcal{H} := \mathcal{H}'$;
end;
$(l, u) := (1, 0)$;
for each $H \in \mathcal{H}$ **do begin**
  compute $c, d \in [0, 1]$ such that $R \cup H \models_{tight} (\beta|\top)[c, d]$;
  $(l, u) := (\min(l, c), \max(u, d))$
end;
return $\theta = \{x/l,\ y/u\}$.

Figure 3: Algorithm tight-lex-consequence

7 Computational Complexity

In this section, we characterize the computational complexity of the problems CONSISTENCY and TIGHT S-CONSEQUENCE in the propositional case.

7.1 Complexity Classes

We briefly describe the complexity classes that occur in our complexity results. See [9, 29] for further background.

The class NP contains all decision problems that can be solved in nondeterministic polynomial time. The class Δ_2^{P} contains all decision problems that can be solved in deterministic polynomial time with an oracle for NP.

To classify problems that compute an output value, rather than a Yes / No-answer, function classes have been introduced. In particular, $\mathrm{F}\Delta_2^{\mathrm{P}}$ is the functional analog to Δ_2^{P}.

7.2 Complexity Results

We consider the general propositional case as well as the restriction to the 1-conjunctive propositional case. In both cases, the given probabilistic logic programs P are ground and the given probabilistic queries Q are object-ground. In the 1-conjunctive propositional case, we then additionally assume that P and Q are 1-conjunctive.

The results on the propositional complexity of CONSISTENCY and TIGHT S-CONSEQUENCE are shown in Tables 1–2. In detail, CONSISTENCY and TIGHT S-CONSEQUENCE are NP- and $\mathrm{F}\Delta_2^{\mathrm{P}}$-complete, respectively, in the general and the 1-conjunctive propositional case.

That is, they have the same complexity as the problems SATISFIABILITY and TIGHT LOGICAL CONSEQUENCE, respectively, in the respective propositional cases [20]. Intuitively, adding inheritance with overriding to probabilistic logic programming does not increase its complexity.

Table 1: Propositional Complexity of CONSISTENCY

	general case	1-conjunctive case
consistency	NP-complete	NP-complete

Table 2: Propositional Complexity of TIGHT S-CONSEQUENCE

	general case	1-conjunctive case
$\Vdash\!\sim^{z}_{tight}$	$\mathrm{F}\Delta_2^{\mathrm{P}}$-complete	$\mathrm{F}\Delta_2^{\mathrm{P}}$-complete
$\Vdash\!\sim^{lex}_{tight}$	$\mathrm{F}\Delta_2^{\mathrm{P}}$-complete	$\mathrm{F}\Delta_2^{\mathrm{P}}$-complete

8 Summary and Outlook

We presented probabilistic logic programming under inheritance with overriding, which is based on recent approaches to probabilistic default reasoning. We described some general nonmonotonic properties of entailment under inheritance with overriding. Moreover, we presented algorithms for probabilistic logic programming under inheritance with overriding, and we analyzed its propositional complexity.

A very interesting topic of future research is to investigate the relationship to probabilistic logic programming under maximum entropy as presented in [24], where we also have some form of inheritance with overriding.

A Appendix: Proofs

Theorems 5.1, 5.2, 5.3, and 6.1 are special cases of Theorems 6.1, 6.2, 6.3, and 7.1, respectively, in [23]. The membership results shown in Tables 1 and 2 follow from respective membership results for more general problems given in Tables 5 and 7, respectively, in [23].

It thus remains to prove the hardness results for the 1-conjunctive case in Tables 1 and 2.

Theorem A.1 *Given a ground 1-conjunctive probabilistic logic program P, deciding whether P is consistent is* NP*-hard.*

Proof. We give a polynomial transformation from the NP-complete *graph 3-colorability problem* [9]: Given a finite undirected graph $G = (V, E)$, decide whether G is 3-colorable.

We construct a ground 1-conjunctive probabilistic logic program P such that G is 3-colorable iff P is consistent. The vocabulary Φ contains only 0-ary predicate symbols and is defined by:

$$\Phi \;=\; \{p_{i,j} \mid i \in V,\ j \in \{1, 2, 3\}\}\,.$$

The program $P = P_1 \cup P_2 \cup P_3$ is defined as follows:

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

Observe that P is a ground 1-conjunctive probabilistic logic program. As shown by Georgakopoulos et al. [10], G is 3-colorable iff P is satisfiable. It is now easy to see that P is satisfiable iff the probabilistic default theory $T(P) = (P_1 \cup P_2, P_3)$ is consistent, that is, iff P is consistent. In summary, G is 3-colorable iff P is consistent. □

Theorem A.2 *Given a ground 1-conjunctive probabilistic logic program P, which is assumed to be consistent, and an object-ground 1-conjunctive probabilistic query $Q = \exists(\beta|\alpha)[x, y]$, computing the tight answer substitution for Q to P under some fixed $s \in \{z, lex\}$ is* $\mathrm{F}\Delta_2^{\mathrm{P}}$*-hard.*

Proof. We give a polynomial transformation from the $\mathrm{F}\Delta_2^{\mathrm{P}}$-complete *traveling salesman cost problem* [29]: 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 loss of generality, we can assume $n \geq 3$.

Let d denote the smallest length of a tour, and let a be the sum of all $d_{i,j}$ with $i, j \in V$ and $i < j$. We construct P and $Q = \exists(\beta|\alpha)[x, y]$ as in the statement of the theorem such that $d = l \cdot a$, where $\theta = \{x/l,\ y/u\}$ is the tight answer substitution for Q to P under $s \in \{z, lex\}$.

Let $E = \{\{i, j\} \subseteq V \mid i \neq j\}$, $w = d/a$, and $w_{\{i,j\}} = d_{i,j}/a$ 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 by:

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

We next define a probabilistic logic program 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 $Pr(I) > 0$ for some model Pr of P_1 corresponds to a permutation of the members in V, and vice versa. We next define a probabilistic logic program 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 $Q = \exists(p \mid \top)[x, y]$. Observe that P and Q are 1-conjunctive, P is ground, and Q is object-ground. Moreover, notice that P is satisfiable. It is then easy to see that $T(P)$ is consistent, that is, P is consistent. As P is satisfiable, it follows from Theorem 6.1 that the tight answer substitution for Q to P under $s \in \{z, lex\}$ is given by the tight answer substitution for Q to P under 0-entailment. As shown in [20], the latter is given by $\theta = \{x/w,\ y/u\}$. $\Box$

References

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

[2] S. Benferhat, A. Saffiotti, and P. Smets. Belief functions and default reasoning. *Artif. Intell.*, 122:1–69, 2000.

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

[4] A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. In *Proceedings ICLP-97*, pp. 391–405, 1997.

[5] M. I. Dekhtyar, A. Dekhtyar, and V. S. Subrahmanian. Hybrid probabilistic programs: Algorithms and complexity. In *Proceedings UAI-99*, pp. 160–169, 1999.

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

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

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

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

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

[11] M. Goldszmidt and J. Pearl. On the consistency of defeasible databases. *Artif. Intell.*, 52(2):121–149, 1991.

[12] M. Goldszmidt and J. Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artif. Intell.*, 84(1–2):57–112, 1996.

[13] P. Haddawy. Generating Bayesian networks from probability logic knowledge bases. In *Proceedings UAI-94*, pp. 262–269. 1994.

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

[15] M. Jaeger. Relational Bayesian networks. In *Proceedings UAI-97*, pp. 266–273. 1997.

[16] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artif. Intell.*, 14(1):167–207, 1990.

[17] H. E. Kyburg, Jr. *The Logical Foundations of Statistical Inference*. D. Reidel, Dordrecht, 1974.

[18] H. E. Kyburg, Jr. The reference class. *Philos. Sci.*, 50:374–397, 1983.

[19] D. Lehmann. Another perspective on default reasoning. *Ann. Math. Artif. Intell.*, 15(1):61–82, 1995.

[20] T. Lukasiewicz. Probabilistic logic programming with conditional constraints. *ACM Trans. Computat. Logic*. 2(3):289–337, July 2001. To appear.

[21] T. Lukasiewicz. Credal networks under maximum entropy. In *Proceedings UAI-2000*, pp. 363–370, 2000.

[22] T. Lukasiewicz. Probabilistic default reasoning with conditional constraints. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning*, Breckenridge, CO, USA, April 2000.

[23] T. Lukasiewicz. Probabilistic default reasoning with strict and defeasible conditional constraints. Technical Report INFSYS RR-1843-00-02, Institut für Informationssysteme, TU Wien, 2000.

[24] T. Lukasiewicz and G. Kern-Isberner. Probabilistic logic programming under maximum entropy. In *Proc. ECSQARU-99*, *LNCS* 1638, pp. 279–292. 1999.

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

[26] 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.

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

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

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

[30] J. Pearl. System Z: A natural ordering of defaults with tractable applications to default reasoning. In *Proceedings TARK-90*, pp. 121–135, 1990.

[31] J. L. Pollock. *Nomic Probabilities and the Foundations of Induction*. Oxford University Press, 1990.

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

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

[34] H. Reichenbach. *Theory of Probability*. University of California Press, Berkeley, CA, 1949.

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