LP\textsuperscript{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

LP\textsuperscript{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\textsuperscript{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 LP\textsuperscript{MLN} into programs with weak constraints and a translation of P-log into LP\textsuperscript{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\textsuperscript{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 LP\textsuperscript{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\textsuperscript{MLN}, which yields a way to compute P-log using standard ASP solvers and MLN solvers.

Introduction

LP\textsuperscript{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 LP\textsuperscript{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\textsuperscript{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 LP\textsuperscript{MLN} and these two extensions of answer set programs. It shows how different weight schemes of LP\textsuperscript{MLN} and weak constraints are related, and how the probabilistic reasoning in P-log can be expressed in LP\textsuperscript{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 LP\textsuperscript{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\textsuperscript{MLN} program can be turned into a usual ASP program with weak constraints so that the most probable stable models of the LP\textsuperscript{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 LP\textsuperscript{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\textsuperscript{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 LP\textsuperscript{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 LP^{MLN} into P-log in (Balai and Gelfond 2016). In conjunction 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: LP^{MLN}
We review the definition of 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 \( r(x) \leftarrow p(x), \neg q(x) \) is identified with formula \( \forall x (p(x) \land \neg q(x) \rightarrow r(x)) \). An 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 \( \alpha \) for denoting the infinite weight (in which case it is called hard). An LP^{MLN} program is called ground if its formulas contain no variables. We assume a finite Herbrand Universe. Any 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 LP^{MLN} program \( \Pi \) and any interpretation \( I \), \( \Pi[I] \) denotes the set of weighted formula \( w : F \) in \( \Pi \) such that \( I \models F \), and \( \text{SM}[\Pi] \) denotes the set \( \{ I \mid I \text{ is a stable model of } \Pi[I] \} \) (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 \( \Pi \) is defined as

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

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

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

\( I \) is called a (probabilistic) stable model of \( \Pi \) if \( P_{\Pi}(I) \neq 0 \).

Review: Weak Constraints
A weak constraint has the form

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

where \( F \) is a ground formula, \( \text{Weight} \) is a real number and \( \text{Level} \) is a nonnegative integer. Note 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 \( \Pi \) be a program \( \Pi_1 \cup \Pi_2 \), where \( \Pi_1 \) is a set of ground formulas and \( \Pi_2 \) is a set of weak constraints. We call \( I \) a stable model of \( \Pi \) if it is a stable model of \( \Pi_1 \) (in the sense of (Ferraris, Lee, and Lifschitz 2011)). For every stable model \( I \) of \( \Pi \) and any nonnegative integer \( l \), the penalty of \( I \) at level \( l \), denoted by \( \text{Penalty}_{\Pi}(I, l) \), is defined as

\[
\sum_{F \in \Pi_2, \sim \sim F} w.
\]

For any two stable models \( I \) and \( I' \) of \( \Pi \), we say \( I \) is dominated by \( I' \) if

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

A stable model of \( \Pi \) is called optimal if it is not dominated by another stable model of \( \Pi \).

Turning LP^{MLN} into Programs with Weak Constraints

General Translation
We define a translation that turns an LP^{MLN} program into a program with weak constraints. For any ground LP^{MLN} program \( \Pi \), the translation \( \text{lpm}^{\Pi} \) is simply defined as follows. We turn each weighted formula \( w : F \) in \( \Pi \) into \( \{ F \}^\text{ch} \), where \( \{ F \}^\text{ch} \) is a choice formula, standing for \( F \lor \neg F \). We define a translation that turns an answer set program with weak constraints. For any ground \( \text{lpm}^{\Pi} \) program into \( \text{lpm}^{\Pi} \), the translation \( \text{lpm}^{\Pi} \) is simply defined as follows. We turn each weighted formula \( w : F \) in \( \Pi \) into \( \{ F \}^\text{ch} \), where \( \{ F \}^\text{ch} \) is a choice formula, standing for \( F \lor \neg F \). Further, we add

\[
\therefore F \quad [-1@1]
\]

if \( w \) is \( \alpha \), and

\[
\therefore F \quad [-w@0]
\]

otherwise.

Intuitively, choice formula \( \{ F \}^\text{ch} \) 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” \( e^\alpha \) or \( e^w \) that it receives under the LP^{MLN} semantics.

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

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

The following theorem follows from Proposition 1. As the probability of a stable model of an LP^{MLN} program \( \Pi \) increases, the penalty of the corresponding stable model of \( \text{lpm}^{\Pi} \) 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 \( \neg p \).

\(^{2}\)This view of choice formulas was used in PrASP (Nickles and Mileo 2014) in defining spanning programs.
Theorem 1 For any LP^MLN program \( \Pi \), the most probable stable models (i.e., the stable models with the highest probability) of \( \Pi \) are precisely the optimal stable models of the program with weak constraints \( lpmln2wc(\Pi) \).

Example 1 For program \( \Pi \):

\[
10 : \quad p \rightarrow q \\
1 : \quad p \rightarrow r \\
5 : \quad p \\
-20 : \quad \neg r \rightarrow \bot
\] (3)

SM[\( \Pi \)] has 5 elements: \( \emptyset \), \( \{p\} \), \( \{p, q\} \), \( \{p, r\} \), \( \{p, q, r\} \). Among them, \( \{p, q, r\} \) 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

\[
\begin{align*}
\{p \rightarrow q\}^{ch} & : \quad p \rightarrow q \quad [-10 @ 0] \\
\{p \rightarrow r\}^{ch} & : \quad p \rightarrow r \quad [-1 @ 0] \\
\{p\}^{ch} & : \quad p \quad [-5 @ 0] \\
\{\neg r \rightarrow \bot\}^{ch} & : \quad \neg r \rightarrow \bot \quad [20 @ 0]
\end{align*}
\]

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 LP^MLN program \( \Pi \) in Example 1 from (Lee and Wang 2016).

\[
\begin{align*}
\alpha & : \quad Bird(Jo) \leftarrow ResidentBird(Jo) \quad (r1) \\
\alpha & : \quad Bird(Jo) \leftarrow MigratoryBird(Jo) \quad (r2) \\
\alpha & : \quad \bot \leftarrow ResidentBird(Jo), MigratoryBird(Jo) \quad (r3) \\
2 : \quad ResidentBird(Jo) \quad (r4) \\
1 : \quad MigratoryBird(Jo) \quad (r5)
\end{align*}
\]

The translation \( lpmln2wc(\Pi) \) is

\[
\begin{align*}
\{Bird(Jo) \leftarrow ResidentBird(Jo)\}^{ch} & \quad [-10 @ 1] \\
\{Bird(Jo) \leftarrow MigratoryBird(Jo)\}^{ch} & \quad [-10 @ 1] \\
\{\bot \leftarrow ResidentBird(Jo), MigratoryBird(Jo)\}^{ch} & \quad [-10 @ 1] \\
\{ResidentBird(Jo)\}^{ch} & \quad [-2 @ 0] \\
\{MigratoryBird(Jo)\}^{ch} & \quad [-1 @ 0]
\end{align*}
\]

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

In some applications, one may not want any hard rules to be violated assuming that hard rules encode definite knowledge. For that, \( lpmln2wc(\Pi) \) can be modified by simply turning hard rules into the usual ASP rules. Then the stable models of \( lpmln2wc(\Pi) \) satisfy all hard rules. For example,

The program in Example 2 can be translated into programs with weak constraints as follows.

\[
\begin{align*}
Bird(Jo) & \leftarrow ResidentBird(Jo) \quad [-10 @ 0] \\
Bird(Jo) & \leftarrow MigratoryBird(Jo) \quad [-10 @ 0] \\
\bot & \leftarrow ResidentBird(Jo), MigratoryBird(Jo) \quad [-10 @ 0] \\
\bot & \leftarrow ResidentBird(Jo) \quad [-2 @ 0] \\
\bot & \leftarrow MigratoryBird(Jo) \quad [-1 @ 0]
\end{align*}
\]

Also note that while the most probable stable models of \( \Pi \) and the optimal stable models of \( lpmln2wc(\Pi) \) 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 LP^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 \( \Pi \), let \( mln2wc(\Pi) \) be the union of \( lpmln2wc(\Pi) \) plus choice rules \( \{A\}^{ch} \) for all atoms \( A \).

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

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 LP^MLN (Theorem 4 from (Lee and Wang 2016)) and the reduction of Causal Models to LP^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 \( lpmln2wc^{\min}(\Pi) \) be a modification of \( lpmln2wc(\Pi) \) by replacing (1) with

\[\neg F \quad [1 @ 1]\]

and (2) with

\[\neg F \quad [w @ 0].\]

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 \( lpmln2wc(\Pi) \) is replaced by \( lpmln2wc^{\min}(\Pi) \).

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

---

1Recall 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 $lpmln2wc^{\text{pmt, rule}}(\Pi)$ be the translation by turning each weighted formula $w_i : F_i$ in $\Pi$ into

$$\neg F_i \rightarrow \text{unsat}(i)$$

$$\neg \text{unsat}(i) \rightarrow F_i$$

$$\therefore \text{unsat}(i) \ [w_i @ l].$$

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

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

The corollary allows us to compute the most probable stable models (MAP estimates) of the LP$^{\text{MLN}}$ 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$^{\text{MLN}}$ program is already in the rule form $\text{Head} \leftarrow \text{Body}$ that is allowed in the input languages of CLINGO and DLV, we may avoid the use of F2LP by slightly modifying the translation $lpmln2wc^{\text{pmt, rule}}$ by turning each weighted rule $w_i : \text{Head}_i \leftarrow \text{Body}_i$ instead into

$$\text{unsat}(i) \leftarrow \text{Body}_i, \not\text{Head}_i$$

$$\text{Head}_i \leftarrow \text{Body}_i, \not\text{unsat}(i)$$

$$\therefore \text{unsat}(i) \ [w_i @ l].$$

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

In the case when $\text{Head}_i$ is $\bot$, the translation can be further simplified: we simply turn $w_i : \bot \leftarrow \text{Body}_i$ into

$$\therefore \text{Body}_i \ [w_i @ l].$$

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

$$\begin{align*}
\text{unsat}(1) \leftarrow p, \not q & \quad q \leftarrow p, \not \text{unsat}(1) \quad \therefore \text{unsat}(1) \ [10@0] \\
\text{unsat}(2) \leftarrow p, \not r & \quad r \leftarrow p, \not \text{unsat}(2) \quad \therefore \text{unsat}(2) \ [1@0] \\
\text{unsat}(3) \leftarrow \not p & \quad p \leftarrow \not \text{unsat}(3) \quad \therefore \text{unsat}(3) \ [5@0] \\
\end{align*}$$

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

**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 $\text{Range}(c)$), where each of $s_0, \ldots, s_n$ 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 $c : s_1 \times \cdots \times s_n \to s_0$, and $\vec{u} \in s_1 \times \cdots \times s_n$, and $v \in s_0$. Symbol $c(\vec{u})$ is called an attribute and $v$ is called its value. If the range $s_0$ of $c$ is $\{f, t\}$ then $c$ is called Boolean, and $c(\vec{u}) = t$ can be abbreviated as $c(\vec{u})$ and $c(\vec{u}) = f$ as $\neg c(\vec{u})$.

The signature of a P-log program is the union of two propositional signatures $\sigma_1$ and $\sigma_2$, where $\sigma_1$ is a sorted propositional signature, and $\sigma_2$ is a usual propositional signature consisting of atoms $\text{Do}(c(\vec{u}) = v)$, $\text{Obs}(c(\vec{u}) = v)$ and $\text{Obs}(c(\vec{u}) \neq v)$ for all atoms $c(\vec{u}) = v$ in $\sigma_1$.

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

$$\Pi = (R, S, P, \text{Obs, Act})$$

where the signature of each of $R$, $S$, and $P$ is $\sigma_1$ and the signature of each of $\text{Obs}$ and $\text{Act}$ is $\sigma_2$ such that

- **R** is a set of normal rules of the form

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

where $A, B_1, \ldots, B_n$ are atoms $(0 \leq m \leq n)$.

- **S** is a set of random selection rules of the form

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

where $r$ is a unique identifier, $p$ is a boolean constant with a unary argument, and $\text{Body}$ 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 $\text{Body}$ is true, the value of $c(\vec{u})$ is selected at random from the set $\text{Range}(c) \cap \{x : p(x)\}$ unless this value is fixed by a deliberate action, i.e., $\text{Do}(c(\vec{u}) = v)$ for some value $v$.

- **P** is a set of so-called probability atoms (pr-atoms) of the form

$$\text{pr}_r(c(\vec{u}) = v \mid C) = p$$

where $r$ is the identifier of some random selection rule for $c(\vec{u})$ in $S$; $c(\vec{u}) = v \in \sigma_1$; $C$ 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 $r$.

- **Obs** is a set of atomic facts for representing “observation”: $\text{Obs}(c(\vec{u}) = v)$ and $\text{Obs}(c(\vec{u}) \neq v)$.

- **Act** is a set of atomic facts for representing a deliberate action: $\text{Do}(c(\vec{u}) = v)$.

**Semantics** Let $\Pi$ be a P-log program (4) of signature $\sigma_1 \cup \sigma_2$. The possible worlds of $\Pi$, denoted by $\omega(\Pi)$, are the stable models of $\tau(\Pi)$, a (standard) ASP program with the propositional signature

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

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 $\tau(\Pi)$.

---

⁴http://reasoning.eas.asu.edu/f2lp/
⁵http://potassco.sourceforge.net
⁶Alternatively, we may turn it into the “reward” way, i.e., turning it into $\therefore \not\text{Body}_i[\not w_i]$, but the rule may not be in the input language of CLINGO.

⁷Note that here “$\not=$” 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 $W \models \text{Body} \land p(v) \land \neg \text{Intervene}(c(\vec{u})). \quad (5)$ 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 II satisfy the following conditions:

- **Condition 1 [Unique random selection rule]:** If a P-log program II contains two random selection rules for $c(\vec{u})$:
  
  \begin{align*}
  [r_1] & \quad \text{random}(c(\vec{u}) : \{x : p_1(x)\}) \leftarrow \text{Body}_1, \\
  [r_2] & \quad \text{random}(c(\vec{u}) : \{x : p_2(x)\}) \leftarrow \text{Body}_2,
  \end{align*}

  then no possible world of II satisfies both $\text{Body}_1$ and $\text{Body}_2$.

- **Condition 2 [Unique probability assignment]:** If a P-log program II contains a random selection rule for $c(\vec{u})$:
  
  \begin{align*}
  [r] & \quad \text{random}(c(\vec{u}) : \{x : p(x)\}) \leftarrow \text{Body}
  \end{align*}

  along with two different pr-atoms:

  \begin{align*}
  pr_r(c(\vec{u}) = v | C_1) &= p_1, \\
  pr_r(c(\vec{u}) = v | C_2) &= p_2,
  \end{align*}

  then no possible world of II satisfies $\text{Body}$, $C_1$, and $C_2$ together.

Given a P-log program II, a possible world $W \in \omega(II)$, 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 $W \models \text{Body}$. Let $W_{rW,c(\vec{u})}$ denote this random selection rule, and let $AV_W(c(\vec{u})) = \{v' | \text{there exists a pr-atom } pr_{rW,c(\vec{u})}(c(\vec{u}) = v' | C) = p \text{ that is applied in } W \text{ for some } C \text{ and } p \}$.

We then define the following notations:

- If $v \in AV_W(c(\vec{u}))$, there exists a pr-atom $pr_{rW,c(\vec{u})}(c(\vec{u}) = v | C) = p$ in II for some $C$ and $p$ such that $W \models C$. By **Condition 2**, for any other $pr_{rW',c(\vec{u})}(c(\vec{u}) = v | C') = p'$ in II, it follows that $W \notmodels C'$. So there is only one pr-atom that is applied in $W$ for $c(\vec{u}) = v$, and we define

  \begin{align*}
  \text{PossWithAssPr}(W, c(\vec{u}) = v) &= p.
  \end{align*}

  ("$c(\vec{u}) = v$ is possible in $W$ with assigned probability $p$")

- If $v \not\in AV_W(c(\vec{u}))$, we define

  \begin{align*}
  \text{PossWithDefPr}(W, c(\vec{u}) = v) &= \max \{p, 0\},
  \end{align*}

  where $p$ is

  \begin{align*}
  1 - \sum_{v' \in AV_W(c(\vec{u}))} \text{PossWithAssPr}(W, c(\vec{u}) = v')
  \end{align*}

  \begin{align*}
  \left|\{v'' | c(\vec{u}) = v'' \text{ is possible in } W \text{ and } v'' \not\in AV_W(c(\vec{u}))\}\right|^1.
  \end{align*}

  ("$c(\vec{u}) = v$ is possible in $W$ with the default probability.")

The max function is used to ensure that the default probability is nonnegative. \footnote{Note that this is slightly different from the original definition of P-log from (Baral, Gelfond, and Rushton 2009), according to which, if Intervene(c(\vec{u})) is true, the probability of c(\vec{u}) = v is determined by the default probability, which is a bit unintuitive.}

For each possible world $W \in \omega(II)$, and each atom $c(\vec{u}) = v$ possible in $W$, the probability of $c(\vec{u}) = v$ to happen in $W$ is defined as:

\begin{align*}
P(W, c(\vec{u}) = v) &= \left\{ \begin{array}{ll}
\text{PossWithAssPr}(W, c(\vec{u}) = v) & \text{if } v \in AV_W(c(\vec{u})); \\
\text{PossWithDefPr}(W, c(\vec{u}) = v) & \text{otherwise}.
\end{array} \right.
\end{align*}

The unnormalized probability of a possible world $W$ is defined as

\begin{align*}
\hat{\mu}_W(W) &= \prod_{c(\vec{u}) \in W} P(W, c(\vec{u}) = v),
\end{align*}

and, assuming II has at least one possible world with nonzero unnormalized probability, the normalized probability of $W$ is defined as

\begin{align*}
\mu_W(W) &= \frac{\hat{\mu}_W(W)}{\sum_{W, \in \omega(II)} \hat{\mu}_W(W)}.
\end{align*}

We say II is consistent if II 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 II, using both assigned probability and default probability, as

\begin{align*}
\neg \text{CanOpen}(d) &\leftarrow \text{Selected} = d. \quad (d \in \{1, 2, 3, 4\}), \\
\neg \text{CanOpen}(d) &\leftarrow \text{Prize} = d. \\
\text{CanOpen}(d) &\leftarrow \text{not} \neg \text{CanOpen}(d). \\
\text{random(Prize)}. &\quad \text{random(Selected)}. \\
\text{random(Open) : (x : CanOpen(c))}. \\
pr(\text{Prize} = 1) &= 0.3, \quad pr(\text{Prize} = 3) = 0.2. \\
\text{Obs(Selected} \leftarrow 1), \quad \text{Obs(Open} \leftarrow 2), \quad \text{Obs(Prize} \leftarrow 2).
\end{align*}

The possible worlds of II are as follows:

- $W_1 = \{\text{Obs(Selected} \leftarrow 1), \text{Obs(Open} \leftarrow 2), \text{Obs(Prize} \leftarrow 2), \text{Selected} \leftarrow 1, \text{Open} \leftarrow 2, \text{Prize} \leftarrow 1, \text{CanOpen(1)} = \text{t}, \text{CanOpen(2)} = \text{t}, \text{CanOpen(3)} = \text{t}, \text{CanOpen(4)} = \text{t}\}$
- $W_2 = \{\text{Obs(Selected} \leftarrow 1), \text{Obs(Open} \leftarrow 2), \text{Obs(Prize} \leftarrow 2), \text{Selected} \leftarrow 1, \text{Open} \leftarrow 2, \text{Prize} \leftarrow 3, \text{CanOpen(1)} = \text{t}, \text{CanOpen(2)} = \text{t}, \text{CanOpen(3)} = \text{t}, \text{CanOpen(4)} = \text{t}\}$
- $W_3 = \{\text{Obs(Selected} \leftarrow 1), \text{Obs(Open} \leftarrow 2), \text{Obs(Prize} \leftarrow 2), \text{Selected} \leftarrow 1, \text{Open} \leftarrow 2, \text{Prize} \leftarrow 4, \text{CanOpen(1)} = \text{t}, \text{CanOpen(2)} = \text{t}, \text{CanOpen(3)} = \text{t}, \text{CanOpen(4)} = \text{t}\}$

The probability of each atom to happen is

\begin{align*}
P(W_1, \text{Selected} \leftarrow 1) &= \text{PossWithDefPr}(W_1, \text{Selected} \leftarrow 1) = 1/4 \quad (6) \\
P(W_1, \text{Open} \leftarrow 2) &= \text{PossWithDefPr}(W_1, \text{Open} \leftarrow 2) = 1/3 \\
P(W_2, \text{Open} \leftarrow 2) &= \text{PossWithDefPr}(W_2, \text{Open} \leftarrow 2) = 1/2 \\
P(W_3, \text{Open} \leftarrow 2) &= \text{PossWithDefPr}(W_3, \text{Open} \leftarrow 2) = 1/2 \quad (7)
\end{align*}

\begin{align*}
P(W_1, \text{Prize} \leftarrow 1) &= \text{PossWithAssPr}(W_1, \text{Prize} \leftarrow 1) = 0.3 \\
P(W_2, \text{Prize} \leftarrow 3) &= \text{PossWithAssPr}(W_2, \text{Prize} \leftarrow 3) = 0.2 \\
P(W_3, \text{Prize} \leftarrow 4) &= \text{PossWithDefPr}(W_3, \text{Prize} \leftarrow 4) = 0.25
\end{align*}

So,
• $\mu(W_1) = 1/4 \times 1/3 \times 0.3 = 1/40$
• $\mu(W_2) = 1/4 \times 1/2 \times 0.2 = 1/40$
• $\mu(W_3) = 1/4 \times 1/2 \times 0.25 = 1/32$.
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 LP$^{MLN}$
We define translation plog2pmln($\Pi$) that turns a P-log program $\Pi$ into an 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 plog2pmln($\Pi$). In addition, plog2pmln($\Pi$) contains the following rules to associate probability to each possible world of $\Pi$. Below $x$, $y$ denote schematic variables, and $W$ is a possible world of $\Pi$.

Possible Atoms: For each random selection rule (5) for $c(\overline{u})$ in $\mathcal{S}$ and for each $v \in \text{Range}(c)$, plog2pmln($\Pi$) includes

$$\text{Poss}_r(c(\overline{u}) = v) \leftarrow \text{Body}, p(v), \text{not Intervene}(c(\overline{u}))$$ (8)

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

Assigned Probability: For each pr-atom (6) in $\mathcal{P}$, plog2pmln($\Pi$) contains the following rules:

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

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

$$\alpha : \text{AssPr}_{r,C}(c(\overline{u}) = v) \leftarrow \text{PossWithAssPr}_{r,C}(c(\overline{u}) = v)$$ (11)

$$\alpha : \text{PossWithAssPr}_{r,C}(c(\overline{u}) = v) \leftarrow \text{PossWithAssPr}_{r,C}(c(\overline{u}) = v)$$ (12)

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

Denominator for Default Probability: For each random selection rule (5) for $c(\overline{u})$ in $\mathcal{S}$ and for each $v \in \text{Range}(c)$, plog2pmln($\Pi$) includes

$$\alpha : \text{PossWithDefPr}(c(\overline{u}) = v) \leftarrow \text{Poss}_r(c(\overline{u}) = v), \text{not PossWithAssPr}(c(\overline{u}) = v)$$ (13)

$$\alpha : \text{NumDefPr}(c(\overline{u}), x) \leftarrow c(\overline{u}) = v, \text{PossWithDefPr}(c(\overline{u}) = v), x = \#\text{count}(y : \text{PossWithDefPr}(c(\overline{u}) = y))$$ (14)

Rule (12) asserts that $c(\overline{u}) = v$ 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 $\text{NumDefPr}(c(\overline{u}), x)$ is true if there are exactly $x$ different values $v$ such that $c(\overline{u}) = v$ 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$(c(\overline{u})) : \{x : p(x)\}) \leftarrow \text{Body}$ for $c(\overline{u})$ in $\mathcal{S}$ along with all pr-atoms associated with it in $\mathcal{P}$:

$$p_r(c(\overline{u}) = v_1 | C_1) = p_1$$
$$\ldots$$
$$p_r(c(\overline{u}) = v_n | C_n) = p_n$$

where $n \geq 1$, and $v_1$ and $v_j$ ($i \neq j$) may be equal. For each $v \in \text{Range}(c)$, plog2pmln($\Pi$) contains the following rules:

$$\alpha : \text{DomPr}(c(\overline{u}), 1 - y) \leftarrow \text{Body}$$

$$c(\overline{u}) = v, \text{PossWithDefPr}(c(\overline{u}) = v), y = \#\text{sum}(p_1 : \text{PossWithAssPr}_{r,C_1}(c(\overline{u}) = v_1); \ldots ; p_n : \text{PossWithAssPr}_{r,C_n}(c(\overline{u}) = v_n)).$$ (15)

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-negative default probability to a possible world.

Note that most rules in plog2pmln($\Pi$) are hard rules. The soft rules (11), (14), (17) cannot be simplified as atomic facts, e.g., $\ln\left(\frac{1}{m}\right) : \text{NumDefPr}(c(\overline{u}), m)$ 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 world 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 LP$^{MLN}$ (Lee and Wang 2016), it is interesting to note that both kinds of probabilities can be captured in LP$^{MLN}$ using different kinds of rules.

Example 3 Continued For the program $\Pi$ in Example 3, plog2pmln($\Pi$) consists of the rules $\alpha : R$ for each rule $R$ in $\tau(\Pi)$ and the following rules.

Possible Atoms:

$$\alpha : \text{Poss}(\text{Price} = d) \leftarrow \neg \text{Intervene}(\text{Price})$$

$$\alpha : \text{Poss}(<\text{Selected} = d) \leftarrow \neg \text{Intervene}(\text{Selected})$$

$$\alpha : \text{Poss}(\text{Open} = d) \leftarrow \text{CanOpen}(d), \neg \text{Intervene}(\text{Open})$$

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:

\[ \alpha : \text{PossWithAssPr}(\text{Prize} = 1) \leftarrow \text{Poss}(\text{Prize} = 1) \]
\[ \alpha : \text{AssPr}(\text{Prize} = 1) \leftarrow \text{Prize} = 1, \text{PossWithAssPr}(\text{Prize} = 1) \ln(0.3) : \quad \perp \not\rightarrow \text{not AssPr}(\text{Prize} = 1) \]
\[ \alpha : \text{PossWithAssPr}(\text{Prize} = 3) \leftarrow \text{Poss}(\text{Prize} = 3) \]
\[ \alpha : \text{AssPr}(\text{Prize} = 3) \leftarrow \text{Prize} = 3, \text{PossWithAssPr}(\text{Prize} = 3) \ln(0.2) : \quad \perp \not\rightarrow \text{not AssPr}(\text{Prize} = 3) \]

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

Denominator for Default Probability:

\[ \alpha : \text{PossWithDefPr}(\text{Prize} = d) \leftarrow \text{Poss}(\text{Prize} = d), \text{not PossWithAssPr}(\text{Prize} = d) \]
\[ \alpha : \text{PossWithDefPr}(\text{Selected} = d) \leftarrow \text{Poss}(\text{Selected} = d), \text{not PossWithAssPr}(\text{Selected} = d) \]
\[ \alpha : \text{PossWithDefPr}(\text{Open} = d) \leftarrow \text{Poss}(\text{Open} = d), \text{not PossWithAssPr}(\text{Open} = d) \]
\[ \alpha : \text{NumDefPr}(\text{Prize}, x) \leftarrow \text{Prize} = d, \text{PossWithDefPr}(\text{Prize} = d), x = \#\text{count}\{y : \text{PossWithDefPr}(\text{Prize} = y)\} \]
\[ \alpha : \text{NumDefPr}(\text{Selected}, x) \leftarrow \text{Selected} = d, \text{PossWithDefPr}(\text{Selected} = d), x = \#\text{count}\{y : \text{PossWithDefPr}(\text{Selected} = y)\} \]
\[ \alpha : \text{NumDefPr}(\text{Open}, x) \leftarrow \text{Open} = d, \text{PossWithDefPr}(\text{Open} = d), x = \#\text{count}\{y : \text{PossWithDefPr}(\text{Open} = y)\} \]
\[ \ln(\frac{1}{m}) : \quad \perp \not\rightarrow \text{not NumDefPr}(c, m) \]
\[ (c \in \{\text{Prize, Selected, Open}\}, m \in \{2, 3, 4\}) \]

Numerator for Default Probability:

\[ \alpha : \text{RemPr}(\text{Prize}, 1 - x) \leftarrow \text{Prize} = d, \text{PossWithDefPr}(\text{Prize} = d), x = \#\text{sum}\{0.3 : \text{PossWithAssPr}(\text{Prize} = 1)\} \]
\[ 0.2 : \text{PossWithAssPr}(\text{Prize} = 3) \]
\[ \alpha : \text{TotalDefPr}(\text{Prize}, x) \leftarrow \text{RemPr}(\text{Prize}, x), x > 0 \]
\[ \ln(x) : \quad \perp \not\rightarrow \text{not TotalDefPr}(\text{Prize}, x) \]
\[ \alpha : \quad \perp \not\rightarrow \text{RemDefPr}(\text{Prize}, x), x \leq 0 \]

Clearly, the signature of plog2pmln(II) is a superset of the signature of II. Further, plog2pmln(II) 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 II and the set of the stable models of plog2pmln(II) such that each stable model is an extension of the possible world, and the probability of each possible world of II coincides with the probability of the corresponding stable model of plog2pmln(II).

Theorem 3 Let II be a consistent P-log program. There is a 1-1 correspondence \( \phi \) between the set of the possible worlds of II with non-zero probabilities and the set of probabilistic stable models of plog2pmln(II) such that

- For every possible world W of II that has a non-zero probability, \( \phi(W) \) is a probabilistic stable model of plog2pmln(II), and \( \mu_{II}(W) = P_{\text{plog2pmln(II)}}(\phi(W)) \).
- For every probabilistic stable model I of plog2pmln(II), the restriction of I onto the signature of II, denoted \( I_{\sigma(\Pi)} \), is a possible world of II and \( \mu_{II}(I_{\sigma(\Pi)}) > 0 \).

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

1. \( \phi(W) \models \text{Poss}_{r,c}(\vec{u}) = v \) 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 II, \( \phi(W) \models \text{PossWithAssPr}_{r,C}(c(\vec{u}) = v) \) iff this pr-atom is applied in W.
3. For each pr-atom \( pr_r(c(\vec{u}) = v \mid C) = p \) in II, \( \phi(W) \models \text{AssPr}_{r,C}(c(\vec{u}) = v) \) iff this pr-atom is applied in W, and \( W \models \text{c} = \vec{u} \).
4. \( \phi(W) \models \text{PossWithDefPr}(c(\vec{u}) = v) \) iff \( v \in AV_W(c(\vec{u})) \).
5. \( \phi(W) \models \text{PossWithDefPr}(c(\vec{u}) = v) \) iff \( c(\vec{u}) = v \) is possible in W and \( v \not\in AV_W(c(\vec{u})) \).
6. \( \phi(W) \models \text{NumDefPr}(c(\vec{u}), m) \) iff there exist exactly \( m \) different values \( v \) such that \( c(\vec{u}) = v \) is possible in W; \( v \not\in AV_W(c(\vec{u})) \); and, for one of such \( v \), \( W \models c(\vec{u}) = v \).
7. \( \phi(W) \models \text{RemPr}(c(\vec{u}), k) \) iff there exists a value \( v \) such that \( W \models c(\vec{u}) = v; c(\vec{u}) = v \) is possible in W; \( v \not\in AV_W(c(\vec{u})) \); and
\[ k = 1 - \sum_{v \in AV_W(c(\vec{u}))} \text{PossWithAssPr}(W, c(\vec{u}) = v). \]
8. \( \phi(W) \models \text{TotalDefPr}(c(\vec{u}), k) \) iff \( \phi(W) \models \text{RemPr}(c(\vec{u}, k)) \) and \( k > 0 \).

To check that \( \mu_{II}(W) = P_{\text{plog2pmln(II)}}(\phi(W)) \), 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 II and \( c(\vec{u}) = v \) that is satisfied by II. 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, \( \phi(W) \models \text{AssPr}_{r,C}(c(\vec{u}) = v) \), so that from (11), the same \( p \) is a factor of \( TW \).

If \( c(\vec{u}) = v \) is possible in W and \( v \not\in AV_W(c(\vec{u})) \), the default probability is applied: \( P(W, c(\vec{u}) = v) = p \) is computed by (7). By Condition 8, \( \phi(W) \models \text{TotalDefPr}(c(\vec{u}), x) \) where \( x = 1 - \sum_{v \in AV_W(c(\vec{u}))} \text{PossWithAssPr}(W, c(\vec{u}) = v') \).

Since \( \phi(W) \models (17) \), it is a factor of \( TW \), which is the same as the numerator of (7). Furthermore, by Condition 6, \( \phi(W) \models \text{NumDefPr}(c(\vec{u}), m) \), 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 II for the Monty Hall problem, II = plog2pmln(II) 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 $i = 1, 3$: $\text{PossWithDefPr}(\text{Prize} = i)$ for $i = 2, 4$; $\text{PossWithDefPr}(\text{Selected} = i)$ for $i = 1, 2, 3, 4$; $\text{NumDefPr}(\text{Selected}, 4)$. In addition,

- $I_1 \models \{ \text{AssPr}(\text{Prize} = 1), \text{Poss}(\text{Open} = 2), \\
\text{Poss}(\text{Open} = 3), \text{Poss}(\text{Open} = 4), \\
\text{PossWithDefPr}(\text{Open} = 2), \text{PossWithDefPr}(\text{Open} = 3), \\
\text{PossWithDefPr}(\text{Open} = 4), \text{NumDefPr}(\text{Open}, 3) \}$

- $I_2 \models \{ \text{AssPr}(\text{Prize} = 3), \text{Poss}(\text{Open} = 2), \\
\text{Poss}(\text{Open} = 4), \text{PossWithDefPr}(\text{Open} = 2), \\
\text{PossWithDefPr}(\text{Open} = 4), \text{NumDefPr}(\text{Open}, 2) \}$

- $I_3 \models \{ \text{Poss}(\text{Open} = 2), \text{Poss}(\text{Open} = 3), \\
\text{PossWithDefPr}(\text{Open} = 2), \text{PossWithDefPr}(\text{Open} = 3), \\
\text{NumDefPr}(\text{Open}, 2), \text{NumDefPr}(\text{Prize}, 2), \\
\text{RemPr}(\text{Prize}, 0.5), \text{TotalDefPr}(\text{Prize}, 0.5) \}$

The unnormalized weight $W_{\text{II}}(I_i)$ of each probabilistic stable model $I_i$ is shown below. $w(\text{AssPr}_{c}(c(i) = v))$ denotes the exponentiated weight of rule (11); $w(\text{NumDefPr}(c(i), m))$ denotes the exponentiated weight of rule (14); $w(\text{TotalDefPr}(c(i), x))$ denotes the exponentiated weight of rule (17).

- $W_{\text{II}}(I_1) = w(\text{NumDefPr}(\text{Selected}, 4)) \times \\
w(\text{AssPr}(\text{Prize} = 1)) \times w(\text{NumDefPr}(\text{Open}, 3)) = \\
\frac{1}{3} \times \frac{4}{10} \times \frac{5}{10} = \frac{2}{15}$.

- $W_{\text{II}}(I_2) = w(\text{NumDefPr}(\text{Selected}, 4)) \times \\
w(\text{AssPr}(\text{Prize} = 3)) \times w(\text{NumDefPr}(\text{Open}, 2)) = \\
\frac{1}{4} \times \frac{2}{3} \times \frac{1}{2} = \frac{1}{12}$.

- $W_{\text{II}}(I_3) = w(\text{NumDefPr}(\text{Selected}, 4)) \times \\
x w(\text{TotalDefPr}(\text{Open}, 2)) \times \times w(\text{NumDefPr}(\text{Prize}, 2) \times \\
w(\text{TotalDefPr}(\text{Prize}, 0.5)) = \frac{1}{4} \times \frac{1}{2} \times \frac{5}{10} = \frac{1}{8}$.

Combining the translations plog2pmln and pmln2wc, one can compute P-log MAP inference using standard ASP solvers.

**Conclusion**

In this paper, we show how L$\text{P}^{\text{MLN}}$ 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. L$\text{P}^{\text{MLN}}$ 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**


