Equivalence of Logic Programs under Updates

Katsumi Inoue

National Institute of Informatics

Chiaki Sakama

Wakayama University

JELIA 2004. 09. 29

Answer Set Programming

A **program** is regarded as the ***constraints*** to be satisfied by **solutions**. Each solution is obtained by computing an ***answer set*** of the program.

Program: $r \leftarrow p$

$r \leftarrow q$

$p \leftarrow$ ***not*** q

$q \leftarrow$ ***not*** p

Answer Sets: $\{ p , r \}$, $\{ q , r \}$

Strong Equivalence in ASP

[Maher 88], [Lifschitz, Pearce & Valverde 01]

- Programs P_1 and P_2 are **strongly equivalent** if for any program R, $P_1 \cup R$ and $P_2 \cup R$ have the same answer sets.
- *Strong equivalence can be used to simplify a part of logic program without looking at the other part.*
- E.g. P_1 : $p \leftarrow \textbf{not}\ q.$ $\quad \leftarrow q.$

  P_2 : $p \leftarrow.$ $\quad \leftarrow q.$

  are strongly equivalent.

Equivalence under Negative Influence

P_1 : *suspect* ← *motivated*, ***not*** *alibi* .

motivated ← .

alibi ← .

is strongly equivalent to

P_2 : *motivated* ← .

alibi ← .

- Suppose *alibi* is proved to be false. From the simplified program P_2 we cannot infer *suspect* any more.
- Simplifying programs should be more careful.

Asymmetry in Strong Equivalence

P_1 : $q \leftarrow p.$ $p \leftarrow .$

P_2 : $q \leftarrow p.$ $p \leftarrow$ ***not*** $r.$

P_3 : $q \leftarrow .$ $p \leftarrow .$

- P_1 , P_2 , P_3 are all equivalent (= have the same answer sets).
- P_1 and P_2 are not strongly equivalent.
- P_1 and P_3 are strongly equivalent.
- With strong equivalence, negative dependency is sensitive to difference, while positive dependency may be ignored.
- We consider an equivalence criterion which does not distinguish positive and negative dependency.

Equivalence under Updates

- The process of program development is dynamic. Strong equivalence is not suitable for program simplification in dynamic environments.
- We provide an equivalence criterion which is tolerant to both assertion and removal of rules: **update equivalence**.
- We want to analyze the effect of removal in equivalence criteria. E.g., how two update equivalent programs are different from each other?
- Often, the language of updates is restricted to a subset of the whole language: **relative equivalence**.

Contents

- ASP and Equivalence of Logic Programs
- Equivalence in Dynamic Environments
- Relative Equivalence
- Strong/Common/Uniform Update Equivalence
- Relative Update Equiv. ⇒ Relative Strong Equiv.
- Complexity Results

(General Extended Disjunctive) Programs

⌘ Rule ***r*** :

$$L_1 ; \ldots ; L_k ; \textbf{not } L_{k+1} ; \ldots ; \textbf{not } L_l \leftarrow L_{l+1}, \ldots, L_m, \textbf{not } L_{m+1}, \ldots, \textbf{not } L_n$$

⌘ $\text{head}^+(r) = \{L_1, \ldots, L_k\}$, $\text{head}^-(r) = \{L_{k+1}, \ldots, L_l\}$,

$\text{body}^+(r) = \{L_{l+1}, \ldots, L_m\}$, $\text{body}^-(r) = \{L_{m+1}, \ldots, L_n\}$.

⌘ Integrity constraint (**IC**): $k=0$.

⌘ Fact: $k=l=m=n$.

⌘ Extended disjunctive program (**EDP**): $\forall r$: $k=l$.

⌘ Normal logic program (**NLP**): $\forall r$: $k=l=1$ & $\forall L_i$: atom.

Answer Sets (1)

⌘ *Answer set semantics* [Gelfond, Lifschitz, Woo]

I. When P is a GEDP without ***not*** ($k=l$ & $m=n$) ,

S is an **answer set** of P if

S is a minimal set satisfying the conditions:

1. For each ground rule ***r*** from P :

$$L_1 ; \ldots ; L_l \leftarrow L_{l+1} , \ldots , L_m ,$$

$\{L_{l+1} , \ldots , L_m\} \subseteq \boldsymbol{S}$ implies $\{L_1 , \ldots , L_l\} \cap \boldsymbol{S} \neq \phi$;

2. If ***S*** contains a pair of complementary literals,

then ***S*** = ***Lit***.

Answer Sets (2)

II. When P is any GEDP,

the GEDP (without ***not***) P^S is obtained as follows:

A rule

$$L_1 ; \ldots ; L_k \leftarrow L_{l+1}, \ldots, L_m$$

is in P^S iff there is a ground rule from P of the form

$$L_1 ; \ldots ; L_k ; \textbf{not}\ L_{k+1} ; \ldots ; \textbf{not}\ L_l \leftarrow L_{l+1}, \ldots, L_m, \textbf{not}\ L_{m+1}, \ldots, \textbf{not}\ L_n$$

s.t. $\{L_{k+1}, \ldots, L_l\} \subseteq \boldsymbol{S}$ and $\{L_{m+1}, \ldots, L_n\} \cap \boldsymbol{S} = \phi$.

$\boldsymbol{S}$ is an **answer set** of P if $\boldsymbol{S}$ is an answer set of P^S.

Answer Sets (3)

- An answer set is **consistent** if it is not ***Lit***.
- A program is **consistent** if it has a consistent answer set.
- The set of all answer sets of P is denoted as $AS(P)$.
- $\boldsymbol{S} \in AS(P)$ is **minimal** if $\forall \boldsymbol{S'} \in AS(P).\ \boldsymbol{S'} \subseteq \boldsymbol{S} \rightarrow \boldsymbol{S} \subseteq \boldsymbol{S'}$.
- Every answer set of any EDP is minimal [Gelfond & Lifschitz].
- Answer sets may not be minimal for GEDPs.

Equivalence between Programs

- Let P_1, P_2 and R be programs.
- P_1 and P_2 are **(weakly) equivalent** if $AS(P_1) = AS(P_2)$.
- P_1 and P_2 are **equivalent relative to** R if $AS(P_1 \cup R) = AS(P_2 \cup R)$.
- P_1 and P_2 are **strongly equivalent** [Lifschitz, Pearce & Valverde] if they are equivalent relative to any program R.
- Strong equivalence implies weak equivalence.

Relative Update Equivalence

- Let P_1 and P_2 be programs.
- Let $\mathcal{Q}$ and $\mathcal{R}$ be sets of rules called **updatable rules**.

  $\mathcal{Q}$ is a set of **removable rules**, and $\mathcal{R}$ is a set of **insertable rules**.
- P_1 and P_2 are **update equivalent with respect to** $(\mathcal{Q}, \mathcal{R})$

  if $AS((P_1 \setminus Q) \cup R) = AS((P_2 \setminus Q) \cup R)$ for any $Q \subseteq \mathcal{Q}$ and $R \subseteq \mathcal{R}$.
- *Update equivalence* represents equivalence in dynamic settings where common changes are made for two programs.
- *Relative equivalence* restricts the language of updates.
- P_1 and P_2 are **strong equivalent with respect to** $\mathcal{R}$

  if $AS(P_1 \cup R) = AS(P_2 \cup R)$ for any $R \subseteq \mathcal{R}$.

Equivalence in the Literature

- Update equivalence generalizes strong equivalence, in both *dynamic* and *relative* settings.
- Eiter *et al.* [LPAR'01]
  - generalize strong equivalence in the context of updates;
  - do not consider the effect of removal.
- Leite [2003]
  - considers **update equivalence in Dynamic LP**;
  - does not generalize strong equivalence in ASP.
- Lin [KR'02], Woltran [JELIA'04]
  - consider **relativized equivalence**, i.e., strong equivalence wrt programs mentioning only a distinguished set of atoms;
  - do not consider the effect of removal.

Strong Update Equivalence

- Let P_1, P_2, Q and R be programs.
- P_1 and P_2 are **strongly update equivalent** (S-update equiv.)
  if $AS((P_1 \setminus Q) \cup R) = AS((P_2 \setminus Q) \cup R)$ for any Q and R.
- E.g. $\{ p \leftarrow p \}$ and Φ are S-update equiv.
- S-update equivalence implies strong equivalence: $Q = \Phi$.
- P_1 and P_2 are S-update equivalent
  iff P_1 and P_2 are update equivalent wrt $(\mathcal{P}, \mathcal{P})$
  iff P_1 and P_2 are update equivalent wrt $(P_1 \cup P_2, \mathcal{P})$
  where $\mathcal{P}$ is the set of all rules in the language.

Characterizing S-update Equiv.

- A rule r is **valid** if $\{r\}$ and Φ are strongly equivalent.
- **Symmetric difference:** $P_1 \Delta P_2 = (P_1 \setminus P_2) \cup (P_2 \setminus P_1)$.
- **Theorem.** P_1 and P_2 are strongly update equivalent iff $P_1 \Delta P_2$ is a set of valid rules.
- **Theorem.** A rule r is valid iff either of the following holds:
  - $\text{head}^+(r) \cap \text{body}^+(r) \neq \emptyset$ [tautology]
  - $\text{head}^-(r) \cap \text{body}^-(r) \neq \emptyset$ [tautology*]
  - $\text{body}^+(r) \cap \text{body}^-(r) \neq \emptyset$ [contradictory]
  - $\text{head}^+(r) \cup \text{body}^-(r) \neq \emptyset$ and $\exists L,M \in \text{head}^-(r) \cup \text{body}^+(r)$ such that $L = \neg M$.
- E.g. $p \leftarrow p.$ ***not*** $p \leftarrow$ ***not*** $p.$ $q \leftarrow p,$ ***not*** $p.$ $q \leftarrow p, \neg p.$ are valid.

Update Equivalence on Common Rules

- Let P_1, P_2, Q and R be programs.
- P_1 and P_2 are **update equivalent on common rules** (C-update equiv.) if $AS((P_1 \setminus Q) \cup R) = AS((P_2 \setminus Q) \cup R)$ for any R and any Q such that $Q \subseteq P_1 \cap P_2$.
- E.g. { ***not*** $p \leftarrow q$ } and { $\leftarrow p,q$ } are C-update equiv.
- S-update equivalence implies C-update equivalence.
- C-update equivalence implies strong equivalence: $Q = \Phi$.
- P_1 and P_2 are C-update equivalent
  iff P_1 and P_2 are update equivalent wrt ($P_1 \cap P_2$, $\mathcal{P}$)
  where $\mathcal{P}$ is the set of all rules in the language.

Characterizing C-update Equiv.

- **Theorem.** P_1 and P_2 are C-update equivalent iff $P_1 \setminus P_2$ and $P_2 \setminus P_1$ are strongly equivalent.
- **Corollary.** Suppose P_1 and P_2 such that $P_1 \cap P_2 = \Phi$. Then, P_1 and P_2 are C-update equivalent iff P_1 and P_2 are strongly equivalent.

Uniform / Extensional Update Equiv.

- Let P_1 and P_2 be programs, and $\mathcal{E}$ be the set of extensional literals, which is assumed to be common to P_1 and P_2 .
- P_1 and P_2 are **extensionally update equivalent** if they are update equivalent wrt $(\mathcal{E}, \mathcal{E})$.
- P_1 and P_2 are **uniformly update equivalent** if they are update equivalent wrt (***Lit***, ***Lit***).
- Extensionally update equivalence generalizes Sagiv's equivalence for Datalog programs.
- Uniformly update equivalence generalizes uniform equivalence for Datalog programs [Sagiv], NLPs and EDPs [Eiter & Fink].

Reduction to Non-removal Updates

- Let P_1 and P_2 be programs, and $\mathcal{Q}$ and $\mathcal{R}$ be sets of updatable rules.
- Translating relative update equiv. to relative strong equiv.:

  1. $P_i' = (P_i \setminus \mathcal{Q}) \cup \{ (H \leftarrow B, \textbf{\textit{not}}\ \delta_r) \mid r = (H \leftarrow B) \in P_i \cap \mathcal{Q} \}$.
  2. $\mathcal{R}' = \mathcal{R} \cup \{ \delta_r \mid r \in \mathcal{Q} \}$.

- δ_r represents the *deletion* of r.
- The translation can be computed in linear time.
- **Theorem.** P_1 and P_2 are update equivalent with respect to $(\mathcal{Q}, \mathcal{R})$ iff P_1' and P_2' are strongly equivalent with respect to $\mathcal{R}'$.

Complexity Results

- Deciding weak equivalence is Π^P_2-hard [Turner].
- Relative update equivalence includes weak equivalence. Hence,
- **Theorem.** Deciding relative update equivalence is Π^P_2-hard.
- **Theorem.** Deciding S-update equivalence can be done in polynomial time, by checking if each rule in $P_1 \Delta P_2$ is valid.
- Deciding strong equivalence is coNP-complete [Turner].
- C-update equivalence is reduced to strong equivalence. Hence,
- **Theorem.** Deciding C-update equivalence is coNP-complete.

Discussion

- Relative update equivalence is very general.
- S-update equivalence is too strong since only valid rules can appear in the symmetric difference.
- The merit of S-update equivalence is its low computational complexity.
- C-update equivalence is more practical, and can be reduced to strong equivalence.
- Relative update/strong equivalence is useful for characterizing various notions of equivalence, including weak, strong, uniform, and extensional.

Future Work

- Relation to SE-models
- Generalization for the class of nested programs
- Transformation preserving various types of relative update equiv.
- Equivalence of abductive LP (ongoing)