LP^{MLN}, Weak Constraints, and P-log

Joohyung Lee and Zhun Yang
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, USA

Abstract

LP^{MLN} is a recently introduced formalism that extends answer set programs by adopting the log-linear weight scheme of Markov Logic. This paper investigates the relationships between LP^{MLN} and two other extensions of answer set programs: weak constraints to express quantitative preference among answer sets, and P-log to incorporate probabilistic uncertainty. We present a translation of LP^{MLN} into programs with weak constraints, and a translation of P-log into LP^{MLN}, which complement the existing translations in the opposite directions. The first translation allows us to compute the most probable stable models (i.e., MAP estimates) of LP^{MLN} programs using standard ASP solvers, the result that can be extended to other formalisms, such as Markov Logic, ProbLog, and Pearl's Causal Models, that are shown to be translatable to LP^{MLN}. The second translation tells us how probabilistic nonmonotonicity (the ability of the reasoner to change his probabilistic model as a result of new information) of P-log can be represented in LP^{MLN}, which yields a way to compute P-log using standard answer set solvers and MLN solvers.

Introduction

LP^{MLN} (Lee and Wang 2016) is a recently introduced probabilistic logic programming language that extends answer set programs (Gelfond and Lifschitz 1988) with the concept of weighed rules, whose weight scheme is adopted from that of Markov Logic (Richardson and Domingos 2006). It is shown in (Lee and Wang 2016; Lee, Meng, and Wang 2015) that LP^{MLN} is expressive enough to embed Markov Logic and several other probabilistic logic languages, such as ProbLog (De Raedt, Kimmig, and Toivonen 2007), Pearls' Causal Models (Pearl 2000), and a fragment of P-log (Baral, Gelfond, and Rushton 2009).

Among several extensions of answer set programs to overcome the deterministic nature of the stable model semantics, LP^{MLN} is one of the few languages that incorporate the concept of weights as a first-class citizen. A closest one is weak constraints (Buccafurri, Leone, and Rullo 2000) although their main role is to assign quantitative preference over the stable models of non-weak constraint rules: weak constraints cannot be used for deriving stable models. It is relatively a simple extension of the stable model semantics, but has turned out to be useful in many practical applications. Weak constraints are part of the ASP Core 2 language (Calimeri et al. 2013), and are implemented in standard ASP solvers, such as CLINGO and DLV.

P-log is a probabilistic extension of answer set programs. In contrast to weak constraints, it is highly structured, and has quite a sophisticated semantics. One of its distinct features is *probabilistic nonmonotonicity* (the ability of the reasoner to change his probabilistic model as a result of new information) whereas in most other probabilistic logic languages, new information can only cause the reasoner to abandon some of his possible worlds, making the effect of an update *monotonic*.

This paper reveals interesting relationships between LP^{MLN} and these two extensions of answer set programs. It shows how different weight schemes of LP^{MLN} and weak constraints are related, and how the probabilistic reasoning in P-log can be expressed in LP^{MLN}. The work helps us understand these languages better as well as other related languages, and also provides new, effective computational methods based on the translations.

It is shown in (Lee and Wang 2016) that programs with weak constraints can be easily viewed as a special case of LP^{MLN} programs. In the first part of this paper, we show that an inverse translation is also possible under certain conditions, i.e., an LP^{MLN} program can be turned into a usual ASP program with weak constraints so that the most probable stable models of the LP^{MLN} program are exactly the (optimal) stable models of the program with weak constraints. This allows for using ASP solvers for computing MAP estimates of LP^{MLN} programs. Interestingly, the translation is quite simple, so it can be easily applied in practice. Further, the result implies that MAP inference in other probabilistic logic languages, such as Markov Logic, ProbLog, and Pearl's Causal Models, can be computed by standard ASP solvers because they are known to be embeddable in LP^{MLN}, thereby allowing us to apply combinatorial optimization in standard ASP solvers to MAP inference in those languages.

In the second part of the paper, we show how P-log can be completely characterized in LP^{MLN}. Unlike the translation in (Lee and Wang 2016), which was limited to a subset of P-log that does not allow dynamic default probability, our translation applies to full P-log, and complements the translation from LP^{MLN} into P-log in (Balai and Gelfond 2016). In combination with the embedding of LP^{MLN} in answer set programs with weak constraints, our work shows how MAP estimates of P-log can be computed by standard ASP solvers, which provides a highly efficient way to compute P-log.

Preliminaries

Review: $\mathrm{LP}^{\mathrm{MLN}}$

We review the definition of $\mathrm{LP}^{\mathrm{MLN}}$ from (Lee and Wang 2016). In fact, we consider a more general syntax of programs than the one from (Lee and Wang 2016), but this is not an essential extension. We follow the view of (Ferraris, Lee, and Lifschitz 2011) by identifying typical logic program rules as a special class of first-order formulas under the stable model semantics. For example, rule $r(x) \leftarrow p(x), \mathit{not}\ q(x)$ is identified with formula $\forall x(p(x) \wedge \neg q(x) \rightarrow r(x))$. An $\mathrm{LP}^{\mathrm{MLN}}$ program is a finite set of weighted first-order formulas $w : F$ where w is a real number or α for denoting the infinite weight. An $\mathrm{LP}^{\mathrm{MLN}}$ program is called *ground* if its formulas contain no variables. Any $\mathrm{LP}^{\mathrm{MLN}}$ program can be turned into a ground program by replacing the quantifiers with multiple conjunctions and disjunctions over the Herbrand Universe. Each of the ground instances of a formula with free variables receives the same weight as the original formula.

For any ground $\mathrm{LP}^{\mathrm{MLN}}$ program Π and any interpretation I, $\overline{\Pi}$ denotes the unweighted formula obtained from Π, and Π_I denotes the set of $w : F$ in Π such that $I \models F$, and $\mathrm{SM}[\Pi]$ denotes the set $\{I \mid I \text{ is a stable model of } \overline{\Pi_I}\}$ (We refer to the stable model semantics of first-order formulas in (Ferraris, Lee, and Lifschitz 2011)). A member of $\mathrm{SM}[\Pi]$ is called a *soft stable model* of Π. The *unnormalized weight* of an interpretation I under Π is defined as

$$W_\Pi(I) = \begin{cases} exp\left(\sum_{w:F \in \Pi_I} w\right) & \text{if } I \in \mathrm{SM}[\Pi]; \\ 0 & \text{otherwise.} \end{cases}$$

The *normalized weight* (a.k.a. *probability*) of an interpretation I under Π is defined as

$$P_\Pi(I) = \lim_{\alpha \to \infty} \frac{W_\Pi(I)}{\sum_{J \in \mathrm{SM}[\Pi]} W_\Pi(J)}.$$

I is called a *(probabilistic) stable model* of Π if $P_\Pi(I) \neq 0$.

Review: Weak Constraints

A *weak constraint* has the form

$$:\sim F \quad [\mathit{Weight}]. \tag{1}$$

where F is a ground formula, and *Weight* is a real number. Note that the syntax is more general than the one from the literature (Buccafurri, Leone, and Rullo 2000), where F was restricted to conjunctions of literals.[1] We will see the generalization is more convenient for stating our result, but will also present translations that conform to the restrictions imposed in the input language of ASP solvers.

The stable models of a program $\Pi_1 \cup \Pi_2$, where Π_1 is a set of ground formulas and Π_2 is a set of weak constraints, are the stable models of Π_1 with the minimum *penalty*, where the penalty of a stable model is defined as the sum of the weights of all weak constraints (1) in Π_2 whose bodies are satisfied by the stable model.

[1] A literal is either an atom p or its negation *not* p.

Turning $\mathrm{LP}^{\mathrm{MLN}}$ into Programs with Weak Constraints

General Translation

We define a translation that turns an $\mathrm{LP}^{\mathrm{MLN}}$ program into a program with weak constraints. For any ground $\mathrm{LP}^{\mathrm{MLN}}$ program Π, the translation $\mathrm{lpmln2wc}(\Pi)$ is simply defined as follows. We turn each hard formula $\alpha : F$ in Π into F and turn each soft formula $w : F$ (where $w \neq \alpha$) into

$$\begin{array}{c} \{F\}^{\mathrm{ch}} \\ :\sim F \ \ [-w]. \end{array} \tag{2}$$

where $\{F\}^{\mathrm{ch}}$ is a choice formula, standing for $F \vee \neg F$ (Ferraris, Lee, and Lifschitz 2011).[2] Traditionally, choice formulas are restricted to the special case when F is an atom, and this is the case that is implemented in standard ASP solvers. Intuitively, choice formula (2) allows F to be either included or not in deriving a stable model. When F is included, the stable model gets the (negative) penalty $-w$, which corresponds to the (positive) "reward" e^w that it receives under the $\mathrm{LP}^{\mathrm{MLN}}$ semantics.

Theorem 1 *For any $\mathrm{LP}^{\mathrm{MLN}}$ program Π that has a soft stable model that satisfies all hard formulas in Π, the most probable stable models (i.e., the stable models with the highest probability) of Π are precisely the stable models of the program with weak constraints $\mathrm{lpmln2wc}(\Pi)$.*

Example 1 *For program*

$$\begin{array}{rl|rl} 10: & p \rightarrow q & 5: & p \\ 1: & p \rightarrow r & -20: & \neg r \rightarrow \bot \end{array} \tag{3}$$

The translation yields

$$\begin{array}{rcll} \{p \rightarrow q\}^{\mathrm{ch}} & :\sim & p \rightarrow q & [-10] \\ \{p \rightarrow r\}^{\mathrm{ch}} & :\sim & p \rightarrow r & [-1] \\ \{p\}^{\mathrm{ch}} & :\sim & p & [-5] \\ \{\neg r \rightarrow \bot\}^{\mathrm{ch}} & :\sim & \neg r \rightarrow \bot & [20] \end{array}$$

whose stable model is $\{p, q\}$ with the penalty -15.

The condition in Theorem 1 that Π has a soft stable model that satisfies all hard formulas in Π is required because $\mathrm{LP}^{\mathrm{MLN}}$ program Π in general may have stable models that do not satisfy some hard formulas, in which case the corresponding $\mathrm{lpmln2wc}(\Pi)$ has no stable models.

Also note that while the most probable stable models of Π and the stable models of $\mathrm{lpmln2wc}(\Pi)$ coincide, their weight vs. penalty are not proportional to each other. The former is defined by an exponential function whose exponent is the sum of the weights of the satisfied formulas, while the latter simply adds up the penalties of the satisfied formulas. On the other hand, both are monotonically increasing/decreasing as more formulas are satisfied.

In view of Theorem 2 from (Lee and Wang 2016), which tells us how to embed Markov Logic into $\mathrm{LP}^{\mathrm{MLN}}$, it follows from Theorem 1 that MAP inference in MLN can be also reduced to stable model finding of programs with weak constraints. For any Markov Logic Network Π, let $\mathrm{mln2wc}(\Pi)$ be the union of $\mathrm{lpmln2wc}(\Pi)$ plus choice rules $\{A\}^{\mathrm{ch}}$ for all atoms A.

[2] This view of choice formulas was already used in PrASP (Nickles and Mileo 2014) in defining *spanning* programs.

Theorem 2 *For any Markov Logic Network N that has a model that satisfies all hard formulas in N, the most probable models of N are precisely the most probable stable models of the program with weak constraints* $\mathrm{mln2wc}(N)$.

Similarly, MAP inference in ProbLog and Pearl's Causal Models can be reduced to stable model finding of programs with weak constraints in view of the reduction of ProbLog to $\mathrm{LP}^{\mathrm{MLN}}$ (Theorem 4 from (Lee and Wang 2016) and the reduction of Causal Models to $\mathrm{LP}^{\mathrm{MLN}}$ (Theorem 4 from (Lee, Meng, and Wang 2015)) thereby allowing us to apply combinatorial optimization methods in standard ASP solvers to these languages.

Alternative Translations

Instead of assigning a "reward" to each stable model in $\mathrm{lpmln2wc}(\Pi)$, we may alternatively assign a "penalty". Let $\mathrm{lpmln2wc}^{\mathrm{pnt}}(\Pi)$ be a modification of $\mathrm{lpmln2wc}(\Pi)$ by turning each soft formula $w : F$ into

$$\begin{array}{rl} & \{F\}^{\mathrm{ch}} \\ :\sim & \neg F \ \ [w]. \end{array}$$

Intuitively, when F is not included, the stable model gets the penalty w.

Corollary 1 *Theorem 1 remains true when* $\mathrm{lpmln2wc}(\Pi)$ *is replaced by* $\mathrm{lpmln2wc}^{\mathrm{pnt}}(\Pi)$.

This alternative view of assigning weights to stable models in fact originates from Probabilistic Soft Logic (PSL) (Bach et al. 2015), where the probability density function of an interpretation is obtained from the sum over the "penalty" from the formulas that are "distant" from satisfaction. This view will lead to a slight advantage when we further turn the translation into the input language of ASP solvers (See Footnote 5).

The current ASP solvers do not allow arbitrary formulas to appear in weak contraints. For computation, let $\mathrm{lpmln2wc}^{\mathrm{pnt,rule}}(\Pi)$ be the translation modified from $\mathrm{lpmln2wc}^{\mathrm{pnt}}$ by turning each soft formula

$$w_i : F_i \tag{4}$$

instead into

$$\begin{array}{rcll} \neg F_i & \to & \mathit{unsat}(i) & \\ \neg \mathit{unsat}(i) & \to & F_i & \\ & :\sim & \mathit{unsat}(i) & [w_i]. \end{array}$$

where $\mathit{unsat}(i)$ is a new atom.

Corollary 2 *Theorem 1 remains true when* $\mathrm{lpmln2wc}(\Pi)$ *in the statement is replaced by* $\mathrm{lpmln2wc}^{\mathrm{pnt,rule}}(\Pi)$.

The corollary allows us to compute the most probable stable models (MAP estimates) of the $\mathrm{LP}^{\mathrm{MLN}}$ program using the combination of F2LP [3] and CLINGO [4] (assuming that the weights are approximated to integers). System F2LP turns this program with formulas into the input language of CLINGO, so CLINGO can be used to compute the theory.

If the unweighted $\mathrm{LP}^{\mathrm{MLN}}$ program is already in the rule form *Head* $\leftarrow$ *Body* that is allowed in the input languages of CLINGO and DLV, we may avoid the use of F2LP by slightly modifying the translation $\mathrm{lpmln2wc}^{\mathrm{pnt,rule}}$ by turning each soft rule

$$w_i : \mathit{Head}_i \leftarrow \mathit{Body}_i \tag{5}$$

instead into

$$\begin{array}{rcl} \mathit{unsat}(i) & \leftarrow & \mathit{Body}_i, \mathit{not}\ \mathit{Head}_i \\ \mathit{Head}_i & \leftarrow & \mathit{Body}_i, \mathit{not}\ \mathit{unsat}(i) \\ & :\sim & \mathit{unsat}(i) \quad [w_i] \end{array}$$

In the case when Head_i is $\bot$, the translation can be further simplified. We turn $w_i : \bot \leftarrow \mathit{Body}_i$ into $:\sim \mathit{Body}_i \quad [w_i]$.[5]

Example 1 continued: *For program* (3)*, the simplified translation* $\mathrm{lpmln2wc}^{\mathrm{pnt,rule}}$ *yields*

$$\begin{array}{l|l|ll} \mathit{unsat}(1) \leftarrow p, \mathit{not}\ q & q \leftarrow p, \mathit{not}\ \mathit{unsat}(1) & :\sim \mathit{unsat}(1) & [10] \\ \mathit{unsat}(2) \leftarrow p, \mathit{not}\ r & r \leftarrow p, \mathit{not}\ \mathit{unsat}(2) & :\sim \mathit{unsat}(2) & [1] \\ \mathit{unsat}(3) \leftarrow \mathit{not}\ p & p \leftarrow \mathit{not}\ \mathit{unsat}(3) & :\sim \mathit{unsat}(3) & [5] \\ & & :\sim \mathit{not}\ r & [-20] \end{array}$$

Turning P-log into $\mathrm{LP}^{\mathrm{MLN}}$

Review: P-log

Syntax A *sort* is a set of symbols. A *constant* c maps an element in the *domain* $s_1 \times \cdots \times s_n$ to an element in the *range* s_0 (denoted by $\mathit{Range}(c)$), where each of $s_0, \ldots, s_n$ is a sort. A *sorted propositional signature* is constructed from a set of functions and their associated sorts, consisting of all *atoms* $c(\vec{u}) = v$ where $c : s_1 \times \cdots \times s_n \to s_0$, $\vec{u} \in s_1 \times \cdots \times s_n$, and $v \in s_0$. [6] $c(\vec{u})$ is called an *attribute* and v its *value*. If the range s_0 of c is $\{\mathbf{f}, \mathbf{t}\}$ then c is called *Boolean*, and $c(\vec{u}) = \mathbf{t}$ can be abbreviated as $c(\vec{u})$ and $c(\vec{u}) = \mathbf{f}$ as $\sim c(\vec{u})$.

The signature of P-log program is the union of two propositional signatures σ_1 and σ_2, where σ_1 is a sorted propositional signature, and σ_2 is a standard propositional signature consisting of atoms $\mathit{Do}(c(\vec{u}) = v)$, $\mathit{Obs}(c(\vec{u}) = v)$ and $\mathit{Obs}(c(\vec{u}) \neq v)$ for all atoms $c(\vec{u}) = v$ in σ_1.

A P-log program Π of signature $\sigma_1 \cup \sigma_2$ is a tuple

$$\Pi = \langle \mathbf{R}, \mathbf{S}, \mathbf{P}, \mathbf{Obs}, \mathbf{Act} \rangle \tag{6}$$

where the signature of each of $\mathbf{R}$, $\mathbf{S}$, and $\mathbf{P}$ is σ_1 and the signature of each of $\mathbf{Obs}$ and $\mathbf{Act}$ is σ_2 such that

- $\mathbf{R}$ is a set of *normal rules* of the form

  $$A \leftarrow B_1, \ldots, B_m, \mathit{not}\ B_{m+1}, \ldots, \mathit{not}\ B_n$$

  where $A, B_1, \ldots, B_n$ are atoms $(0 \leq m \leq n)$.
- $\mathbf{S}$ is a set of *random selection rules* of the form

  $$[r] \ \ \mathit{random}(c(\vec{u}) : \{x : p(x)\}) \leftarrow \mathit{Body} \tag{7}$$

  where r is a unique identifier, p is a boolean function with unary argument, and *Body* is a set of literals. x is a schematic variable ranging over the argument sort of p. Rule (7) is called a *random selection* rule for $c(\vec{u})$. Intuitively, rule (7) says that if *Body* is true, the value of $c(\vec{v})$ is selected at random from the set $\mathit{Range}(c) \cap \{x : p(x)\}$ unless this value is fixed by a deliberate action.

[3] http://reasoning.eas.asu.edu/f2lp/

[4] http://potassco.sourceforge.net

[5] Alternatively, we may turn it into the "reward" way, i.e., turning it into $:\sim \mathit{not}\ \mathit{Body}_i[-w_i]$, but the rule may not be in the input language of CLINGO.

[6] Note that here "=" is just a part of the symbol for propositional atoms, and is not equality in first-order logic.

- $\mathbf{P}$ is a set of *probability atoms (pr-atoms)* of the form
$$pr_r(c(\vec{u}) = v \mid C) = p \tag{8}$$
where r is the identifier of some random selection rule in $\mathbf{S}$, $c(\vec{u}) = v \in \sigma_1$, C is a set of literals, and $p \in [0, 1]$. We say pr-atom (8) is *associated* with the random selection rule whose identifier is r.
- $\mathbf{Obs}$ is a set of atomic facts for representing "observation": $Obs(c(\vec{u}) = v)$ and $Obs(c(\vec{u}) \neq v)$.
- $\mathbf{Act}$ is a set of atomic facts for representing deliberate action: $Do(c(\vec{u}) = v)$.

Semantics Let Π be a P-log program (6) of signature $\sigma_1 \cup \sigma_2$. The possible worlds of Π, denoted by $\omega(\Pi)$, are the stable models of $\tau(\Pi)$, a (standard) ASP program with the propositional signature

$\sigma_1 \cup \sigma_2 \cup \{\mathit{Intervene}(c(\vec{u})) \mid c(\vec{u})$ is an attribute occurring in $\mathbf{S}\}$.

Due to lack of space we refer the reader to (Baral, Gelfond, and Rushton 2009) for the definition of $\tau(\Pi)$. (Also reviewed in the supplemental material.)

An atom $c(\vec{u}) = v$ is called *possible* in a possible world W due to a random selection rule (7) if Π contains (7) such that $W \models \mathit{Body} \wedge p(v) \wedge \neg \mathit{Intervene}(c(\vec{u}))$. Pr-atom (8) is *applied* in W if $c(\vec{u}) = v$ is possible in W due to r and $W \models C$.

As in (Baral, Gelfond, and Rushton 2009), we assume that all P-log programs Π satisfy the following conditions:

- **Condition 1 [Unique random selection rule]:** If a P-log program Π contains two random selection rules for $c(\vec{u})$
$$[r_1]\ random(c(\vec{u}) : \{x : p_1(x)\}) \leftarrow \mathit{Body}_1,$$
$$[r_2]\ random(c(\vec{u}) : \{x : p_2(x)\}) \leftarrow \mathit{Body}_2,$$
then no possible world of Π satisfies both Body_1 and Body_2.
- **Condition 2 [Unique probability assignment]:** If a P-log program Π contains a random selection rule for $c(\vec{u})$
$$[r]\ random(c(\vec{u}) : \{x : p(x)\}) \leftarrow \mathit{Body}$$
along with two different pr-atoms :
$$pr_r(c(\vec{u}) = v \mid C_1) = p_1,$$
$$pr_r(c(\vec{u}) = v \mid C_2) = p_2,$$
then no possible world of Π satisfies *Body*, C_1, and C_2 together.

Given a P-log program Π, a possible world $W \in \omega(\Pi)$, and an atom $c(\vec{u}) = v$ possible in W, by **Condition 1**, it follows that there is exactly one random selection rule (7) such that $W \models \mathit{Body}$. Let $r_{W,c(\vec{u})}$ denote this random selection rule, and let $AV_W(c(\vec{u})) = \{v' \mid$ there exists a pr-atom $pr_{r_{W,c(\vec{u})}}(c(\vec{u}) = v' \mid C) = p$ that is applied in W for some C and $p\}$. We then define the following notations:

- If $v \in AV_W(c(\vec{u}))$, there exists a pr-atom $pr_{r_{W,c(\vec{u})}}(c(\vec{u}) = v \mid C) = p \in \Pi$ for some C and p such that $W \models C$. By **Condition 2**, for any other $pr_{r_{W,c(\vec{u})}}(c(\vec{u}) = v \mid C') = p' \in \Pi$, it follows that $W \not\models C'$. So there is only one pr-atom that is applied for $c(\vec{u}) = v$, and we define
$$\mathit{PossWithAssPr}(W, c(\vec{u}) = v) = p.$$
("$c(\vec{u}) = v$ is possible in W with assigned probability p.")
- If $v \notin AV_W(c(\vec{u}))$, we define
$$\mathit{PossWithDefPr}(W, c(\vec{u}) = v) = \frac{1 - \sum_{v' \in AV_W(c(\vec{u}))} \mathit{PossWithAssPr}(W, c(\vec{u}) = v')}{|\{v'' \mid c(\vec{u}) = v'' \text{ is possible in } W \text{ and } v'' \notin AV_W(c(\vec{u}))\}|}. \tag{9}$$
("$c(\vec{u}) = v$ is possible in W with the default probability")

For each possible world $W \in \omega(\Pi)$, and each atom $c(\vec{u}) = v$ possible in W, the probability of $c(\vec{u}) = v$ to *happen* in W is defined as:
$$P(W, c(\vec{u}) = v) = \begin{cases} \mathit{PossWithAssPr}(W, c(\vec{u}) = v) & \text{if } v \in AV_W(c(\vec{u})) \\ \mathit{PossWithDefPr}(W, c(\vec{u}) = v) & \text{otherwise.} \end{cases}$$

The *unnormalized probability* of a possible world W is defined as
$$\hat{\mu}_\Pi(W) = \prod_{\substack{c(\vec{u})=v \in W \text{ and} \\ c(\vec{u})=v \text{ is possible in } W}} P(W, c(\vec{u}) = v).$$

Assuming Π has at least one possible world with nonzero unnormalized probability, the *normalized probability* of W is defined as
$$\mu_\Pi(W) = \frac{\hat{\mu}_\Pi(W)}{\sum_{W_i \in \omega(\Pi)} \hat{\mu}_\Pi(W_i)}.$$

We say Π is *consistent* if Π has at least one possible world with non-zero probability.

Example 2 *Consider a variant of the Monty Hall Problem encoding in P-log from (Baral, Gelfond, and Rushton 2009) to illustrate the probabilistic nonmonotonicity in the presence of assigned probabilities. There are 4 doors, behind which are three goats and one car. The guest picks door 1, and Monty, the show host who always opens one of the doors with a goat, opens door 2. Further, while the guest and Monty are unaware, the statistics is that in the past, with 30% chance the prize was behind door 1, and with 20% chance the prize was behind door 3. Is it still better to switch to another door? This example can be formalized in P-log program* Π*, using both assigned probability and default probability, as*

$\sim \mathit{CanOpen}(d) \leftarrow \mathit{Selected} = d. \quad (d \in \{1, 2, 3, 4\})$,
$\sim \mathit{CanOpen}(d) \leftarrow \mathit{Prize} = d.$
$\mathit{CanOpen}(d) \leftarrow \mathit{not}\ \sim \mathit{CanOpen}(d).$
$random(\mathit{Prize}). \quad random(\mathit{Selected}).$
$random(\mathit{Open} : \{x : \mathit{CanOpen}(x)\}).$
$pr(\mathit{Prize} = 1) = 0.3.\ pr(\mathit{Prize} = 3) = 0.2.$
$Obs(\mathit{Selected} = 1).\ Obs(\mathit{Open} = 2).\ Obs(\mathit{Prize} \neq 2).$

There are three possible worlds:

- $W_1 = \{\mathit{Selected} = 1, \mathit{Open} = 2, \mathit{Prize} = 1, \dots\}$
- $W_2 = \{\mathit{Selected} = 1, \mathit{Open} = 2, \mathit{Prize} = 3, \dots\}$
- $W_3 = \{\mathit{Selected} = 1, \mathit{Open} = 2, \mathit{Prize} = 4, \dots\}$.

The probability of each atom to happen is

$P(W_i, \mathit{Selected} = 1) = \mathit{PossWithDefPr}(W, \mathit{Selected} = 1) = 1/4$

$P(W_1, \mathit{Open} = 2) = \mathit{PossWithDefPr}(W_1, \mathit{Open} = 2) = 1/3$
$P(W_2, \mathit{Open} = 2) = \mathit{PossWithDefPr}(W_2, \mathit{Open} = 2) = 1/2$
$P(W_3, \mathit{Open} = 2) = \mathit{PossWithDefPr}(W_3, \mathit{Open} = 2) = 1/2$

$P(W_1, \mathit{Prize} = 1) = \mathit{PossWithAssPr}(W_1, \mathit{Prize} = 1) = 0.3$
$P(W_2, \mathit{Prize} = 3) = \mathit{PossWithAssPr}(W_2, \mathit{Prize} = 3) = 0.2$
$P(W_3, \mathit{Prize} = 4) = \mathit{PossWithDefPr}(W_3, \mathit{Prize} = 4) = 0.25$

So,

- $\hat{\mu}_\Pi(W_1) = 1/4 \times 1/3 \times 0.3 = 1/40$
- $\hat{\mu}_\Pi(W_2) = 1/4 \times 1/2 \times 0.2 = 1/40$
- $\hat{\mu}_\Pi(W_3) = 1/4 \times 1/2 \times 0.25 = 1/32$.

Thus, in comparison with staying (W_1), switching to door 3 (W_2) does not affect the chance, but switching to door 4 (W_3) increase the chance by about 25%.

Turning P-log into $\mathrm{LP}^{\mathrm{MLN}}$

We define translation $\mathrm{plog2lpmln}(\Pi)$ that turns a P-log program Π into an $\mathrm{LP}^{\mathrm{MLN}}$ program in a modular way. First, every rule R in $\tau(\Pi)$ (that is used in defining the semantics of P-log) is turned into a hard rule $\alpha : R$ in $\mathrm{plog2lpmln}(\Pi)$. Further, $\mathrm{plog2lpmln}(\Pi)$ contains the following rules. Below x, y denote schematic variables.

- For each random selection rule for $c(\vec{u})$ in **S**:

$$[r]\ random(c(\vec{u}) : \{x : p(x)\}) \leftarrow \mathit{Body}$$

and for each $v \in \mathit{Range}(c)$, $\mathrm{plog2lpmln}(\Pi)$ includes

$$\alpha :\ \mathit{PossWithDefPr}(c(\vec{u}){=}v) \leftarrow \mathit{Body}, p(v), \mathit{not\ Intervene}(c(\vec{u})), \mathit{not\ PossWithAssPr}(c(\vec{u}){=}v). \tag{10}$$

Rule (10) asserts that $c(\vec{u}) = v$ is possible with a default probability if the randomness of $c(\vec{u})$ is not intervened, and no pr-atom is applied to assign a probability.

Further, $\mathrm{plog2lpmln}(\Pi)$ contains the following rules: [7]

$$\alpha :\ \mathit{NumDefPr}(c(\vec{u}), x) \leftarrow c(\vec{u}){=}v, \mathit{PossWithDefPr}(c(\vec{u}) = v), x = \#count\{y : \mathit{PossWithDefPr}(c(\vec{u}){=}y)\} \quad (v \in \mathit{Range}(c)) \tag{11}$$

$$ln(\tfrac{1}{k}) :\quad \bot \leftarrow \mathit{not\ NumDefPr}(c(\vec{u}), k) \quad (k = 2, \ldots, |\mathit{Range}(c)|) \tag{12}$$

Intuitively, $\mathit{NumDefPr}(c(\vec{u}), x)$ is true if there are exactly x different values v such that $c(\vec{u}){=}v$ is possible in W with a default probability, and there is at least one of them that is also true in W. This value x is the denominator of (9). A stable model I that satisfies $\mathit{NumDefPr}(c(\vec{u}), m)$ receives $1/m$ as a factor of $W_{\mathrm{plog2lpmln}(\Pi)}(I)$ due to rule (12).

- For each pr-atom

$$pr_r(c(\vec{u}){=}v \mid C) = p$$

in **P**, $\mathrm{plog2lpmln}(\Pi)$ contains the following rules:

$$\alpha :\ \mathit{PrAtomApplied}_{r,C}(c(\vec{u}){=}v) \leftarrow \mathit{Body}, p(v), \mathit{not\ Intervene}(c(\vec{u})), C \tag{13}$$

where *Body* and $p(\cdot)$ is from the random selection rule (7) associated with the pr-atom;

$$\alpha :\ \mathit{PossWithAssPr}(c(\vec{u}){=}v) \leftarrow \mathit{PrAtomApplied}_{r,C}(c(\vec{u}){=}v) \tag{14}$$

$$\alpha :\ \mathit{AssPr}_{r,C}(c(\vec{u}){=}v) \leftarrow \mathit{PrAtomApplied}_{r,C}(c(\vec{u}){=}v), c(\vec{u}){=}v \tag{15}$$

$$ln(p) :\ \bot \leftarrow \mathit{not\ AssPr}_{r,C}(c(\vec{u}){=}v) \quad (p > 0) \tag{16}$$

$$\alpha :\ \bot \leftarrow \mathit{AssPr}_{r,C}(c(\vec{u}){=}v) \quad (p = 0)$$

[7] Count aggregates can be represented by first-order formulas under the stable model semantics (Lee, Lifschitz, and Palla 2008).

Rule (13) asserts that $\mathit{PrAtomApplied}_{r,C}(c(\vec{u}){=}v)$ is true when the pr-atom $pr_r(c(\vec{u}){=}v \mid C) = p$ is applied. Rule (14) asserts that if this pr-atom is applied, the event $c(\vec{u}){=}v$ is possible with an assigned probability, and rule (15) assigns a probability to this event if $c(\vec{u}){=}v$ is also true in W. Rule (16) asserts that this probability is the one that is specified in the pr-atom.

- Consider each random selection rule (17) in **S**:

$$[r]\ random(c(\vec{u}) : \{x : p(x)\}) \leftarrow \mathit{Body} \tag{17}$$

along with all pr-atoms associated with it in **P**:

$$\begin{aligned} &pr_r(c(\vec{u}){=}v_1 \mid C_1) = p_1 \\ &\ldots \\ &pr_r(c(\vec{u}){=}v_n \mid C_n) = p_n \end{aligned} \tag{18}$$

where $n \geq 1$, and v_i and v_j $(i \neq j)$ may be equal. For every $v \in \mathit{Range}(c)$ and every $b_i \in \{\mathbf{t}, \mathbf{f}\}$ where $1 \leq i \leq n$, $\mathrm{plog2lpmln}(\Pi)$ contains the following rules:

$$\alpha :\ \mathit{UnAssPr}_r(c(\vec{u}), b_1, ..., b_n) \leftarrow \mathit{Body}, c(\vec{u}){=}v, \mathit{PossWithDefPr}(c(\vec{u}){=}v), \left[\mathit{PrAtomApplied}_{r,C_1}(c(\vec{u}){=}v_1)\right]^{b_1}, \ldots, \left[\mathit{PrAtomApplied}_{r,C_n}(c(\vec{u}){=}v_n)\right]^{b_n} \tag{19}$$

where $\left[\mathit{PrAtomApplied}_{r,C_i}(c(\vec{u}){=}v_i)\right]^{b_i}$ is $\mathit{PrAtomApplied}_{r,C_i}(c(\vec{u}){=}v_i)$ if $b_i = \mathbf{t}$ and $\mathit{not\ PrAtomApplied}_{r,C_i}(c(\vec{u}){=}v_i)$ if $b_i = \mathbf{f}$.

For every $b_i \in \{\mathbf{t}, \mathbf{f}\}$, $\mathrm{plog2lpmln}(\Pi)$ contains

–

$$ln(1 - \sum_{i \in \{1,\ldots,n\}} [p_i]^{b_i}) :\ \bot \leftarrow \mathit{not\ UnAssPr}_r(c(\vec{u}), b_1, \ldots, b_n) \tag{20}$$

where $[p_i]^{b_i} = \begin{cases} p_i & \text{if } b_i = \mathbf{t} \\ 0 & \text{otherwise} \end{cases}$

if $\sum_{i \in \{1,\ldots,n\}} [p_i]^{b_i} < 1$, and

–

$$\alpha : \bot \leftarrow \mathit{UnAssPr}_r(c(\vec{u}), b_1, \ldots b_n)$$

otherwise.

Rules (19) and (20) are to account for the numerator of (9). Rule (19) is used to record which pr-atoms are not applied in W when there is an atom $c(\vec{u}){=}v$ that is true in W and is possible with a default probability. For instance, $\mathit{UnAssPr}_r(c(\vec{u}), \mathbf{t}, \mathbf{f}, \mathbf{t})$ records the bit-vector that only the first and third pr-atoms are applied. Rule (20) records each probability in the numerator of (9) in accordance with the list of pr-atoms that are applied.

Note that the number of rules in (19) and (20) is exponential to the number of associated pr-atoms, while all other rules in the translation $\mathrm{plog2lpmln}(\Pi)$ are linear to the size of the input.

Certain conditions can be imposed to avoid the exponential blowup. For instance, if $C_1, \ldots, C_n$ in (18) is known to be true or false, we can only consider the corresponding bit-vector only. For instance, for the Monty Hall variant above, we can only consider the case when both pr-atoms are applied.

Also, note that most rules in $\mathrm{plog2lpmln}(\Pi)$ are hard rules. The soft rules (12), (16), (20) cannot be simplified as atomic

facts, e.g., $ln(\frac{1}{k})$: *NumDefPr*$(c(\vec{u}), k)$ in place of (12), which is in contrast with the use of probabilistic choice atoms in the distribution-based probabilistic logic programming language, such as ProbLog. This is related to the fact that the probability of each atom to happen in a possible word in P-log is derived from assigned and default probability, and not from independent probabilistic choices.

Example 2 Continued *Due to lack of space, we show only a part of the translation* $\mathrm{plog2lpmln}(\Pi)$.[8] *For attribute Open, rules* (10), (11), (12) *are the following* ($d \in \{1, 2, 3, 4\}$).

$$\begin{aligned}
\alpha: \quad & \textit{PossWithDefPr}(\textit{Open}=d) \leftarrow \\
& \qquad \textit{CanOpen}(d), \textit{not Intervene}(\textit{Open}) \\
& \qquad \textit{not PossWithAssPr}(\textit{Open}=d), \\
\alpha: \quad & \textit{NumDefPr}(\textit{Open}, x) \leftarrow \textit{Open}=y, \\
& \qquad \textit{PossWithDefPr}(\textit{Open}=y) \\
& \qquad x = \#count\{z : \textit{PossWithDefPr}(\textit{Open}=z)\} \\
ln(\tfrac{1}{d}): \quad & \bot \leftarrow \textit{not NumDefPr}(\textit{Open}, d) \qquad (d \geq 2)
\end{aligned}$$

For attribute Prize, rules (19), (20) *are the following:*

$$\begin{aligned}
& \alpha: \textit{UnAssPr}(\textit{Prize}, \mathbf{t}, \mathbf{t}) \leftarrow \textit{Prize}=x, \textit{PossWithDefPr}(\textit{Prize}=x), \\
& \qquad \textit{PrAtomApplied}(\textit{Prize}=1), \textit{PrAtomApplied}(\textit{Prize}=3). \\
& ln(0.5): \bot \leftarrow \textit{not UnAssPr}(\textit{Prize}, \mathbf{t}, \mathbf{t})
\end{aligned}$$

The following theorem tells us that there is a 1-1 correspondence between the possible worlds of Π and the stable models of $\mathrm{plog2lpmln}(\Pi)$, and the probability of each possible world Π coincides with the probability of the corresponding stable model of $\mathrm{plog2lpmln}(\Pi)$.

Theorem 3 *Let* Π *be a consistent P-log program such that all its possible worlds have non-zero probabilities. There is a 1-1 correspondence* ϕ *between the possible worlds of* Π *and the stable models of* $\mathrm{plog2lpmln}(\Pi)$ *such that*

- *For every possible world* W *of* Π, $\phi(W)$ *is a probabilistic stable model of* $\mathrm{plog2lpmln}(\Pi)$, *and* $\mu_\Pi(W) = P_{\mathrm{plog2lpmln}(\Pi)}(\phi(W))$.
- *For every probabilistic stable model* I *of* $\mathrm{plog2lpmln}(\Pi)$, *there exists a possible world* W *of* Π *such that* $I = \phi(W)$.

Proof. (Sketch) [9] We can check that the following mapping ϕ is the 1-1 correspondence.

1. $\phi(W) \models \textit{PossWithAssPr}(c(\vec{u})=v)$ iff $v \in AV_W(c(\vec{u}))$.
2. $\phi(W) \models \textit{PossWithDefPr}(c(\vec{u})=v)$ iff $c(\vec{u})=v$ is possible in W and $v \notin AV_W(c(\vec{u}))$.
3. $\phi(W) \models \textit{NumDefPr}(c(\vec{u}), n)$ iff there exist exactly n different values v such that $c(\vec{u}) = v$ is possible in W, $v \notin AV_W(c(\vec{u}))$, and, for one of such v, $W \models c(\vec{u})=v$.
4. For each pr-atom $pr_r(c(\vec{u})=v \mid C) = p$ in Π, $\phi(W) \models \textit{PrAtomApplied}_{r,C}(c(\vec{u}) = v)$ iff this pr-atom is applied in W.
5. For each pr-atom $pr_r(c(\vec{u}) = v \mid C) = p$ in Π, $\phi(W) \models \textit{AssPr}_{r,C}(c(\vec{u}) = v)$ iff this pr-atom is applied in W, and $W \models c(\vec{u})=v$.
6. For each $b_1, \ldots, b_n \in \{\mathbf{t}, \mathbf{f}\}$, $\phi(W) \models \textit{UnAssPr}_r(c(\vec{u}), b_1, \ldots, b_n)$ iff r is a random selection rule (17) for $c(\vec{u})$ whose *Body* is true in W, along with n pr-atoms (18) such that there is a v where $c(\vec{u})=v$ is possible in W, $v \notin AV_W(c(\vec{u}))$, $W \models c(\vec{u})=v$, and the i-th pr-atom is applied iff b_i is $\mathbf{t}$.

[8] The full translation is given in the supplemental material.

[9] Detailed proofs are available in the supplemental material.

To check that $\mu_\Pi(W) = P_{\mathrm{plog2lpmln}(\Pi)}(\phi(W))$, note first that the weight of $\phi(W)$ is computed by multiplying the weights of rules (12), (16), (20) that are satisfied by $\phi(W)$. Let's call this product TW.

Consider a possible world W of Π and $c(\vec{u}) = v$ that is satisfied by W. If $c(\vec{u}) = v$ is possible in W and pr-atom $pr_r(c(\vec{u} = v \mid C)) = p$ is applied in W (i.e., $v \in AV_W(c(\vec{u}))$), then the assigned probability is applied: $P(W, c(\vec{u}) = v) = p$. On the other hand, by condition 5, $\phi(W) \models \textit{AssPr}_{r,C}(c(\vec{u})=v)$, so that from (16), the same p is a factor of TW.

If $c(\vec{u}) = v$ is possible in W and $v \notin AV_W(c(\vec{u}))$, then the default probability is applied: $P(W, c(\vec{u}) = v) = p$ is computed by (9). By Condition 6, $\phi(W) \models \textit{UnAssPr}_r(c(\vec{u}, b_1, \ldots, b_n))$ where $b_1, \ldots, b_n$ is a bit-vector representing that i-th pr-atom is applied or not. Since $\phi(W) \models$ (20), $1 - \sum_{i \in \{1,\ldots,n\}} [p_i]^{b^i}$ is a factor of TW, which is the same as the numerator of (9). Furthermore, by Condition 3, $\phi(W) \models \textit{NumDefPr}(c(\vec{u}), m)$, where m is the denominator of (9). Since $\phi(W) \models$ (12), $\frac{1}{m}$ is a factor of TW.

Example 2 Continued *For the P-log program* Π *for the Monty Hall problem,* $\Pi' = \mathrm{plog2lpmln}(\Pi)$ *has three probabilistic stable models* I_1, I_2, *and* I_3, *each of which is an extension of* W_1, W_2, *and* W_3 *respectively, and satisfies the following atoms:* *PrAtomApplied*$(\textit{Prize} = i)$ *for* $i = 1, 3$; *PossWithAssPr*$(\textit{Prize} = i)$ *for* $i = 1, 3$ *PossWithDefPr*$(\textit{Prize} = i)$ *for* $i = 2, 4$; *PossWithDefPr*$(\textit{Selected} = i)$ *for* $i = 1, 2, 3, 4$; *PossWithDefPr*$(\textit{Open} = 2)$; *NumDefPr*$(\textit{Selected}, 4)$. *In addition,*

- $I_1 \models \{\textit{AssPr}(\textit{Prize} = 1), \textit{PossWithDefPr}(\textit{Open} = 3), \textit{PossWithDefPr}(\textit{Open}=4), \textit{NumDefPr}(\textit{Open}, 3)\}$
- $I_2 \models \{\textit{AssPr}(\textit{Prize} = 3), \textit{PossWithDefPr}(\textit{Open} = 4), \textit{NumDefPr}(\textit{Open}, 2)\}$
- $I_3 \models \{\textit{UnAssProb}(\textit{Prize}, \mathbf{t}, \mathbf{t}), \textit{NumDefPr}(\textit{Prize}, 2), \textit{PossWithDefPr}(\textit{Open=3}), \textit{NumDefPr}(\textit{Open}, 2)\}$.

The unnormalized weight $W_{\Pi'}(I_i)$ *of each probabilistic stable model* I_i *is shown below.* $w(\textit{NumDefPr}(c(\vec{u}), k))$ *denotes the weight of rule* (12), *and* $w(\textit{AssPr}(c(\vec{u}) = v))$ *denotes the weight of rule* (16). $W_{\Pi'}(I_1) = w(\textit{NumDefPr}(\textit{Selected}, 4) \times w(\textit{NumDefPr}(\textit{Open}, 3) \times w(\textit{AssPr}(\textit{Prize} = 1) = \frac{1}{4} \times \frac{1}{3} \times \frac{3}{10} = \frac{1}{40}$. *Similarly,* $W_{\Pi'}(I_2) = \frac{1}{4} \times \frac{1}{2} \times \frac{2}{10} = \frac{1}{40}$; $W_{\Pi'}(I_3) = \frac{1}{4} \times \frac{1}{2} \times \frac{1}{4} = \frac{1}{32}$.

Combining the translations plog2lpmln and lpmln2wc, one can compute P-log MAP inference using standard ASP solvers.

Conclusion

In this paper we show how $\mathrm{LP}^{\mathrm{MLN}}$ is related to weak constraints and P-log. Weak constraints are relatively simple extension to ASP programs, while P-log is highly structured but a more complex extension. $\mathrm{LP}^{\mathrm{MLN}}$ is shown to be a good middle ground language that clarifies the relationships. We expect the relationships will help us to apply the mathematical and computational results developed for one language to another language.

References

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2015. Hinge-loss markov random fields and probabilistic soft logic. arXiv:1505.04406 [cs.LG].

Balai, E., and Gelfond, M. 2016. On the relationship between P-log and $\mathrm{LP}^{\mathrm{MLN}}$. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Baral, C.; Gelfond, M.; and Rushton, J. N. 2009. Probabilistic reasoning with answer sets. *TPLP* 9(1):57–144.

Buccafurri, F.; Leone, N.; and Rullo, P. 2000. Enhancing disjunctive datalog by constraints. *Knowledge and Data Engineering, IEEE Transactions on* 12(5):845–860.

Calimeri, F.; Faber, W.; Gebser, M.; Ianni, G.; Kaminski, R.; Krennwallner, T.; Leone, N.; Ricca, F.; and Schaub, T. 2013. Asp-core-2 input language format.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, 2462–2467.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175:236–263.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Proceedings of International Logic Programming Conference and Symposium*, 1070–1080. MIT Press.

Lee, J., and Wang, Y. 2016. Weighted rules under the stable model semantics. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Lee, J.; Lifschitz, V.; and Palla, R. 2008. A reductive semantics for counting and choice in answer set programming. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 472–479.

Lee, J.; Meng, Y.; and Wang, Y. 2015. Markov logic style weighted rules under the stable model semantics. In Technical Communications of the 31st International Conference on Logic Programming.

Nickles, M., and Mileo, A. 2014. Probabilistic inductive logic programming based on answer set programming. In *15th International Workshop on Non-Monotonic Reasoning (NMR 2014)*.

Pearl, J. 2000. *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.