Handling Preferences in LP$^{\mathrm{MLN}}$: A Preliminary Report

Bin Wang[1], Shutao Zhang[1], Hongxiang Xu[1], Zhizheng Zhang[1], Wei Wu[2], and Xiaodong Li[2]

[1] School of Computer Science and Engineering
Southeast University, Nanjing, China
{kse.wang, shutao_zhang, xhx1693, seu_zzz}@seu.edu.cn
[2] Science and Technology on Information System Engineering Lab
The 28th Research Institute of China Electronics Technology Group Corparation
Nanjing, China
xmuww2004@126.com

Abstract. In this paper, we present a new knowledge representation and reasoning tool to handle uncertainty, inconsistencies, and preferences by combining the ideas of LP$^{\mathrm{MLN}}$ and logic programming with ordered disjunction (LPOD). The former is an extension of Answer Set Programming (ASP) that is designed for handling uncertainty and inconsistencies in knowledge representation by incorporating the method in Markov Logic Networks (MLN). The latter is a logic formalism to represent preferences in ASP, which is a simple yet effective way to handle preferences. Firstly, we present a new knowledge representation language o-LP$^{\mathrm{MLN}}$ that extends LP$^{\mathrm{MLN}}$ by introducing ordered disjunctions. Then, we present an example to show the use of the language. Finally, we present an approach to computing o-LP$^{\mathrm{MLN}}$ programs by translating them into regular LP$^{\mathrm{MLN}}$ programs. Results obtained in this paper provide an alternative tool to handle inconsistencies, uncertainty, and preferences in a unified framework. Moreover, a by-product of the paper is that we present an approach to implementing LPOD via using LP$^{\mathrm{MLN}}$ solvers.

Keywords: LP$^{\mathrm{MLN}}$ · LPOD · Preferences

1 Introduction

LP$^{\mathrm{MLN}}$ [12] is an extension of Answer Set Programming (ASP) [10], which is designed to handle uncertainty and inconsistencies in knowledge representation by incorporating the methods in Markov Logic Networks [16]. Intuitively, an LP$^{\mathrm{MLN}}$ program extended ASP by assigning each rule a weight degree, which means an ASP rule can be violated with a loss of the weight. Therefore, a stable model of an LP$^{\mathrm{MLN}}$ program can only satisfy a subprogram, and a stable model is a better solution if the sum of the weight degrees of rules satisfied by the stable model is greater than that of another stable model. Through this way, uncertain and inconsistent knowledge can be encoded and computed in the framework of LP$^{\text{MLN}}$. So far, there is few effort that studies the applications of LP$^{\text{MLN}}$, most of the researchers focus on the implementations and theoretical properties of LP$^{\text{MLN}}$ [12, 1, 13, 11, 17, 18].

In many real-world applications, preferences are inevitable. For example, it has been shown that many problems can be formulated as a qualitative optimization problem [4] in ASP, by which the solutions of the problems are associated with the optimal or suboptimal stable models of corresponding ASP programs. To handle preferences, large body of extensions of ASP has been presented, even LP$^{\text{MLN}}$ can be viewed as such an extension. For example, "a is preferred to b" can be trivially encoded by following LP$^{\text{MLN}}$ program M

$$2 : a. \tag{1}$$
$$1 : b. \tag{2}$$

Among these extensions, logic programs with ordered disjunction (LPOD) [3] is a simple yet effective one by introducing ordered disjunction in the head of a regular ASP rule (called LPOD rule). An LPOD rule "$a \times b \leftarrow body$" means if X and Y are the stable models satisfying "$body$" such that $a \in X$, $a \notin Y$ and $b \in Y$, then X is preferred to Y w.r.t. the rule. The other meaning behind the LPOD rule is that if we believe "$body$" and "a" are true, then we are allowed but not necessarily to believe "b" is true in the same stable model, which makes the LPOD different from LP$^{\text{MLN}}$. For example, the same knowledge "a is preferred to b" can be encoded by following LPOD program P

$$a \times b. \tag{3}$$

It is easy to check that $X = \{a\}$ and $Y = \{b\}$ are two (candidate) stable models of P, but $Z = \{a, b\}$ is not a stable model of P, and X is preferable to Y w.r.t. P. By contrast, we can observe that all of X, Y and Z are stable models of above LP$^{\text{MLN}}$ program M, and the weight degree of Z w.r.t. M is higher than others. Obviously, M cannot capture the intuition of P, which motivates us to present a new language o-LP$^{\text{MLN}}$ by combining LP$^{\text{MLN}}$ and LPOD.

In this paper, we present a new language o-LP$^{\text{MLN}}$, which can be regarded as an extension of LP$^{\text{MLN}}$ and LPOD. Since there are several usable LP$^{\text{MLN}}$ solvers, the extension is expected to be a powerful knowledge representation and reasoning tool for handling uncertainty, inconsistencies, and preferences. Firstly, we define the syntax and semantics of the language o-LP$^{\text{MLN}}$. Secondly, we present an example to show the use of the language o-LP$^{\text{MLN}}$. Thirdly, we present a modular and linear-time constructible translation from o-LP$^{\text{MLN}}$ programs to regular LP$^{\text{MLN}}$ programs, which provides an alternative approach to implementing o-LP$^{\text{MLN}}$ by using existing LP$^{\text{MLN}}$ solvers [11, 17]. As a by-product, the translation can also be used to implement LPOD. Finally, we discuss some related and further work of the paper.

2 Preliminaries

In this section, we review the language of LP$^{\text{MLN}}$ and LPOD. Note that throughout this paper, we consider only finite ground logic programs, that is, logic programs containing no variables. And we assume the reader is familiar with the notions in ASP.

2.1 LP$^{\text{MLN}}$

An LP$^{\text{MLN}}$ program is a finite set of weighted rules $w : r$, where w is either a symbol α denoting the "infinite weight" or a real number, and r is an ASP rule of the form

$$h_1 \ \vee \ ... \ \vee \ h_k \ \leftarrow \ b_1, ..., b_m, \ not \ c_1, ..., \ not \ c_n. \tag{4}$$

where hs, bs and cs are literals, $\vee$ is epistemic disjunction, and *not* is default negation. For an ASP rule r of the form (4), the head of r is $head(r) = \{h_i \mid 1 \leq i \leq k\}$, the positive body of r is $body^+(r) = \{b_i \mid 1 \leq i \leq m\}$, the negative body of r is $body^-(r) = \{c_i \mid 1 \leq i \leq n\}$, and the literals occurred in r is $lit(r) = head(r) \cup body^+(r) \cup body^-(r)$. Therefore, an ASP rule r of the form (4) can also be abbreviated as "$head(r) \leftarrow body^+(r), not \ body^-(r)$" or "$head(r) \leftarrow body(r)$". An LP$^{\text{MLN}}$ rule $w : r$ is called soft if w is a real number, and hard if w is α. By M^s and M^h we denote the sets of all soft rules and hard rules in an LP$^{\text{MLN}}$ program M respectively. By $\overline{M}$ we denote the set of unweighted ASP counterpart of an LP$^{\text{MLN}}$ program M, i.e. $\overline{M} = \{r \mid w : r \in M\}$. An LP$^{\text{MLN}}$ program is called *ground* if its rules contain no variables. For a ground LP$^{\text{MLN}}$ program M, we use $W(M)$ to denote the weight degree of M, i.e. $W(M) = exp\left(\sum_{w:r \in M} w\right)$.

A ground LP$^{\text{MLN}}$ rule $w : r$ is satisfied by a consistent set X of ground literals, denoted by $X \models w : r$, if $X \models r$ by the notion of satisfiability in ASP. An LP$^{\text{MLN}}$ program M is satisfied by X, denoted by $X \models M$, if X satisfies all rules in M. By M_X we denote the LP$^{\text{MLN}}$ reduct of an LP$^{\text{MLN}}$ program M w.r.t. X, i.e. $M_X = \{w : r \in M \mid X \models w : r\}$. X is a stable model of the program M if X is a stable model of $\overline{M_X}$ and $M^h \subseteq M_X$. And by $SM(M)$ we denote the set of all stable models of an LP$^{\text{MLN}}$ program M. For an interpretation I, the weight degree $W(M, I)$ of I w.r.t. M is defined as

$$W(M, I) = W(M_I^s) \tag{5}$$

and if $I \in SM(M)$, the probability degree $P(M, I)$ of I w.r.t. M is defined as

$$P(M, I) = \frac{W(M, I)}{\Sigma_{X' \in SM(M)} W(M, X')} \tag{6}$$

For a literal l, the probability degree $P(M, l)$ of l w.r.t. M is defined as

$$P(M, l) = \sum_{l \in X, X \in SM(M)} P(M, X) \tag{7}$$

Note that in its original definitions [12], a stable model is allowed to violate some hard rules, which is slightly different from our defintions here but does not affect the generality of the results obtained in the paper.

2.2 LPOD

Following Lee and Yang's point [14], we distinguish regular rules and LPOD rules in a logic program with ordered disjunction (LPOD). An LPOD P consists of two parts: the regular part P^r and the ordered disjunction part P^o, where P^r is an ASP program consisting of rules of the form (4), and P^o is a finite set of LPOD rules r of the form (8),

$$h_1 \times ... \times h_n \leftarrow \mathit{body}(r). \tag{8}$$

where hs ($n > 1$) are literals satisfying $h_i = h_j$ iff $i = j$. By $o(r)$ we denote the number of literals occurred in the head of an LPOD rule r, i.e. $o(r) = |\mathit{head}(r)|$. An LPOD rule r of the form (8) means if $\mathit{body}(r)$ is true, for any postive integers $i < j$, we prefer to believe h_i rather than h_j, and if we believe h_i, we are not necessary to believe h_j.

Definition 1. *For an LPOD rule r, its i-th option ($1 \leq i \leq o(r)$), denoted by r^i, is defined as*

$$h_i \leftarrow \mathit{body}(r),\ \mathit{not}\ h_1, ...,\ \mathit{not}\ h_{i-1}. \tag{9}$$

Definition 2. *A split program of an LPOD P is obtained from P by replacing each rule in P^o by one of its options.*

Based on the above definitions, the semantics of an LPOD P is defined as follows. A consistent set S of literals is a *candidate stable model* of P if it is a stable model of a split program of P. By $CSM(P)$ we denote the set of all candidate stable models of P. The *satisfaction degree* $\mathit{deg}(I, r)$ of an LPOD rule r w.r.t. an interpretation I is defined as

$$\mathit{deg}(I, r) = \begin{cases} 1, & \text{if } I \not\models \mathit{body}(r); \\ \mathit{min}\{k \mid h_k \in \mathit{head}(r) \cap I\}, & \mathit{otherwise}. \end{cases} \tag{10}$$

And the *satisfaction degree* $\mathit{deg}(I, P)$ of an LPOD P w.r.t. an interpretation I is defined as the sum of satisfaction degrees of LPOD rules in P w.r.t. I, i.e. $\mathit{deg}(I, P) = \sum_{r \in P^o} \mathit{deg}(I, r)$. For a candidate stable model S of P, by $S^i(P)$ we denote the set of LPOD rules in P^o that are satisfied by S at degree i. Based on the notion of satisfaction degree, for two candidate stable model X and Y of P, Brewka [5] introduces four preference criteria:

1. **Cardinality-Preferred:** X is cardinality-preferred to Y, denoted by $X >^c Y$, if there is a positive integer i such that $|X^i(P)| > |Y^i(P)|$, and $|X^j(P)| = |Y^j(P)|$ for all $j < i$;
2. **Inclusion-Preferred:** X is inclusion-preferred to Y, denoted by $X >^i Y$, if there is a positive integer i such that $Y^i(P) \subset X^i(P)$, and $X^j(P) = Y^j(P)$ for all $j < i$;
3. **Pareto-Preferred:** X is pareto-preferred to Y, denoted by $X >^p Y$, if there is a rule $r \in P_o$ such that $\mathit{deg}(X, r) < \mathit{deg}(Y, r)$, and there does not exist a rule $r \in P_o$ such that $\mathit{deg}(Y, r) < \mathit{deg}(X, r)$.

4. **Penalty-Sum-Preferred:** X is penalty-sum-preferred to Y, denoted by $X >^{ps} Y$, if $deg(P, X) < deg(P, Y)$.

For each $pr \in \{c, i, p, ps\}$, a candidate stable model X of P is a *pr*-preferred stable model if there is no candidate stable model Y of P such that $Y >^{pr} X$.

3 *o*-LP$^{\text{MLN}}$

To handle preferences in LP$^{\text{MLN}}$ naturally, we extend LP$^{\text{MLN}}$ by introducing ordered disjunctions in this section, the obtained language is called *o*-LP$^{\text{MLN}}$.

Syntactically, an *o*-LP$^{\text{MLN}}$ program M consists of two parts: the regular part M^r and the ordered disjunction part M^o, where M^r is a regular LP$^{\text{MLN}}$ program, and M^o consists of rules of the form (8) preceded by "α", called weighted ordered disjunctive rules (*wo*-rules for simplicity). A *wo*-rule is a hard rule, because we treat the preferences as some conditions that must be satisfied in solving problems. As a consequence, the preference criteria are used prior to certainty degrees in computing inference results of an *o*-LP$^{\text{MLN}}$ program, which will be shown later.

Similar to LPOD, we have the following definitions.

Definition 3. *For a wo-rule $\alpha : r$, its i-th option ($1 \leq i \leq o(r)$) is defined as $\alpha : r^i$, where r^i is the i-th option of rule r.*

Definition 4. *For an o-LP^{MLN} program M, a split program of M is obtained by replacing each rule in M^o by one of its options.*

Semantically, a consistent set X of literals is a candidate stable model of M, if X is a stable model of a split program of M. By $CSM(M)$ we denote the set of all candidate stable models of M. The satisfaction degree $deg(I, \alpha : r)$ of a *wo*-rule $\alpha : r$ w.r.t. an interpretation I and the satisfaction degree $deg(I, M)$ of an *o*-LP$^{\text{MLN}}$ program M w.r.t. I are defined the same as in LPOD. Therefore, the preferred stable models of an *o*-LP$^{\text{MLN}}$ program M are defined the same as in LPOD, that is, for each $pr \in \{c, i, p, ps\}$, a candidate stable model X of M is a *pr*-preferred stable model if there is no candidate stable models Y of M such that $Y >^{pr} X$. By *pr*-$SM(M)$ we denote the set of all *pr*-preferred stable models of M under a preference criterion $pr \in \{c, i, p, ps\}$.

In addition, the weight degree $W(M, X)$ of a candidate stable model X w.r.t. an *o*-LP$^{\text{MLN}}$ program M is defined the same as in LP$^{\text{MLN}}$, i.e. $W(M, X) = W(M^s_X)$. And the probability degree $P_{pr}(M, X)$ of a *pr*-preferred stable model X w.r.t. an *o*-LP$^{\text{MLN}}$ M is defined as follows

$$P_{pr}(M, X) = \frac{W(M, X)}{\sum_{Y \in pr\text{-}SM(M)} W(M, Y)} \tag{11}$$

where $pr \in \{c, i, p, ps\}$. Here, we do not define the probability degree for any candidate stable model for two main reasons. Firstly, preferred stable models are used to model intended solutions of a problem, while non-preferred stable models are usually discarded. Hence, considering the probability of all candidate stable models is not useful practically. Secondly, many candidate stable models may satisfy the same subprogram, which means the probability defined for these stable models cannot measure the "importance" of a stable model appropriately. For the same reason, we define the probability degree $P_{pr}(M, l)$ of a literal l w.r.t. an o-LP$^{\text{MLN}}$ program M as follows

$$P_{pr}(M, l) = \sum_{l \in X, X \in pr\text{-}SM(M)} P_{pr}(M, X) \tag{12}$$

Following the above definitions, the steps of computing the inference results of an o-LP$^{\text{MLN}}$ program M are as follows

1. compute all candidate stable models of M;
2. find all preferred stable models of M;
3. compute weight and probability degrees of all preferred stable models;
4. compute the probability degrees of literals w.r.t. M.

Now we use an example to show the use of o-LP$^{\text{MLN}}$ in handling uncertain and inconsistent knowledge with preferences.

Example 1. This example is from the Example 1 in [8], in which we use weight to measure the possibility of knowledge. A personal assistant agent has following knowledge (we assume the weight degree each soft knowledge ranges from 1 - 5.)

1. If I go to the cinema and it rains, then I will be not wet;
2. If I go to the beach and it rains, then I will be wet;
3. If I go to the cinema and there is sultriness, then I will be fresh;
4. If I go to the beach and there is sultriness, then I will be not fresh;
5. It is almost-certain that if it is humid, there will be sultriness (weight degree: 4);
6. It is quasi-certain that if it is cloudy, then it will rain (weight degree: 2);
7. The weather forcast said that today is a humid day, cloudy with weight degree 4, not cloudy with weight degree 1;
8. My preferecnes today are that going to beach is more important than being fresh; and being fresh is more important than being not wet.

The question is where should I go today? The above eight sentences can be encoded in the following o-LP$^{\text{MLN}}$ program M, in which w stand for wet, r for rain, h for humid, c for cloudy, go_c for going to cinema, go_b for going to beach, s for sultriness, f for being fresh.

$$\alpha : \neg w \leftarrow go\_c, r. \tag{r1}$$

$$\alpha : w \leftarrow go\_b, r. \tag{r2}$$

$$\alpha : f \leftarrow go\_c, s. \tag{r3}$$

$$\alpha : \neg f \leftarrow go\_b, s. \tag{r4}$$

$$4 : s \leftarrow h. \tag{r5}$$

$$2 : r \leftarrow c. \tag{r6}$$

$$\alpha : h. \tag{r7}$$

$$4 : c. \tag{r8}$$

$$1 : \neg c. \tag{r9}$$

$$\alpha : go\_b \times f \times \neg w. \tag{r10}$$

$$\alpha : go\_b \vee go\_c. \tag{r11}$$

$$\alpha : \leftarrow go\_b, go\_c. \tag{r12}$$

Each of rule $(r1)$ - $(r10)$ in M expresses some knowledge in above sentences, where $(r5)$ - $(r9)$ handle uncertain and inconsistent knowledge in a regular LP$^{\text{MLN}}$ style, and $(r10)$ handles preferences in an LPOD style. Rules $(r11)$ and $(r12)$ means "I have to go to cinema or beach". All candidate stable models of M are shown in Table 1. From the table, it is easy to check that the candidate stable models containing go_b (denoted by SM_b) are preferred to those containing f and $\neg w$ (denoted by SM_f and SM_{nw} respectively), i.e. for each $pr \in \{c, i, p, ps\}$, $SM_b >^{pr} SM_f$, $SM_b >^{pr} SM_{nw}$, and $SM_f >^{pr} SM_{nw}$. Although both of candidate stable models $X = \{r, s, c, f, h, \neg w, go\_c\}$ and $Y = \{r, s, c, w, h, \neg f, go\_b\}$ are the most probable stable models, we choose to go to beach, which shows the effect of preferences in decision making. If we arrange our trip based on the preferred stable models, the probability degree we are wet is greater than 0.83. Note that the exact probability degree is unknown, because we are not sure if it rains in some cases, which shows the abilty of LP$^{\text{MLN}}$ in reasoning over uncertain knowledge.

Table 1. Candidate Stable Models of M in Example 1

CSM	$W(M,*)$	$deg(*, r10)$	is preferred
$\{r, c, h, \neg w, go\_c\}$	e^6	3	
$\{c, h, \neg w, go\_c\}$	e^4	3	
$\{\neg c, h, \neg w, go\_c\}$	e^3	3	
$\{h, \neg w, go\_c\}$	e^2	3	
$\{r, s, c, f, h, \neg w, go\_c\}$	e^{10}	2	
$\{s, c, f, h, go\_c\}$	e^8	2	
$\{s, f, \neg c, h, go\_c\}$	e^7	2	
$\{s, f, h, go\_c\}$	e^6	2	
$\{r, c, f, h, \neg w, go\_c\}$	e^6	2	
$\{c, f, h, go\_c\}$	e^4	2	
$\{f, \neg c, h, go\_c\}$	e^3	2	
$\{f, h, go\_c\}$	e^2	2	
$\{r, s, c, w, h, \neg f, go\_b\}$	e^{10}	1	c, i, p, ps
$\{s, c, h, \neg f, go\_b\}$	e^8	1	c, i, p, ps
$\{s, \neg c, h, \neg f, go\_b\}$	e^7	1	c, i, p, ps
$\{s, h, \neg f, go\_b\}$	e^6	1	c, i, p, ps
$\{r, c, w, h, go\_b\}$	e^6	1	c, i, p, ps
$\{c, h, go\_b\}$	e^4	1	c, i, p, ps
$\{\neg c, h, go\_b\}$	e^3	1	c, i, p, ps
$\{h, go\_b\}$	e^2	1	c, i, p, ps

4 Computing *o*-LP$^{\text{MLN}}$ Programs

In this section, we present a translation from o-LP$^{\text{MLN}}$ programs to LP$^{\text{MLN}}$ programs, which provides an alternative approach to implementing o-LP$^{\text{MLN}}$.

In previous section, we rewrite an LPOD rule "$a \times b$." in LP$^{\text{MLN}}$ as follows

$$2 : a. \tag{13}$$

$$1 : b. \tag{14}$$

And we show that the above LP$^{\text{MLN}}$ program fails to capture the meaning of original LPOD rule. The gap between above LPOD and LP$^{\text{MLN}}$ rules is caused by following property of LPOD rules, called *satisfaction chain property*.

Proposition 1. *Let r be an LPOD rule in an LPOD P, if X is a candidate stable model of P, then $X \models r^k$ for any $deg(X, r) \leq k \leq o(r)$, where r^k is the k-th option of r.*

Obviously, our LP$^{\text{MLN}}$ encoding of an LPOD rule cannot express the satisfaction chain property. Therefore, to compute LPOD rules in LP$^{\text{MLN}}$, we need to encode not only the preferences but also satisfaction chain property expressed by the rule, which is the intuition of our translation method presented in this section.

Definition 5. *Given an o-LP^{MLN} program M, its LP^{MLN} counterpart $\tau(M)$ consists of three parts, i.e. $\tau(M) = M^r \cup \tau_1(M^o) \cup \tau_2(M^o)$, where M^r is the regular part of M, and the other two parts of $\tau(M)$ are defined as follows*

- $\tau_1(M^o) = \{\alpha : sh(r) \mid \alpha : r \in M^o\}$, *where $sh(r)$ is a complete shift of r defined as follows*

$$\leftarrow body(r),\ not\ h_1, ...,\ not\ h_{o(r)}. \tag{15}$$

- $\tau_2(M^o) = \{-1 : r^k \mid \alpha : r \in M^o,\ and\ 1 \leq k \leq o(r)\}$.

Definition 5 presents a linear and modular translation from o-LP$^{\text{MLN}}$ to LP$^{\text{MLN}}$, which uses the ideas in split program to express the satisfaction chain property of o-LP$^{\text{MLN}}$ rules. Lemma 1 shows that an o-LP$^{\text{MLN}}$ program and its LP$^{\text{MLN}}$ counterpart coincide on their (candidate) stable models.

Lemma 1. *Given an o-LP^{MLN} program M, $\tau(M)$ is the LP^{MLN} counterpart defined in Definition 5, a consistent set X of literals is a candidate stable model of M iff it is a stable model of $\tau(M)$.*

Proof. The proof is divided into two parts, in the first part, we show that if $X \in CSM(M)$ then $X \in SM(\tau(M))$; in the second part, we show that if $X \in SM(\tau(M))$ then $X \in CSM(M)$. For an o-LP$^{\text{MLN}}$ program M, let M^* be a split program of M. For each wo-rule $\alpha : r \in M^o$, let $\alpha : r^*$ be the corresponding LP$^{\text{MLN}}$ rule in M^*, and $\alpha : sh(r)$ be the corresponding rule in $\tau_1(M^o)$.

Part 1: For the split program M^*, without loss of generality, suppose X is a stable model of M^*. By the definition, for each wo-rule $\alpha : r \in M^o$, we have $X \models$

$head(r^*)$ or $X \not\models body(r^*)$, which means $X \not\models body(sh(r))$, i.e. $X \models \alpha : sh(r)$. Therefore, we can infer that $X \models \tau_1(M^o)$, which means $\tau_1(M^o) \subseteq \tau(M)_X$. By the definition of split program, we have $\overline{(M^r)^h} \subseteq \overline{M^*_X} \subseteq \overline{\tau(M)_X}$. Assume there is a proper subset X' of X such that $X' \models \tau(M)_X$, by above results, we have $X' \models M^*_X$, which contradicts with the premise that X is a stable model of M^*. Hence, X is a stable model of $\tau(M)$.

Part 2: Suppose X is a stable model of $\tau(M)$. By the definition, it is obvious that $\tau(M)_X = M^r_X \cup \tau_1(M^o) \cup \{-1 : r^k \in \tau_2(M^o) \mid deg(X, \alpha : r) \leq k \leq o(r)\}$. Let $M(X)$ be a split program of M w.r.t. X, $M(X)$ is constructed as follows

- add all rules in M^r to $M(X)$;
- for each o-LP$^{\text{MLN}}$ rule $\alpha : r \in M^o$, add $\alpha : r^k$ to $M(X)$, where $k = deg(X, \alpha : r)$.

It is clear that $M(X)$ is a split program of M and $X \models M(X)$. Assume there is a proper subset X' of X such that $X' \models M(X)$. From the proof in Part 1, we have $X' \models M^r_X \cup \tau_1(M^o)$. By Proposition 1, it is clear that $X' \models \tau(M)_X$, which contradicts with $X \in SM(\tau(M))$. Hence, we can infer that $X \in CSM(M)$.

Combining the above results, we have shown that $CSM(M) = SM(\tau(M))$, Lemma 1 is proven.

Now we investigate the preference criteria in the context of LP$^{\text{MLN}}$. Firstly, we introduce some notations. For an o-LP$^{\text{MLN}}$ program M and its LP$^{\text{MLN}}$ counterpart $\tau(M)$, the weight degree $W(\tau(M), X, r)$ of X w.r.t. a *wo*-rule $\alpha : r \in M^o$ is defined as

$$W(\tau(M), X, \alpha : r) = W(\tau_2(\{\alpha : r\})_X) \tag{16}$$

According to the satisfaction chain property for LPOD rules, we can obtain the relationship between satisfaction degree and weight degree defined in o-LP$^{\text{MLN}}$, which is shown in Proposition 2.

Proposition 2. *Given an o-LP^{MLN} program M and its LP^{MLN} counterpart $\tau(M)$, for a candidate stable model $X \in CSM(M)$ and a wo-rule $\alpha : r \in M^o$, the satisfaction degree $deg(X, \alpha : r)$ of $\alpha : r$ w.r.t. X can be derived from the weight degree $W(\tau(M), X, \alpha : r)$ of X w.r.t. $\alpha : r$, i.e.*

$$deg(X, \alpha : r) = o(r) + 1 + ln(W(\tau(M), X, \alpha : r)) \tag{17}$$

Based on the results in Proposition 2, it is clear that the preferred stable model of an o-LP$^{\text{MLN}}$ program M can be computed from its LP$^{\text{MLN}}$ counterpart, which is shown in Theorem 1.

Theorem 1. *Given an o-LP^{MLN} program M and its LP^{MLN} counterpart $\tau(M)$, all computing results of M can be derived from the computing results of $\tau(M)$, that is,*

- *$CSM(M) = SM(\tau(M))$;*
- *for each candidate stable model X, $W(M, X) = W(M^r, X)$;*

– *for each $pr \in \{c, i, p, ps\}$, pr-preferred stable models of M can be derived from the computing results of $\tau(M)$.*

Theorem 1 directly implies an approach to computing o-LP$^{\text{MLN}}$s via translating it into LP$^{\text{MLN}}$ programs and using existing LP$^{\text{MLN}}$ solvers, such as LPMLN2ASP [11], LPMLN-Models [17] etc. Among different preference criteria, the penalty-sum criterion especially relates to the weight of a stable model defined in LP$^{\text{MLN}}$. Therefore, we have a direct corollary of Theorem 1, which is shown in Corollary 1.

Corollary 1. *Given an o-LP^{MLN} program M and its LP^{MLN} counterpart $\tau(M)$, a consistent set X of literals is a ps-preferred stable model of M iff X is a stable model of $\tau(M)$, and there does not exist a stable model $Y \in SM(\tau(M))$ such that $W(\tau_2(M^o), Y) < W(\tau_2(M^o), X)$.*

Until now, we have shown an approach to computing o-LP$^{\text{MLN}}$ programs via using existing LP$^{\text{MLN}}$ solvers. Actually, it is easy to observe that our approach presented here can also be used to computing LPODs, since an LPOD can be viewed as an o-LP$^{\text{MLN}}$ program without soft rules. It is worth noting that although LPOD rules can be encoded in LP$^{\text{MLN}}$, our extension of LP$^{\text{MLN}}$ cannot only be seen as a syntactic sugar. On the one hand, the ordered disjunction provides a way to express preferences in LP$^{\text{MLN}}$, which is different from uncertain and inconsistent knowledge essentially. On the other hand, o-LP$^{\text{MLN}}$ introduces new criteria to evaluate stable models of an LP$^{\text{MLN}}$ program, which enriches the expressivity of LP$^{\text{MLN}}$.

In addition, the LP$^{\text{MLN}}$ counterpart of an o-LP$^{\text{MLN}}$ program is linear-time constructible, which means the computational complexity of o-LP$^{\text{MLN}}$ is the same as that of LP$^{\text{MLN}}$, and our extension of LP$^{\text{MLN}}$ does not increase the computational complexity.

5 Related Work

Concerning related work, since LPOD is an earlier work on representing preferences in ASP, there have been several formalisms presented after it, such as Answer Set Optimization [4], meta-programming technique [9] etc. All these formalisms can serve as a foundation to handle preferences in LP$^{\text{MLN}}$.

LPPOD [7, 8] is another extension of ASP that handles uncertainty and preferences in a unified framework by combining LPOD and possibilistic ASP (PASP) [15]. But PASP only handles qualitative uncertainty, while LP$^{\text{MLN}}$ can handle both qualitative and quantitative uncertainty. Therefore, o-LP$^{\text{MLN}}$ can handle qualitative as well as quantitative uncertainty.

In addition, Lee and Yang [14] present an "almost" modular translation from LPOD to ASP by introducing some auxiliary atoms, while our translation from LPOD to LP$^{\text{MLN}}$ does not introduce any new atom, and it is completely modular and linear-time constructible. All of other implementations of LPOD make iterative calls of ASP solvers to find preferred stable models [2, 6], while our translation only needs to call LP$^{\text{MLN}}$ solvers one time.

6 Conclusion and Future Work

In this paper, we present an alternative knowledge representation and reasoning tool for handling inconsistencies, uncertainty, and preferences in a unified framework, which is an extension of LP$^{\mathrm{MLN}}$ by introducing ordered disjunctions, called o-LP$^{\mathrm{MLN}}$. Our contributions are as follows. Firstly, we present the syntax and semantics of language o-LP$^{\mathrm{MLN}}$. Secondly, we present a translation from o-LP$^{\mathrm{MLN}}$ to regular LP$^{\mathrm{MLN}}$ programs. Using existing LP$^{\mathrm{MLN}}$ solvers, the translation provides a method to implement o-LP$^{\mathrm{MLN}}$. As a by-product, the translation also provides a one-shot approach to implementing LPOD.

For the future, we plan to further study the combination of LP$^{\mathrm{MLN}}$ and LPOD. For example, in this paper, we treat LPOD rules as hard rules in LP$^{\mathrm{MLN}}$, while this restriction should be relaxed in some cases, i.e. the preferences should be allowed to be dissatisfied, which will be discussed in further work. Moreover, we plan to develop an efficient o-LP$^{\mathrm{MLN}}$ solver, and use o-LP$^{\mathrm{MLN}}$ in more practical scenarios.

7 Acknowledgments

We are grateful to the anonymous referees for their useful comments. This work is supported by the National Key Research and Development Plan of China (Grant No.2017YFB1002801).

References

1. Balai, E., Gelfond, M.: On the Relationship between P-log and LP$^{\mathrm{MLN}}$. In: Kambhampati, S. (ed.) Proceedings of the 25th International Joint Conference on Artificial Intelligence. pp. 915–921 (2016)
2. Brewka, G., Niemelä, I., Syrjänen, T.: Implementing ordered disjunction using answer set solvers for normal programs. In: Proceedings of the 8th European Conference On Logics In Artificial Intelligence. pp. 444–456. Cosenza, Italy (2002). https://doi.org/10.1007/3-540-45757-7-37
3. Brewka, G.: Logic Programming with Ordered Disjunction. In: Proceedings of the 9th International Workshop on Non-Monotonic Reasoning. pp. 100–105 (2002)
4. Brewka, G.: Answer Sets: From Constraint Programming Towards Qualitative Optimization. In: Proceedings of the 7th Logic Programming and Nonmonotonic Reasoning. vol. 2923, pp. 34–46. Fort Lauderdale, USA (2004)
5. Brewka, G.: Preferences in Answer Set Programming. In: Proceedings of the 11th Conference of the Spanish Association for Artificial Intelligence on Current Topics in Artificial Intelligence. pp. 1–10 (2005)
6. Brewka, G., Delgrande, J.P., Romero, J.: asprin: Customizing Answer Set Preferences without a Headache. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence. pp. 1467—-1474 (2015)
7. Confalonieri, R., Nieves, J.C., Osorio, M., Vázquez-Salceda, J.: Dealing with explicit preferences and uncertainty in answer set programming. Annals of Mathematics and Artificial Intelligence **65**(2-3), 159–198 (2012). https://doi.org/10.1007/s10472-012-9311-0

8. Confalonieri, R., Prade, H.: Using possibilistic logic for modeling qualitative decision: Answer Set Programming algorithms. International Journal of Approximate Reasoning **55**(2), 711–738 (2014). https://doi.org/10.1016/j.ijar.2013.11.002
9. Gebser, M., Kaminski, R., Schaub, T.: Complex optimization in answer set programming. Theory and Practice of Logic Programming **11**(4-5), 821–839 (2011). https://doi.org/10.1017/S1471068411000329
10. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: Kowalski, R.A., Bowen, K.A. (eds.) Proceedings of the Fifth International Conference and Symposium on Logic Programming. pp. 1070–1080. MIT Press (1988)
11. Lee, J., Talsania, S., Wang, Y.: Computing LP$^{\text{MLN}}$ using ASP and MLN solvers. Theory and Practice of Logic Programming **17**(5-6), 942–960 (sep 2017). https://doi.org/10.1017/S1471068417000400
12. Lee, J., Wang, Y.: Weighted Rules under the Stable Model Semantics. In: Baral, C., Delgrande, J.P., Wolter, F. (eds.) Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning:. pp. 145–154. AAAI Press (2016)
13. Lee, J., Yang, Z.: LP$^{\text{MLN}}$, Weak Constraints, and P-log. In: Singh, S.P., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. pp. 1170–1177. AAAI Press (2017)
14. Lee, J., Yang, Z.: Translating LPOD and CR-Prolog2 into Standard Answer Set Programs **2** (2018)
15. Nicolas, P., Garcia, L., Stéphan, I., Lefèvre, C.: Possibilistic uncertainty handling for answer set programming. Annals of Mathematics and Artificial Intelligence **47**(1-2), 139–181 (2006). https://doi.org/10.1007/s10472-006-9029-y
16. Richardson, M., Domingos, P.M.: Markov Logic Networks. Machine learning **62**(1-2), 107–136 (2006). https://doi.org/10.1007/s10994-006-5833-1
17. Wang, B., Zhang, Z.: A Parallel LP$^{\text{MLN}}$ Solver: Primary Report. In: Bogaerts, B., Harrison, A. (eds.) Proceedings of the 10th Workshop on Answer Set Programming and Other Computing Paradigms. pp. 1–14. CEUR-WS, Espoo, Finland (2017)
18. Wang, B., Zhang, Z., Xu, H., Shen, J.: Splitting an LP$^{\text{MLN}}$ Program. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. pp. 1997–2004 (2018)