Probabilistic and Truth-Functional Many-Valued Logic Programming

Thomas Lukasiewicz
Institut für Informatik, Universität Gießen
Arndtstraße 2, D-35392 Gießen, Germany

Abstract

We introduce probabilistic many-valued logic programs in which the implication connective is interpreted as material implication. We show that probabilistic many-valued logic programming is computationally more complex than classical logic programming. More precisely, some deduction problems that are P-complete for classical logic programs are shown to be co-NP-complete for probabilistic many-valued logic programs. We then focus on many-valued logic programming in $\mathrm{Pr}_n^\star$ *as an approximation of probabilistic many-valued logic programming. Surprisingly, many-valued logic programs in* $\mathrm{Pr}_n^\star$ *have both a probabilistic semantics in probabilities over a set of possible worlds and a truth-functional semantics in the finite-valued Łukasiewicz logics* $Ł_n$*. Moreover, many-valued logic programming in* $\mathrm{Pr}_n^\star$ *has a model and fixpoint characterization, a proof theory, and computational properties that are very similar to those of classical logic programming.*

1. Introduction

We start by presenting probabilistic many-valued logic programs in which the implication connective is interpreted as material implication. We show that probabilistic many-valued logic programming in this framework is computationally more complex than classical logic programming. More precisely, some deduction problems that are P-complete for classical logic programs are shown to be co-NP-complete for probabilistic many-valued logic programs (see also [15] and [14] for other work on the subtleties and the computational complexity of probabilistic deduction).

We then focus on many-valued logic programming in $\mathrm{Pr}_n^\star$ as an approximation of probabilistic many-valued logic programming. Crucially, many-valued logic programming in $\mathrm{Pr}_n^\star$ has a model and fixpoint characterization and a proof theory that are very similar to those of classical logic programming. Furthermore, special cases of many-valued logic programming in $\mathrm{Pr}_n^\star$ have the same computational complexity like their classical counterparts.

Surprisingly (and at first sight even paradoxically), many-valued logic programs in $\mathrm{Pr}_n^\star$ have both a probabilistic semantics in probabilities over a set of possible worlds and a truth-functional semantics in the finite-valued Łukasiewicz logics $Ł_n$. That is, many-valued logic programming in $\mathrm{Pr}_n^\star$ lies in the intersection between probabilistic logics and truth-functional many-valued logics.

The literature already contains quite extensive work on probabilistic and on truth-functional many-valued logic programming separately. However, to the best of our knowledge, an integration of both has never been studied so far.

Probabilistic propositional logics and their various dialects are thoroughly studied in the literature (see, for example, [18] and [5]). Their extensions to probabilistic first-order logics can be classified into first-order logics in which probabilities are defined over a set of possible worlds and those in which probabilities are given over the domain (see, for example, [8] and [2]). The first ones are suitable for representing degrees of belief, while the latter are appropriate for describing statistical knowledge. The same classification holds for probabilistic logic programming (see, for example, [17], [14], and [16]).

Many approaches to truth-functional finite-valued logic programming are restricted to three or four truth values (see, for example, [10], [6], and [4]). Among these approaches, the one closest in spirit to many-valued logic programming in $\mathrm{Pr}_n^\star$ is perhaps the three-valued one in [10].

Many-valued logic programming in $\mathrm{Pr}_n^\star$ is closely related to van Emden's infinite-valued quantitative deduction [19]. More precisely, it is an approximation of probabilistic logic programming under the material implication, while van Emden's quantitative deduction can be understood as an approximation of probabilistic logic programming under the conditional probability implication [14].

Moreover, many-valued logic programming in $\mathrm{Pr}_n^\star$ is related to the work on generalized annotated logic programming [9] and to signed formula logic programming [11].

Many-valued logic programming in $\mathrm{Pr}_n^\star$ itself was initiated in [12], where we already presented a model and fixpoint characterization.

The rest of this paper is organized as follows. Section 2 deals with probabilistic many-valued logic programming. In Section 3, we concentrate on many-valued logic programming in $\mathrm{Pr}_n^\star$. Section 4 summarizes the main results.

This paper is an extract from a longer version [13], which includes in full detail all the proofs missing here.

2. Many-valued logic programming in $\mathbf{Pr}_n$

2.1. Technical preliminaries

We briefly summarize how classical first-order logics can be given a probabilistic n-valued semantics with $n \geq 3$ in which probabilities are defined over a set of possible worlds. We basically follow the important work of Halpern [8], which we adapt and restrict to the n-valued setting.

Let $TV = \{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}$ with $n \geq 3$ denote the set of *truth values*. Let Φ be a first-order vocabulary that contains a set of function symbols and a set of predicate symbols (as usual, *constant symbols* are function symbols of arity zero; we say Φ is *function-free* if it does not contain any function symbols of arity greater than zero). Let $\mathcal{X}$ be a set of *object* and *truth variables*.

We define *object terms* by induction as follows. An object term is an object variable from $\mathcal{X}$ or an expression of the kind $f(t_1, \ldots, t_k)$, where f is a function symbol of arity $k \geq 0$ from Φ and $t_1, \ldots, t_k$ are object terms. A *truth term* is a truth value from TV or a truth variable from $\mathcal{X}$. We define *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 formula (called *atom*). If F and G are formulas, then $\neg F$, $(F \wedge G)$, $(F \vee G)$, and $(F \leftarrow G)$ are formulas. If F is a formula and x is an object variable from $\mathcal{X}$, then $\forall x\, F$ and $\exists x\, F$ are formulas. An *n-valued formula* is an expression $\mathsf{tv}(F) \geq t$, where F is a classical formula and t is a truth term.

An *interpretation* $I = (D, \pi)$ consists of a nonempty set D, called *domain*, and a mapping π that assigns to each function symbol from Φ a function of right arity over D and to each predicate symbol from Φ a predicate of right arity over D. A *variable assignment* σ is a mapping that assigns to each object variable from $\mathcal{X}$ an element from D and to each truth variable from $\mathcal{X}$ a truth value from TV. For an object variable x from $\mathcal{X}$ and an element d from D, we write $\sigma[x/d]$ to denote the variable assignment that is identical to σ except that it assigns d to x (for a truth variable x from $\mathcal{X}$ and a truth value c from TV, the notation $\sigma[x/c]$ has an analogous meaning). The variable assignment σ is by induction extended to all object and truth terms by defining $\sigma(f(t_1, \ldots, t_k)) = \pi(f)(\sigma(t_1), \ldots, \sigma(t_k))$ for all object terms $f(t_1, \ldots, t_k)$ and $\sigma(c) = c$ for all truth values c from TV. The *truth* of formulas F in I under σ, denoted $I \models_\sigma F$, is inductively defined as follows:

- $I \models_\sigma p(t_1, \ldots, t_k)$ iff $(\sigma(t_1), \ldots, \sigma(t_k)) \in \pi(p)$.
- $I \models_\sigma \neg F$ iff not $I \models_\sigma F$.
- $I \models_\sigma (F \wedge G)$ iff $I \models_\sigma F$ and $I \models_\sigma G$.
- $I \models_\sigma \forall x\, F$ iff $I \models_{\sigma[x/d]} F$ for all $d \in D$.
- The truth of the remaining formulas in I under σ is defined by expressing $\vee$, $\leftarrow$, and $\exists$ in terms of $\neg$, $\wedge$, and $\forall$ as usual.

A formula F *is true* in I, or I is a *model* of F, denoted $I \models F$, iff F is true in I under all variable assignments σ.

A *probabilistic interpretation* (Pr_n*-interpretation*) Pr is a triple $(D, \mathcal{I}, \mu)$, where D is a nonempty set (called *domain*), $\mathcal{I}$ is a set of classical interpretations over D (which are called *possible worlds*) such that $\pi_i(f) = \pi_j(f)$ for all function symbols f from Φ and all interpretations $(D, \pi_i), (D, \pi_j) \in \mathcal{I}$, and μ is a mapping from $\mathcal{I}$ to the set of truth values TV such that all $\mu(I)$ with $I \in \mathcal{I}$ sum up to 1. The *truth value* $Pr_\sigma(F)$ of a formula F in the Pr_n-interpretation Pr under a variable assignment σ is defined as follows (we write $Pr(F)$ if F is variable-free):

$$Pr_\sigma(F) \;=\; \textstyle\sum_{I \in \mathcal{I},\, I \models_\sigma F} \mu(I)\,. \qquad (1)$$

An n-valued formula $\mathsf{tv}(F) \geq t$ *is true* in Pr under σ iff $Pr_\sigma(F) \geq \sigma(t)$. An n-valued formula P *is true* in Pr, or Pr is a *model* of P, denoted $Pr \models P$, iff P is true in Pr under all variable assignments σ. Pr is a *model* of a set of n-valued formulas $\mathcal{P}$, denoted $Pr \models \mathcal{P}$, iff Pr is a model of all n-valued formulas in $\mathcal{P}$. $\mathcal{P}$ is *satisfiable* iff a model of $\mathcal{P}$ exists. P is a *logical consequence* of $\mathcal{P}$, denoted $\mathcal{P} \models P$, iff each model of $\mathcal{P}$ is also a model of P.

For an n-valued formula $\mathsf{tv}(F) \geq c$ with a truth value c from TV and a set of n-valued formulas $\mathcal{P}$, let $\boldsymbol{c}$ denote the set of all truth values $Pr_\sigma(F)$ in models Pr of $\mathcal{P}$ under variable assignments σ. It is easy to see that $\mathsf{tv}(F) \geq c$ is a logical consequence of $\mathcal{P}$ iff $c \leq \min \boldsymbol{c}$. Hence, we get a natural notion of tightness for logical consequences: the n-valued formula $\mathsf{tv}(F) \geq c$ is a *tight logical consequence* of $\mathcal{P}$, denoted $\mathcal{P} \models_{\mathit{tight}} \mathsf{tv}(F) \geq c$, iff $c = \min \boldsymbol{c}$.

A *Herbrand* Pr_n*-interpretation* $(\mathcal{I}, \mu)$ consists of a set $\mathcal{I}$ of classical Herbrand interpretations over Φ (that is, subsets of the Herbrand base HB_Φ over Φ) and a mapping μ from $\mathcal{I}$ to TV such that all $\mu(I)$ with $I \in \mathcal{I}$ sum up to 1.

Terms, formulas, n-valued formulas, and sets of n-valued formulas are *ground* iff they do not contain any variables. The notions of substitutions, ground substitutions, instances of formulas, and ground instances of formulas are defined as usual. The last two are assumed to be canonically extended to n-valued formulas. Finally, we also adopt the usual conventions to eliminate parentheses.

2.2. Many-valued logic programs

We now introduce probabilistic many-valued logic programs. We start by defining many-valued program clauses, which are special many-valued formulas.

An *n-valued program clause* is an n-valued formula $\mathsf{tv}(H \vee \neg B_1 \vee \cdots \vee \neg B_k) \geq c$, where $H, B_1, \ldots, B_k$ with

$k \geq 0$ are atoms and c is a truth value from TV. It is abbreviated by $(H \leftarrow B_1, \ldots, B_k)[c, 1]$. Note that all object variables in an n-valued program clause are implicitly universally quantified. An *n-valued logic program* $\mathcal{P}$ is a finite set of n-valued program clauses. We use $ground(\mathcal{P})$ to denote the set of all ground instances of clauses in $\mathcal{P}$.

Many-valued program clauses can be classified into facts and rules: *facts* are of the kind $(H \leftarrow)[c, 1]$, while *rules* have the form $(H \leftarrow B_1, \ldots, B_k)[c, 1]$ with $k > 0$. They can also be divided into logical and purely many-valued program clauses: *logical* program clauses are of the kind $(H \leftarrow B_1, \ldots, B_k)[1, 1]$, while *purely many-valued* ones have the form $(H \leftarrow B_1, \ldots, B_k)[c, 1]$ with $c < 1$.

Next, we introduce many-valued queries, answer substitutions, and answers. An *n-valued query* to an n-valued logic program $\mathcal{P}$ is an expression $\exists(A_1, \ldots, A_l)[t, 1]$, where $A_1, \ldots, A_l$ with $l \geq 1$ are atoms and t is a truth term. An n-valued query is *object-ground* iff it does not contain any object variables. Given an n-valued query $Q_c = \exists(A_1, \ldots, A_l)[c, 1]$ with $c \in TV$, we are interested in its *correct answer substitutions*, which are substitutions θ such that $\mathcal{P} \models \mathsf{tv}((A_1 \wedge \cdots \wedge A_l)\theta) \geq c$ and that θ acts only on variables in Q_c. The *correct answer* for Q_c is Yes if a correct answer substitution exists and No otherwise. Given an n-valued query $Q_x = \exists(A_1, \ldots, A_l)[x, 1]$ with $x \in \mathcal{X}$, we are interested in its *tight answer substitutions*, which are substitutions θ such that $\mathcal{P} \models_{\mathit{tight}} \mathsf{tv}((A_1 \wedge \cdots \wedge A_l)\theta) \geq x\theta$, that θ acts only on variables in Q_x, and that $x\theta$ is a truth value from TV. Note that such n-valued queries Q_x always have a tight answer substitution.

Example 2.1 Let $n = 101$ and let the n-valued logic program $\mathcal{P}$ contain the following rules and facts (R, S, and T are object variables; *h*, *a*, *b*, and *o* are constants):

$$
\begin{aligned}
&(\mathit{re}(R, S) \leftarrow \mathit{ro}(R, S))[.7, 1] \\
&(\mathit{re}(R, S) \leftarrow \mathit{ro}(R, S), \mathit{so}(R, S))[.9, 1] \\
&(\mathit{re}(R, S) \leftarrow \mathit{ro}(R, S), \mathit{ad}(R, S))[1, 1] \\
&(\mathit{re}(R, S) \leftarrow \mathit{re}(R, T), \mathit{re}(T, S))[1, 1] \\
&(\mathit{ro}(h, a) \leftarrow\ )[1, 1], (\mathit{ad}(h, a) \leftarrow\ )[1, 1] \\
&(\mathit{ro}(a, b) \leftarrow\ )[1, 1], (\mathit{ad}(a, b) \leftarrow\ )[.8, 1] \\
&(\mathit{ro}(b, o) \leftarrow\ )[1, 1], (\mathit{so}(b, o) \leftarrow\ )[1, 1]
\end{aligned}
$$

Then, some many-valued queries are $\exists(\mathit{re}(h, o))[.99, 1]$, $\exists(\mathit{re}(h, U))[.8, 1]$, and $\exists(\mathit{re}(h, o))[X, 1]$, where U is an object variable and X is a truth variable. The correct answer for $\exists(\mathit{re}(h, o))[.99, 1]$ to $\mathcal{P}$ is No, whereas the correct answer for $\exists(\mathit{re}(h, U))[.8, 1]$ to $\mathcal{P}$ is Yes (all the correct answer substitutions for $\exists(\mathit{re}(h, U))[.8,1]$ to $\mathcal{P}$ are given by $\{U/a\}$ and $\{U/b\}$). Finally, the unique tight answer substitution for $\exists(\mathit{re}(h, o))[X, 1]$ to $\mathcal{P}$ is given by $\{X/.7\}$.

Like classical logic programs, many-valued logic programs have the nice property that they are always satisfiable [13]. Furthermore, ground many-valued formulas are logically entailed in Pr_n-interpretations iff they are logically entailed in Herbrand Pr_n-interpretations [13].

In the sequel, we use *probabilistic many-valued logic programming* as a synonym for the problem of deciding whether Yes is the correct answer for a given ground many-valued query to a many-valued logic program.

2.3. Computational complexity

We now analyze the computational complexity of two decidable special cases of probabilistic many-valued logic programming. The first one is a generalization of propositional logic programming, while the second one generalizes the decision problem that defines the data complexity of datalog. These two special cases are of special interest, since their classical counterparts have the nice property that they are P-complete (see, for example, [3] for a survey).

Crucially, the P-completeness does not carry over to the two probabilistic many-valued generalizations:

Theorem 2.2 *a) The problem of deciding whether* Yes *is the correct answer for a ground n-valued query $\exists(A_1, \ldots, A_l)[c, 1]$ to a ground n-valued logic program $\mathcal{P}$ is co-NP-complete. b) Let Φ be function-free. Let $\mathcal{P}$ be a fixed n-valued logic program and let $\mathcal{F}$ be a varying finite set of ground logical facts. Let $\mathcal{P} \cup \mathcal{F}$ contain all constant symbols from Φ. The problem of deciding whether* Yes *is the correct answer for a ground n-valued query $\exists(A_1, \ldots, A_l)[c, 1]$ to $\mathcal{P} \cup \mathcal{F}$ is co-NP-complete.*

Hence, restricted deduction problems that are computationally tractable for classical logic programs are presumably intractable for many-valued logic programs. Thus, any attempt towards efficient probabilistic many-valued logic programming should be guided by looking for efficient special-case, average-case, or approximation techniques.

3. Many-valued logic programming in $\mathbf{Pr}_n^\star$

3.1. $\mathbf{Pr}_n^\star$-interpretations

Probabilistic many-valued logic programming as introduced in Section 2.2 has a well-defined probabilistic semantics. However, its increased computational complexity compared to classical logic programming is quite discouraging for a broad use in practice, especially for a possible application in large knowledge-base systems.

This increase in complexity seems to be mainly due to the probabilistic semantics in its full generality. In fact, we now provide a truth-functional approach to many-valued logic programming that approximates our probabilistic one and that is less computationally complex. The main idea is to focus on a special kind of Pr_n-interpretations:

A $\mathrm{Pr}_n^\star$-*interpretation* is a Pr_n-interpretation Pr with

$$Pr_\sigma(A \wedge B) = \min(Pr_\sigma(A), Pr_\sigma(B)) \tag{2}$$

for all variable assignments σ and all atoms A and B.

Interestingly, (2) is equivalently expressed as follows.

Theorem 3.1 *Let $Pr = (D, \mathcal{I}, \mu)$ be a Pr_n-interpretation. It holds $Pr_\sigma(A \wedge B) = \min(Pr_\sigma(A), Pr_\sigma(B))$ for all variable assignments σ and all atoms A and B iff all the interpretations $I \in \mathcal{I}$ with $\mu(I) > 0$ can be written in a sequence $(D, \pi_1), \ldots, (D, \pi_k)$ such that for all predicate symbols p from Φ: $\pi_1(p) \supseteq \pi_2(p) \supseteq \cdots \supseteq \pi_k(p)$.*

A $\mathrm{Pr}_n^\star$-*model* of a set of n-valued formulas $\mathcal{P}$ is a $\mathrm{Pr}_n^\star$-interpretation that is a model of $\mathcal{P}$. The set of n-valued formulas $\mathcal{P}$ is *satisfiable in* $\mathrm{Pr}_n^\star$ iff a $\mathrm{Pr}_n^\star$-model of $\mathcal{P}$ exists. The n-valued formula P is a *logical consequence in* $\mathrm{Pr}_n^\star$ of $\mathcal{P}$ iff each $\mathrm{Pr}_n^\star$-model of $\mathcal{P}$ is also a model of P. The n-valued formula $\mathsf{tv}(F) \geq c$ is a *tight logical consequence in* $\mathrm{Pr}_n^\star$ of $\mathcal{P}$ iff c is the minimum of all truth values $Pr_\sigma(F)$ in $\mathrm{Pr}_n^\star$-models Pr of $\mathcal{P}$ under variable assignments σ.

The next theorem shows that tight logical consequences in $\mathrm{Pr}_n^\star$ approximate logical and tight logical consequences in Pr_n. In particular, for many-valued logic programs $\mathcal{P}$ and formulas F, this theorem shows that $\mathcal{P} \models_{\textit{tight}} \mathsf{tv}(F) \geq 0$ in $\mathrm{Pr}_n^\star$ immediately entails $\mathcal{P} \models_{\textit{tight}} \mathsf{tv}(F) \geq 0$ in Pr_n.

Theorem 3.2 *Let $\mathcal{F}$ be a set of n-valued formulas, let F be a formula, and let $c \in TV$. If $\mathcal{F} \models_{\textit{tight}} \mathsf{tv}(F) \geq c$ in $\mathrm{Pr}_n^\star$, then all truth values $d \in TV$ with $\mathcal{F} \models \mathsf{tv}(F) \geq d$ in Pr_n are contained in $\{0, \ldots, c\} \subseteq TV$.*

3.2. Comparison with Ł$_n$-interpretations

We now focus on the relationship between $\mathrm{Pr}_n^\star$-interpretations and interpretations in Ł$_n$. We first define Ł$_n$-interpretations and the truth value of classical formulas in Ł$_n$-interpretations under variable assignments.

An Ł$_n$-*interpretation* $L = (D, \boldsymbol{\pi})$ consists of a nonempty domain D and a mapping $\boldsymbol{\pi}$ that assigns to each k-ary function symbol from Φ a mapping from D^k to D and to each k-ary predicate symbol from Φ a mapping from D^k to the set of truth values TV. The *truth value* $L_\sigma(F)$ of a formula F in the Ł$_n$-interpretation L under a variable assignment σ is inductively defined by:

- $L_\sigma(p(t_1, \ldots, t_k)) = \boldsymbol{\pi}(p)(\sigma(t_1), \ldots, \sigma(t_k))$.
- $L_\sigma(\neg F) = 1 - L_\sigma(F)$.
- $L_\sigma(F \wedge G) = \min(L_\sigma(F), L_\sigma(G))$.
- $L_\sigma(F \vee G) = \max(L_\sigma(F), L_\sigma(G))$.
- $L_\sigma(F \leftarrow G) = \min(1, L_\sigma(F) - L_\sigma(G) + 1)$.
- $L_\sigma(\forall x\, F) = \min\{L_{\sigma[x/d]}(F) \mid d \in D\}$.
- $L_\sigma(\exists x\, F) = \max\{L_{\sigma[x/d]}(F) \mid d \in D\}$.

We next show that for logical combinations of certain formulas, the truth value in $\mathrm{Pr}_n^\star$-interpretations under variable assignments is defined like the truth value in Ł$_n$-interpretations under variable assignments.

Lemma 3.3 *Let $Pr = (D, \mathcal{I}, \mu)$ be a $\mathrm{Pr}_n^\star$-interpretation and let σ be a variable assignment. For all object variables $x \in \mathcal{X}$, all formulas F, and all formulas G and H that are built without the logical connectives $\neg$ and $\leftarrow$:*

$$\begin{aligned}
Pr_\sigma(\neg F) &= 1 - Pr_\sigma(F) & (3)\\
Pr_\sigma(G \wedge H) &= \min(Pr_\sigma(G), Pr_\sigma(H)) & (4)\\
Pr_\sigma(G \vee H) &= \max(Pr_\sigma(G), Pr_\sigma(H)) & (5)\\
Pr_\sigma(G \leftarrow H) &= \min(1, Pr_\sigma(G) - Pr_\sigma(H) + 1) & (6)\\
Pr_\sigma(\forall x\, G) &= \min\{Pr_{\sigma[x/d]}(G) \mid d \in D\} & (7)\\
Pr_\sigma(\exists x\, G) &= \max\{Pr_{\sigma[x/d]}(G) \mid d \in D\}\,. & (8)
\end{aligned}$$

This means that $\mathrm{Pr}_n^\star$- and Ł$_n$-interpretations give the same truth value to all formulas built without the logical connectives $\neg$ and $\leftarrow$, and to all logical combinations of these formulas (thus, also to classical program clauses):

Theorem 3.4 *Let Pr be a $\mathrm{Pr}_n^\star$-interpretation, let L be an Ł$_n$-interpretation, and let σ be a variable assignment. If $Pr_\sigma(A) = L_\sigma(A)$ for all atoms A, then $Pr_\sigma(G) = L_\sigma(G)$, $Pr_\sigma(\neg G) = L_\sigma(\neg G)$, and $Pr_\sigma(G \leftarrow H) = L_\sigma(G \leftarrow H)$ for all formulas G and H built without $\neg$ and $\leftarrow$.*

Note that there also exist formulas with different truth values in $\mathrm{Pr}_n^\star$-interpretations and in Ł$_n$-interpretations:

Theorem 3.5 *There are $\mathrm{Pr}_n^\star$-interpretations Pr, Ł$_n$-interpretations L, variable assignments σ, and formulas G with $Pr_\sigma(A) = L_\sigma(A)$ for all atoms A and $Pr_\sigma(G) \neq L_\sigma(G)$.*

This last theorem is not surprising, since $\mathrm{Pr}_n^\star$-interpretations still satisfy the axioms of probability. That is, $\mathrm{Pr}_n^\star$-interpretations always give the same truth value to formulas that are logically equivalent in the classical sense. Ł$_n$-interpretations, in contrast, do not have this property.

3.3. Many-valued logic programs

We keep the definitions of many-valued program clauses and many-valued programs from Section 2.2. In particular, the semantics of many-valued program clauses in $\mathrm{Pr}_n^\star$-interpretations is already given by the semantics of many-valued formulas in Pr_n-interpretations. The truth of many-valued program clauses in $\mathrm{Pr}_n^\star$-interpretations is then additionally characterized as follows.

Lemma 3.6 *For all $\mathrm{Pr}_n^\star$-interpretations $Pr = (D, \mathcal{I}, \mu)$, all variable assignments σ, and all n-valued program clauses $(H \leftarrow B_1, \ldots, B_k)[c, 1]$:*

$$(H \leftarrow B_1, \ldots, B_k)[c, 1] \textit{ is true in } Pr \textit{ under } \sigma \textit{ iff}$$
$$Pr_\sigma(H) \geq c - 1 + \min(Pr_\sigma(B_1), \ldots, Pr_\sigma(B_k))\,.$$

Given an n-valued query $Q_c = \exists(A_1, \ldots, A_l)[c, 1]$ with $c \in TV$, we are interested in its *correct answer substitutions in* $\mathrm{Pr}_n^\star$, which are substitutions θ such that $\mathcal{P} \models \mathsf{tv}(A_1\theta \wedge \cdots \wedge A_l\theta) \geq c$ in $\mathrm{Pr}_n^\star$ and that θ acts only on variables in Q_c. The *correct answer in* $\mathrm{Pr}_n^\star$ for Q_c is Yes if a correct answer substitution in $\mathrm{Pr}_n^\star$ exists and No otherwise. Given an n-valued query $Q_x = \exists(A_1, \ldots, A_l)[x, 1]$ with $x \in \mathcal{X}$, we are interested in its *tight answer substitutions in* $\mathrm{Pr}_n^\star$, which are substitutions θ such that $\mathcal{P} \models_{\mathit{tight}} \mathsf{tv}(A_1\theta \wedge \cdots \wedge A_l\theta) \geq x\theta$ in $\mathrm{Pr}_n^\star$, that θ acts only on variables in Q_x, and that $x\theta$ is a truth value from TV.

Example 3.7 Let $n = 101$ and let $\mathcal{P}$ be the n-valued logic program from Example 2.1. The correct answer in $\mathrm{Pr}_n^\star$ for the n-valued query $\exists(re(h, o))[.99, 1]$ to $\mathcal{P}$ is No, whereas the correct answer in $\mathrm{Pr}_n^\star$ for $\exists(re(h, U))[.8, 1]$ to $\mathcal{P}$ is Yes (note that all the correct answer substitutions in $\mathrm{Pr}_n^\star$ for $\exists(re(h, U))[.8,1]$ to $\mathcal{P}$ are given by $\{U/a\}$, $\{U/b\}$, and $\{U/o\}$). Finally, the unique tight answer substitution in $\mathrm{Pr}_n^\star$ for $\exists(re(h, o))[X, 1]$ to $\mathcal{P}$ is given by $\{X/.8\}$.

Note that many-valued logic programs are always satisfiable in $\mathrm{Pr}_n^\star$ [13]. Moreover, ground many-valued formulas are logically entailed in $\mathrm{Pr}_n^\star$-interpretations iff they are logically entailed in Herbrand $\mathrm{Pr}_n^\star$-interpretations [13].

In the sequel, we use *many-valued logic programming in* $\mathrm{Pr}_n^\star$ as a synonym for the problem of deciding whether Yes is the correct answer in $\mathrm{Pr}_n^\star$ for a given ground many-valued query to a many-valued logic program.

3.4. Model and fixpoint semantics

We briefly discuss the model and fixpoint semantics of many-valued logic programs in $\mathrm{Pr}_n^\star$ [12]. In the sequel, let $\mathcal{P}$ be an n-valued logic program.

We focus on Herbrand $\mathrm{Pr}_n^\star$-interpretations, which we identify with fuzzy sets. In detail, each Herbrand $\mathrm{Pr}_n^\star$-interpretation $(\mathcal{I}, \mu)$ is identified with the fuzzy set $\boldsymbol{I}\colon HB_\Phi \to TV$, where $\boldsymbol{I}[A]$, for all $A \in HB_\Phi$, is the sum of all $\mu(I)$ with $I \in \mathcal{I}$ and $I \models A$. We subsequently use bold symbols to denote such fuzzy sets. The fuzzy sets $\emptyset$ and $\boldsymbol{HB}_\Phi$ are defined by $\emptyset[A] = 0$ and $\boldsymbol{HB}_\Phi[A] = 1$ for all $A \in HB_\Phi$. Finally, we define the intersection, the union, and the subset relation for fuzzy sets $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ as usual by $\boldsymbol{S}_1 \cap \boldsymbol{S}_2 = \min(\boldsymbol{S}_1, \boldsymbol{S}_2)$, $\boldsymbol{S}_1 \cup \boldsymbol{S}_2 = \max(\boldsymbol{S}_1, \boldsymbol{S}_2)$, and $\boldsymbol{S}_1 \subseteq \boldsymbol{S}_2$ iff $\boldsymbol{S}_1 = \boldsymbol{S}_1 \cap \boldsymbol{S}_2$, respectively.

We define the immediate consequence operator $\boldsymbol{T}_\mathcal{P}$ as follows. For all $\boldsymbol{I} \subseteq \boldsymbol{HB}_\Phi$ and $H \in HB_\Phi$:

$$\boldsymbol{T}_\mathcal{P}(\boldsymbol{I})[H] \;=\; \max(\{c - 1 + \min(\boldsymbol{I}[B_1], \ldots, \boldsymbol{I}[B_k]) \mid (H \leftarrow B_1, \ldots, B_k)[c, 1] \in \mathit{ground}(\mathcal{P})\} \cup \{0\})\,.$$

Note that we define $\min(\boldsymbol{I}[B_1], \ldots, \boldsymbol{I}[B_k]) = 1$ for $k = 0$.

For all $\boldsymbol{I} \subseteq \boldsymbol{HB}_\Phi$, we define $\boldsymbol{T}_\mathcal{P}\!\uparrow\omega(\boldsymbol{I})$ as the union of all $\boldsymbol{T}_\mathcal{P}\!\uparrow l(\boldsymbol{I})$ with $l < \omega$, where $\boldsymbol{T}_\mathcal{P}\!\uparrow 0(\boldsymbol{I}) = \boldsymbol{I}$ and $\boldsymbol{T}_\mathcal{P}\!\uparrow(l+1)(\boldsymbol{I}) = \boldsymbol{T}_\mathcal{P}(\boldsymbol{T}_\mathcal{P}\!\uparrow l(\boldsymbol{I}))$ for all $l < \omega$. Finally, we abbreviate $\boldsymbol{T}_\mathcal{P}\!\uparrow\alpha(\emptyset)$ by $\boldsymbol{T}_\mathcal{P}\!\uparrow\alpha$.

The model and fixpoint semantics of many-valued logic programs in $\mathrm{Pr}_n^\star$ is now expressed as follows.

Theorem 3.8

$$\bigcap\{\boldsymbol{I} \mid \boldsymbol{I} \subseteq \boldsymbol{HB}_\Phi,\, \boldsymbol{I} \models \mathcal{P}\} \;=\; \mathit{lfp}(\boldsymbol{T}_\mathcal{P}) \;=\; \boldsymbol{T}_\mathcal{P}\!\uparrow\omega\,.$$

Thus, tight answer substitutions for object-ground many-valued queries can be characterized as follows.

Theorem 3.9 *Let $\mathcal{P}$ be an n-valued logic program and let $\exists(A_1, \ldots, A_l)[x, 1]$ be an object-ground n-valued query with $x \in \mathcal{X}$. The tight answer substitution in $\mathrm{Pr}_n^\star$ for $\exists(A_1, \ldots, A_l)[x, 1]$ to $\mathcal{P}$ is given by $\{x/c\}$, where c is the minimum of all $\boldsymbol{T}_\mathcal{P}\!\uparrow\omega[A_i]$ with $i \in [1\!:\!l]$.*

3.5. Proof theory

We now present $\mathrm{SLDPr}_n^\star$-resolution for many-valued logic programs in $\mathrm{Pr}_n^\star$, which is an extension of the classical SLD-resolution (see, for example, [1]). In the sequel, many-valued facts $(A \leftarrow\ )[c, 1]$ are abbreviated by $(A)[c, 1]$.

A *subgoal list* is a finite list $(A_1)[a_1, 1] \ldots (A_m)[a_m, 1]$ of n-valued facts $(A_1)[a_1, 1], \ldots, (A_m)[a_m, 1]$ such that $a_1, \ldots, a_m > 0$ and $m \geq 0$. A substitution θ is applied to a subgoal list by replacing each contained atom A_i by $A_i\theta$. For n-valued program clauses P_1 and P_2, we say P_1 is a *variant* of P_2 iff P_1 is an instance of P_2 and P_2 is an instance of P_1. The notions of unifiers and most general unifiers (mgu) are defined as usual.

The subgoal list $(\alpha\,(B_1)[b, 1] \ldots (B_k)[b, 1]\omega)\theta$ is a *resolvent* of the subgoal list $\alpha\,(A)[a, 1]\omega$ and the n-valued program clause $(H \leftarrow B_1, \ldots, B_k)[c, 1]$ *with* mgu θ iff A and H unify with mgu θ, $a \leq c$, and $b = a - c + 1$.

Note that, for subgoal lists $\alpha\,(A)[a, 1]\omega$ and n-valued program clauses $(H \leftarrow B_1, \ldots, B_k)[c, 1]$, the resolvent $(\alpha\,(B_1)[b, 1] \ldots (B_k)[b, 1]\omega)\theta$ is a subgoal list, since $0 < a \leq c \leq 1$ and $b = a - c + 1$ entails $0 < b \leq 1$.

An $\mathrm{SLDPr}_n^\star$*-derivation* of a subgoal list R_0 from an n-valued logic program $\mathcal{P}$ is a maximal sequence R_0, $(C_0, \theta_0), R_1, (C_1, \theta_1), \ldots$, where $R_0, R_1, \ldots$ is a sequence of subgoal lists, $C_0, C_1, \ldots$ is a sequence of variants of clauses from $\mathcal{P}$, and $\theta_0, \theta_1, \ldots$ is a sequence of substitutions such that R_{i+1} is a resolvent of R_i and C_i with mgu θ_i and such that C_i does not have any variables in common with $R_0, C_0, \ldots, R_{i-1}$. If a subgoal list R_j is empty, then it is the last one in a derivation. Such an $\mathrm{SLDPr}_n^\star$-derivation is called *successful*.

The presented $\mathrm{SLDPr}_n^\star$-resolution is a sound and complete technique for correct query answering in $\mathrm{Pr}_n^\star$. That is, for n-valued logic programs $\mathcal{P}$ and n-valued queries $Q_c = \exists(A_1, \ldots, A_l)[c,1]$ with $c > 0$, the correct answer in $\mathrm{Pr}_n^\star$ for Q_c to $\mathcal{P}$ is Yes iff a successful $\mathrm{SLDPr}_n^\star$-derivation of $(A_1)[c, 1] \ldots (A_l)[c, 1]$ from $\mathcal{P}$ exists. Moreover, each successful $\mathrm{SLDPr}_n^\star$-derivation of $(A_1)[c, 1] \ldots (A_l)[c, 1]$ from

$\mathcal{P}$ with the sequence of substitutions $\theta_0, \theta_1, \ldots, \theta_j$ provides a correct answer substitution in $\mathrm{Pr}_n^\star$ for Q_c to $\mathcal{P}$ by the substitution $\theta_0\theta_1 \ldots \theta_j$ restricted to the variables in Q_c.

More precisely, the soundness and the completeness of $\mathrm{SLDPr}_n^\star$-resolution is expressed as follows.

Theorem 3.10 *a) Let $\mathcal{P}$ be an n-valued logic program and $Q_c = \exists(A_1, \ldots, A_l)[c, 1]$ be an n-valued query with $c > 0$. If there exists a successful $\mathrm{SLDPr}_n^\star$-derivation of $(A_1)[c, 1] \ldots (A_l)[c, 1]$ from $\mathcal{P}$ with the sequence of substitutions $\theta_0, \theta_1, \ldots, \theta_j$, then the substitution $\theta_0\theta_1 \ldots \theta_j$ restricted to the variables in Q_c is a correct answer substitution in $\mathrm{Pr}_n^\star$ for Q_c to $\mathcal{P}$. b) Let $\mathcal{P}$ be an n-valued logic program and $Q_c = \exists(A_1, \ldots, A_l)[c,1]$ be an n-valued query with $c > 0$. If* Yes *is the correct answer in $\mathrm{Pr}_n^\star$ for Q_c to $\mathcal{P}$, then a successful $\mathrm{SLDPr}_n^\star$-derivation of $(A_1)[c, 1] \ldots (A_l)[c, 1]$ from $\mathcal{P}$ exists.*

3.6. Computational complexity

We now focus on the computational complexity of many-valued logic programming in $\mathrm{Pr}_n^\star$. Like in Section 2.3, we concentrate on the two decidable special cases that generalize propositional logic programming and the decision problem that defines the data complexity of datalog. Crucially, in contrast to the probabilistic many-valued generalizations, the truth-functional ones are P-complete.

Theorem 3.11 *a) The optimization problem of computing the tight answer substitution in $\mathrm{Pr}_n^\star$ for an object-ground n-valued query $\exists(A_1, \ldots, A_l)[x, 1]$, with $x \in \mathcal{X}$, to a ground n-valued logic program $\mathcal{P}$ is P-complete. b) Let Φ be function-free. Let $\mathcal{P}$ be a fixed n-valued logic program, let $\mathcal{F}$ be a varying finite set of ground n-valued facts. Let $\mathcal{P} \cup \mathcal{F}$ contain all constant symbols from Φ. The optimization problem of computing the tight answer substitution in $\mathrm{Pr}_n^\star$ for an object-ground n-valued query $\exists(A_1, \ldots, A_l)[x, 1]$, with $x \in \mathcal{X}$, to $\mathcal{P} \cup \mathcal{F}$ is P-complete.*

4. Summary and conclusion

We introduced probabilistic many-valued logic programs in which the implication connective is interpreted as material implication. We showed that probabilistic many-valued logic programming is computationally more complex than classical logic programming. We then focused on the approximation of probabilistic many-valued logic programming by many-valued logic programming in $\mathrm{Pr}_n^\star$. In particular, we introduced a sound and complete proof theory for many-valued logic programming in $\mathrm{Pr}_n^\star$.

Crucially, many-valued logic programs in $\mathrm{Pr}_n^\star$ have both a probabilistic semantics in probabilities over a set of possible worlds and a truth-functional semantics in the finite-valued Łukasiewicz logics $Ł_n$. Furthermore, many-valued logic programming in $\mathrm{Pr}_n^\star$ has a model and fixpoint characterization, a proof theory, and computational properties that are very similar to those of classical logic programming. Hence, it is well worth being studied more deeply.

Finally, this paper showed how presumably intractable probabilistic deduction problems in artificial intelligence can be tackled by efficient approximation techniques based on truth-functional many-valued logics.

References

[1] 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.

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

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

[4] J. P. Delahaye and V. Thibau. Programming in three-valued logic. *Theor. Comput. Sci.*, 78:189–216, 1991.

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

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

[7] R. Hähnle and G. Escalada-Imaz. Deduction in many-valued logics: a survey. *Mathware Soft Comput.*, IV(2):69–97, 1997.

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

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

[10] J.-L. Lassez and M. J. Maher. Optimal fixedpoints of logic programs. *Theor. Comput. Sci.*, 39:15–25, 1985.

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

[12] T. Lukasiewicz. Many-valued first-order logics with probabilistic semantics. In *Proc. of the Annual Conference of the European Association for Computer Science Logic*, 1998.

[13] T. Lukasiewicz. Probabilistic and truth-functional many-valued logic programming. Technical Report 9809, Institut für Informatik, Universität Gießen, 1998.

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

[15] T. Lukasiewicz. Local probabilistic deduction from taxonomic and probabilistic knowledge-bases over conjunctive events. *Int. J. Approx. Reasoning*, 1999. To appear.

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

[17] 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.

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

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