

# Strong Equivalence for $LP^{MLN}$ Programs

Joohyung Lee and Man Luo

Arizona State University, USA

{joolee, mluo26}@asu.edu

$LP^{MLN}$  is a probabilistic extension of answer set programs with the weight scheme adapted from Markov Logic. We study the concept of strong equivalence in  $LP^{MLN}$ , which is a useful mathematical tool for simplifying a part of an  $LP^{MLN}$  program without looking at the rest of it. We show that the verification of strong equivalence in  $LP^{MLN}$  can be reduced to equivalence checking in classical logic via a reduct and choice rules as well as to equivalence checking under the “soft” logic of here-and-there. The result allows us to leverage an answer set solver for  $LP^{MLN}$  strong equivalence checking. The study also suggests us a few reformulations of the  $LP^{MLN}$  semantics using choice rules, the logic of here-and-there, and classical logic.

## 1 Introduction

$LP^{MLN}$  is a probabilistic extension of answer set programs with the weight scheme adapted from Markov Logic [18]. An  $LP^{MLN}$  program defines a probability distribution over all “soft” stable models, which do not necessarily satisfy all rules in the program, but the more rules with the bigger weights they satisfy, the bigger probabilities they get.

The language turns out to be highly expressive to embed several other probabilistic logic languages, such as P-log [2], ProbLog [5], Markov Logic, and Causal Models [17], as described in [11, 1, 14]. Inference engines for  $LP^{MLN}$ , such as LPMLN2ASP, LPMLN2MLN [10], and LPMLN-MODELS [19], have been developed based on the reduction of  $LP^{MLN}$  to answer set programs and Markov Logic.  $LP^{MLN}$  is a basis of probabilistic action language  $p\mathcal{BC}+$  [12], which is defined as a high-level notation of  $LP^{MLN}$  to describe probabilistic transition systems.

As more results are built upon  $LP^{MLN}$ , it becomes more critical to identify the equivalence between  $LP^{MLN}$  programs. Similar to answer set programs,  $LP^{MLN}$  programs  $F$  and  $G$  that have the same soft stable models with the same probability distribution are not necessarily equivalent in a stronger sense: when we add the same program  $H$  to each of  $F$  and  $G$ , the resulting programs may have different soft stable models and different probability distributions.

As in standard answer set programs, strong equivalence in  $LP^{MLN}$  is important in  $LP^{MLN}$  programming to simplify a part of an  $LP^{MLN}$  program without looking at the rest of it and to verify the correctness of  $LP^{MLN}$  for the representation. For instance, the following rules appearing in any program

$$\begin{aligned} -(w_1 + w_2) : & \quad a \vee b \\ w_1 : & \quad a \leftarrow b \\ w_2 : & \quad b \leftarrow a \end{aligned}$$

can be replaced by a simpler rule

$$\begin{aligned} w_1 : & \quad a \\ w_2 : & \quad b \end{aligned}$$

without affecting the probability distribution over soft stable models.

However, because of the semantic differences, strong equivalence for answer set programs does not simply carry over to  $\text{LP}^{\text{MLN}}$ . First, the weights of rules play a role. Even for the same structure of rules, different assignments of weights make the programs no longer strongly equivalent. Also, due to the fact that soft stable models do not have to satisfy all rules, strongly equivalent answer set programs do not simply translate to strongly equivalent  $\text{LP}^{\text{MLN}}$  programs. For instance,  $\{a \vee b, \perp \leftarrow a, b\}$  is strongly equivalent to  $\{a \leftarrow \text{not } b, b \leftarrow \text{not } a, \perp \leftarrow a, b\}$ , but its  $\text{LP}^{\text{MLN}}$  counterpart  $\{\alpha : a \vee b, \alpha : \perp \leftarrow a, b\}$  is not strongly equivalent to  $\{\alpha : a \leftarrow \text{not } b, \alpha : b \leftarrow \text{not } a, \alpha : \perp \leftarrow a, b\}$ : if we add  $\{\alpha : a \leftarrow b, \alpha : b \leftarrow a\}$  to each of them,  $\{a, b\}$  is a soft stable model of the former (by disregarding the rule  $\alpha : \perp \leftarrow a, b$ ) but not of the latter (c.f. Example 1).

In this paper, we extend the notion of strong equivalence to  $\text{LP}^{\text{MLN}}$ , and show that the verification of strong equivalence in  $\text{LP}^{\text{MLN}}$  can be reduced to equivalence checking in classical logic plus weight consideration. We also extend the logic of here-and-there to weighted rules, which provides a monotonic basis of checking  $\text{LP}^{\text{MLN}}$  strong equivalence. The characterization of strong equivalence suggests us a few reformulations of the  $\text{LP}^{\text{MLN}}$  semantics using choice rules, the logic of here-and-there, and classical logic, which present us useful insights into the semantics.

The paper is organized as follows. After reviewing some preliminaries in Section 2, we present the definition of strong equivalence and some characterization of strong equivalence in terms of classical logic in Section 3. Then, we define the soft logic of here-and-there and soft equilibrium models, and show how soft logic of HT is related to strong equivalence in Section 4. Then, we show another way to characterize strong equivalence in the style of second-order logic in Section 5 and use it to design a way to check strong equivalence using ASP solvers in Section 6.

## 2 Preliminaries

### 2.1 Review: Language $\text{LP}^{\text{MLN}}$

We first review the definition of a (deterministic) stable model for a propositional formula [6]. For any propositional formula  $F$  and any set  $X$  of atoms, the reduct  $F^X$  is obtained from  $F$  by replacing every maximal subformula of  $F$  that is not satisfied by  $X$  with  $\perp$ . Set  $X$  is a *stable model* of  $F$  if  $X$  is a minimal model of the reduct  $F^X$ .

We next review the definition of  $\text{LP}^{\text{MLN}}$  from [11]. An  $\text{LP}^{\text{MLN}}$  program is a finite set of weighted formulas  $w : R$  where  $R$  is a propositional formula<sup>1</sup> and  $w$  is a real number (in which case, the weighted rule is called *soft*) or  $\alpha$  for denoting the infinite weight (in which case, the weighted rule is called *hard*).

For any  $\text{LP}^{\text{MLN}}$  program  $F$  and any set  $X$  of atoms,  $\bar{F}$  denotes the set of usual (unweighted) formulas obtained from  $F$  by dropping the weights, and  $F_X$  denotes the set of  $w : R$  in  $F$  such that  $X \models R$ .

Given an  $\text{LP}^{\text{MLN}}$  program  $F$ ,  $\text{SM}[F]$  denotes the set of *soft stable models*:

$$\{X \mid X \text{ is a (standard) stable model of } \bar{F}_X\}.$$

By  $TW(F)$  (“Total Weight” of  $F$ ) we denote the expression  $\exp(\sum_{w:R \in F} w)$ . For any interpretation  $X$ , the weight of an interpretation  $X$ , denoted  $W_F(X)$ , is defined as<sup>2</sup>

$$W_F(X) = \begin{cases} TW(F_X) & \text{if } X \in \text{SM}[F]; \\ 0 & \text{otherwise,} \end{cases}$$

<sup>1</sup>Same as in Markov Logic, we could allow schematic variables that range over the Herbrand Universe, and define the process of grounding accordingly. The result of this paper can be straightforwardly extended to that case.

<sup>2</sup>We identify an interpretation with the set of atoms true in it.

and the probability of  $X$ , denoted  $P_F(X)$ , is defined as

$$P_F(X) = \lim_{\alpha \rightarrow \infty} \frac{W_F(X)}{\sum_{Y \in \text{SM}[F]} W_F(Y)}.$$

Alternatively, the weight can be defined by counting the penalty of the interpretation [10]. More precisely, the penalty based weight of an interpretation  $X$  is defined as the exponentiated negative sum of the weights of the rules that are not satisfied by  $X$  (when  $X$  is a stable model of  $\overline{F_X}$ ). Let

$$W_F^{\text{pnt}}(X) = \begin{cases} (TW(F \setminus F_X))^{-1} & \text{if } X \in \text{SM}[F]; \\ 0 & \text{otherwise,} \end{cases}$$

and

$$P_F^{\text{pnt}}(X) = \lim_{\alpha \rightarrow \infty} \frac{W_F^{\text{pnt}}(X)}{\sum_{Y \in \text{SM}[F]} W_F^{\text{pnt}}(Y)}.$$

The following theorem tells us that the LP<sup>MLN</sup> semantics can be reformulated using the concept of a penalty-based weight.

**Theorem 1** *For any LP<sup>MLN</sup> program  $F$  and any interpretation  $X$ ,*

$$\begin{aligned} W_F(X) &= TW(F) \times W_F^{\text{pnt}}(X), \\ P_F(X) &= P_F^{\text{pnt}}(X). \end{aligned}$$

## 2.2 Review: Logic of Here and There

Logic of here and there (Logic *HT*) is proven to be useful as a monotonic basis for checking strong equivalence [15], and equilibrium models [16] are defined as a special class of minimal models in logic *HT*.

An *HT* interpretation is an ordered pair  $\langle Y, X \rangle$  of sets of atoms such that  $Y \subseteq X$ , which describe “two worlds”: the atoms in  $Y$  are true “here” ( $h$ ) and the atoms in  $X$  are true “there ( $t$ ).” The worlds are ordered by  $h < t$ .

For any *HT* interpretation  $\langle Y, X \rangle$ , any world  $w$ , and any propositional formula  $F$ , we define when the triple  $\langle Y, X, w \rangle$  satisfies  $F$  recursively, as follows:

- for any atom  $F$ ,  $\langle Y, X, h \rangle \models_{\text{ht}} F$  if  $F \in Y$ ;  $\langle Y, X, t \rangle \models_{\text{ht}} F$  if  $F \in X$ .
- $\langle Y, X, w \rangle \not\models_{\text{ht}} \perp$ .
- $\langle Y, X, w \rangle \models_{\text{ht}} F \wedge G$  if  $\langle Y, X, w \rangle \models_{\text{ht}} F$  and  $\langle Y, X, w \rangle \models_{\text{ht}} G$ .
- $\langle Y, X, w \rangle \models_{\text{ht}} F \vee G$  if  $\langle Y, X, w \rangle \models_{\text{ht}} F$  or  $\langle Y, X, w \rangle \models_{\text{ht}} G$ .
- $\langle Y, X, w \rangle \models_{\text{ht}} F \rightarrow G$  if for every world such that  $w \leq w'$ ,  $\langle Y, X, w' \rangle \not\models_{\text{ht}} F$  or  $\langle Y, X, w' \rangle \models_{\text{ht}} G$ .

**Definition 1** *We say that an *HT* interpretation  $\langle Y, X \rangle$  satisfies  $F$  (symbolically,  $\langle Y, X \rangle \models_{\text{ht}} F$ ) if  $\langle Y, X, h \rangle$  satisfies  $F$ . An *HT* model of  $F$  is an *HT* interpretation that satisfies  $F$ .*

Equilibrium models are defined as a special class of minimal models in logic *HT* as follows.

**Definition 2** An *HT* interpretation  $\langle Y, X \rangle$  is total if  $Y = X$ . A total *HT* interpretation  $\langle X, X \rangle$  is an equilibrium model of a propositional formula  $F$  if

- $\langle X, X \rangle \models_{\text{HT}} F$ , and
- for any proper subset  $Y$  of  $X$ ,  $\langle Y, X \rangle \not\models_{\text{HT}} F$ .

A natural deduction system for logic *HT* can be obtained from the natural deduction system for classical logic by dropping the law of excluded middle  $F \vee \neg F$  from the list of deduction rules and by adding the axiom schema  $F \vee (F \rightarrow G) \vee \neg G$ . From the deduction system, we can derive the weak law of excluded middle  $\neg F \vee \neg \neg F$ .

Theorem 1 from [15] shows that strong equivalence between two answer set programs coincides with equivalence in logic *HT*. The deduction rules above can be used for checking strong equivalence.

### 3 Strong Equivalence in $\text{LP}^{\text{MLN}}$

We define the notions of weak and strong equivalences, naturally extended from those for the standard stable model semantics.

**Definition 3**  $\text{LP}^{\text{MLN}}$  programs  $F$  and  $G$  are called weakly equivalent to each other if

$$P_F(X) = P_G(X)$$

for all interpretations  $X$ .

**Definition 4**  $\text{LP}^{\text{MLN}}$  programs  $F$  and  $G$  are called strongly equivalent to each other if, for any  $\text{LP}^{\text{MLN}}$  program  $H$ ,

$$P_{F \cup H}(X) = P_{G \cup H}(X)$$

for all interpretations  $X$ .

Note that strong equivalence implies weak equivalence, but not vice versa.

**Example 1** Consider two programs  $F$  and  $G$ <sup>3</sup>

$$\begin{array}{ll} F & 2: a \vee b \\ & 1: \leftarrow a \wedge b \\ G & 1: a \leftarrow \neg b \\ & 1: b \leftarrow \neg a \\ & 1: \leftarrow a \wedge b. \end{array}$$

The programs are weakly equivalent, but not strongly equivalent. One can check their probability distributions over soft stable models are identical. However, for

$$H = \{1: a \leftarrow b, 1: b \leftarrow a\},$$

set  $\{a, b\}$  is a soft stable model of  $F \cup H$  but not of  $G \cup H$ , so that  $P_{F \cup H}(\{a, b\})$  is  $e^4/Z$  ( $Z$  is a normalization factor) but  $P_{G \cup H}(\{a, b\})$  is 0.

<sup>3</sup>We identify  $F \leftarrow G$  with  $G \rightarrow F$  and  $\leftarrow F$  with  $F \rightarrow \perp$ .

We call an expression of the form  $e^{c_1+c_2\alpha}$ , where  $c_1$  is a real number accounting for the weight of soft rules and  $c_2$  is an integer accounting for the weight of hard rules, a *w-expression*. Then Definition 4 can be equivalently rewritten as follows: F and G are strongly equivalent to each other if there is a *w-expression*  $c$  such that for any LP<sup>MLN</sup> program H,

$$W_{F \cup H}(X) = c \times W_{G \cup H}(X) \quad (1)$$

for all interpretations  $X$ . The *w-expression*  $c$  accounts for the fact that the weights are “proportional” to each other, so the probability distribution remains the same.

In view of Theorem 1, it is also possible to use the penalty based weights, i.e., the equation

$$W_{F \cup H}^{\text{pnt}}(X) = c \times W_{G \cup H}^{\text{pnt}}(X)$$

can be used in place of (1).

By definition, every interpretation that has a non-zero weight is a soft stable model. Thus Definition 4 implies that the LP<sup>MLN</sup> programs are “structurally equivalent” to each other, which is defined as follows.

**Definition 5** LP<sup>MLN</sup> programs F and G are structurally equivalent if, for any LP<sup>MLN</sup> program H, programs  $F \cup H$  and  $G \cup H$  have the same set of soft stable models.

Strong equivalence implies structural equivalence, but not vice versa.

**Proposition 1** If LP<sup>MLN</sup> programs F and G are strongly equivalent, then they are structurally equivalent as well.

The fact that LP<sup>MLN</sup> programs F and G are structurally equivalent does not follow from the fact that ASP programs  $\bar{F}$  and  $\bar{G}$  are strongly equivalent.

**Example 1 Continued** In Example 1, two ASP programs  $\bar{F}$  and  $\bar{G}$  are strongly equivalent (in the sense of standard answer set programs) but F and G are not structurally equivalent, and consequently not strongly equivalent. If we add  $H = \{1 : a \leftarrow b, 1 : b \leftarrow a\}$  to each program,  $X = \{a, b\}$  is a soft stable model of  $F \cup H$  but not of  $G \cup H$ .

The following theorem shows a characterization of strong equivalence that does not need to consider adding all possible LP<sup>MLN</sup> programs H. Similar to Proposition 2 from [6], it shows that the verification of strong equivalence in LP<sup>MLN</sup> can be reduced to equivalence checking in classical logic plus weight checking.

**Theorem 2** For any LP<sup>MLN</sup> programs F and G, program F is strongly equivalent to G if and only if there is a *w-expression*  $c$  such that for every interpretation  $X$ ,

1.  $TW(F_X) = c \times TW(G_X)$ , and
2.  $(\bar{F}_X)^X$  and  $(\bar{G}_X)^X$  are classically equivalent.

Recall that  $TW(F_X)$  is simply an exponentiated sum of the weights of the rules that are true in  $X$ . Condition 1 of Theorem 2 does not require to check whether  $X$  is a soft stable model or not.<sup>4</sup> In view of Theorem 1, the condition can be replaced with

$$1'. TW(F \setminus F_X) = c \times TW(G \setminus G_X).$$

without affecting the correctness of Theorem 2.

<sup>4</sup>Instead, Condition 2 ensures that they are structurally equivalent as shown in **Theorem on Soft Stable Models** below.

**Example 2** Consider two programs

$$\begin{array}{ll} \text{F} & 0: \quad \neg a \\ & 2: \quad b \leftarrow a \\ & 3: \quad a \leftarrow \neg\neg a \\ \text{G} & 2: \quad \neg a \vee b \\ & 1: \quad a \vee \neg a \end{array}$$

The programs are strongly equivalent to each other. The following table shows that Conditions 1,2 of Theorem 2 are true in accordance with the theorem.

$X$	$TW(F_X)$	$TW(G_X)$	$(\overline{F_X})^X$	$(\overline{G_X})^X$
$\emptyset$	$e^5$	$e^3$	$\top$	$\top$
$\{a\}$	$e^3$	$e^1$	$a$	$a$
$\{b\}$	$e^5$	$e^3$	$\top$	$\top$
$\{a,b\}$	$e^5$	$e^3$	$a \wedge b$	$a \wedge b$

Table 1:  $(\overline{F_X})^X$  and  $(\overline{G_X})^X$

Note that  $TW(F_X) = e^2 \times TW(G_X)$ . However, if we replace rule 3:  $a \leftarrow \neg\neg a$  in F with 3:  $a \leftarrow a$  to result in  $F'$ , then  $F'$  and G are not strongly equivalent: for

$$H = \{1 : a \leftarrow b, 1 : b \leftarrow a\}$$

$\{a,b\}$  is a soft stable model of  $G \cup H$  with the weight  $e^5$ , but it is not a soft stable model  $F' \cup H$ , so its weight is 0. In accordance with Theorem 2,  $(\overline{F'_{\{a,b\}}})^{\{a,b\}}$  is not equivalent to  $(\overline{G_{\{a,b\}}})^{\{a,b\}}$ . The former is equivalent to  $\{b \leftarrow a\}$ , and the latter is equivalent to  $\{a \wedge b\}$ .

Even if the programs have the same soft stable models, the different weight assignments may make them not strongly equivalent. For instance, replacing the first rule in G by 3:  $\neg a \vee b$  to result in  $G'$ , we have  $TW(F_\emptyset) = e^1 \times TW(G'_\emptyset)$  and  $TW(F_{\{a\}}) = e^2 \times TW(G'_{\{a\}})$ , so there is no single  $w$ -expression  $c$  such that  $TW(F_X) = c \times TW(G'_X)$ .

Choice rules are useful constructs in answer set programming, and they turn out to have an interesting position in the semantics of  $LP^{MLN}$ . We consider a general form of choice rules that is not limited to atoms. For any propositional formula  $F$ , by  $\{F\}^{ch}$  we denote the formula  $F \vee \neg F$ . The following proposition tells us that choice rules can be alternatively represented in  $LP^{MLN}$  with the weight 0 rule.

**Proposition 2** For any formula  $F$ , the weighted formula  $0 : F$  is strongly equivalent to  $w : \{F\}^{ch}$ , where  $w$  is any real number or  $\alpha$ .

The following fact can also be useful for simplification.

**Proposition 3** Let  $H$  be an  $LP^{MLN}$  program that is structurally equivalent to  $w : \top$  or  $w : \perp$  ( $w$  is a real number or  $\alpha$ ). For any  $LP^{MLN}$  program  $F$ , program  $F \cup H$  is strongly equivalent to  $F$ .

For example, adding  $H = \{w_1 : a \wedge \neg a, w_2 : a \leftarrow a\}$  to  $F$ , one can easily see  $F$  and  $F \cup H$  are strongly equivalent.

Interestingly, some facts about strong equivalence known in answer set programs do not simply carry over to  $LP^{MLN}$  strong equivalence. The fact that, for any propositional formulas  $F, G$ , and  $K$ ,

$$(F \rightarrow G) \rightarrow K$$

is strongly equivalent to

$$\begin{aligned} & (G \vee \neg F) \rightarrow K \\ & K \vee F \vee \neg G \end{aligned}$$

is a key lemma to prove that any propositional formulas can be turned into the logic program syntax [3]. The result is significant because it allows stable models of general syntax of formulas to be computed by converting into rule forms and computed by standard answer set solvers, as done in system F2LP. However, it turns out that the transformation does not work under LP<sup>MLN</sup>, i.e., there are some formulas  $F, G, K$  such that

$$w : (F \rightarrow G) \rightarrow K \tag{2}$$

is not strongly equivalent to

$$\begin{aligned} w_1 : & (G \vee \neg F) \rightarrow K \\ w_2 : & K \vee F \vee \neg G \end{aligned} \tag{3}$$

regardless of weights  $w, w_1, w_2$ . For example, assuming  $F, G, K$  are atoms, and take interpretation  $X = \{F, G\}$ .

$$\begin{aligned} & (((F \rightarrow G) \rightarrow K)_X)^X \Leftrightarrow \perp \\ & ((\{(G \vee \neg F) \rightarrow K, K \vee F \vee \neg G\}_X)^X \Leftrightarrow F^X. \end{aligned}$$

So Condition 2 of Theorem 2 does not hold, and it follows that (2) is not strongly equivalent to (3).<sup>5</sup>

### 3.1 Reformulation of LP<sup>MLN</sup> Using Choice Rules

The second condition of Theorem 2 is equivalent to the fact that for any LP<sup>MLN</sup> program  $H$ , programs  $F \cup H$  and  $G \cup H$  have the same soft stable models. Throughout the paper, we show that the condition can be represented in several different ways. We start with the following version that uses choice rules.

We extend the notion of choice rules to a set of formulas as follows: for a set  $\Gamma$  of propositional formulas,  $\{\Gamma\}^{\text{ch}}$  denotes the set of choice formulas  $\{\{F\}^{\text{ch}} \mid F \in \Gamma\}$ .

**Theorem on Soft Stable Models** *For any LP<sup>MLN</sup> program  $F$  and  $G$ , the following conditions are equivalent.*

- (a)  $F$  and  $G$  are structurally equivalent.
- (b) For any set  $X$  of atoms,  $(\overline{F}_X)^X$  and  $(\overline{G}_X)^X$  are classically equivalent.
- (c) For any set  $X$  of atoms,  $(\{\overline{F}\}^{\text{ch}})^X$  and  $(\{\overline{G}\}^{\text{ch}})^X$  are classically equivalent.

Thus, Theorem 2 remains valid if we replace Condition 2 in it with

- 2'.  $(\{\overline{F}\}^{\text{ch}})^X$  and  $(\{\overline{G}\}^{\text{ch}})^X$  are classically equivalent.

As a side remark, **Theorem on Soft Stable Models** also tells us an equivalent characterization of soft stable models, which in turn leads to a reformulation of LP<sup>MLN</sup> semantics.

**Proposition 4** *For any LP<sup>MLN</sup> program  $F$ ,  $X$  is a soft stable model of  $F$  iff  $X$  is a (standard) stable model of  $\{\overline{F}\}^{\text{ch}}$ .*

---

<sup>5</sup> Of course, (2) is strongly equivalent to

$$w : ((G \vee \neg F) \rightarrow K) \wedge (K \vee F \vee \neg G)$$

but the latter is not in a rule form.

## 4 Soft Logic of Here and There

We extend the logic of here-and-there and the concept of equilibrium models to  $\text{LP}^{\text{MLN}}$  programs as follows.

**Definition 6** An HT interpretation  $\langle Y, X \rangle$  is called a soft HT model of an  $\text{LP}^{\text{MLN}}$  program  $F$  if, for every rule  $w : R$  in  $F_X$ ,  $\langle Y, X \rangle$  satisfies  $R$ . In other words,  $\langle Y, X \rangle$  is a soft HT model of  $F$  iff  $\langle Y, X \rangle$  is an HT model of  $\overline{F}_X$ .

We extend the **Theorem on Soft Stable Models** to consider HT models as follows. We omit repeating conditions (b), (c).

**Theorem on Soft Stable Models** For any  $\text{LP}^{\text{MLN}}$  program  $F$  and  $G$ , the following conditions are equivalent.

- (a)  $F$  and  $G$  are structurally equivalent.
- (d)  $F$  and  $G$  have the same set of soft HT models.
- (e) For any set  $X$  of atoms,  $\overline{F}_X \leftrightarrow \overline{G}_X$  is provable in logic HT.
- (f)  $(\{\overline{F}\}^{\text{ch}}) \leftrightarrow (\{\overline{G}\}^{\text{ch}})$  is provable in logic HT.

Again, any of the conditions (d), (e), (f) can replace Condition 2 of Theorem 2 without affecting the correctness.

**Example 2 Continued** We consider soft HT models of  $F$ ,  $G$  and  $F'$  in Example 2.

$X$	$F$	$G$	$F'$
$\langle \phi, \phi \rangle$	Yes	Yes	Yes
$\langle \phi, \{a\} \rangle$	No	No	Yes
$\langle \{a\}, \{a\} \rangle$	Yes	Yes	Yes
$\langle \phi, \{b\} \rangle$	Yes	Yes	Yes
$\langle \{b\}, \{b\} \rangle$	Yes	Yes	Yes
$\langle \phi, \{a, b\} \rangle$	No	No	Yes
$\langle \{a\}, \{a, b\} \rangle$	No	No	No
$\langle \{b\}, \{a, b\} \rangle$	No	No	Yes
$\langle \{a, b\}, \{a, b\} \rangle$	Yes	Yes	Yes

Table 2: Soft HT models of  $F$ ,  $G$ , and  $F'$

From Table 2, we see that  $F$  and  $G$  have the same set of soft HT models.

Condition (f) allows us to prove the structural equivalence between two  $\text{LP}^{\text{MLN}}$  programs by using deduction rules in logic HT.

**Example 3** Consider  $\text{LP}^{\text{MLN}}$  programs  $F$  and  $G$ :

$$F \quad 2: \quad \neg a \vee b \qquad G \quad 2: \quad \neg \neg a \rightarrow b$$

We check that  $\{\overline{F}\}^{\text{ch}} \leftrightarrow \{\overline{G}\}^{\text{ch}}$  is provable in logic HT. Recall that



$$\begin{aligned}\{\bar{F}\}^{ch} &= (\neg a \vee b) \vee \neg(\neg a \vee b) \\ \{\bar{G}\}^{ch} &= (\neg\neg a \rightarrow b) \vee \neg(\neg\neg a \rightarrow b).\end{aligned}$$

**Left-to-right:** Assume  $(\neg a \vee b) \vee \neg(\neg a \vee b)$ .

Case 1: Assume  $(\neg a \vee b)$ . Then  $\neg\neg a \rightarrow b$  is intuitionistically derivable, so derivable in logic HT as well.

Case 2: Assume  $\neg(\neg a \vee b)$ . Then  $\neg(\neg\neg a \rightarrow b)$  is intuitionistically derivable (Glivenko's Theorem).

**Right-to-left:** Assume  $(\neg\neg a \rightarrow b) \vee \neg(\neg\neg a \rightarrow b)$ .

Case 1: Assume  $\neg\neg a \rightarrow b$ . Then  $\neg a \vee b$  can be derived from the weak law of excluded middle  $\neg a \vee \neg\neg a$ .

Case 2: Assume  $\neg(\neg\neg a \rightarrow b)$ . Then  $\neg(\neg a \vee b)$  is intuitionistically derivable (Glivenko's Theorem).

In view of the equivalence between Conditions (a) and (f) of **Theorem on Soft Stable Models**, we conclude that  $F$  and  $G$  are structurally equivalent.

## 4.1 Soft Equilibrium Models

**Definition 7** A soft HT interpretation is called total if  $Y = X$ . A total soft HT interpretation  $\langle X, X \rangle$  is a soft equilibrium model of an  $LP^{MLN}$  program  $F$  if, for any proper subset  $Y$  of  $X$ ,  $\langle Y, X \rangle$  is not a soft HT model of  $F$ .

In comparison with Definition 2, Definition 7 omits the condition that  $\langle X, X \rangle$  satisfies  $\bar{F}_X$  because the condition is trivially satisfied by the definition of  $F_X$ .

The following lemma tells us how soft HT models are related to the reducts in  $LP^{MLN}$ .

**Lemma 1** For any  $LP^{MLN}$  program  $F$  and any sets  $Y, X$  of atoms such that  $Y \subseteq X$ , the following conditions are equivalent:

- (a)  $\langle Y, X \rangle$  is a soft HT model of  $F$ .
- (b)  $Y$  satisfies  $(\bar{F}_X)^X$ .
- (c)  $Y$  satisfies  $(\{\bar{F}\}^{ch})^X$ .

From the lemma, we conclude:

**Proposition 5** A set  $X$  of atoms is a soft stable model of  $F$  iff  $\langle X, X \rangle$  is a soft equilibrium model of  $F$ .

**Example 2 Continued** Table 1 shows that  $F$  and  $G$  have three soft stable models, which are  $\emptyset$ ,  $\{a\}$ ,  $\{a, b\}$ . Table 2 shows that  $F$  and  $G$  have three equilibrium models, which are  $\langle \emptyset, \emptyset \rangle$ ,  $\langle \{a\}, \{a\} \rangle$ ,  $\langle \{a, b\}, \{a, b\} \rangle$ . On the other hand,  $F'$  has only one equilibrium model,  $\langle \emptyset, \emptyset \rangle$ , which provides another account for the fact that  $F'$  and  $G$  have different soft stable models.

The weight of a soft equilibrium model can be defined the same as the weight of a soft stable model as defined in Section 2.1.

## 5 Strong Equivalence by Reduction to Classical Logic

We extend the theorem on stable models as follows. Let  $\mathbf{p}$  be the propositional signature. Let  $\mathbf{p}'$  be the set of new atoms  $p'$  where  $p \in \mathbf{p}$ . For any formula  $F$ ,  $\Delta_{\mathbf{p}'}(F)$  is defined recursively:

- $\Delta_{\mathbf{p}'}(p) = p'$  for any atomic formula  $p \in \mathbf{p}$ ;
- $\Delta_{\mathbf{p}'}(\neg F) = \neg F$ ;

- $\Delta_{\mathbf{p}'}(F \wedge G) = \Delta_{\mathbf{p}'}(F) \wedge \Delta_{\mathbf{p}'}(G)$ ;
- $\Delta_{\mathbf{p}'}(F \vee G) = \Delta_{\mathbf{p}'}(F) \vee \Delta_{\mathbf{p}'}(G)$ ;
- $\Delta_{\mathbf{p}'}(F \rightarrow G) = (\Delta_{\mathbf{p}'}(F) \rightarrow \Delta_{\mathbf{p}'}(G)) \wedge (F \rightarrow G)$ .

Lemma 1 is extended to  $\Delta$  as follows.

**Lemma 1'** *Let  $X, Y \subseteq \mathbf{p}$  and  $Y' = \{p' \in \mathbf{p}' \mid p \in Y\}$ . Each of the following conditions is equivalent to each of Conditions (a),(b),(c) of Lemma 1.*

- (d)  $Y' \cup X$  satisfies  $\Delta_{\mathbf{p}'}(\overline{F_X})$ .
- (e)  $Y' \cup X$  satisfies  $\Delta_{\mathbf{p}'}(\{\overline{F}\}^{\text{ch}})$ .

**Theorem on Soft Stable Models** *For any  $\text{LP}^{\text{MLN}}$  programs  $F$  and  $G$ , the following conditions are equivalent.*

- (a)  $F$  and  $G$  are structurally equivalent.
- (g) For any set  $X$  of atoms,  $\{p' \rightarrow p \mid p \in \mathbf{p}\}$  entails  $\Delta_{\mathbf{p}'}(\overline{F_X}) \leftrightarrow \Delta_{\mathbf{p}'}(\overline{G_X})$  (in the sense of classical logic).
- (h)  $\{p' \rightarrow p \mid p \in \mathbf{p}\}$  entails  $\Delta_{\mathbf{p}'}(\{\overline{F}\}^{\text{ch}}) \leftrightarrow \Delta_{\mathbf{p}'}(\{\overline{G}\}^{\text{ch}})$  (in the sense of classical logic).

The equivalence between Conditions (a) and (h) of **Theorem on Soft Stable Models** tells us the structural equivalence checking reduces to satisfiability checking. It also indicates the structural equivalence checking between  $\text{LP}^{\text{MLN}}$  programs is no harder than checking strong equivalence between standard answer set programs. In conjunction with Condition 1 of Theorem 2, the complexity of  $\text{LP}^{\text{MLN}}$  strong equivalence checking is no harder than checking strong equivalence for standard answer set programs.

**Theorem 3** *The problem of determining if two  $\text{LP}^{\text{MLN}}$  programs are strongly equivalent is co-NP-complete.*

## 5.1 Reformulation of $\text{LP}^{\text{MLN}}$ in Classical Logic

The following proposition relates  $\Delta$  to soft stable models.

**Proposition 6** *For any  $\text{LP}^{\text{MLN}}$  program  $F$ , set  $X$  is a soft stable model of  $F$  iff there is no strict subset  $Y$  of  $X$  such that  $Y' \cup X$  satisfies  $\Delta_{\mathbf{p}'}(\{\overline{F}\}^{\text{ch}})$ .*

The definition of  $\Delta$  is similar to the definition of  $F^*$  used in the second-order logic based definition of a stable model from [7]. This leads to the following reformulation of  $\text{LP}^{\text{MLN}}$  in second-order logic.

Let  $\mathbf{p}$  be a list of distinct atoms,  $p_1, \dots, p_n$ , and let  $\mathbf{u}$  be a list of distinct propositional variables  $u_1, \dots, u_n$ . By  $\mathbf{u} \leq \mathbf{p}$  we denote the conjunction of the formulas  $\forall \mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$  for all  $i = 1, \dots, n$ , where  $\mathbf{x}$  is a list of distinct object variables whose length is the same as the arity of  $p_i$ . Expression  $\mathbf{u} < \mathbf{p}$  stands for  $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{p} \leq \mathbf{u})$ .

**Proposition 7** *For any  $\text{LP}^{\text{MLN}}$  program  $F$ , a set  $X$  of atoms is a soft stable model of  $F$  iff  $X$  satisfies*

$$\neg \exists \mathbf{u}(\mathbf{u} < \mathbf{p}) \wedge \Delta_{\mathbf{u}}(\{\overline{F}\}^{\text{ch}}).$$

## 6 Checking Strong Equivalence Using ASP Solver

Based on the **Theorem on Soft Stable Models**, we use the following variant of Theorem 2 to leverage an ASP solver for checking LP<sup>MLN</sup> strong equivalence.

**Theorem 2'** For any LP<sup>MLN</sup> programs  $F$  and  $G$ , program  $F$  is strongly equivalent to  $G$  if and only if there is a  $w$ -expression  $c_1 + c_2\alpha$  such that for every interpretation  $X$ ,

- 1a.  $\sum_{\substack{w:R \in F, w \neq \alpha, \\ \text{and } X \not\models R}} w = c_1 + \sum_{\substack{w:R \in G, w \neq \alpha, \\ \text{and } X \not\models R}} w;$
- 1b.  $|\{\alpha : R \in F \mid X \not\models R\}| = c_2 + |\{\alpha : R \in G \mid X \not\models R\}|;$
2.  $\{p' \rightarrow p \mid p \in \mathbf{p}\}$  entails  $\Delta_{\mathbf{p}'}(\{\bar{F}\}^{\text{ch}}) \leftrightarrow \Delta_{\mathbf{p}'}(\{\bar{G}\}^{\text{ch}})$  (in the sense of classical logic).

In each of the following subsections, we show how to check the conditions using CLINGO together with F2LP [9]. We need F2LP to turn propositional formulas under the stable model semantics into the input language of CLINGO. We assume weights are given in integers as required by the input language of CLINGO.

### 6.1 Checking Conditions 1a, 1b of Theorem 2'

In order to check Conditions 1(a),1(b) of Theorem 2', we start by finding potential values for  $c_1$  and  $c_2$ . For that, we arbitrarily set  $X = \emptyset$  and find the values. If the same values of  $c_1$  and  $c_2$  make the equations true for all other interpretations as well, the conditions hold.

The checking is done by using the program  $\mathbf{P}$  in the input language of F2LP, constructed as follows. For any soft rule  $w_i : R_i$  in  $F$ , where  $w_i$  is an integer,  $\mathbf{P}$  contains

$$\begin{aligned} \text{f\_unsat\_s}(w_i, i) &\leftarrow \text{not } R_i \\ R_i &\leftarrow \text{not f\_unsat\_s}(w_i, i) \end{aligned} \quad (4)$$

and for any hard rule  $\alpha : F$  in  $F$ ,  $\mathbf{P}$  contains

$$\begin{aligned} \text{f\_unsat\_h}(i) &\leftarrow \text{not } R_i \\ R_i &\leftarrow \text{not f\_unsat\_h}(i). \end{aligned} \quad (5)$$

Or if  $R_i$  is already in the form

$$\text{Head}_i \leftarrow \text{Body}_i$$

allowed in the input language of CLINGO, instead of (4), we can also use <sup>6</sup>

$$\begin{aligned} \text{f\_unsat\_s}(w_i, i) &\leftarrow \text{Body}_i, \text{not Head}_i \\ \text{Head}_i &\leftarrow \text{not f\_unsat\_s}(w_i, i), \text{Body}_i \end{aligned}$$

and instead of (5),

$$\begin{aligned} \text{f\_unsat\_h}(1, i) &\leftarrow \text{Body}_i, \text{not Head}_i \\ \text{Head}_i &\leftarrow \text{not f\_unsat\_h}(1, i), \text{Body}_i. \end{aligned}$$

<sup>6</sup> In the case  $\text{Head}_i$  is a disjunction  $l_1; \dots; l_n$ , expression  $\text{not Head}_i$  stands for  $\text{not } l_1, \dots, \text{not } l_n$ .

$\mathbf{P}$  contains similar rules for each weighted formula in  $G$  using  $\text{g\_unsat\_s}(\dots)$  and  $\text{g\_unsat\_h}(\dots)$  atoms, as well as

$$\begin{aligned} \text{f\_pw\_s}(S) &\leftarrow S = \#sum\{X, Y : \text{f\_unsat\_s}(X, Y), Y = 1..i_f\} \\ \text{g\_pw\_s}(S) &\leftarrow S = \#sum\{X, Y : \text{g\_unsat\_s}(X, Y), Y = 1..i_g\} \\ \text{f\_pw\_h}(S) &\leftarrow S = \#count\{W : \text{f\_unsat\_h}(W), W = 1..i_f\} \\ \text{g\_pw\_h}(S) &\leftarrow S = \#count\{W : \text{g\_unsat\_h}(W), W = 1..i_g\}. \end{aligned}$$

( $i_f$  is the total number of rules in  $F$ , and  $i_g$  is the total number of rules in  $G$ ), and furthermore,

$$\neg p \tag{6}$$

for each atom  $p$  in  $\mathbf{p}$  to ensure that we consider  $X = \emptyset$ .

For example, for  $F$  and  $G$  in Example 2,  $\mathbf{P}$  is

```
not a :- not f_unsat_s(0, 1).
f_unsat_s(0, 1) :- not not a.
a :- not not a, not f_unsat_s(3, 2).
f_unsat_s(3, 2) :- not not a, not a.
b :- a, not f_unsat_s(2, 3).
f_unsat_s(2, 3) :- a, not b.
not a | b :- not g_unsat_s(2, 1).
g_unsat_s(2, 1) :- not not a, not b.
a :- not not a, not g_unsat_s(1, 2).
g_unsat_s(1, 2) :- not not a, not a.
f_pw_s(S) :- S = #sum{X, Y: f_unsat_s(X, Y), Y=1..3}.
g_pw_s(S) :- S = #sum{X, Y: g_unsat_s(X, Y), Y=1..2}.
not a.
not b.
```

$\mathbf{P}$  has a unique answer set, which tells us the potential parameters  $c_1$  and  $c_2$  for Conditions 1a and 1b each. If the answer set contains  $\{\text{f\_pw\_s}(x_1), \text{f\_pw\_h}(x_2), \text{g\_pw\_s}(y_1), \text{g\_pw\_h}(y_2)\}$  then let

$$c_1 = x_1 - y_1 \quad c_2 = x_2 - y_2.$$

Below we show how to check Condition 1 given  $c_1$  and  $c_2$  computed as above. Let  $\mathbf{P}^*$  is the program obtained from  $\mathbf{P}$  by removing rules (6) for all atom  $p \in \mathbf{p}$  and adding the following rules

$$\begin{aligned} &\leftarrow \text{f\_pw\_s}(X), \text{g\_pw\_s}(Y), X = Y + c_1 \\ &\leftarrow \text{f\_pw\_h}(X), \text{g\_pw\_h}(Y), X = Y + c_2. \end{aligned}$$

**Proposition 8** *Conditions 1a, 1b of Theorem 2' hold iff  $\mathbf{P}^*$  has no stable models.*

## 6.2 Checking the second condition of Theorem 2'

We check the second condition of Theorem 2' by checking if each of the following ASP program is unsatisfiable. Let  $\mathbf{p}$  be the set of all atoms occurring in  $F$  and  $G$ .

$\mathbf{P}_1^{**}$  is the following set of rules:

$$\{\{p\}^{\text{ch}} \mid p \in \mathbf{p}\} \cup \{\{p'\}^{\text{ch}} \mid p \in \mathbf{p}\} \cup \{p' \rightarrow p \mid p \in \mathbf{p}\} \cup \Delta_{\mathbf{p}'}(\{\bar{F}\}^{\text{ch}}) \cup \neg \Delta_{\mathbf{p}'}(\{\bar{G}\}^{\text{ch}}).$$

$\mathbf{P}_2^{**}$  is the following set of rules:

$$\{\{p\}^{\text{ch}} \mid p \in \mathbf{p}\} \cup \{\{p'\}^{\text{ch}} \mid p \in \mathbf{p}\} \cup \{p' \rightarrow p \mid p \in \mathbf{p}\} \cup \Delta_{\mathbf{p}'}(\{\overline{G}\}^{\text{ch}}) \cup \neg\Delta_{\mathbf{p}'}(\{\overline{F}\}^{\text{ch}}).$$

For example, for  $F$  in Example 2,  $\mathbf{P}_1^{**}$  in the input language of F2LP is as follows.

```
{a; aa; b; bb}.
```

```
aa -> a.
```

```
bb -> b.
```

```
% \Delta(\{F\}^{\text{ch}})
```

```
not a | not not a.
```

```
(aa -> bb) & ( a-> b) | not ( a-> b) .
```

```
(not not a->aa) & (not not a->a) | not (not not a->a) .
```

```
% not \Delta(\{G\}^{\text{ch}})
```

```
not ((not a | bb | not (not a | b)) & ((aa| not a) | not (a | not a))) .
```

**Proposition 9** *Condition 2 of Theorem 2' is true iff neither  $\mathbf{P}_1^{**}$  nor  $\mathbf{P}_2^{**}$  has stable models.*

The structural equivalence checking method is related to the strong equivalence checking method using SAT solvers in [4]. Paper [8] reports another system for automated equivalence checking.

## 7 Conclusion

In this paper, we defined the concept of strong equivalence for LP<sup>MLN</sup> programs and provide several equivalent characterizations. On the way, we have presented a few reformulations of LP<sup>MLN</sup> that give us useful insight.

The strong equivalence checking in Section 6 restricts soft rules' weights to integers only. We expect that this restriction can be removed if we use an external function call in CLINGO.

Building upon the results presented here, we plan to extend the work to approximate strong equivalence, where the probability distributions may not necessarily be identical but allowed to be slightly different with some error bounds. This would be more practically useful for LP<sup>MLN</sup> programs whose weights are learned from the data [13].

**Acknowledgements:** We are grateful to the anonymous referees for their useful comments. This work was partially supported by the National Science Foundation under Grant IIS-1815337.

## References

- [1] Evgenii Balai & Michael Gelfond (2016): *On the Relationship between P-log and LP<sup>MLN</sup>*. In: *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 915–921.
- [2] Chitta Baral, Michael Gelfond & J. Nelson Rushton (2009): *Probabilistic reasoning with answer sets*. *Theory and Practice of Logic Programming* 9(1), pp. 57–144, doi:10.1016/j.artint.2003.09.003.
- [3] Pedro Cabalar & Paolo Ferraris (2007): *Propositional theories are strongly equivalent to logic programs*. *Theory and Practice of Logic Programming* 7(6), pp. 745–759, doi:10.1016/S0743-1066(97)10001-2.

- [4] Yin Chen, Fangzhen Lin & Lei Li (2005): *SELP - A System for Studying Strong Equivalence Between Logic Programs*. In: *LPNMR*, pp. 442–446.
- [5] Luc De Raedt, Angelika Kimmig & Hannu Toivonen (2007): *ProbLog: A Probabilistic Prolog and Its Application in Link Discovery*. In: *IJCAI*, 7, pp. 2462–2467.
- [6] Paolo Ferraris (2005): *Answer sets for propositional theories*. In: *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pp. 119–131, doi:10.1007/11546207\_10.
- [7] Paolo Ferraris, Joohyung Lee & Vladimir Lifschitz (2011): *Stable models and circumscription*. *Artificial Intelligence* 175, pp. 236–263, doi:10.1016/j.artint.2010.04.011.
- [8] Tomi Janhunen & Emilia Oikarinen (2004): *LPEQ and DLPEQ? translators for automated equivalence testing of logic programs*. In: *International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer, pp. 336–340, doi:10.1007/978-3-540-24609-1\_30.
- [9] Joohyung Lee & Ravi Palla (2009): *System F2LP – Computing Answer Sets of First-Order Formulas*. In: *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pp. 515–521, doi:10.1007/11546207\_36.
- [10] Joohyung Lee, Samidh Talsania & Yi Wang (2017): *Computing LPMLN using ASP and MLN solvers*. *Theory and Practice of Logic Programming*, doi:10.1017/S1471068417000400.
- [11] Joohyung Lee & Yi Wang (2016): *Weighted Rules under the Stable Model Semantics*. In: *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 145–154.
- [12] Joohyung Lee & Yi Wang (2018): *A Probabilistic Extension of Action Language  $\mathcal{BC}^+$* . *Theory and Practice of Logic Programming* 18(3–4), pp. 607–622, doi:10.1017/S1471068403001807.
- [13] Joohyung Lee & Yi Wang (2018): *Weight Learning in a Probabilistic Extension of Answer Set Programs*. In: *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 22–31.
- [14] Joohyung Lee & Zhun Yang (2017): *LPMLN, Weak Constraints, and P-log*. In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1170–1177.
- [15] Vladimir Lifschitz, David Pearce & Agustin Valverde (2001): *Strongly equivalent logic programs*. *ACM Transactions on Computational Logic* 2, pp. 526–541, doi:10.1145/383779.383783.
- [16] David Pearce (2006): *Equilibrium logic*. *Annals of Mathematics and Artificial Intelligence* 47(1-2), pp. 3–41, doi:10.1007/s10472-006-9028-z.
- [17] Judea Pearl (2000): *Causality: models, reasoning and inference*. 29, Cambridge Univ Press, doi:10.1017/CBO9780511803161.
- [18] Matthew Richardson & Pedro Domingos (2006): *Markov logic networks*. *Machine Learning* 62(1-2), pp. 107–136, doi:10.1007/s10994-006-5833-1.
- [19] Bin Wang & Zhizheng Zhang (2017): *A Parallel LPMLN Solver: Primary Report*. In: Working Notes of the Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP).