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

Joohyung Lee and Zhun Yang
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, USA
{joolee, zyang90}@asu.edu

Abstract

$\mathrm{LP}^{\mathrm{MLN}}$ 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 $\mathrm{LP}^{\mathrm{MLN}}$ and two other extensions of answer set programs: weak constraints to express a quantitative preference among answer sets, and P-log to incorporate probabilistic uncertainty. We present a translation of $\mathrm{LP}^{\mathrm{MLN}}$ into programs with weak constraints and a translation of P-log into $\mathrm{LP}^{\mathrm{MLN}}$, 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 $\mathrm{LP}^{\mathrm{MLN}}$ programs using standard ASP solvers. This result can be extended to other formalisms, such as Markov Logic, ProbLog, and Pearl's Causal Models, that are shown to be translatable into $\mathrm{LP}^{\mathrm{MLN}}$. 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 $\mathrm{LP}^{\mathrm{MLN}}$, which yields a way to compute P-log using standard ASP solvers and MLN solvers.

Introduction

$\mathrm{LP}^{\mathrm{MLN}}$ (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 weighted 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 $\mathrm{LP}^{\mathrm{MLN}}$ 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, $\mathrm{LP}^{\mathrm{MLN}}$ is one of the few languages that incorporate the concept of weights into the semantics. Another one is weak constraints (Buccafurri, Leone, and Rullo 2000), which are to assign a 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 $\mathrm{LP}^{\mathrm{MLN}}$ and these two extensions of answer set programs. It shows how different weight schemes of $\mathrm{LP}^{\mathrm{MLN}}$ and weak constraints are related, and how the probabilistic reasoning in P-log can be expressed in $\mathrm{LP}^{\mathrm{MLN}}$. The result 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 $\mathrm{LP}^{\mathrm{MLN}}$ programs. In the first part of this paper, we show that an inverse translation is also possible under certain conditions, i.e., an $\mathrm{LP}^{\mathrm{MLN}}$ program can be turned into a usual ASP program with weak constraints so that the most probable stable models of the $\mathrm{LP}^{\mathrm{MLN}}$ program are exactly the optimal stable models of the program with weak constraints. The result allows for using ASP solvers for computing Maximum A Posteriori probability (MAP) estimates of $\mathrm{LP}^{\mathrm{MLN}}$ 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 $\mathrm{LP}^{\mathrm{MLN}}$, thereby allowing us to apply combinatorial optimization in standard ASP solvers to MAP inference in these languages.

In the second part of the paper, we show how P-log can be completely characterized in $\mathrm{LP}^{\mathrm{MLN}}$. 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 recent translation from $\mathrm{LP^{MLN}}$ into P-log in (Balai and Gelfond 2016). In conjunction with the embedding of $\mathrm{LP^{MLN}}$ 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: $\mathrm{LP^{MLN}}$

We review the definition of $\mathrm{LP^{MLN}}$ 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 logic program rules as a special case of first-order formulas under the stable model semantics. For example, rule $r(x) \leftarrow p(x), \textit{not}\ q(x)$ is identified with formula $\forall x(p(x) \wedge \neg q(x) \rightarrow r(x))$. An $\mathrm{LP^{MLN}}$ program is a finite set of weighted first-order formulas $w : F$ where w is a real number (in which case the weighted formula is called *soft*) or α for denoting the infinite weight (in which case it is called *hard*). An $\mathrm{LP^{MLN}}$ program is called *ground* if its formulas contain no variables. We assume a finite Herbrand Universe. Any $\mathrm{LP^{MLN}}$ 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 $\mathrm{LP^{MLN}}$ 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 the reader to the stable model semantics of first-order formulas in (Ferraris, Lee, and Lifschitz 2011)). The *unnormalized weight* of an interpretation I under Π is defined as

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

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

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

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

Review: Weak Constraints

A *weak constraint* has the form

$$:\sim F \quad [\textit{Weight} \ @\ \textit{Level}].$$

where F is a ground formula, *Weight* is a real number and *Level* is a nonnegative integer. Note that the syntax is more general than the one from the literature (Buccafurri, Leone, and Rullo 2000; Calimeri et al. 2013), 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 on the input language of ASP solvers.

Let Π be 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. We call I a stable model of Π if it is a stable model of Π_1 (in the sense of (Ferraris, Lee, and Lifschitz 2011)). For every stable model I of Π and any nonnegative integer l, the *penalty* of I at level l, denoted by $\textit{Penalty}_\Pi(I, l)$, is defined as

$$\sum_{\substack{:\sim\ F[w@l] \in \Pi_2, \\ I \models F}} w.$$

For any two stable models I and I' of Π, we say I is *dominated* by I' if

- there is some nonnegative integer l such that $\textit{Penalty}_\Pi(I', l) < \textit{Penalty}_\Pi(I, l)$ and
- for all integers $k > l$, $\textit{Penalty}_\Pi(I', k) = \textit{Penalty}_\Pi(I, k)$.

A stable model of Π is called *optimal* if it is not dominated by another stable model of Π.

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

General Translation

We define a translation that turns an $\mathrm{LP^{MLN}}$ program into a program with weak constraints. For any ground $\mathrm{LP^{MLN}}$ program Π, the translation $\mathrm{lpmln2wc}(\Pi)$ is simply defined as follows. We turn each weighted formula $w : F$ in Π into $\{F\}^{\rm ch}$, where $\{F\}^{\rm ch}$ is a choice formula, standing for $F \vee \neg F$ (Ferraris, Lee, and Lifschitz 2011). Further, we add

$$:\sim F \quad [-1@1] \tag{1}$$

if w is α, and

$$:\sim F \quad [-w@0] \tag{2}$$

otherwise.

Intuitively, choice formula $\{F\}^{\rm ch}$ allows F to be either included or not in deriving a stable model.[2] When F is included, the stable model gets the (negative) penalty -1 at level 1 or $-w$ at level 0 depending on the weight of the formula, which corresponds to the (positive) "reward" e^α or e^w that it receives under the $\mathrm{LP^{MLN}}$ semantics.

The following proposition tells us that choice formulas can be used for generating the members of $\mathrm{SM}[\Pi]$.

Proposition 1 *For any* $\mathrm{LP^{MLN}}$ *program* Π*, the set* $\mathrm{SM}[\Pi]$ *is exactly the set of the stable models of* $\mathrm{lpmln2wc}(\Pi)$.

The following theorem follows from Proposition 1. As the probability of a stable model of an $\mathrm{LP^{MLN}}$ program Π increases, the penalty of the corresponding stable model of $\mathrm{lpmln2wc}(\Pi)$ decreases, and the distinction between hard rules and soft rules can be simulated by the different levels of weak constraints, and vice versa.

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

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

Theorem 1 *For any* $\mathrm{LP}^{\mathrm{MLN}}$ *program* Π*, the most probable stable models (i.e., the stable models with the highest probability) of* Π *are precisely the optimal stable models of the program with weak constraints* $\mathrm{lpmln2wc}(\Pi)$.

Example 1 *For program* Π*:*

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

$\mathrm{SM}[\Pi]$ *has* 5 *elements:* $\emptyset$, $\{p\}$, $\{p,q\}$, $\{p,r\}$, $\{p,q,r\}$. *Among them,* $\{p,q\}$ *is the most probable stable model, whose weight is* e^{15}*, while* $\{p,q,r\}$ *is a probabilistic stable model whose weight is* e^{-4}*. The translation yields*

$$\begin{array}{c|clr} \{p \to q\}^{\mathrm{ch}} & :\sim & p \to q & [-10 \ @\ 0] \\ \{p \to r\}^{\mathrm{ch}} & :\sim & p \to r & [-1 \ @\ 0] \\ \{p\}^{\mathrm{ch}} & :\sim & p & [-5 \ @\ 0] \\ \{\neg r \to \bot\}^{\mathrm{ch}} & :\sim & \neg r \to \bot & [20 \ @\ 0] \end{array}$$

whose optimal stable model is $\{p,q\}$ *with the penalty at level* 0 *being* -15*, while* $\{p,q,r\}$ *is a stable model whose penalty at level* 0 *is* 4.

The following example illustrates how the translation accounts for the difference between hard rules and soft rules by assigning different levels.

Example 2 *Consider the* $\mathrm{LP}^{\mathrm{MLN}}$ *program* Π *in Example 1 from (Lee and Wang 2016).*

$$\begin{array}{lll} \alpha: & \mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{ResidentBird}(\mathit{Jo}) & (r1) \\ \alpha: & \mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{MigratoryBird}(\mathit{Jo}) & (r2) \\ \alpha: & \bot \leftarrow \mathit{ResidentBird}(\mathit{Jo}), \mathit{MigratoryBird}(\mathit{Jo}) & (r3) \\ 2: & \mathit{ResidentBird}(\mathit{Jo}) & (r4) \\ 1: & \mathit{MigratoryBird}(\mathit{Jo}) & (r5) \end{array}$$

The translation $\mathrm{lpmln2wc}(\Pi)$ *is* [3]

$$\begin{array}{l} \{\mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{ResidentBird}(\mathit{Jo})\}^{\mathrm{ch}} \\ \{\mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{MigratoryBird}(\mathit{Jo})\}^{\mathrm{ch}} \\ \{\bot \leftarrow \mathit{ResidentBird}(\mathit{Jo}), \mathit{MigratoryBird}(\mathit{Jo})\}^{\mathrm{ch}} \\ \{\mathit{ResidentBird}(\mathit{Jo})\}^{\mathrm{ch}} \\ \{\mathit{MigratoryBird}(\mathit{Jo})\}^{\mathrm{ch}} \end{array}$$

$$\begin{array}{lr} :\sim \mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{ResidentBird}(\mathit{Jo}) & [-1@1] \\ :\sim \mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{MigratoryBird}(\mathit{Jo}) & [-1@1] \\ :\sim \bot \leftarrow \mathit{ResidentBird}(\mathit{Jo}), \mathit{MigratoryBird}(\mathit{Jo})\} & [-1@1] \\ :\sim \mathit{ResidentBird}(\mathit{Jo}) & [-2@0] \\ :\sim \mathit{MigratoryBird}(\mathit{Jo}) & [-1@0] \end{array}$$

The three probabilistic stable models of Π*,* $\emptyset$*,* $\{\mathit{Bird}(Jo), \mathit{ResidentBird}(Jo)\}$*, and* $\{\mathit{Bird}(Jo), \mathit{MigratoryBird}(Jo)\}$*, have the same penalty* -3 *at level* 1*. Among them,* $\{\mathit{Bird}(Jo), \mathit{ResidentBird}(Jo)\}$ *has the least penalty at level* 0*, and thus is an optimal stable model of* $\mathrm{lpmln2wc}(\Pi)$.

In some applications, one may not want any hard rules to be violated assuming that hard rules encode definite knowledge. For that, $\mathrm{lpmln2wc}(\Pi)$ can be modified by simply turning hard rules into the usual ASP rules. Then the stable models of $\mathrm{lpmln2wc}(\Pi)$ satisfy all hard rules. For example, the program in Example 2 can be translated into programs with weak constraints as follows.

$$\begin{array}{l} \mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{ResidentBird}(\mathit{Jo}) \\ \mathit{Bird}(\mathit{Jo}) \leftarrow \mathit{MigratoryBird}(\mathit{Jo}) \\ \bot \leftarrow \mathit{ResidentBird}(\mathit{Jo}), \mathit{MigratoryBird}(\mathit{Jo}) \\ \{\mathit{ResidentBird}(\mathit{Jo})\}^{\mathrm{ch}} \\ \{\mathit{MigratoryBird}(\mathit{Jo})\}^{\mathrm{ch}} \end{array}$$

$$\begin{array}{lr} :\sim \mathit{ResidentBird}(\mathit{Jo}) & [-2@0] \\ :\sim \mathit{MigratoryBird}(\mathit{Jo}) & [-1@0] \end{array}$$

Also note that while the most probable stable models of Π and the optimal stable models of $\mathrm{lpmln2wc}(\Pi)$ coincide, their weights and penalties 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, they 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 also be reduced to the optimal 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.

Theorem 2 *For any Markov Logic Network* Π*, the most probable models of* Π *are precisely the optimal stable models of the program with weak constraints* $\mathrm{mln2wc}(\Pi)$.

Similarly, MAP inference in ProbLog and Pearl's Causal Models can be reduced to finding an optimal stable model of a program 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 aggregating the weights of satisfied formulas, we may aggregate the weights of formulas that are not satisfied. Let $\mathrm{lpmln2wc}^{\mathrm{pnt}}(\Pi)$ be a modification of $\mathrm{lpmln2wc}(\Pi)$ by replacing (1) with

$$:\sim \ \neg F \ \ [1@1]$$

and (2) with

$$:\sim \ \neg F \ \ [w@0].$$

Intuitively, when F is not satisfied, the stable model gets the penalty 1 at level 1, or w at level 0 depending on whether F is a hard or soft formula.

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

[3] Recall that we identify the rules with the corresponding first-order formulas.

view will lead to a slight advantage when we further turn the translation into the input language of ASP solvers (See Footnote 6).

The current ASP solvers do not allow arbitrary formulas to appear in weak constraints. For computation using the ASP solvers, let $\text{lpmln2wc}^{\text{pnt,rule}}(\Pi)$ be the translation by turning each weighted formula $w_i : F_i$ in Π into

$$\begin{array}{rcl} \neg F_i & \rightarrow & \mathit{unsat}(i) \\ \neg \mathit{unsat}(i) & \rightarrow & F_i \\ & :\sim & \mathit{unsat}(i) \quad [w_i@l]. \end{array}$$

where $\mathit{unsat}(i)$ is a new atom, and $l = 1$ if w_i is α and $l = 0$ otherwise.

Corollary 2 *Let Π be an LP^{MLN} program. There is a 1-1 correspondence ϕ between the most probable stable models of Π and the optimal stable models of $\text{lpmln2wc}^{\text{pnt,rule}}(\Pi)$, where $\phi(I) = I \cup \{\mathit{unsat}(i) \mid w_i : F_i \in \Pi,\ I \not\models F_i\}$.*

The corollary allows us to compute the most probable stable models (MAP estimates) of the LP^{MLN} program using the combination of F2LP [4] and CLINGO [5] (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 LP^{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 $\text{lpmln2wc}^{\text{pnt,rule}}$ by turning each weighted rule

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

instead into

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

where $l = 1$ if w_i is α and $l = 0$ otherwise.

In the case when Head_i is $\bot$, the translation can be further simplified: we simply turn $w_i : \bot \leftarrow \mathit{Body}_i$ into $:\sim \mathit{Body}_i \quad [w_i@l]$.[6]

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

$$\begin{array}{l|l|lr} \mathit{unsat}(1) \leftarrow p, \mathit{not}\ q & q \leftarrow p, \mathit{not}\ \mathit{unsat}(1) & :\sim \mathit{unsat}(1) & [10@0] \\ \mathit{unsat}(2) \leftarrow p, \mathit{not}\ r & r \leftarrow p, \mathit{not}\ \mathit{unsat}(2) & :\sim \mathit{unsat}(2) & [1@0] \\ \mathit{unsat}(3) \leftarrow \mathit{not}\ p & p \leftarrow \mathit{not}\ \mathit{unsat}(3) & :\sim \mathit{unsat}(3) & [5@0] \\ & & :\sim \mathit{not}\ r & [-20@0] \end{array}$$

Turning P-log into LP^{MLN}

Review: P-log

Syntax A *sort* is a set of symbols. A *constant* c maps an element in the *domain* $s_1 \times \dots \times s_n$ to an element in the *range* s_0 (denoted by *Range*(c)), where each of $s_0, \dots, s_n$ is a sort. A *sorted propositional signature* is a special case of propositional signatures constructed from a set of constants and their associated sorts, consisting of all propositional atoms $c(\vec{u}) = v$ where $c : s_1 \times \dots \times s_n \rightarrow s_0$, and $\vec{u} \in s_1 \times \dots \times s_n$, and $v \in s_0$.[7] Symbol $c(\vec{u})$ is called an *attribute* and v is called 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 a 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 usual 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

$$\Pi = \langle \mathbf{R}, \mathbf{S}, \mathbf{P}, \mathbf{Obs}, \mathbf{Act} \rangle \tag{4}$$

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

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

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

  $$[r]\ \ random(c(\vec{u}) : \{x : p(x)\}) \leftarrow \mathit{Body} \tag{5}$$

  where r is a unique identifier, p is a boolean constant with a unary argument, and *Body* is a set of literals. x is a schematic variable ranging over the argument sort of p. Rule (5) is called a *random selection* rule for $c(\vec{u})$. Intuitively, rule (5) says that if *Body* is true, the value of $c(\vec{u})$ is selected at random from the set *Range*$(c) \cap \{x : p(x)\}$ unless this value is fixed by a deliberate action, i.e., $\mathit{Do}(c(\vec{u}) = v)$ for some value v.
- $\mathbf{P}$ is a set of so-called *probability atoms (pr-atoms)* of the form

  $$pr_r(c(\vec{u}) = v \mid C) = p \tag{6}$$

  where r is the identifier of some random selection rule for $c(\vec{u})$ in $\mathbf{S}$; $c(\vec{u}) = v \in \sigma_1$; C is a set of literals; and p is a real number in $[0, 1]$. We say pr-atom (6) is *associated* with the random selection rule whose identifier is r.
- $\mathbf{Obs}$ is a set of atomic facts for representing "observation": $\mathit{Obs}(c(\vec{u}) = v)$ and $\mathit{Obs}(c(\vec{u}) \neq v)$.
- $\mathbf{Act}$ is a set of atomic facts for representing a deliberate action: $\mathit{Do}(c(\vec{u}) = v)$.

Semantics Let Π be a P-log program (4) 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

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

that accounts for the logical part of P-log. Due to lack of space we refer the reader to (Baral, Gelfond, and Rushton 2009) for the definition of $\tau(\Pi)$.

[4] `http://reasoning.eas.asu.edu/f2lp/`

[5] `http://potassco.sourceforge.net`

[6] 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.

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

An atom $c(\vec{u}) = v$ is called *possible* in a possible world W due to a random selection rule (5) if Π contains (5) such that $W \models \mathit{Body} \wedge p(v) \wedge \neg \mathit{Intervene}(c(\vec{u}))$.[8] Pr-atom (6) 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 $c(\vec{u})$:
$$[r_1]\ \mathit{random}(c(\vec{u}) : \{x : p_1(x)\}) \leftarrow \mathit{Body}_1,$$
$$[r_2]\ \mathit{random}(c(\vec{u}) : \{x : p_2(x)\}) \leftarrow \mathit{Body}_2,$$
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 $c(\vec{u})$:
$$[r]\ \mathit{random}(c(\vec{u}) : \{x : p(x)\}) \leftarrow \mathit{Body}$$
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 (5) 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 Π 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 Π, it follows that $W \not\models C'$. So there is only one pr-atom that is applied in W 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
$$\mathit{PossWithDefPr}(W, c(\vec{u}) = v) = \max\big(p, 0\big),$$
where p is
$$\frac{1 - \sum_{v' \in AV_W(c(\vec{u}))} \mathit{PossWithAssPr}(W, c(\vec{u}) = v')}{|\{v'' \mid c(\vec{u}) = v'' \text{ is possible in } W \text{ and } v'' \notin AV_W(c(\vec{u}))\}|}. \quad (7)$$
("$c(\vec{u}) = v$ is possible in W with the default probability.")
The max function is used to ensure that the default probability is nonnegative.[9]

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
$$\hat{\mu}_\Pi(W) = \prod_{\substack{c(\vec{u}) = v \in W \text{ and} \\ c(\vec{u}) = v \text{ is possible in } W}} P(W, c(\vec{u}) = v),$$
and, 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 a non-zero probability.

Example 3 *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 four 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*

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

The possible worlds of Π are as follows:

- $W_1 = \{\mathit{Obs}(\mathit{Selected} = 1), \mathit{Obs}(\mathit{Open} = 2), \mathit{Obs}(\mathit{Prize} \neq 2), \mathit{Selected} = 1, \mathit{Open} = 2, \mathit{Prize} = 1, \mathit{CanOpen}(1) = \mathbf{f}, \mathit{CanOpen}(2) = \mathbf{t}, \mathit{CanOpen}(3) = \mathbf{t}, \mathit{CanOpen}(4) = \mathbf{t}\}$
- $W_2 = \{\mathit{Obs}(\mathit{Selected} = 1), \mathit{Obs}(\mathit{Open} = 2), \mathit{Obs}(\mathit{Prize} \neq 2), \mathit{Selected} = 1, \mathit{Open} = 2, \mathit{Prize} = 3, \mathit{CanOpen}(1) = \mathbf{f}, \mathit{CanOpen}(2) = \mathbf{t}, \mathit{CanOpen}(3) = \mathbf{f}, \mathit{CanOpen}(4) = \mathbf{t}\}$
- $W_3 = \{\mathit{Obs}(\mathit{Selected} = 1), \mathit{Obs}(\mathit{Open} = 2), \mathit{Obs}(\mathit{Prize} \neq 2), \mathit{Selected} = 1, \mathit{Open} = 2, \mathit{Prize} = 4, \mathit{CanOpen}(1) = \mathbf{f}, \mathit{CanOpen}(2) = \mathbf{t}, \mathit{CanOpen}(3) = \mathbf{t}, \mathit{CanOpen}(4) = \mathbf{f}\}$.

The probability of each atom to happen is

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

$$
\begin{array}{l}
P(W_1, \mathit{Open}\!=\!2) = \mathit{PossWithDefPr}(W_1, \mathit{Open}\!=\!2) = 1/3 \\
P(W_2, \mathit{Open}\!=\!2) = \mathit{PossWithDefPr}(W_2, \mathit{Open}\!=\!2) = 1/2 \\
P(W_3, \mathit{Open}\!=\!2) = \mathit{PossWithDefPr}(W_3, \mathit{Open}\!=\!2) = 1/2
\end{array}
$$

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

So,

[8] Note that this is slightly different from the original definition of P-log from (Baral, Gelfond, and Rushton 2009), according to which, if $\mathit{Intervene}(c(\vec{u}))$ is true, the probability of $c(\vec{u}) = v$ is determined by the default probability, which is a bit unintuitive.

[9] In (Baral, Gelfond, and Rushton 2009), a stronger condition of "unitariness" is imposed to prevent (7) from being negative.

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

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) increases the chance by 25%.

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

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

Possible Atoms: For each random selection rule (5) for $c(\vec{u})$ in $\mathbf{S}$ and for each $v \in \mathit{Range}(c)$, $\mathrm{plog2lpmln}(\Pi)$ includes

$$\mathit{Poss}_r(c(\vec{u}) = v) \leftarrow \mathit{Body}, p(v), \mathit{not}\ \mathit{Intervene}(c(\vec{u})) \quad (8)$$

Rule (8) expresses that $c(\vec{u}) = v$ is possible in W due to r if $W \models \mathit{Body} \wedge p(v) \wedge \neg \mathit{Intervene}(c(\vec{u}))$.

Assigned Probability: For each pr-atom (6) in $\mathbf{P}$, $\mathrm{plog2lpmln}(\Pi)$ contains the following rules:

$$\alpha : \mathit{PossWithAssPr}_{r,C}(c(\vec{u}) = v) \leftarrow \mathit{Poss}_r(c(\vec{u}) = v), C \quad (9)$$

$$\alpha : \mathit{AssPr}_{r,C}(c(\vec{u}) = v) \leftarrow c(\vec{u}) = v, \mathit{PossWithAssPr}_{r,C}(c(\vec{u}) = v) \quad (10)$$

$$ln(p) : \bot \leftarrow \mathit{not}\ \mathit{AssPr}_{r,C}(c(\vec{u}) = v) \qquad (p > 0) \quad (11)$$

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

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

Rule (9) expresses the condition under which pr-atom (6) is applied in a possible world W. Further, if $c(\vec{u}) = v$ is true in W as well, rules (10) and (11) contribute the assigned probability $e^{ln(p)} = p$ to the unnormalized probability of W as a factor when $p > 0$.

Denominator for Default Probability: For each random selection rule (5) for $c(\vec{u})$ in $\mathbf{S}$ and for each $v \in \mathit{Range}(c)$, $\mathrm{plog2lpmln}(\Pi)$ includes

$$\alpha : \mathit{PossWithDefPr}(c(\vec{u}) = v) \leftarrow \mathit{Poss}_r(c(\vec{u}) = v), \mathit{not}\ \mathit{PossWithAssPr}(c(\vec{u}) = v) \quad (12)$$

$$\alpha : \mathit{NumDefPr}(c(\vec{u}), x) \leftarrow c(\vec{u}) = v, \mathit{PossWithDefPr}(c(\vec{u}) = v), x = \#\mathit{count}\{y : \mathit{PossWithDefPr}(c(\vec{u}) = y)\} \quad (13)$$

$$ln(\tfrac{1}{m}) : \bot \leftarrow \mathit{not}\ \mathit{NumDefPr}(c(\vec{u}), m) \qquad (m = 2, \dots, |\mathit{Range}(c)|) \quad (14)$$

Rule (12) asserts that $c(\vec{u}) = v$ is possible in W with a default probability if it is possible in W and not possible with an assigned probability. Rule (13) expresses, intuitively, that $\mathit{NumDefPr}(c(\vec{u}), x)$ 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 (7). Then rule (14) contributes the factor $1/x$ to the unnormalized probability of W as a factor.

Numerator for Default Probability:

- Consider each random selection rule $[r]\ random(c(\vec{u}) : \{x : p(x)\}) \leftarrow \mathit{Body}$ for $c(\vec{u})$ in $\mathbf{S}$ along with all pr-atoms associated with it in $\mathbf{P}$:

  $$pr_r(c(\vec{u}) = v_1 \mid C_1) = p_1$$
  $$\dots$$
  $$pr_r(c(\vec{u}) = v_n \mid C_n) = p_n$$

  where $n \geq 1$, and v_i and v_j $(i \neq j)$ may be equal. For each $v \in \mathit{Range}(c)$, $\mathrm{plog2lpmln}(\Pi)$ contains the following rules:[10]

  $$\alpha : \mathit{RemPr}(c(\vec{u}), 1-y) \leftarrow \mathit{Body}\ c(\vec{u}) = v, \mathit{PossWithDefPr}(c(\vec{u}) = v), y = \#\mathrm{sum}\{p_1 : \mathit{PossWithAssPr}_{r,C_1}(c(\vec{u}) = v_1); \dots ; p_n : \mathit{PossWithAssPr}_{r,C_n}(c(\vec{u}) = v_n)\}. \quad (15)$$

  $$\alpha : \mathit{TotalDefPr}(c(\vec{u}), x) \leftarrow \mathit{RemPr}(c(\vec{u}), x), x > 0 \quad (16)$$

  $$ln(x) : \bot \leftarrow \mathit{not}\ \mathit{TotalDefPr}(c(\vec{u}), x) \quad (17)$$

  $$\alpha : \bot \leftarrow \mathit{RemPr}(c(\vec{u}), x), x \leq 0. \quad (18)$$

  In rule (15), y is the sum of all assigned probabilities. Rules (16) and (17) are to account for the numerator of (7) when $n > 0$. The variable x stands for the numerator of (7). Rule (18) is to avoid assigning a non-positive default probability to a possible world.

Note that most rules in $\mathrm{plog2lpmln}(\Pi)$ are hard rules. The soft rules (11), (14), (17) cannot be simplified as atomic facts, e.g., $ln(\frac{1}{m}) : \mathit{NumDefPr}(c(\vec{u}), m)$ in place of (14), which is in contrast with the use of probabilistic choice atoms in the distribution semantics 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 probabilities, and not from independent probabilistic choices like the other probabilistic logic programming languages. In conjunction with the embedding of ProbLog in $\mathrm{LP^{MLN}}$ (Lee and Wang 2016), it is interesting to note that both kinds of probabilities can be captured in $\mathrm{LP^{MLN}}$ using different kinds of rules.

Example 3 Continued For the program Π in Example 3, $\mathrm{plog2lpmln}(\Pi)$ consists of the rules $\alpha : R$ for each rule R in $\tau(\Pi)$ and the following rules.

Possible Atoms:

$$\alpha : \mathit{Poss}(\mathit{Prize} = d) \leftarrow \mathit{not}\ \mathit{Intervene}(\mathit{Prize})$$
$$\alpha : \mathit{Poss}(\mathit{Selected} = d) \leftarrow \mathit{not}\ \mathit{Intervene}(\mathit{Selected})$$
$$\alpha : \mathit{Poss}(\mathit{Open} = d) \leftarrow \mathit{CanOpen}(d), \mathit{not}\ \mathit{Intervene}(\mathit{Open})$$

[10] The sum aggregate can be represented by ground first-order formulas under the stable model semantics under the assumption that the Herbrand Universe is finite (Ferraris 2011). In the general case, it can be represented by generalized quantifiers (Lee and Meng 2012) or infinitary propositional formulas (Harrison, Lifschitz, and Yang 2014). In the input language of ASP solvers, which does not allow real number arguments, p_i can be approximated to integers of some fixed interval.

Assigned Probability:

α : *PossWithAssPr*(*Prize* $= 1) \leftarrow$ *Poss*(*Prize* $= 1$)
α : *AssPr*(*Prize* $= 1) \leftarrow$ *Prize* $= 1$, *PossWithAssPr*(*Prize* $= 1$)
$ln(0.3) : \bot \leftarrow$ *not AssPr*(*Prize* $= 1$)

α : *PossWithAssPr*(*Prize* $= 3) \leftarrow$ *Poss*(*Prize* $= 3$)
α : *AssPr*(*Prize* $= 3) \leftarrow$ *Prize* $= 3$, *PossWithAssPr*(*Prize* $= 3$)
$ln(0.2) : \bot \leftarrow$ *not AssPr*(*Prize* $= 3$)

(We simplified slightly not to distinguish *PossWithAssPr*$(\cdot)$ and *PossWithAssPr*$_{r,C}(\cdot)$ because there is only one random selection rule for *Prize* and both pr-atoms for *Prize* has empty conditions.)

Denominator for Default Probability:

α : *PossWithDefPr*(*Prize* $= d) \leftarrow$
Poss(*Prize* $= d$), *not PossWithAssPr*(*Prize* $= d$)
α : *PossWithDefPr*(*Selected* $= d) \leftarrow$
Poss(*Selected* $= d$), *not PossWithAssPr*(*Selected* $= d$)
α : *PossWithDefPr*(*Open* $= d) \leftarrow$
Poss(*Open* $= d$), *not PossWithAssPr*(*Open* $= d$)

α : *NumDefPr*(*Prize*, $x) \leftarrow$
Prize $= d$, *PossWithDefPr*(*Prize* $= d$),
$x = \#count\{y$: *PossWithDefPr*(*Prize* $= y)\}$
α : *NumDefPr*(*Selected*, $x) \leftarrow$
Selected $= d$, *PossWithDefPr*(*Selected* $= d$),
$x = \#count\{y$: *PossWithDefPr*(*Selected* $= y)\}$
α : *NumDefPr*(*Open*, $x) \leftarrow$
Open $= d$, *PossWithDefPr*(*Open* $= d$),
$x = \#count\{y$: *PossWithDefPr*(*Open* $= y)\}$
$ln(\frac{1}{m}) :\leftarrow$ *not NumDefPr*(c, m)
($c \in \{$*Prize*, *Selected*, *Open*$\}$, $m \in \{2, 3, 4\}$)

Numerator for Default Probability:

α : *RemPr*(*Prize*, $1-x) \leftarrow$ *Prize* $= d$, *PossWithDefPr*(*Prize* $= d$),
$x = \#\text{sum}\{0.3$: *PossWithAssPr*(*Prize*$=1$);
0.2 : *PossWithAssPr*(*Prize*$=3$)$\}$
α : *TotalDefPr*(*Prize*, $x) \leftarrow$ *RemPr*(*Prize*, x), $x > 0$
$ln(x) : \bot \leftarrow$ *not TotalDefPr*(*Prize*, x)
$\alpha : \bot \leftarrow$ *RemDefPr*(*Prize*, x), $x \leq 0$

Clearly, the signature of $\text{plog2lpmln}(\Pi)$ is a superset of the signature of Π. Further, $\text{plog2lpmln}(\Pi)$ is linear-time constructible. The following theorem tells us that there is a 1-1 correspondence between the set of the possible worlds with non-zero probabilities of Π and the set of the stable models of $\text{plog2lpmln}(\Pi)$ such that each stable model is an extension of the possible world, and the probability of each possible world of Π coincides with the probability of the corresponding stable model of $\text{plog2lpmln}(\Pi)$.

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

- *For every possible world W of Π that has a non-zero probability, $\phi(W)$ is a probabilistic stable model of $\text{plog2lpmln}(\Pi)$, and $\mu_\Pi(W) = P_{\text{plog2lpmln}(\Pi)}(\phi(W))$.*
- *For every probabilistic stable model I of $\text{plog2lpmln}(\Pi)$, the restriction of I onto the signature of $\tau(\Pi)$, denoted $I|_{\sigma(\tau(\Pi))}$, is a possible world of Π and $\mu_\Pi(I|_{\sigma(\tau(\Pi))}) > 0$.*

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

1. $\phi(W) \models$ *Poss*$_r(c(\vec{u}) = v)$ iff $c(\vec{u}) = v$ is possible in W due to r.
2. For each pr-atom $pr_r(c(\vec{u})=v \mid C) = p$ in Π,
$\phi(W) \models$ *PossWithAssPr*$_{r,C}(c(\vec{u})=v)$ iff this pr-atom is applied in W.
3. For each pr-atom $pr_r(c(\vec{u})=v \mid C) = p$ in Π,
$\phi(W) \models$ *AssPr*$_{r,C}(c(\vec{u}) = v)$ iff this pr-atom is applied in W, and $W \models c(\vec{u})=v$.
4. $\phi(W) \models$ *PossWithAssPr*$(c(\vec{u})=v)$ iff $v \in AV_W(c(\vec{u}))$.
5. $\phi(W) \models$ *PossWithDefPr*$(c(\vec{u})=v)$ iff $c(\vec{u})=v$ is possible in W and $v \notin AV_W(c(\vec{u}))$.
6. $\phi(W) \models$ *NumDefPr*$(c(\vec{u}), m)$ iff there exist exactly m 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, $W \models c(\vec{u})=v$.
7. $\phi(W) \models$ *RemPr*$(c(\vec{u}), k)$ iff there exists a value v such that $W \models c(\vec{u}) = v$; $c(\vec{u}) = v$ is possible in W; $v \notin AV_W(c(\vec{u}))$; and
$$k = 1 - \sum_{v \in AV_W(c(\vec{u}))} \textit{PossWithAssPr}(W, c(\vec{u})=v).$$
8. $\phi(W) \models$ *TotalDefPr*$(c(\vec{u}), k)$ iff $\phi(W) \models$ *RemPr*$(c(\vec{u}, k))$ and $k > 0$.

To check that $\mu_\Pi(W) = P_{\text{plog2lpmln}(\Pi)}(\phi(W))$, note first that the weight of $\phi(W)$ is computed by multiplying e to the power of the weights of rules (11), (14), (17) that are satisfied by $\phi(W)$. Let's call this product TW.

Consider a possible world W with a non-zero probability 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 3, $\phi(W) \models$ *AssPr*$_{r,C}(c(\vec{u})=v)$, so that from (11), the same p is a factor of TW.

If $c(\vec{u}) = v$ is possible in W and $v \notin AV_W(c(\vec{u}))$, the default probability is applied: $P(W, c(\vec{u}) = v) = p$ is computed by (7). By Condition 8, $\phi(W) \models$ *TotalDefPr*$(c(\vec{u}), x)$ where $x = 1 - \sum_{v' \in AV_W(c(\vec{u}))} \textit{PossWithAssPr}(W, c(\vec{u}) = v')$. Since $\phi(W) \models$ (17), it is a factor of TW, which is the same as the numerator of (7). Furthermore, by Condition 6, $\phi(W) \models$ *NumDefPr*$(c(\vec{u}), m)$, where m is the denominator of (7). Since $\phi(W) \models$ (14), $\frac{1}{m}$ is a factor of TW. ∎

Example 3 Continued *For the P-log program Π for the Monty Hall problem, $\Pi' = \text{plog2lpmln}(\Pi)$ 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: Poss$(Prize = i)$ for $i = 1, 2, 3, 4$; Poss$(Selected=i)$ for $i = 1, 2, 3, 4$; PossWithAssPr(Prize$=$*

$i)$ for $i = 1, 3$; *PossWithDefPr*$($*Prize* $= i)$ for $i = 2, 4$; *PossWithDefPr*$($*Selected* $= i)$ for $i = 1, 2, 3, 4$; *NumDefPr*$($*Selected*$, 4)$. In addition,

- $I_1 \models \{\mathit{AssPr}(\mathit{Prize}{=}1), \mathit{Poss}(\mathit{Open}{=}2), \mathit{Poss}(\mathit{Open}{=}3), \mathit{Poss}(\mathit{Open}{=}4), \mathit{PossWithDefPr}(\mathit{Open}{=}2), \mathit{PossWithDefPr}(\mathit{Open}{=}3), \mathit{PossWithDefPr}(\mathit{Open}{=}4), \mathit{NumDefPr}(\mathit{Open}, 3)\}$
- $I_2 \models \{\mathit{AssPr}(\mathit{Prize}{=}3), \mathit{Poss}(\mathit{Open}{=}2), \mathit{Poss}(\mathit{Open}{=}4), \mathit{PossWithDefPr}(\mathit{Open}{=}2), \mathit{PossWithDefPr}(\mathit{Open}{=}4), \mathit{NumDefPr}(\mathit{Open}, 2)\}$
- $I_3 \models \{\mathit{Poss}(\mathit{Open}{=}2), \mathit{Poss}(\mathit{Open}{=}3), \mathit{PossWithDefPr}(\mathit{Open}{=}2), \mathit{PossWithDefPr}(\mathit{Open}{=}3), \mathit{NumDefPr}(\mathit{Open}, 2), \mathit{NumDefPr}(\mathit{Prize}, 2), \mathit{RemPr}(\mathit{Prize}, 0.5), \mathit{TotalDefPr}(\mathit{Prize}, 0.5)\}$.

The unnormalized weight $W_{\Pi'}(I_i)$ of each probabilistic stable model I_i is shown below. $w(\mathit{AssPr}_{r,C}(c(\vec{u}) = v))$ denotes the exponentiated weight of rule (11); $w(\mathit{NumDefPr}(c(\vec{u}), m))$ denotes the exponentiated weight of rule (14); $w(\mathit{TotalDefPr}(c(\vec{u}), x))$ denotes the exponentiated weight of rule (17).

- $W_{\Pi'}(I_1) = w(\mathit{NumDefPr}(\mathit{Selected}, 4)) \times w(\mathit{AssPr}(\mathit{Prize} = 1)) \times w(\mathit{NumDefPr}(Open, 3)) = \frac{1}{4} \times \frac{3}{10} \times \frac{1}{3} = \frac{1}{40}$.
- $W_{\Pi'}(I_2) = w(\mathit{NumDefPr}(\mathit{Selected}, 4)) \times w(\mathit{AssPr}(\mathit{Prize} = 3)) \times w(\mathit{NumDefPr}(Open, 2)) = \frac{1}{4} \times \frac{2}{10} \times \frac{1}{2} = \frac{1}{40}$;
- $W_{\Pi'}(I_3) = w(\mathit{NumDefPr}(\mathit{Selected}, 4)) \times \times w(\mathit{NumDefPr}(Open, 2)) \times \times w(\mathit{NumDefPr}(\mathit{Prize}, 2) \times w(\mathit{TotalDefPr}(\mathit{Prize}, 0.5) = \frac{1}{4} \times \frac{1}{2} \times \frac{1}{2} \times \frac{5}{10} = \frac{1}{32}$.

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

Conclusion

In this paper, we show how $\mathrm{LP^{MLN}}$ is related to weak constraints and P-log. Weak constraints are a relatively simple extension to ASP programs, while P-log is highly structured but a more complex extension. $\mathrm{LP^{MLN}}$ 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.

Acknowledgments We are grateful to Evgenii Balai, Yi Wang and anonymous referees for their useful comments on the draft of this paper. This work was partially supported by the National Science Foundation under Grants IIS-1319794 and IIS-1526301.

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 $\mathrm{LP^{MLN}}$. 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.

Ferraris, P. 2011. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic (TOCL)* 12(4):25.

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.

Harrison, A. J.; Lifschitz, V.; and Yang, F. 2014. The semantics of gringo and infinitary propositional formulas. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014*.

Lee, J., and Meng, Y. 2012. Stable models of formulas with generalized quantifiers (preliminary report). In *Technical Communications of the 28th International Conference on Logic Programming*, 61–71.

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)*, 145–154.

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.