Strong Equivalence for $\mathrm{LP}^{\mathrm{MLN}}$ Programs

Joohyung Lee and Man Luo

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, USA
{joolee,mluo26}@asu.edu

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

1 Introduction

$\mathrm{LP}^{\mathrm{MLN}}$ is a probabilistic extension of answer set programs with the weight scheme adapted from Markov Logic [1]. An $\mathrm{LP}^{\mathrm{MLN}}$ program defines the 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 their probabilities.

The language turns out to be highly expressive to embed several other probabilistic logic languages, such as P-log [2], ProbLog [3], Markov Logic, and Causal Models [4], as described in [5–7]. Inference engines for $\mathrm{LP}^{\mathrm{MLN}}$, such as LPMLN2ASP, LPMLN2MLN [8], and LPMLN-MODELS [9], have been developed based on the reduction of $\mathrm{LP}^{\mathrm{MLN}}$ to answer set programs and Markov Logic. $\mathrm{LP}^{\mathrm{MLN}}$ is a basis of probabilistic action language $p\mathcal{BC}+$ [10], which is defined as a high-level notation of $\mathrm{LP}^{\mathrm{MLN}}$ to describe the probabilistic transition systems.

As more results are built upon $\mathrm{LP}^{\mathrm{MLN}}$, it becomes more important to consider the equivalence between different $\mathrm{LP}^{\mathrm{MLN}}$ programs. As with answer set programs, $\mathrm{LP}^{\mathrm{MLN}}$ 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 $\mathrm{LP}^{\mathrm{MLN}}$ is important for $\mathrm{LP}^{\mathrm{MLN}}$ programming to simplify a part of an $\mathrm{LP}^{\mathrm{MLN}}$ program without looking at the rest of it and to verify the correctness of $\mathrm{LP}^{\mathrm{MLN}}$ for the representation. For instance, the following rules appearing in any program

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

can be replaced by a simpler rule

$$w_1 : a$$
$$w_2 : b$$

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 LP$^{\mathrm{MLN}}$. First, weights 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 LP$^{\mathrm{MLN}}$ programs. For instance, $\{a \vee b,\ \ \bot \leftarrow a, b\}$ is strongly equivalent to $\{a \leftarrow \mathit{not}\ b,\ \ b \leftarrow \mathit{not}\ a,\ \ \bot \leftarrow a, b\}$, but its LP$^{\mathrm{MLN}}$ counterpart $\{\alpha : a \vee b,\ \ \alpha : \bot \leftarrow a, b\}$ is not strongly equivalent to $\{\alpha : a \leftarrow \mathit{not}\ b,\ \ \alpha : b \leftarrow \mathit{not}\ a,\ \ \alpha : \bot \leftarrow a, b\}$: 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 $\alpha : \bot \leftarrow a, b$) but not of the latter (**c.f.** Example 1).

We extend the notion of strong equivalence to LP$^{\mathrm{MLN}}$, and show that the verification of strong equivalence in LP$^{\mathrm{MLN}}$ 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 strong equivalence. The study of strong equivalence suggests us a few reformulations of the LP$^{\mathrm{MLN}}$ semantics using choice rules, logic of here and there, and second-order 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 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 LP$^{\mathrm{MLN}}$

We first review the definition of a (deterministic) stable model for a propositional formula [11]. 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 $\bot$. 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 LP$^{\mathrm{MLN}}$ from [5]. An LP$^{\mathrm{MLN}}$ program is a finite set of weighted formulas $w : R$ where R is a propositional formula [1] and w is a real number (in which case, the weighted rule is called *soft*) or α for denoting the infinite weight (in which case, the weighted rule is called *hard*).

For any LP$^{\mathrm{MLN}}$ program F and any set X of atoms, $\overline{\mathsf{F}}$ 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 $X \models R$.

[1] 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.

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

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

By $TW(\mathsf{F})$ ("Total Weight" of F) we denote the expression $exp(\sum_{w:R\in\mathsf{F}} w)$. For any interpretation X, the weight of an interpretation X, denoted $W_{\mathsf{F}}(X)$, is defined as[2]

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

and the probability of X, denoted $P_{\mathsf{F}}(X)$, is defined as

$$P_{\mathsf{F}}(X) = \lim_{\alpha\to\infty} \frac{W_{\mathsf{F}}(X)}{\sum_{Y\in\mathrm{SM}[\mathsf{F}]} W_{\mathsf{F}}(Y)}.$$

Alternatively, the weight can be defined by counting the penalty of the interpretation [8]. 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 $\overline{\mathsf{F}_X}$). Let

$$W^{\mathrm{pnt}}_{\mathsf{F}}(X) = \begin{cases} (TW(\mathsf{F} \setminus \mathsf{F}_X))^{-1} & \text{if } X \in \mathrm{SM}[\mathsf{F}]; \\ 0 & \text{otherwise} \end{cases}$$

and

$$P^{\mathrm{pnt}}_{\mathsf{F}}(X) = \lim_{\alpha\to\infty} \frac{W^{\mathrm{pnt}}_{\mathsf{F}}(X)}{\sum_{Y\in\mathrm{SM}[\mathsf{F}]} W^{\mathrm{pnt}}_{\mathsf{F}}(Y)}.$$

The following theorem tells us that the $\mathrm{LP}^{\mathrm{MLN}}$ semantics can be reformulated using the concept of a penalty-based weight.

Theorem 1 ?*pty For any* $\mathrm{LP}^{\mathrm{MLN}}$ *program* F *and any interpretation* X*,*

$$W_{\mathsf{F}}(X) = TW(\mathsf{F}) \times W^{\mathrm{pnt}}_{\mathsf{F}}(X),$$
$$P_{\mathsf{F}}(X) = P^{\mathrm{pnt}}_{\mathsf{F}}(X).$$

2.2 Review: Logic of Here and There

Logic of here and there (Logic HT) is proven to be useful for a monotonic basis for checking strong equivalence [12] and equilibrium models [13] 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:

[2] We identify an interpretation with the set of atoms true in it.

- 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 $F \in X$.
- $\langle Y, X, w\rangle \not\models_{\text{ht}} \bot$.
- $\langle Y, X, w\rangle \models_{\text{ht}} F \wedge G$ if

$$\langle Y, X, w\rangle \models_{\text{ht}} F \text{ and } \langle Y, X, w\rangle \models_{\text{ht}} G.$$

- $\langle Y, X, w\rangle \models_{\text{ht}} F \vee G$ if

$$\langle Y, X, w\rangle \models_{\text{ht}} F \text{ or } \langle Y, X, w\rangle \models_{\text{ht}} G.$$

- $\langle Y, X, w\rangle \models_{\text{ht}} F \to G$ if for every world such that $w \leq w'$,

$$\langle Y, X, w'\rangle \not\models_{\text{ht}} F \text{ or } \langle Y, X, w'\rangle \models_{\text{ht}} G.$$

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 $F \vee (F \to G) \vee \neg G$. From the deduction system, we can derive the weak law of excluded middle $\neg F \vee \neg\neg F$.

Theorem 1 from [12] shows that strong equivalence between two answer set programs coincides with equivalence in the logic HT. The deduction rules above can be used for checking strong equivalence.

3 Strong Equivalence in LP^{MLN}

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

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

$$P_{\mathsf{F}}(X) = P_{\mathsf{G}}(X)$$

for all interpretations X.

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

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

for all interpretations X.

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

Example 1. Consider two programs F and G [3]

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

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

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

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

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 $\mathrm{LP}^{\mathrm{MLN}}$ program H,

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

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., refer to the equation

$$W^{\mathrm{pnt}}_{\mathsf{F}\cup\mathsf{H}}(X) = c \times W^{\mathrm{pnt}}_{\mathsf{G}\cup\mathsf{H}}(X)$$

in place of (1).

Due to the weight scheme, every interpretation that has a non-zero weight is a soft stable model. Thus Definition 4 implies that the $\mathrm{LP}^{\mathrm{MLN}}$ programs are "structurally equivalent" to each other, which is defined as follows.

Definition 5. $\mathrm{LP}^{\mathrm{MLN}}$ *programs* F *and* G *are* structurally equivalent *if, for any* $\mathrm{LP}^{\mathrm{MLN}}$ *program* H*, programs* $\mathsf{F} \cup \mathsf{H}$ *and* $\mathsf{G} \cup \mathsf{H}$ *have the same set of soft stable models.*

Strong equivalence implies structural equivalence, but not vice versa.

Proposition 1 *If* $\mathrm{LP}^{\mathrm{MLN}}$ *programs* F *and* G *are strongly equivalent, then they are structurally equivalent as well.*

The fact that $\mathrm{LP}^{\mathrm{MLN}}$ programs F and G are structurally equivalent does not follow from the fact that ASP programs $\overline{\mathsf{F}}$ and $\overline{\mathsf{G}}$ are strongly equivalent.

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

The following theorem shows a characterization of strong equivalence that does not need to consider adding all possible $\mathrm{LP}^{\mathrm{MLN}}$ programs H. Similar to Proposition 2 from [11], it shows that the verification of strong equivalence in $\mathrm{LP}^{\mathrm{MLN}}$ can be reduced to equivalence checking in classical logic plus weight checking.

[3] We identify $F \leftarrow G$ with $G \rightarrow F$ and $\leftarrow F$ with $F \rightarrow \bot$.

Theorem 2 *For any* $\mathrm{LP}^{\mathrm{MLN}}$ *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. $TW(\mathsf{F}_X) = c \times TW(\mathsf{G}_X)$*, and*
2. $(\overline{\mathsf{F}_X})^X$ *and* $(\overline{\mathsf{G}_X})^X$ *are classically equivalent.*

Recall that $TW(\mathsf{F}_X)$ is simply an exponentiated sum of the weights of the rules that are true in X. Thus checking the first condition of Theorem 2 does not need to consider whether X is a soft stable model or not. In view of Theorem 1, one could also equivalently add up the weights of rules that are not true in X and exponentiate the sum, i.e., Condition 1 of Theorem 2 can be replaced with

1'. $TW(\mathsf{F} \setminus \mathsf{F}_X) = c \times TW(\mathsf{G} \setminus \mathsf{G}_X)$.

Example 2. Consider two programs

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

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	$TW(\mathsf{F}_X)$	$TW(\mathsf{G}_X)$	$(\overline{\mathsf{F}_X})^X$	$(\overline{\mathsf{G}_X})^X$
ϕ	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. $(\overline{\mathsf{F}_X})^X$ and $(\overline{\mathsf{G}_X})^X$

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

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

$\{a, b\}$ is a soft stable model of $\mathsf{G} \cup \mathsf{H}$ with the weight e^5, but it is not a soft stable model $\mathsf{F}' \cup \mathsf{H}$, so its weight is 0. In accordance with Theorem 2, $(\overline{\mathsf{F}'_{\{a,b\}}})^{\{a,b\}}$ is not equivalent to $(\overline{\mathsf{G}_{\{a,b\}}})^{\{a,b\}}$. 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 $TW(\mathsf{F}_\phi) = e^1 \times TW(\mathsf{G}'_\phi)$ and $TW(\mathsf{F}_{\{a\}}) = e^2 \times TW(\mathsf{G}'_{\{a\}})$, so there is no constant c such that $TW(\mathsf{F}_X) = c \times TW(\mathsf{G}'_X)$.

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

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

The following fact can also be useful for simplification.

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

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

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

$$(F \to G) \to K$$

is strongly equivalent to

$$\begin{array}{l}(G \vee \neg F) \to K \\ K \vee F \vee \neg G\end{array}$$

is a key lemma to prove that any propositional formulas can be turned into the logic program syntax [14]. 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 $\mathrm{LP}^{\mathrm{MLN}}$, i.e., there are some formulas F, G, K such that

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

is not strongly equivalent to

$$\begin{array}{l}w_1 : (G \vee \neg F) \to K \\ w_2 : K \vee F \vee \neg G\end{array} \tag{3}$$

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

$$\begin{array}{c}(((F \to G) \to K)_X)^X \Leftrightarrow \bot \\ ((\{(G \vee \neg F) \to K, K \vee F \vee \neg G\})_X)^X \Leftrightarrow F^X.\end{array}$$

So condition 2 of Theorem 2 does not hold, and it follows that (2) is not strongly equivalent to (3).

3.1 Reformulation of $\mathrm{LP}^{\mathrm{MLN}}$ Using Choice Rules

The second condition of Theorem 2 is equivalent to the fact that for any $\mathrm{LP}^{\mathrm{MLN}}$ program H, programs $\mathsf{F} \cup \mathsf{H}$ and $\mathsf{G} \cup \mathsf{H}$ 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 Γ of propositional formulas, $\{\Gamma\}^{\mathrm{ch}}$ denotes the set of choice formulas $\{\{F\}^{\mathrm{ch}} \mid F \in \Gamma\}$.

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

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

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

2'. $(\{\overline{\mathsf{F}}\}^{\mathrm{ch}})^X$ and $(\{\overline{\mathsf{G}}\}^{\mathrm{ch}})^X$ 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 $\mathrm{LP}^{\mathrm{MLN}}$ semantics.

Proposition 4 *For any* $\mathrm{LP}^{\mathrm{MLN}}$ *program* F*,* X *is a soft stable model of* F *iff* X *is a soft stable model of* $\{\overline{\mathsf{F}}\}^{\mathrm{ch}}$.

4 Soft Logic of Here and There

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

Definition 6. *An* HT *interpretation* $\langle Y, X\rangle$ *is called a* soft HT model *of an* $\mathrm{LP}^{\mathrm{MLN}}$ *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* $\overline{\mathsf{F}_X}$.

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* $\mathrm{LP}^{\mathrm{MLN}}$ *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,* $\overline{\mathsf{F}_X} \leftrightarrow \overline{\mathsf{G}_X}$ *is provable in logic* HT.
(f) $(\{\overline{\mathsf{F}}\}^{\mathrm{ch}}) \leftrightarrow (\{\overline{\mathsf{G}}\}^{\mathrm{ch}})$ *is provable in* HT.

Again, one 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.*
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 $\mathrm{LP}^{\mathrm{MLN}}$ programs by using deduction rules in HT.

Example 3. Consider $\mathrm{LP}^{\mathrm{MLN}}$ programs F and G:

$$\mathsf{F}\ \ 2 : \neg a \vee b \qquad \mathsf{G}\ \ 2 : \neg\neg a \rightarrow b$$

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

$$\{\overline{\mathsf{F}}\}^{\mathrm{ch}} = (\neg a \vee b) \vee \neg(\neg a \vee b)$$
$$\{\overline{\mathsf{G}}\}^{\mathrm{ch}} = (\neg\neg a \rightarrow b) \vee \neg(\neg\neg a \rightarrow b)$$

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'

Left-to-right: Assume $(\neg a \vee b) \vee \neg(\neg a \vee b)$.
Case 1: $(\neg a \vee b)$. Then $\neg\neg a \to b$ is intuitionistically derivable, so derivable in logic HT as well.
Case 2: $\neg(\neg a \vee b)$. Then $\neg(\neg\neg a \to b)$ is intuitionistically derivable (Glivenko's Theorem).

Right-to-left: Assume $(\neg\neg a \to b) \vee \neg(\neg\neg a \to b)$.
Case 1: $\neg\neg a \to b$. Then $\neg a \vee b$ can be derived from the weak law of excluded middle $\neg a \vee \neg\neg a$.
Case 2: $\neg(\neg\neg a \to b)$. 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* $\mathrm{LP}^{\mathrm{MLN}}$ *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 $\overline{\mathsf{F}_X}$ because this condition is trivially satisfied.

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

Lemma 1. *For any* $\mathrm{LP}^{\mathrm{MLN}}$ *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 $(\overline{\mathsf{F}_X})^X$.
(c) Y satisfies $(\{\overline{\mathsf{F}}\}^{\mathrm{ch}})^X$.

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* $\phi,\ \{a\},\ \{a,b\}$. *And Table 2 shows that* F *and* G *have three equilibrium models, which are* $\langle\phi,\phi\rangle,\ \langle\{a\}\,\{a\}\rangle,\ \langle\{a,b\},\{a,b\}\rangle$. *On the other hand,* F' *has only one equilibrium model,* $\langle\phi,\phi\rangle$, *which provides another aspect to show* F' *and* G *have different soft stable models.*

The weight of the soft equilibrium models could be defined in the same way as in soft stable models as 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 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 $p \in \mathbf{p}$;
- $\Delta_{\mathbf{p}'}(\neg F) = \neg F$;
- $\Delta_{\mathbf{p}'}(F \wedge G) = \Delta_{\mathbf{p}'}(F) \wedge \Delta_{\mathbf{p}'}(G)$;
- $\Delta_{\mathbf{p}'}(F \vee G) = \Delta_{\mathbf{p}'}(F) \vee \Delta_{\mathbf{p}'}(G)$;
- $\Delta_{\mathbf{p}'}(F \to G) = (\Delta_{\mathbf{p}'}(F) \to \Delta_{\mathbf{p}'}(G)) \wedge (F \to G)$.

Lemma 1 is extended to Δ 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* $\Delta_{\mathbf{P}'}(\overline{\mathsf{F}_X})$.
(e) $Y' \cup X$ *satisfies* $\Delta_{\mathbf{P}'}(\{\overline{\mathsf{F}}\}^{\mathrm{ch}})$.

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

(a) For any $\mathrm{LP}^{\mathrm{MLN}}$ *program* H, *programs* $\mathsf{F} \cup \mathsf{H}$ *and* $\mathsf{G} \cup \mathsf{H}$ *have the same soft stable models.*
(g) For any set X *of atoms,* $\{p' \to p \mid p \in \mathbf{p}\}$ *entails* $\Delta_{\mathbf{P}'}(\overline{\mathsf{F}_X}) \leftrightarrow \Delta_{\mathbf{P}'}(\overline{\mathsf{G}_X})$ *(in the sense of classical logic).*
(h) $\{p' \to p \mid p \in \mathbf{p}\}$ *entails* $\Delta_{\mathbf{P}'}(\{\overline{\mathsf{F}}\}^{\mathrm{ch}}) \leftrightarrow \Delta_{\mathbf{P}'}(\{\overline{\mathsf{G}}\}^{\mathrm{ch}})$ *(in the sense of classical logic).*

The equivalence between Condition (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 $\mathrm{LP}^{\mathrm{MLN}}$ programs is no harder than checking strong equivalence between standard logic programs. In conjunction with Condition 1 of Theorem 2, the complexity of $\mathrm{LP}^{\mathrm{MLN}}$ strong equivalence checking is no harder than checking strong equivalence for standard logic programs.

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

5.1 Reformulation of $\mathbf{LP^{MLN}}$ Using $\boldsymbol{\Delta}$

The following proposition relates Δ to soft stable models.

Proposition 6 *For any* $\mathrm{LP}^{\mathrm{MLN}}$ *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* $\Delta_{\mathbf{p}'}(\{\overline{\mathsf{F}}\}^{\mathrm{ch}})$.

The definition of Δ is similar to the definition of F^* used in the second-order logic based definition of a stable model from [15]. This leads to the following reformulation of $\mathrm{LP}^{\mathrm{MLN}}$ in terms of 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* $\mathrm{LP}^{\mathrm{MLN}}$ *program* F*, a set* X *of atoms is a soft stable model of* F *iff* X *satisfies*

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

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

Theorem 2′ *For any* $\mathrm{LP}^{\mathrm{MLN}}$ *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. $\displaystyle\sum_{\substack{w:R \in \mathsf{F}, w\neq\alpha, \\ \text{and } X\not\models R}} w \;=\; c_1 + \sum_{\substack{w:R \in \mathsf{G}, w\neq\alpha, \\ \text{and } X\not\models R}} w;$

1b. $|\{\alpha : R \in \mathsf{F} \mid X \not\models R\}| = c_2 \times |\{\alpha : R \in \mathsf{G} \mid X \not\models R\}|$;

2. $\{p' \rightarrow p \mid p \in \mathbf{p}\}$ *entails* $\Delta(\{\overline{\mathsf{F}}\}^{\mathrm{ch}}) \leftrightarrow \Delta(\{\overline{\mathsf{G}}\}^{\mathrm{ch}})$ *(in the sense of classical logic).*

In each of the following subsections, we show how to check the conditions using CLINGO together with F2LP [16]. 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 Condition 1, we need to find a potential value for c. For that, we choose $X = \emptyset$ and find the value of c. If the same c makes the equation true for all other interpretations as well, the condition is true.

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

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

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

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

Or if R_i is already in the form

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

that in the input language of CLINGO, instead of (4), we can also use [4]

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

and instead of (5),

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

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

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

(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.
```

[4] In the case Head_i is a disjunction $l_1; \cdots; l_n$, expression $\mathit{not}\ \mathit{Head}_i$ stands for $\mathit{not}\ l_1, \cdots, \mathit{not}\ l_n$.

$\mathbf{P}$ 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 $\{\mathtt{f\_pw\_s}(x_1), \mathtt{f\_pw\_h}(x_2), \mathtt{g\_pw\_s}(y_1), \mathtt{g\_pw\_h}(y_2)\}$ then let

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

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

$$\leftarrow \mathtt{f\_pw\_s}(X), \mathtt{g\_pw\_s}(Y), X = Y + c_1$$
$$\leftarrow \mathtt{f\_pw\_h}(X), \mathtt{g\_pw\_h}(X)), X = Y + c_2.$$

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 that 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:

$$\{\{p\}^{\mathrm{ch}} \mid p \in \mathbf{p}\} \cup \{\{p'\}^{\mathrm{ch}} \mid p \in \mathbf{p}\} \cup \{p' \to p \mid p \in \mathbf{p}\} \cup\ \Delta_{\mathbf{P}'}(\{\overline{\mathsf{F}}\}^{\mathrm{ch}}) \cup \neg\Delta_{\mathbf{P}'}(\{\overline{G}\}^{\mathrm{ch}}).$$

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

$$\{\{p\}^{\mathrm{ch}} \mid p \in \mathbf{p}\} \cup \{\{p'\}^{\mathrm{ch}} \mid p \in \mathbf{p}\} \cup \{p' \to p \mid p \in \mathbf{p}\} \cup\ \Delta_{\mathbf{P}'}(\{\overline{G}\}^{\mathrm{ch}}) \cup\ \neg\Delta_{\mathbf{P}'}(\{\overline{\mathsf{F}}\}^{\mathrm{ch}}).$$

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}^{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}^{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 [17]. [18] reports another system for automated equivalence checking.

7 Conclusion

In this paper, we defined the concept of strong equivalence for $\mathrm{LP}^{\mathrm{MLN}}$ programs and provide several equivalent characterizations. On the way, we have presented a few reformulations of $\mathrm{LP}^{\mathrm{MLN}}$ 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 bound.

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. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2) (2006) 107–136
2. Baral, C., Gelfond, M., Rushton, J.N.: Probabilistic reasoning with answer sets. Theory and Practice of Logic Programming **9**(1) (2009) 57–144
3. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: IJCAI. Volume 7. (2007) 2462–2467
4. Pearl, J.: Causality: models, reasoning and inference. Volume 29. Cambridge Univ Press (2000)
5. Lee, J., Wang, Y.: Weighted rules under the stable model semantics. In: Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR). (2016) 145–154
6. Balai, E., Gelfond, M.: On the relationship between P-log and $\mathrm{LP}^{\mathrm{MLN}}$. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2016) 915–921
7. Lee, J., Yang, Z.: LPMLN, weak constraints, and P-log. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). (2017) 1170–1177
8. Lee, J., Talsania, S., Wang, Y.: Computing LPMLN using ASP and MLN solvers. Theory and Practice of Logic Programming (2017)
9. Wang, B., Zhang, Z.: A parallel LPMLN solver: Primary report. In: *Working Notes of the Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP)*. (2017)
10. Lee, J., Wang, Y.: A probabilistic extension of action language $\mathcal{BC}$+. Theory and Practice of Logic Programming **18(3–4)** (2018) 607–622
11. Ferraris, P.: Answer sets for propositional theories. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 119–131
12. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2** (2001) 526–541
13. Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence **47**(1-2) (2006) 3–41
14. Cabalar, P., Ferraris, P.: Propositional theories are strongly equivalent to logic programs. Theory and Practice of Logic Programming **7**(6) (2007) 745–759
15. Ferraris, P., Lee, J., Lifschitz, V.: Stable models and circumscription. Artificial Intelligence **175** (2011) 236–263
16. Lee, J., Palla, R.: System F2LP – computing answer sets of first-order formulas. In: Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2009) 515–521

17. Chen, Y., Lin, F., Li, L.: Selp - a system for studying strong equivalence between logic programs. In: LPNMR. (2005) 442–446
18. Janhunen, T., Oikarinen, E.: Lpeq and dlpeqtranslators for automated equivalence testing of logic programs. In: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer (2004) 336–340