The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Splitting an LP^{MLN} Program

Bin Wang, Zhizheng Zhang,* Hongxiang Xu, Jun Shen
School of Computer Science and Engineering
Southeast University, Nanjing 211189, China
{kse.wang, seu_zzz, xhx1693, junshen}@seu.edu.cn

Abstract

The technique called splitting sets has been proven useful in simplifying the investigation of Answer Set Programming (ASP). In this paper, we investigate the splitting set theorem for LP^{MLN} that is a new extension of ASP created by combining the ideas of ASP and Markov Logic Networks (MLN). Firstly, we extend the notion of splitting sets to LP^{MLN} programs and present the splitting set theorem for LP^{MLN}. Then, the use of the theorem for simplifying several LP^{MLN} inference tasks is illustrated. After that, we give two parallel approaches for solving LP^{MLN} programs via using the theorem. The preliminary experimental results show that these approaches are alternative ways to promote an LP^{MLN} solver.

Introduction

The notion of splitting sets introduced by Lifschitz and Turner (1994) has been regarded as one of the most important tools on both theoretical and practical aspects in Answer Set Programming (ASP) (Gelfond and Lifschitz 1988). Intuitively, a splitting set of an ASP program is a set of literals, by which the program can be divided into two parts, called "*bottom part*" and "*top part*". Then, the stable models of the original ASP program can be computed by solving these two parts. It has been shown in several works (Lifschitz and Turner 1994; Ji et al. 2015) that the splitting set theorem can be used to simplify ASP programs and the proofs of some properties of ASP, and to develop more effective ASP solvers. Meanwhile, it is also shown that the splitting set theorem is a useful property for several extensions of ASP such as CR-Prolog (Balduccini 2009), Epistemic Specification (Watson 2000) etc.

LP^{MLN} (Lee and Wang 2016) is a new extension of ASP that incorporates the idea of Markov Logic Networks (MLN) (Richardson and Domingos 2006) and can be viewed as a weighted ASP program. In recent years, several results on LP^{MLN} have been presented. In the theoretical aspect, the relationships between LP^{MLN} and some other knowledge representation languages have been investigated. For example, Lee and Wang (2016) presented the translations from MLN, P-log (Baral, Gelfond, and Rushton 2009), ProbLog (De Raedt, Kimmig, and Toivonen 2007) and ASP with weak constraints (Calimeri et al. 2012) to LP^{MLN} respectively. Lee and Yang (2017) also presented a translation from LP^{MLN} to ASP with weak constraints. Balai and Gelfond (2016) presented a translation from LP^{MLN} to P-log. These results demonstrate the expressivity of LP^{MLN} and open a way to implement an LP^{MLN} solver. In the practical aspect, several solvers of LP^{MLN} have been developed through using the above translations, such as LPMLN2ASP, LPMLN2MLN (Lee, Talsania, and Wang 2017), and LPMLN-Models (Wang and Zhang 2017). In particular, LPMLN-Models is a parallel LP^{MLN} solver. These implementations also stimulate the study and use of LP^{MLN} conversely.

Usually, an LP^{MLN} program is much harder to solve than its unweighted ASP counterpart. Hence, to use LP^{MLN} in more practical scenarios, we need to investigate more useful properties to simplify the inference tasks for an LP^{MLN} program, and develop more effective ways to promote an LP^{MLN} solver. Our goal in this paper is to investigate the splitting set theorem for LP^{MLN} programs and show the usefulness of the theorem in simplifying the inference tasks for LP^{MLN} and promoting an LP^{MLN} solver.

In this paper, we propose the splitting set theorem for LP^{MLN} firstly, which shows that the tasks of computing stable models and their weight degrees of an LP^{MLN} program can be converted into the same kind of tasks of the corresponding "top part" and "bottom part". Secondly, we investigate a special kind of LP^{MLN} programs called independently divisible LP^{MLN} programs, and derive the splitting set theorem for such kind of programs. This special theorem directly implies that all basic LP^{MLN} inference tasks discussed in this paper can be simplified for an independently divisible LP^{MLN} program. Finally, we give two parallel approaches for solving normal and independently divisible LP^{MLN} programs respectively, which is a straightforward application of the splitting set theorem. The preliminary experimental results show that these approaches are alternative ways to promote an LP^{MLN} solver.

Preliminaries

LP^{MLN}

An LP^{MLN} program is a finite set of weighted rules $w : r$, where w is either a real number or the symbol α denoting

*Corresponding author

the "infinite weight", and r is an ASP rule of the form

$$l_1 \vee ... \vee l_k \leftarrow l_{k+1}, ..., l_m, not\ l_{m+1}, ..., not\ l_n. \quad (1)$$

where ls are literals, $\vee$ is epistemic disjunction, and not is default negation. For an ASP rule r of the form (1), $head(r) = \{l_i | 1 \leq i \leq k\}$, $body^+(r) = \{l_i | k+1 \leq i \leq m\}$, $body^-(r) = \{l_i | m+1 \leq i \leq n\}$, and $lit(r) = head(r) \cup body^+(r) \cup body^-(r)$. For an ASP program Π, we have $head(\Pi) = \bigcup_{r\in\Pi} head(r)$, $body^+(\Pi) = \bigcup_{r\in\Pi} body^+(r)$, $body^-(\Pi) = \bigcup_{r\in\Pi} body^-(r)$, and $lit(\Pi) = head(\Pi) \cup body^+(\Pi) \cup body^-(\Pi)$. An LP$^{\text{MLN}}$ rule $w : r$ is called soft if w is a real number, and hard if w is α. We use $\overline{M}$ to denote the set of unweighted ASP counterpart of an LP$^{\text{MLN}}$ program M, i.e. $\overline{M} = \{r | w : r \in M\}$. And 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. $W(M) = exp\left(\sum_{w:r\in M} w\right)$. Usually, a non-ground LP$^{\text{MLN}}$ program is considered as a shorthand for the corresponding ground program. So for simplicity, we restrict attention to ground LP$^{\text{MLN}}$ programs only.

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. We use M_X to 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 we use $SM(M)$ to denote the set of all stable models of an LP$^{\text{MLN}}$ program M. For a stable model X of an LP$^{\text{MLN}}$ program M, the weight degree $W(M, X)$ of X w.r.t. M is defined as

$$W(M, X) = exp\left(\sum_{w:r\in M_X} w\right) \quad (2)$$

and the probability degree $P_s(M, X)$ of X w.r.t. M is defined as

$$P_s(M, X) = \lim_{\alpha\to\infty} \frac{W(M, X)}{\Sigma_{X'\in SM(M)} W(M, X')} \quad (3)$$

For a proposition β, its probability degree $P_p(M, \beta)$ w.r.t. M is defined as

$$P_p(M, \beta) = \sum_{X\in SM(M) \text{ and } X\models\beta} P_s(M, X) \quad (4)$$

For an LP$^{\text{MLN}}$ program M and a literal l, we say that the tuple $(l, P_p(M, l))$ is a *probabilistic consequence* of M, denoted by $M \models (l, P_p(M, l))$, if $P_p(M, l) \neq 0$.

Usually, there are four kinds of basic inference tasks (Lee, Talsania, and Wang 2017). For an LP$^{\text{MLN}}$ program M,

- MAP (Maximum A Posteriori) inference is to compute the most probable stable models of M;
- MAPL (Maximum A Posteriori for Literals) inference is to check if some literals are in the most probable stable models of M, which can be converted to MAP inference;
- MPS (Marginal Probability of Stable models) inference is to compute probability degrees of all stable models of M;
- MPL (Marginal Probability of Literals) inference is to compute the probability degree of a literal w.r.t. M, which is to add the probability degrees of the stable models of M that contains the literal.

Splitting Sets for ASP

The notion of splitting sets for ASP is introduced by Lifschitz and Turner (1994). A splitting set U for a ground ASP program Π is a set of literals such that, for each rule $r \in \Pi$, if $head(r) \cap U \neq \emptyset$ then $lit(r) \subseteq U$. The bottom part of Π w.r.t. U is the set $b(\Pi, U) = \{r \in \Pi \mid head(r) \cap U \neq \emptyset\}$, and the top part of Π w.r.t. U is the set $t(\Pi, U) = \Pi - b(\Pi, U)$. Let U, X be sets of literals, and Π an ASP program, we use $d(\Pi, U, X)$ to denote the set of rules $r \in \Pi$ such that $body^+(r) \cap U \not\subseteq X$ or $body^-(r) \cap U \cap X \neq \emptyset$, called deleting set, and we use $e(\Pi, U, X)$ to denote the set of rules obtained from Π by deleting:

1. all rules in $d(\Pi, U, X)$;
2. all formulas of the form l or $not\ l$ in the bodies of the remaining rules, where $l \in U$.

The process containing above two steps that compute the set $e(\Pi, U, X)$ is called the *partial evaluation* for ASP.

Let U be a splitting set of an ASP program Π, a solution of Π w.r.t. U is a pair $\langle X, Y\rangle$ of the sets of literals such that

- X is a stable model of $b(\Pi, U)$;
- Y is a stable model of $e(t(\Pi, U), U, X)$;
- $X \cup Y$ is consistent.

The splitting set theorem for ASP shows that, for an ASP program Π and a splitting set U of Π, a set S is a stable model of Π iff $S = X \cup Y$ for a solution $\langle X, Y\rangle$ of Π w.r.t. U.

Splitting Sets for LP$^{\text{MLN}}$

In this section, we describe the notations and theorem related to splitting sets for LP$^{\text{MLN}}$. Without loss of generality, we consider only grounded finite LP$^{\text{MLN}}$ programs in rest of the paper.

The Definition of Splitting Sets for LP$^{\text{MLN}}$

Definition 1 (Splitting Sets for LP$^{\text{MLN}}$)**.** For a ground LP$^{\text{MLN}}$ program M, a set U of ground literals is called a splitting set of M, if for each LP$^{\text{MLN}}$ rule $w : r \in M$, $head(r) \cap U \neq \emptyset$ implies $lit(r) \subseteq U$.

It is obvious that the definition of the splitting sets for LP$^{\text{MLN}}$ is quite similar to the definition of the splitting sets for ASP. Analogously, we can define the bottom part $b(M, U)$ and top part $t(M, U)$ of an LP$^{\text{MLN}}$ program M w.r.t. one of its splitting sets U as

$$b(M, U) = \{w : r \in M \mid head(r) \cap U \neq \emptyset\} \quad (5)$$

$$t(M, U) = M - b(M, U) \quad (6)$$

Example 1. Consider the following program M_1:

$$1 : a \leftarrow b, not\ c. \quad \text{(r1)}$$

$$1 : b \leftarrow c, not\ a. \quad \text{(r2)}$$

$$1 : c. \quad \text{(r3)}$$

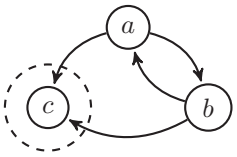

Figure 1: Dependency Graph of the Program in Example 1

Trivially, both $\emptyset$ and $lit\left(\overline{M_1}\right) = \{a, b, c\}$ are splitting sets of M_1. Besides, the set $U_1 = \{c\}$ is also a splitting set of M_1, and we have $b(M_1, U_1) = \{r3\}$ and $t(M_1, U_1) = \{r1, r2\}$.

Let us reconsider the top part and the bottom part of an LP$^{\text{MLN}}$ program from the side of dependency graph. A dependency graph of an LP$^{\text{MLN}}$ program M is a directed graph whose vertices are literals in M, and there is an arc from p to q if there is a rule $w : r \in M$ such that $p \in head(r)$ and $q \in body^+(r) \cup body^-(r)$. For example, the graph shown in Figure 1 is the dependency graph of the program M_1 in Example 1. By the definition of splitting sets, if U is a splitting set of the program M, then $U \cap head(t(M, U)) = \emptyset$. On one hand, it means there are no edges that are from literals in U to literals in the head of the top part. On the other hand, it means literals in the bottom part cannot be derived by any rules in the top part.

Splitting Set Theorem for LP$^{\text{MLN}}$

Now, we explore the use of the splitting sets in dividing the tasks of computing the stable models and their weight degrees of an LP$^{\text{MLN}}$ program. First of all, we define some notions for convenient description.

Definition 2 (Deleting Set)**.** Let U, X be sets of ground literals, and M a ground LP$^{\text{MLN}}$ program, the deleting set $d(M, U, X)$ of M w.r.t. U and X is

$$d(M, U, X) = \{w : r \in M \mid body^+(r) \cap U \not\subseteq X \text{ or } body^-(r) \cap U \cap X \neq \emptyset\} \tag{7}$$

Definition 3 (Partial Evaluation Reduct)**.** For an LP$^{\text{MLN}}$ rule $w : r$ and a set U of literals, the partial evaluation reduct $pred(w : r, U)$ of the rule w.r.t. U is obtained by

- deleting all formulas of the form l and $not\ l$, where $l \in U$;
- dropping the weight w.

Definition 4 (Grouping Set)**.** For an LP$^{\text{MLN}}$ program M and a set U of literals, the grouping set $g(M, U)$ of M w.r.t. U is a set of rule sets defined as follows.

$$\begin{aligned} g(M, U) = \{&M' \subseteq M \mid \forall w_1 : r_1, w_2 : r_2 \in M', \\ &\forall w_3 : r_3 \in M - M', \\ &pred(w_1 : r_1, U) = pred(w_2 : r_2, U), \\ &pred(w_3 : r_3, U) \neq pred(w_1 : r_1, U)\} \end{aligned} \tag{8}$$

We can observe that, for each rule set $M' \in g(M, U)$, all rules in M' have the same partial evaluation reduct $pred(w : r, U)$ w.r.t. U, where $w : r \in M'$.

Definition 5 (Representative Rule)**.** Let M be an LP$^{\text{MLN}}$ program, and U a set of literals. For each rule set $M' \in g(M, U)$, the rule $w' : r'$, denoted by $rep(M', U)$, is the representative rule w.r.t. M', where r' is the partial evaluation reduct of a rule in M' w.r.t. U, and w' is obtained by

$$w' = \sum_{w : r \in M'} w \tag{9}$$

Based on the concepts of deleting set, grouping set and representative rule, we define the partial evaluation for LP$^{\text{MLN}}$.

Definition 6 (Partial Evaluation for LP$^{\text{MLN}}$)**.** For an LP$^{\text{MLN}}$ program M and two sets U, X of literals, the set $e(M, U, X)$ of rules obtained by the partial evaluation for LP$^{\text{MLN}}$ is

$$\begin{aligned} e(M, U, X) = \{&rep(M', U) \mid \\ &M' \in g(M - d(M, U, X), U)\} \end{aligned} \tag{10}$$

Now we define the solutions to an LP$^{\text{MLN}}$ program M w.r.t. a splitting set of M.

Definition 7 (Solutions for LP$^{\text{MLN}}$)**.** For an LP$^{\text{MLN}}$ program M and a splitting set U of M, a solution to M w.r.t. U is a pair $\langle X, Y\rangle$ of sets of literals such that

- X is a stable model of $b(M, U)$;
- Y is a stable model of $e(t(M, U), U, X)$;
- $X \cup Y$ is consistent.

Example 2. Consider the LP$^{\text{MLN}}$ program M_1 in Example 1, the set $U_1 = \{c\}$ is a splitting set of M_1. Firstly, by the definition of the stable models of an LP$^{\text{MLN}}$ program, $X = \{c\}$ is a stable model of the bottom part $b(M_1, U_1)$. Then, we have the deleting set $d(t(M_1, U_1), U_1, X) = \{r1\}$, the grouping set $g(t(M_1, U_1) - d(t(M_1, U_1), U_1, X), U_1) = \{\{r2\}\}$, and by the partial evaluation for LP$^{\text{MLN}}$, we have $e(t(M_1, U_1), U_1, X) = \{b \leftarrow not\ a.\}$. Finally, $Y = \{b\}$ is a stable model of $e(t(M_1, U_1), U_1, X)$, and $X \cup Y$ is consistent. Hence, the pair $\langle X, Y\rangle$ is a solution to the program M_1 w.r.t. the splitting set U_1.

Theorem 1 (Splitting Set Theorem for LP$^{\text{MLN}}$)**.** Let U be a splitting set for an LP$^{\text{MLN}}$ program M. A set S of literals is a stable model of M iff $S = X \cup Y$ for a solution $\langle X, Y\rangle$ to M w.r.t. U. And the weight degree $W(M, S)$ of S w.r.t. M can be reformulated as

$$\begin{aligned} W(M, S) = &W(b(M, U), X) \times \\ &W(e(t(M, U), U, X), Y) \times \\ &W(d(t(M, U), U, X)) \end{aligned} \tag{11}$$

Example 3. Continue Example 2, by Theorem 1, we can derive that $S = X \cup Y = \{c, b\}$ is a stable model of M, and the weight degree $W(b(M_1, U_1), X) = e^1$, $W(e(t(M_1, U_1), U_1, X), Y) = e^1$, and $W(d(t(M_1, U_1), U_1, X)) = e^1$. By the Equation (11), $W(M_1, S) = e^1 \times e^1 \times e^1 = e^3$. It is easy to check that the above computing results coincide with the related definitions of LP$^{\text{MLN}}$.

Apparently, Theorem 1 contains two parts. One is about weight degrees evaluation of the stable models. Another is about stable models generation. In the rest of this subsection, we give the basic ideas of the proof of Theorem 1 (the

complete proof of the theorem can be found in the extended version of the paper[1]).

- For the first part, we give the intuitive meanings of the above definitions such that the soundness of the weight degree evaluation is clear.
- For the second part, we prove that there is a one-to-one mapping between the splitting sets of an LP$^{\text{MLN}}$ program and its ASP translation, which makes the soundness of this part clear.

Here are the intuitive meanings of the above definitions. Consider an LP$^{\text{MLN}}$ program as a set of weighted evidence for decision making and a rule is a piece of weighted evidence, then the strength of a piece of evidence is the sum of all the weights of rules that have the same unweighted form. The partial evaluation reduct of a rule w.r.t. a set U can be viewed as a piece of reduced evidence by removing the factors in U. Under such an assumption, a group in a grouping set w.r.t. U is a set of weighted evidence that are same if reduced by U. Furthermore, the representative rule of a group is just a piece of reduced evidence obtained from the original LP$^{\text{MLN}}$ program w.r.t. U, and its weight is the sum of the strengths of a set of original evidence that can be reduced to it by U. Hence, we can find that the evaluation of weight of a representative rule is compatible with the computation of the strength of a piece of evidence in LP$^{\text{MLN}}$. Now, we use Example 4 to illustrate it.

Example 4. Consider an LP$^{\text{MLN}}$ program M_3 that is obtained by adding following rule ($r6$) into the program M_1 in Example 1.

$$1 \ : \ b \leftarrow not\ a. \tag{r6}$$

The set $U = \{c\}$ is a splitting set of M_3, the set $X = \{c\}$ is a stable model of $b(M_3, U)$, the deleting set $d(t(M_3, U), U, X) = \{r1\}$. Let M' be $t(M_3, U) - d(t(M_3, U), U, X)$, the grouping set $g(M', U) = \{\{r2, r6\}\}$, and the rule set obtained by the partial evaluation for LP$^{\text{MLN}}$ is $e(t(M_3, U), U, X) = \{2 : b \leftarrow not\ a.\}$. It is clear that the weights of the same reduced evidences are accumulated properly in partial evaluation for LP$^{\text{MLN}}$.

Now, let us see Equation (11), by which the computing of weight degree of a stable model is divided into three parts. Suppose $S = X \cup Y$ is a stable model of an LP$^{\text{MLN}}$ program M, and $\langle X, Y\rangle$ is a solution to M. By the definition, the weight degree of S depends on the rules in M satisfied by S, which means, to compute the weight degree of S, we have to consider rules in the bottom part, deleting set, and the set obtained by partial evaluation. It is easy to check that all rules in the corresponding deleting set can be satisfied by the stable model S. Therefore, Equation (11) is provably correct.

From above discussion, we can define the equivalent of an LP$^{\text{MLN}}$ program.

Definition 8. An LP$^{\text{MLN}}$ program M is *equivalent* to another program M', if

- $SM(M) = SM(M')$, and
- for each stable model $S \in SM(M)$, $W(M, S) = W(M', S)$.

Lemma 1. Let M_1 and M_2 be LP$^{\text{MLN}}$ programs. If $\overline{M_1} = \overline{M_2}$, and for each rule $r \in \overline{M_1}$,

$$\sum_{w':r'\in M_1, \text{ and } r'=r} w' = \sum_{w':r'\in M_2, \text{ and } r'=r} w' \tag{12}$$

then the program M_1 is equivalent to M_2.

And we give an equivalent definition of the results obtained by partial evaluation for LP$^{\text{MLN}}$, which is useful for the proof of stable models generation part.

Definition 9. For an LP$^{\text{MLN}}$ program M and two sets U, X of literals, the set $e'(M, U, X)$ of rules is obtained from $e(M, U, X)$ by replacing every rule $w : r \in e(M, U, X)$ with an LP$^{\text{MLN}}$ program M' such that

- $\overline{M'} = \{r\}$ and $w = \sum_{w':r\in M'} w'$;
- $|M'| = |M''|$, where $M'' \in g(M - d(M, U, X), U)$ and $w : r$ is the representative rule of M''.

For the second part, the basic idea of the proof is to translate an LP$^{\text{MLN}}$ program into an ASP program and use the splitting set theorem for ASP programs. Firstly, we give a brief review of the translation from LP$^{\text{MLN}}$ to ASP.

For an LP$^{\text{MLN}}$ program M, its ASP translation $\tau(M)$ is obtained by turning each rule $w_i : r_i \in M$, where r_i is of the form (1), and i is the index of the rule, into following two ASP rules.

$$\begin{aligned} unsat(i) \ \leftarrow \ & l_{k+1}, ..., \ l_m, \ not\ l_{m+1}, ..., \ not\ l_n, \\ & not\ l_1, \ ..., \ not\ l_k. \\ l_1 \ \vee \ ... \ \vee \ l_k \ \leftarrow \ & l_{k+1}, ..., \ l_m, \ not\ l_{m+1}, ..., \\ & not\ l_n, \ not\ unsat(i). \end{aligned} \tag{13}$$

It has been shown in (Lee, Talsania, and Wang 2017) that there is a 1-1 correspondence ϕ between $SM(M)$ and the set of stable models of $\tau(M)$.

The proof of the second part of Theorem 1 is outlined as following Lemma 2 - 4.

Lemma 2. For an LP$^{\text{MLN}}$ program M and a splitting set U of M, there exists a splitting set U' of the translation $\tau(M)$ such that

- $U' = U \cup \{unsat(i) \mid w_i : r_i \in b(M, U)\}$,
- $\tau(b(M, U)) = b(\tau(M), U')$, and
- for a stable model $X \in SM(b(M, U))$, $\tau(e'(t(M, U), U, X)) = e(t(\tau(M), U'), U', \phi(X))$.

Lemma 3. Let U be a splitting set for an LP$^{\text{MLN}}$ program M. A set S of literals is a stable model of M iff $S = X \cup Y$ for a solution $\langle \phi(X), \phi(Y)\rangle$ to the translation $\tau(M)$ w.r.t. U', where U' is defined as in Lemma 2.

Lemma 4. Let U be a splitting set for an LP$^{\text{MLN}}$ program M. A pair $\langle X, Y\rangle$ is a solution to M w.r.t. U iff $\langle \phi(X), \phi(Y)\rangle$ is a solution to the translation $\tau(M)$ w.r.t. U', where U' is defined as in Lemma 2.

It is interesting that the stable models generation part of splitting set theorem for LP$^{\text{MLN}}$ is quite similar to the ones for ASP. By above proof, we can find that such similarity is not a coincidence.

[1] http://cse.seu.edu.cn/PersonalPage/seu_zzz/publications/lpmln-splitting-sets.pdf

Splitting Sequence Theorem for LP$^{\text{MLN}}$

Similar to ASP, we can derive following splitting sequence theorem straightforwardly.

Definition 10 (Splitting Sequence for LP$^{\text{MLN}}$)**.** Let $U_0, \ldots, U_n$ be a finite sequence of sets of literals, for an LP$^{\text{MLN}}$ program M, the sequence is called a splitting sequence of M, if it satisfies

- for any $0 \leq k \leq n$, U_k is a splitting set of M; and
- for any $0 \leq k < n$, $U_k \subset U_{k+1}$.

Definition 11. Let $U_0, \ldots, U_n$ be a splitting sequence of an LP$^{\text{MLN}}$ program M, a solution to M w.r.t. the splitting sequence is a sequence $X_0, \ldots, X_{n+1}$ of sets of literals such that

- X_0 is a stable model of $b(M, U_0)$;
- for any $0 < k \leq n$, X_k is a stable model of the program M_k, where M_k is obtained by

$$M_k = e(b(M, U_k) - b(M, U_{k-1}), U_{k-1}, \bigcup_{0 \leq i < k-1} X_i) \tag{14}$$

- X_{n+1} is a stable model of the program $e(t(M, U_n), U_n, \bigcup_{0 \leq i \leq n} X_i)$;
- $\bigcup_{0 \leq k \leq n+1} X_k$ is consistent.

Theorem 2 (Splitting Sequence Theorem for LP$^{\text{MLN}}$)**.** Let $U_0, \ldots, U_n$ be a finite splitting sequence of an LP$^{\text{MLN}}$ program M. A set S of literals is a stable model of M iff

- a finite sequence $X_0, \ldots, X_{n+1}$ of set of literals is a solution to M w.r.t. the splitting sequence; and
- $S = \bigcup_{0 \leq k \leq n+1} X_k$.

And the weight degree $W(M, S)$ can be reformulated as

$$\begin{aligned} W(M, S) = {} & W(b(M, U_0), X_0) \times \\ & W(e(t(M, U_n), U_n, \bigcup_{0 \leq k \leq n} X_k), X_{n+1}) \times \\ & W(d(t(M, U_n), U_n, \bigcup_{0 \leq k \leq n} X_k)) \times \\ & \prod_{1 \leq k \leq n} W(M_k, X_k) \times \prod_{1 \leq k \leq n} W(M'_k) \end{aligned} \tag{15}$$

where M_k is defined as Equation (14), and M'_k is defined as follows (for any $1 \leq k \leq n$).

$$M'_k = d(b(M, U_k) - b(M, U_{k-1}), U_{k-1}, \bigcup_{0 \leq i < k-1} X_i) \tag{16}$$

Independently Divisible LP$^{\text{MLN}}$ Programs

In this section, we investigate independently divisible LP$^{\text{MLN}}$ programs and derive the corresponding splitting set theorem. Now, we give an example to show the case discussed in this section.

Example 5. Consider the LP$^{\text{MLN}}$ program M that is from Example 1 in (Lee and Wang 2016).

$$\begin{aligned} \alpha &: bird(X) \leftarrow residentBird(X). \\ \alpha &: bird(X) \leftarrow migratoryBird(X). \\ \alpha &: \leftarrow migratoryBird(X), residentBird(X). \\ 2 &: residentBird(joe). \\ 1 &: migratoryBird(joe). \\ 2 &: residentBird(amy). \\ 1 &: migratoryBird(amy). \end{aligned}$$

We use $gr(M)$ to denote the grounded version of M, and M_a and M_j to denote the maximal subset of $gr(M)$ only related to the object amy and joe respectively. It can be observed that $gr(M) = M_a \cup M_j$ and $lit(\overline{M_a}) \cap lit(\overline{M_j}) = \emptyset$. Here, we show some interesting properties of such kind of programs, which will be useful to improve the efficiency of the LP$^{\text{MLN}}$ inference tasks.

Firstly, we introduce a new notation. For a set X of literals, the complement of X, denoted by $\neg X$, is defined as $\neg X = \{\text{the complement of } l \mid l \in X\}$. Now we can give the definition for independently divisible LP$^{\text{MLN}}$ programs.

Definition 12 (Independently Divisible LP$^{\text{MLN}}$ programs)**.** An LP$^{\text{MLN}}$ program M is called independently divisible, if there is a proper subset M_1 of M such that $lit(\overline{M_1}) \cap lit(\overline{M_2}) = \emptyset$ and $\neg head(\overline{M_1}) \cap head(\overline{M_2}) = \emptyset$, where $M_2 = M - M_1$. In addition, above programs M_1 and M_2 are called independent programs w.r.t. M.

For two independent programs M_1 and M_2 w.r.t. $M = M_1 \cup M_2$, $lit(\overline{M_1})$ and $lit(\overline{M_2})$ are two splitting sets for M, and the corresponding computation results are shown in Table 1, where U, $bottom$, top, $deleting$, and $peval$ refer to the splitting set U of M, the bottom part $b(M, U)$, the top part $t(M, U)$, the deleting set $d(t(M, U), U, X)$, and the partial evaluation result $e(t(M, U), U, X)$ for a stable model X of the bottom part respectively.

Table 1: Computation Results

U	$bottom$	top	$deleting$	$peval$
$lit(\overline{M_1})$	M_1	M_2	$\emptyset$	M_2
$lit(\overline{M_2})$	M_2	M_1	$\emptyset$	M_1

Now we give the definition of solutions and the splitting set theorem for independently divisible LP$^{\text{MLN}}$ programs.

Definition 13 (Independent Solutions for LP$^{\text{MLN}}$)**.** Let M_1 and M_2 be independent LP$^{\text{MLN}}$ programs w.r.t. the program $M = M_1 \cup M_2$. If $U = lit(\overline{M_1})$ is a splitting set of M, an independent solution to M w.r.t. U is a pair $\langle X, Y \rangle$ of sets of literals such that

- X is a stable model of M_1;
- Y is a stable model of M_2;

For an independently divisible LP$^{\text{MLN}}$ program M, if $\langle X, Y \rangle$ is a normal solution to M, then $\langle Y, X \rangle$ is not a solution to M usually. But Proposition 1 tells us that the independent solutions are different.

Proposition 1. For an independently divisible LP$^{\text{MLN}}$ program M, if $\langle X, Y\rangle$ is an independent solution to M, then $\langle Y, X\rangle$ is also an independent solution to M.

Theorem 3 (Splitting Set Theorem for Independently Divisible LP$^{\text{MLN}}$ Programs)**.** Let M_1 and M_2 be independent LP$^{\text{MLN}}$ programs w.r.t. the program $M = M_1 \cup M_2$. A set S of literals is a stable model of M iff $S = X \cup Y$ for an independent solution $\langle X, Y\rangle$ to M, where $X \in SM(M_1)$ and $Y \in SM(M_2)$. And the weight degree $W(M, S)$ can be reformulated as

$$W(M, S) = W(M_1, X) \times W(M_2, Y) \tag{17}$$

Theorem 3 tells us that if the bottom part M_1 and the top part M_2 of an LP$^{\text{MLN}}$ program M are independent programs w.r.t. M, then the stable models of M can be computed through solving the programs M_1 and M_2 independently, and for a stable model $S \in SM(M)$, the weight degree $W(M, S)$ only depends on the rules in M_1 and M_2.

For an independently divisible LP$^{\text{MLN}}$ program M, all inference tasks discussed in the preliminaries of the paper can be simplified, which is shown in following four corollaries.

Corollary 1. Let M_1 and M_2 be independent LP$^{\text{MLN}}$ programs w.r.t. the program $M = M_1 \cup M_2$. A set $S \in SM(M)$ is the most probable stable model of M iff $S = X \cup Y$ for an independent solution $\langle X, Y\rangle$ to M, where $X \in SM(M_1)$, $Y \in SM(M_2)$, and both X and Y are the most probable stable models of M_1 and M_2 respectively.

Corollary 2. Let M_1 and M_2 be independent LP$^{\text{MLN}}$ programs w.r.t. the program $M = M_1 \cup M_2$. A literal l is in the most probable stable model of M iff l is in the most probable stable model of M_1, where $l \in lit(\overline{M_1})$.

Corollary 3. Let M_1 and M_2 be independent LP$^{\text{MLN}}$ programs w.r.t. the program $M = M_1 \cup M_2$. A set $S = X \cup Y$ is a stable model of M, where $X \in SM(M_1)$ and $Y \in SM(M_2)$, the probability degree $P_s(M, S)$ can be reformulated as

$$P_s(M, S) = P_s(M_1, X) \times P_s(M_2, Y) \tag{18}$$

Corollary 4. Let M_1 and M_2 be independent LP$^{\text{MLN}}$ programs w.r.t. the program $M = M_1 \cup M_2$. For a literal $l \in lit(\overline{M_1})$, we have the marginal probability $P_p(M, l) = P_p(M_1, l)$.

Example 6. Consider the program M in Example 5. It is easy to check that $gr(M)$ is an independently divisible LP$^{\text{MLN}}$ program, the program M_a and M_j are independent programs w.r.t. M. The set $X = \{residentBird(joe),\ bird(joe)\}$ and $Y = \{residentBird(amy),\ bird(amy)\}$ are the most probable stable model of M_j and M_a respectively. By Corollary 1, the set $S = X \cup Y$ is the most probable stable model of $gr(M)$. By Corollary 2, we can check if the literal $bird(amy)$ is in the most probable stable model of $gr(M)$ by solving only the program M_a instead of the whole program $gr(M)$. By Corollary 3, the probability degree $P_s(gr(M), S) = P_s(M_j, X) \times P_s(M_a, Y) = 0.67 \times 0.67 \approx 0.45$. And by Corollary 4, the probability degree $P_p(M, bird(joe)) = P_p(M_j, bird(joe)) = 0.91$.

From the example, we can observe that all four kinds of inference tasks for an independently divisible LP$^{\text{MLN}}$ program M can be reduced to the same kinds of tasks of independent programs w.r.t. M. These independent programs are easier to solve than M, and they can also be solved concurrently. Therefore, we can use these properties to simplify the inferences for independently divisible LP$^{\text{MLN}}$ programs.

In addition, Corollary 4 also implies that if we extend an LP$^{\text{MLN}}$ program M by a program M' such that M and M' are independent programs w.r.t. $M \cup M'$, then the probabilistic consequences of M are still the probabilistic consequences of $M \cup M'$, and there are not any new probabilistic consequences whose atoms occur in M. As discussed in (Lifschitz and Turner 1994), this property is called *conservative extension property* of LP$^{\text{MLN}}$ programs.

Parallel Algorithms for Solving LP$^{\text{MLN}}$

An LP$^{\text{MLN}}$ program M is much harder to solve than its unweighted ASP counterpart $\overline{M}$, because the solver has to test all possible subsets of the program M, which is the core of several ASP-based LP$^{\text{MLN}}$ solvers. One of the advantages of such kind of solvers are the accurate inference results. There also exists other kinds of LP$^{\text{MLN}}$ sovlers. Lee et al. (Lee, Talsania, and Wang 2017) developed an effective LP$^{\text{MLN}}$ solver LPMLN2MLN that translates an LP$^{\text{MLN}}$ program into an MLN program and uses the MLN solvers as the backend. Due to effectiveness and scalability of the MLN solvers, LPMLN2MLN performs well. While only tight LP$^{\text{MLN}}$ programs can be handled by LPMLN2MLN, and the results computed by the MLN solvers are not accurate. In addition, some advanced ASP constructors such as aggregates are not supported by the MLN solvers. Hence, we need more approaches to improve the ASP-based LP$^{\text{MLN}}$ solver, and the parallelized solving methods are optional ones.

Wang and Zhang (2017) have presented a parallelized solving framework for LP$^{\text{MLN}}$, which focuses on the process of testing subprograms. By the method, the solver can partition an LP$^{\text{MLN}}$ program into several subprograms that are different subsets of the original program essentially, and these subprograms can be solved concurrently. In this paper, we present a new parallelized solving method for LP$^{\text{MLN}}$, which is based on the splitting set theorem for LP$^{\text{MLN}}$.

Let M be an LP$^{\text{MLN}}$ program, U a splitting set of M such that both of the top part and the bottom part of M are not empty. We use b to abbreviate the bottom part of M w.r.t. U, and $SM(b)$ is the set of stable models of the program b. To solve the top part, we need to compute the partial evaluation results, abbreviated by $e(X)$, for each stable model $X \in SM(b)$. And by the splitting set theorem for LP$^{\text{MLN}}$, for each $X \in SM(b)$ and $e(X)$, if the set Y of literals is a stable model of $SM(e(X))$ then $S = X \cup Y$ is a stable model of M, and the weight degree of S can be derived from the weight degrees of X, Y and $d(t(M, U), U, X)$. From the process, we can observe that, for each different stable model $X \in SM(b)$, the computation of $e(X)$ and corresponding stable models and weight degree are independent. An algorithm framework based on this idea is shown in Algorithm 1, which is a straightforward application of the splitting set theorem.

Algorithm 1: A Parallel LP$^{\text{MLN}}$ Solver: lpmln-sst

Input: M: an LP$^{\text{MLN}}$ program
Output: SM: stable models and their weight degrees of M

begin
  $SM = \emptyset$;
  // compute a splitting set
  U =ComputeSplittingSet(M) ;
  // compute bottom and top part
  b =ComputeBottomPart(M, U);
  $t = M - b$;
  // call existing LP$^{\text{MLN}}$ solver
  SM_b = LPMLNSolver(b);
  for $X_i \in SM_b$ **do** // concurrently do
    ...
    // partial evaluation
    $e(X_i)$ =PartialEval(t, U, X_i);
    SM_e =LPMLNSolver($e(X_i)$);
    for $Y_j \in SM_e$ **do**
      $S_{ij} = X_i \cup Y_j$;
      $W(M, S_{ij})$ is obtained by Equation (11);
      $SM = SM \cup \{(S_{ij}, W(M, S_{ij}))\}$;
  return SM;

For an independently divisible LP$^{\text{MLN}}$ program M, we can divide the program M into several pairwise disjoint independent programs that can be solved individually. This property implies a new parallel algorithm for independently divisible programs, which is shown in Algorithm 2.

Now we report the running time statistics for the two algorithms above and another parallel LP$^{\text{MLN}}$ solver LPMLN-Models on the program M in Example 5. The LP$^{\text{MLN}}$ solver used in the implementations of Algorithm 1 and 2 was LPMLN-Models in non-parallel mode. And at most 16 threads were used in these implementations. These experiments were carried out on a Dell PowerEdge R920 server with an Intel Xeon E7-4820@2.00GHz with 32 cores and 24 GB RAM running the Ubuntu 16.04 operating system. We use b-n to denote the program with n birds, hence, the program M is b-2. Table 2 contains the running times for several programs solved by using LPMLN-Models in non-parallel mode (L-M-1), LPMLN-Models with 16 threads (L-M-16), the implementations of Algorithm 1 and 2 respectively.

Although this is a preliminary test, it can be seen from the table that the splitting sets technique is an alternative way to promote the LP$^{\text{MLN}}$ solver, especially to handle independently divisible LP$^{\text{MLN}}$ programs.

Conclusion and Future Work

We summarize the contribution of this paper here. Firstly, we present the splitting set theorem for LP$^{\text{MLN}}$, which shows the tasks of computing stable models and their weight degrees can be turned into the same kind of tasks of the corresponding bottom and top parts. Secondly, we investigate a special kind of LP$^{\text{MLN}}$ programs, called independently divisible LP$^{\text{MLN}}$ program, and show that four kinds of basic LP$^{\text{MLN}}$ inference tasks can be simplified under this case. Finally, we give two parallel approaches for solving LP$^{\text{MLN}}$ programs. The preliminary experimental results show that these approaches are alternative ways to promote an LP$^{\text{MLN}}$ solver. Besides, the proof of the first part of the splitting set theorem for LP$^{\text{MLN}}$ shows that the translation from an LP$^{\text{MLN}}$ program to an ASP program is a very effective way to explore properties of LP$^{\text{MLN}}$ programs.

For the future, we plan to investigate more properties by using the splitting set theorem for LP$^{\text{MLN}}$ and the translation. And we also plan to develop a parallel LP$^{\text{MLN}}$ solver that combines the parallel approaches presented in this paper and Wang and Zhang's paper (2017). In addition, we plan to test these approaches further by modeling with LP$^{\text{MLN}}$ in more real-world applications.

Algorithm 2: A Parallel LP$^{\text{MLN}}$ Solver: lpmln-idp

Input: M: an independently divisible LP$^{\text{MLN}}$ program
Output: SM: stable models and their weight degrees of M

begin
  $SM = \emptyset$;
  // divide M into a set IM of programs
  IM =Divide(M) ;
  for $M_i \in IM$ **do** // concurrently do
    ...
    // call existing LP$^{\text{MLN}}$ solver
    SM_i =LPMLNSolver(M_i);
  For each $S \in SM$, $S = \bigcup_{1 \le i \le n} X_i$, where $X_i \in SM_i$ and $n = |IM|$;
  $W(M, S) = \Pi_{1 \le i \le n} W(M_i, X_i)$;
  return SM;

Table 2: Running Times of Several Ways

Program	L-M-1	L-M-16	Algo. 1	Algo. 2
b-10	0.62	0.94	0.36	0.06
b-11	1.51	1.08	1.12	0.06
b-12	4.75	3.15	3.34	0.15
b-13	17.88	11.15	10.79	0.44
b-14	78.64	37.88	34.51	1.24

Acknowledgments

We are grateful to the anonymous referees for their useful comments. The work was supported by the National Key Research and Development Plan of China (Grant No.2017YFB1002801), the National High Technology Research and Development Program of China (Grant No.2015AA015406), and the National Science Foundation of China (Grant No.60803061).

References

Balai, E., and Gelfond, M. 2016. On the Relationship between P-log and LP^{MLN}. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 915–921.

Balduccini, M. 2009. Splitting a CR-Prolog Program. In Erdem, E.; Lin, F.; and Schaub, T., eds., *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning*, 17–29. Springer-Verlag.

Baral, C.; Gelfond, M.; and Rushton, N. 2009. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming* 9(1):57–144.

Calimeri, F.; Faber, W.; Gebser, M.; Ianni, G.; Kaminski, R.; Krennwallner, T.; Leone, N.; Ricca, F.; and Schaub, T. 2012. ASP-Core-2 Input language format. *ASP Standardization Working Group*.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A probabilistic prolog and its application in link discovery. In Veloso, M. M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2468–2473.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In Kowalski, R. A., and Bowen, K. A., eds., *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, 1070–1080. MIT Press.

Ji, J.; Wang, H.; Huo, Z.; and Yuan, Z. 2015. Splitting a Logic Program Revisited. In Bonet, B., and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 1511–1517. AAAI Press.

Lee, J., and Wang, Y. 2016. Weighted Rules under the Stable Model Semantics. In Baral, C.; Delgrande, J. P.; and Wolter, F., eds., *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning:*, 145–154. AAAI Press.

Lee, J., and Yang, Z. 2017. LP^{MLN}, Weak Constraints, and P-log. In Singh, S. P., and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1170–1177. AAAI Press.

Lee, J.; Talsania, S.; and Wang, Y. 2017. Computing LP^{MLN} using ASP and MLN solvers. *Theory and Practice of Logic Programming* 17(5-6):942–960.

Lifschitz, V., and Turner, H. 1994. Splitting a Logic Program. In Hentenryck, P. V., ed., *Proceedings of the Eleventh International Conference on Logic Programming*, 23–37. MIT Press.

Richardson, M., and Domingos, P. M. 2006. Markov Logic Networks. *Machine learning* 62(1-2):107–136.

Wang, B., and Zhang, Z. 2017. A Parallel LP^{MLN} Solver: Primary Report. In Bogaerts, B., and Harrison, A., eds., *Proceedings of the 10th Workshop on Answer Set Programming and Other Computing Paradigms*, 1–14. Espoo, Finland: CEUR-WS.

Watson, R. 2000. A Splitting Set Theorem for Epistemic Specifications. *CoRR* cs.AI/0003.