Forgetting for Answer Set Programs Revisited

Yisong Wang
Department of Computer Science,
Guizhou University, China
550025

Kewen Wang
School of Information and
Communication Technology,
Griffith University, Australia

Mingyi Zhang
Guizhou Academy of Sciences,
Guiyang, China
550001

Abstract

A new semantic forgetting for answer set programs (ASP), called SM-forgetting, is proposed in the paper. It distinguishes itself from the others in that it preserves not only skeptical and credulous consequences on unforgotten variables, but also strong equivalence – forgetting same variables in strongly equivalent logic programs has strongly equivalent results. The forgetting presents a positive answer to Gabbay, Pearce and Valverde's open question – if ASP has uniform interpolation property. We also investigate some properties, algorithm and computational complexities for the forgetting. It shows that computing the forgetting result is generally intractable even for Horn logic programs.

1 Introduction

The ability of discarding or hiding irrelevant information has been recognized as an important feature for logic-based agent systems and is named *variable forgetting* or *variable elimination* in Artificial Intelligence [Lin and Reiter, 1994]. Forgetting has been applied in cognitive robotics [Lin and Reiter, 1994; Liu and Wen, 2011], resolving conflicts [Lang *et al.*, 2003; Zhang and Foo, 2006; Eiter and Wang, 2008], and ontologies [Wang *et al.*, 2010; Konev *et al.*, 2012]. In particular, forgetting has recently received much attention in logic programs under answer sets/stable models semantics -- Answer Set Programming (ASP) [Zhang and Foo, 2006; Eiter and Wang, 2008; Wong, 2009; Wang *et al.*, 2012], which is one of the major nonmonotonic paradigms for declarative problem solving in Knowledge Representation and Reasoning [Baral, 2003; Brewka *et al.*, 2011].

It is well-known that ASP has two major notions of equivalence: strong equivalence and equivalence. Formally speaking, given two logic programs Π_1 and Π_2, they are *equivalent* if they have the same stable models; they are *strongly equivalent* if $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ have the same stable models for every logic program Π. The notion of strong equivalence plays an important role in ASP. It acts the same role of equivalence in classical logic and allows to simplify logic programs. By treating a logic program Π as a logical theory $\widehat{\Pi}$, the answer sets of Π exactly correspond to the equilibria of $\widehat{\Pi}$ in equilibrium logic; two logic programs Π and Π' are strongly equivalent whenever $\widehat{\Pi}$ and $\widehat{\Pi'}$ are equivalent in the monotonic logic here-and-there (HT) [Lifschitz *et al.*, 2001]. This naturally defines the entailment relationship between logic programs, viz, Π *entails* Π' if $\widehat{\Pi}$ entails $\widehat{\Pi'}$ in HT.

Based on the two notions of equivalence in ASP, several desirable properties have been proposed for forgetting in logic programs, including irrelevance, persistence, existence and so on [Eiter and Wang, 2008; Wong, 2009; Wang *et al.*, 2012], that are outlined in the following. Let $\mathcal{L}$ be an ASP language on a signature $\mathcal{A}$, Π a logic program in $\mathcal{L}$, $V \subseteq \mathcal{A}$ and $\mathsf{f}(\Pi, V)$ a result of *forgetting* about V in Π.

(E) Existence: $\mathsf{f}(\Pi, V)$ is expressible in $\mathcal{L}$.

(IR) Irrelevance: $\mathsf{f}(\Pi, V)$ is irrelevant to V, i.e., it needs not mention any variables in V.

(W) Weakening: Π entails $\mathsf{f}(\Pi, V)$.

(PP) Positive Persistence: if Π entails Π' which is irrelevant to V then $\mathsf{f}(\Pi, V)$ entails Π'.

(NP) Negative Persistence: if Π does not entail Π' which is irrelevant to V then $\mathsf{f}(\Pi, V)$ does not entail Π'.

(SE) Strong Equivalence: If Π and Π' are strongly equivalent, then $\mathsf{f}(\Pi, V)$ and $\mathsf{f}(\Pi', V)$ are strongly equivalent.

(CP) Consequence Persistence: $\mathsf{f}(\Pi, V)$ has an answer set X' whenever Π has an answer set X such that $X' = X \setminus V$.

Zhang and Zhou (2009) firstly proposed **(W)**, **(PP)**, **(NP)** and **(IR)** for knowledge forgetting in modal logic S5. Wang *et al.* (2012) adapted them for HT-forgetting in logic programs, for which both **(E)** and **(SE)** were proved being satisfied. However, HT-forgetting fails for **(SE)**. The property **(CP)** is originally proposed by Eiter and Wang (2008) for a semantical forgetting in disjunctive logic programs in which X' must be minimal under set inclusion in order to guarantee **(E)**. The semantic forgetting satisfies **(IR)**, but none of **(W)**, **(PP)**, **(NP)** and **(SE)**.

Skeptical reasoning and credulous reasoning are two major reasoning tasks in ASP[1]. The property **(CP)** means that the forgetting operator f preserves skeptical and credulous consequences. Recall that strong equivalence allows for simplification of logic programs. The properties **(SE)** and **(CP)** are

[1]An atom p is a credulous (resp. skeptical) consequence of a logic program Π if p belongs to one (resp. all) answer set(s) of Π.

important and useful for various applications, such as conflict resolving [Zhang and Foo, 2006; Eiter and Wang, 2008; Lang and Marquis, 2010].

Since ASP is based on the answer sets semantics which is nonmonotonic, there is no consensus about a suitable set of criteria for forgetting in ASP. This phenomenon is also evidenced by the existence of several different definitions of forgetting in ASP. A fundamental question is then that is there a rational forgetting in logic programs that satisfies all of the seven desirable properties above? The answer is negative, i.e., it is impossible to define a notion of forgetting that satisfies all of the desirable properties listed above. Especially, for any notion of forgetting in ASP, (**W**) and (**NP**) will be violated if (**IR**), (**E**) and (**CP**) are satisfied (cf. Proposition 3 in the paper). So, if we want a notion of forgetting in ASP to satisfy the set of desirable properties

$$\mathcal{F} = \{(\mathbf{E}), (\mathbf{IR}), (\mathbf{CP}), (\mathbf{PP}), (\mathbf{SE})\},$$

then we have to sacrifice (**W**) and (**NP**). Due to the nonmonotonicity of ASP, it would be acceptable for a notion of forgetting in ASP not to satisfy (**W**) and (**NP**).

Therefore, we argue that $\mathcal{F}$ consists of a suitable set of desirable properties for forgetting in ASP. However, none of the extant definitions of forgetting for ASP satisfies all properties in $\mathcal{F}$. For instance, the semantic forgetting in [Eiter and Wang, 2008], the strong and weak forgetting in [Zhang and Foo, 2006] do not enjoy (**SE**). Among these three notions of forgetting, the first preserves only skeptical consequence but the other two notions of forgetting preserve neither skeptical nor credulous consequence in ASP. In this sense they do not satisfy (**CP**). Wong (2009) presented two forgetting operators F_W and F_S for disjunctive logic programs and Wang *et al.* (2012) defined the HT-forgetting for logic programs. While these proposals satisfy the property (**SE**), they do not satisfy the desirable property (**CP**). Some of these issues are illustrated in the following example.

Suppose Eve has a diet recipe which is represented by a logic program Π consisting of

$$\begin{array}{ll} plum \leftarrow banana; & apple \vee pear; \\ apple \leftarrow not\, pear; & pear \leftarrow not\, apple. \end{array}$$

The logic program has two stable models $\{apple\}$ and $\{pear\}$, which correspond to Eve's two diet schemes. In the case that $pear$ is not available any more, her diet schemes naturally turn into $\{apple\}$ and $\{\}$. According to Π, if $banana$ is provided, then $plum$ should be provided too.

In terms of forgetting, if $pear$ is forgotten in Π, we obtain the following results under the semantic forgetting, F_W, F_S and the HT-forgetting, respectively:

$$\begin{array}{ll} \{apple \leftarrow not\, not\, apple\}, & \{plum \leftarrow banana\}, \\ \{plum \leftarrow banana; \quad apple\}, & \{plum \leftarrow banana\}. \end{array}$$

One can see that the information "$plum \leftarrow banana$" is lost in the semantic forgetting; in terms of F_W and HT-forgetting, the only diet scheme is $\{\}$; it is $\{apple\}$ in terms of F_S.

Therefore, it is an interesting but yet open problem to introduce a notion of forgetting for ASP that obeys all criteria in $\mathcal{F}$.

In this paper, we tackle this problem by proposing a new semantic forgetting for general answer set programs, named SM-*forgetting*. Our new notion of (semantic) forgetting fulfils all criteria in $\mathcal{F}$ while it also satisfies some other useful properties. By this forgetting, we present a positive answer to Gabbay, Pearce and Valverde's open question: can general answer set programs have uniform interpolation property [Gabbay *et al.*, 2011]. We also develop an algorithm for computing SM-forgetting and study complexities for several reasoning tasks induced by SM-forgetting.

2 Preliminaries

We assume a propositional language $\mathcal{L}_\mathcal{A}$ on a finite set $\mathcal{A}$ of propositional variables. $\mathcal{A}$ is also referred to as the signature of $\mathcal{L}_\mathcal{A}$. The *formulas* of $\mathcal{L}_\mathcal{A}$ are inductively constructed using connectives $\bot$, $\wedge$, $\vee$ and $\supset$ as the following:

$$\varphi ::= \bot \mid p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \supset \varphi \tag{1}$$

where $p \in \mathcal{A}$. The formula $\neg\varphi$ stands for $\varphi \supset \bot$, while $\top$ for $\bot \supset \bot$. A *theory* is a set of propositional formulas. In what follows we assume the signature of a formula/theory consisting of the atoms occurring it, unless stated otherwise.

Let S be a finite set of formulas. We denote $\bigvee S$ (resp. $\bigwedge S$) the disjunction (resp. conjunction) of all formulas in S. Particularly, $\bigvee \emptyset$ (resp. $\bigwedge \emptyset$) is $\bot$ (resp. $\top$). By $\neg S$ (resp. $\neg\neg S$) we mean $\{\neg\phi \mid \phi \in S\}$ (resp. $\{\neg\neg\phi \mid \phi \in S\}$).

2.1 HT logic and stable models

The syntax of HT is the same as propositional language but its semantics is different from propositional logic. An *HT-interpretation* is a pair $\langle H, T\rangle$ such that $H \subseteq T \subseteq \mathcal{A}$. The *satisfiability relation* in HT, written $\models_{\mathsf{HT}}$, is recursively defined in the following,

- $\langle H,T\rangle \models_{\mathsf{HT}} p$ if $p \in H$; $\langle H,T\rangle \not\models_{\mathsf{HT}} \bot$;
- $\langle H,T\rangle \models_{\mathsf{HT}} \varphi \vee \psi$ if $\langle H,T\rangle \models_{\mathsf{HT}} \varphi$ or $\langle H,T\rangle \models_{\mathsf{HT}} \psi$;
- $\langle H,T\rangle \models_{\mathsf{HT}} \varphi \wedge \psi$ if $\langle H,T\rangle \models_{\mathsf{HT}} \varphi$ and $\langle H,T\rangle \models_{\mathsf{HT}} \psi$;
- $\langle H,T\rangle \models_{\mathsf{HT}} \varphi \supset \psi$ if both (i) $T \models \varphi \supset \psi$, and (ii) $\langle H,T\rangle \models_{\mathsf{HT}} \varphi$ implies $\langle H,T\rangle \models_{\mathsf{HT}} \psi$

where $p \in \mathcal{A}$, φ and ψ are formulas. An HT-interpretation $\langle H,T\rangle$ is an *HT-model* of a formula φ if $\langle H,T\rangle \models_{\mathsf{HT}} \varphi$. We denote $\mathsf{Mod}_{\mathsf{HT}}(\varphi)$ the set of all HT-models of φ. An HT-model $\langle T,T\rangle$ of a formula φ is an *equilibrium model* of φ if there is no $T' \subset T$ such that $\langle T',T\rangle \models_{\mathsf{HT}} \varphi$.

Given two formulas φ and ψ, φ *entails* ψ in HT, written $\varphi \models_{\mathsf{HT}} \psi$, if $\mathsf{Mod}_{\mathsf{HT}}(\varphi) \subseteq \mathsf{Mod}_{\mathsf{HT}}(\psi)$; $\varphi \not\models_{\mathsf{HT}} \psi$ means that φ does not entail ψ in HT; φ and ψ are *HT-equivalent*, written $\psi \equiv_{\mathsf{HT}} \varphi$, if $\mathsf{Mod}_{\mathsf{HT}}(\varphi) = \mathsf{Mod}_{\mathsf{HT}}(\psi)$. A formula φ is *HT-irrelevant to* a set V of atoms, denoted $\mathbf{IR}_{\mathsf{HT}}(\varphi, V)$, if there is a formula ψ not containing any atoms from V such that $\varphi \equiv_{\mathsf{HT}} \psi$. In this case we assume φ does not mention any atoms from V, unless explicitly stated otherwise.

A formula of the following form is called a *rule*:

$$\bigwedge \neg\neg D \wedge \bigwedge \neg C \wedge \bigwedge B \supset \bigvee A \tag{2}$$

where $A = \{a_1, \ldots, a_l\}$, $B = \{b_1, \ldots, b_k\}$, $C = \{c_1, \ldots, c_m\}$ and $D = \{d_1, \ldots, d_n\}$. It is also identified as

$$a_1; \ldots; a_l \leftarrow b_1, \ldots, b_k, not\, c_1, \ldots, not\, c_m, \\ not\, not\, d_1, \ldots, not\, not\, d_n, \quad (3)$$

which is called a nested expression in [Lifschitz *et al.*, 1999]. A rule r of form (3) is *disjunctive* if $n = 0$; it is *normal* if it is disjunctive and $l \leq 1$; it is *Horn* if it is normal and $m = 0$; it is a *constraint* if $l = 0$. A *logic program* Π is a finite set of rules of form (3). It is *disjunctive* (resp. *normal and Horn*) if all of its rules are disjunctive (resp. normal and Horn). The stable models/answer sets semantics of logic programs is attributed to Gelfond and Lifschitz (1988). By identifying every rule r of form (3) as the formula $\widehat{r}$ of the form (2), it is well-known that a set T of atoms is a *stable model/answer set* of a logic program Π iff $\langle T, T\rangle$ is an equilibrium model of $\widehat{\Pi}$, where $\widehat{\Pi} = \{\widehat{r} | r \in \Pi\}$ [Pearce, 1996; Ferraris, 2005].

Let Π be a logic program. By $\mathsf{SM}(\Pi)$ we denote the set of all stable models of Π. Π is *consistent* if $\mathsf{SM}(\Pi) \neq \emptyset$. If Π is disjunctive then $\mathsf{SM}(\Pi)$ forms an *antichain*, i.e., no two elements of $\mathsf{SM}(\Pi)$ are comparable under set inclusion. An atom p is a *skeptical* (resp. *credulous*) consequence of Π, denoted by $\Pi \models_s p$ (resp. $\Pi \models_c p$), if $p \in \bigcap \mathsf{SM}(\Pi)$ (resp. $p \in \bigcup \mathsf{SM}(\Pi)$). A logic programs Π' is *equivalent to* Π, written $\Pi' \equiv_{\mathsf{SM}} \Pi$, if $\mathsf{SM}(\Pi') = \mathsf{SM}(\Pi)$; Π' is *strongly equivalent to* Π if $\mathsf{SM}(\Pi \cup \Pi'') = \mathsf{SM}(\Pi' \cup \Pi'')$ for every logic program Π''. It is well-established that two logic programs are strongly equivalent iff their corresponding theories are HT-equivalent in HT [Lifschitz *et al.*, 2001]. Since very formula in HT can be translated into a conjunction of formulas of the form (2), we will not distinct logic programs from formulas in the following.

2.2 Forgetting and HT-forgetting

By an *interpretation* we mean a set of atoms. The *complement* of an interpretation M, written $\overline{M}$, is the set $\mathcal{A} \setminus M$. The *complement* of a collection $\mathcal{M}$ of (HT-)interpretations, denoted by $\overline{\mathcal{M}}$, is the set of (HT-)interpretations not in $\mathcal{M}$.

Let $V \subseteq \mathcal{A}$ and X an interpretation. An interpretation Y is V*-bisimilar* to X, written $Y \sim_V X$, if $Y \setminus V = X \setminus V$. The V*-extension* of X, denoted by $X_{\dagger V}$, is the collection of interpretations that are V-similar to X. Two HT-interpretations $\langle H, T\rangle$ and $\langle X, Y\rangle$ are V*-bisimilar*, denoted by $\langle H, T\rangle \sim_V \langle X, Y\rangle$, if $H \sim_V X$ and $T \sim_V Y$. The V*-extension* of an HT-interpretation $\langle H, T\rangle$, denoted by $\langle H, T\rangle_{\dagger V}$, is the collection of HT-interpretations that are V-similar to $\langle H, T\rangle$. The V*-extension* of a collection $\mathcal{M}$ of (HT-)interpretations is the collection $\bigcup_{\beta \in \mathcal{M}} \beta_{\dagger V}$.

The forgetting in propositional logic [Lin and Reiter, 1994] can be equivalently reformulated as following:

Definition 1 (forgetting) *Let φ be a formula and $V \subseteq \mathcal{A}$. A formula ψ is a* result of forgetting V in φ *if and only if* $\mathsf{Mod}(\psi) = \mathsf{Mod}(\varphi)_{\dagger V}$.

Such a forgetting result ψ is unique up to equivalence. We denote it by $\mathsf{Forget}(\varphi, V)$.

Similarly, the HT-forgetting in [Wang *et al.*, 2012] can be equivalently reformulated as following:

Definition 2 (HT-forgetting) *Let φ be a formula and $V \subseteq \mathcal{A}$. A formula ψ is a* result of HT-forgetting V in φ *if and only if* $\mathsf{Mod}_{\mathsf{HT}}(\psi) = \mathsf{Mod}_{\mathsf{HT}}(\varphi)_{\dagger V}$.

It is proved (cf. Theorem 1 of [Wang *et al.*, 2012]) that such a formula ψ always exists, it is unique up to HT-equivalence and it is HT-irrelevant to V. It is denoted by $\mathsf{Forget}_{\mathsf{HT}}(\varphi, V)$.

3 SM-Forgetting

In this section we introduce a new notion of semantic forgetting for logic programs, called SM-forgetting, and study its properties, algorithm and computational complexity.

3.1 Definition and properties

Let $V \subseteq \mathcal{A}$ and X an interpretation. The V*-exclusion* of X, written $X_{\|V}$, is the set $X \setminus V$. The V*-exclusion* of a collection $\mathcal{M}$ of interpretations is $\{X_{\|V} | X \in \mathcal{M}\}$, denoted by $\mathcal{M}_{\|V}$.

Definition 3 (SM-forgetting) *Let φ be a formula and $V \subseteq \mathcal{A}$. A formula ψ is a* result of SM-forgetting V *in φ if and only if* $\mathsf{Mod}_{\mathsf{HT}}(\psi)$ *is a maximal subset of* $\mathsf{Mod}_{\mathsf{HT}}(\varphi)_{\dagger V}$ *such that* $\mathsf{SM}(\psi) = \mathsf{SM}(\varphi)_{\|V}$.

Example 1 For the logic program Π in Section 1, the following logic program is a result of SM-forgetting $pear$ in Π:

$$\{plum \leftarrow banana; \qquad apple \leftarrow not\, not\, apple\}.$$

It has two stable models $\emptyset$ and $\{apple\}$, which correspond to the two intended diet schemes for Eve after forgetting $pear$. In particular, we note that the knowledge "$plum \leftarrow banana$" of Π, which is irrelevant to $pear$, is preserved in the result.

The next theorem shows that the result of SM-forgetting a set V of atoms in a formula φ exists, and it is unique up to strong equivalence. We denote it by $\mathsf{Forget}_{\mathsf{SM}}(\varphi, V)$.

Theorem 1 (Existence of SM-forgetting) *Given any formula φ and any set V of atoms, there exists a result of SM-forgetting V in φ. Moreover, if both ψ and ψ' are results of SM-forgetting V in φ, then ψ and ψ' are strongly equivalent.*

Proof sketch: We denote that

$$\mathcal{G} = \{\langle X, Y\rangle | X \subset Y \,\&\, Y \in \mathsf{SM}(\varphi)_{\|V}\}, \quad (4)$$
$$\mathcal{H} = \{\langle Y, Y\rangle | Y \in \mathsf{SM}(\mathsf{Forget}_{\mathsf{HT}}(\varphi, V)) \setminus \mathsf{SM}(\varphi)_{\|V}\}, \quad (5)$$
$$\mathcal{N} = \mathsf{Mod}_{\mathsf{HT}}(\varphi)_{\dagger V} \setminus (\mathcal{G} \cup \mathcal{H}). \quad (6)$$

A collection $\mathcal{M}$ of HT-interpretations is called *HT-valid* if there is a logic program Π such that $\mathcal{M}$ is the set of HT-models of Π. It is shown in [Cabalar and Ferraris, 2007] that a collection $\mathcal{M}$ of HT-interpretations is HT-valid iff

$$\langle X, Y\rangle \in \mathcal{M} \quad \text{implies} \quad \langle Y, Y\rangle \in \mathcal{M}. \quad (7)$$

We can show that $\mathcal{N}$ satisfies (7) and thus, it is HT-valid. Let $\mathsf{Mod}_{\mathsf{HT}}(\psi) = \mathcal{N}$. Removing the HT-interpretations in $\mathcal{G} \cup \mathcal{H}$ from $\mathsf{Mod}_{\mathsf{HT}}(\varphi)_{\dagger V}$ assures that the stable models of ψ are exactly the elements of $\mathsf{SM}(\varphi)_{\|V}$. ■

Recall that $\mathsf{Forget}_{\mathsf{HT}}(\varphi, V) \equiv_{\mathsf{HT}} \{\xi | \varphi \models_{\mathsf{HT}} \xi \,\&\, \mathbf{IR}_{\mathsf{HT}}(\xi, V)\}$ (cf. Theorem 3 of [Wang *et al.*, 2012]). By the construction of $\mathsf{Forget}_{\mathsf{SM}}(\varphi, V)$ in the proof of the above theorem, we have the following proposition.

Proposition 1 *The* SM*-forgetting satisfies the properties (E), (IR), (SE), (CP) and (PP).*

The next proposition shows that the SM-forgetting preserves equivalence. Under an additional condition, it satisfies knowledge weakening in the sense that $\varphi \models_{\mathsf{HT}} \psi$ implies $\mathsf{Forget}_{\mathsf{SM}}(\varphi, V) \models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\psi, V)$.

Proposition 2 *Let φ, ψ be two formulas and $V \subseteq \mathcal{A}$. Then*

(i) $\mathsf{Forget}_{\mathsf{SM}}(\varphi, V) \equiv_{\mathsf{SM}} \mathsf{Forget}_{\mathsf{SM}}(\psi, V)$ *if $\varphi \equiv_{\mathsf{SM}} \psi$;*

(ii) $\mathsf{Forget}_{\mathsf{SM}}(\varphi, V) \models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\psi, V)$ *if $\varphi \models_{\mathsf{HT}} \psi$ and* $\mathsf{SM}(\varphi)_{\|V} = \mathsf{SM}(\psi)_{\|V}$.

The condition $\mathsf{SM}(\varphi)_{\|V} = \mathsf{SM}(\psi)_{\|V}$ is necessary for the statement (ii) in the above proposition. Otherwise, let $\varphi = p \wedge \neg q \supset \bot$ and $\psi = \neg q \supset (p \vee \neg p)$. One can verify that $\varphi \models_{\mathsf{HT}} \psi$, $\mathsf{Forget}_{\mathsf{SM}}(\varphi, \{q\}) \equiv_{\mathsf{HT}} \top$ but $\mathsf{Forget}_{\mathsf{SM}}(\psi, \{q\}) \equiv_{\mathsf{HT}} p \vee \neg p$.

As illustrated in the following example, the properties (**W**) and (**NP**) may not be satisfied by SM-forgetting.

Example 2 Let us consider the following formulas.

- Let $\psi = q \wedge (\neg p \supset p)$. It is not difficult to see that, on the signature $\{p, q\}$, $\mathsf{Mod}_{\mathsf{HT}}(\psi) = \{\langle\{q\}, \{p, q\}\rangle, \langle\{p, q\}, \{p, q\}\rangle\}$ and then ψ has no stable model. One can verify that

  $\mathsf{Forget}_{\mathsf{HT}}(\psi, \{q\}) \equiv_{\mathsf{HT}} \neg p \supset p$, $\mathsf{Forget}_{\mathsf{HT}}(\psi, \{p\}) \equiv_{\mathsf{HT}} q$,
  $\mathsf{Forget}_{\mathsf{SM}}(\psi, \{q\}) \equiv_{\mathsf{HT}} \neg p \supset p$, $\mathsf{Forget}_{\mathsf{SM}}(\psi, \{p\}) \equiv_{\mathsf{HT}} \bot$.

  It is evident that $\psi \models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\psi, \{q\})$, but not vice versa; and $\mathsf{Forget}_{\mathsf{SM}}(\psi, \{p\}) \models_{\mathsf{HT}} \psi$, but not vice versa. It implies that SM-forgetting does not satisfy (**W**) for ψ.

- Let $\varphi = (p \vee \neg p \vee q \vee \neg q) \wedge (p \supset q \vee \neg q) \wedge (q \supset p \vee \neg p)$. One can check that $\mathsf{SM}(\varphi) = \{\emptyset, \{p, q\}\}$, and

  $$\mathsf{Forget}_{\mathsf{HT}}(\varphi, \{p\}) \equiv_{\mathsf{HT}} \top,$$
  $$\varphi' = \mathsf{Forget}_{\mathsf{SM}}(\varphi, \{p\}) \equiv_{\mathsf{HT}} q \vee \neg q.$$

  Since $\langle\emptyset, \{q\}\rangle$ is an HT-model of φ but φ', it follows $\varphi \not\models_{\mathsf{HT}} \varphi'$. Moreover $\langle\{q\}, \{p, q\}\rangle$ is an HT-model of φ' but not an HT-model of φ, it shows $\varphi' \not\models_{\mathsf{HT}} \varphi$. Thus φ is not comparable with φ' in HT. It implies that SM-forgetting falsifies (**W**) and (**NP**) for φ.

  Actually, on the signature $\{q\}$, $q \vee \neg q$ is the only formula (up to HT-equivalence) having the stable models $\emptyset$ and $\{q\}$. In other words, there is no ψ^* (on the signature $\{q\}$) satisfying $\varphi \models_{\mathsf{HT}} \psi^*$ and $\mathsf{SM}(\psi^*) = \{\emptyset, \{q\}\}$.

This example shows that, to ensure (**IR**), (**E**) and (**CP**) of a forgetting in answer set programming, one cannot demand (**W**) or (**NP**), no matter logic programs are consistent or not.

Proposition 3 *There is no forgetting for ASP satisfies (**W**) or (**NP**) that satisfies (**IR**), (**E**) and (**CP**).*

To satisfy the property (**W**), the next proposition presents a sufficient and necessary condition for the SM-forgetting.

Proposition 4 *Let φ be a formula and $V \subseteq \mathcal{A}$. Then we have* $\varphi \models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\varphi, V)$ *iff, for every $Y \subseteq \mathcal{A}$,*

- $\langle X, Y\rangle \not\models_{\mathsf{HT}} \varphi$ *for every $X \subset Y$ if $Y \in \mathsf{SM}(\varphi)_{\|V}$,*
- $Y \models \varphi$ *implies $\langle X, Y\rangle \models_{\mathsf{HT}} \varphi$ for some $X \subset Y$, otherwise.*

The following result guarantees that SM-forgetting a set V of atoms can be done through SM-forgetting one atom by one atom in V and it is irrelevant to the order of atoms.

Theorem 2 *Let φ be a formula, V_1, V_2 two sets of atoms.* $\mathsf{Forget}_{\mathsf{SM}}(\varphi, V_1 \cup V_2) \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\mathsf{Forget}_{\mathsf{SM}}(\varphi, V_1), V_2)$.

Proof sketch: Let's denote

$$\psi \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\varphi, V), \qquad \psi^* \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\varphi, V),$$
$$\psi_1 \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\varphi, V_1), \qquad \psi_1^* \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\varphi, V_1),$$
$$\psi_2 \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\psi_1, V_2), \qquad \psi_2^* \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\psi_1, V_2).$$

We can show that, if $Y \notin \mathsf{SM}(\varphi)_{\|V}$ then

(a) $Y \models \psi^*$ iff $Y \models \psi_2^*$, and

(b) $\langle X, Y\rangle \models_{\mathsf{HT}} \psi^*$ iff $\langle X, Y\rangle \models_{\mathsf{HT}} \psi_2^*$ for every $X \subset Y$.

The remain is to prove $\langle X, Y\rangle \models_{\mathsf{HT}} \psi$ iff $\langle X, Y\rangle \models_{\mathsf{HT}} \psi_2$ by two cases $Y \in \mathsf{SM}(\varphi)_{\|V}$ and $Y \notin \mathsf{SM}(\varphi)_{\|V}$. ■

3.2 Expressibility in disjunctive logic programs

Recall that the results of SM-forgetting are expressible in general (propositional) logic programs by Theorem 1. In this subsection we investigate the validity of this result for smaller classes of logic programs, including disjunctive logic programs and Horn logic programs.

Recall that Eiter *et al.* (2005) and Wong (2009) showed that a collection $\mathcal{M}$ of HT-interpretations is *disjunctive HT-valid*, i.e., there is a disjunctive logic program Π such that $\mathsf{Mod}_{\mathsf{HT}}(\Pi) = \mathcal{M}$, iff $\mathcal{M}$ satisfies (7) and the condition:

$$\langle X, Y\rangle \in \mathcal{M}, \langle Y', Y'\rangle \in \mathcal{M}, \text{ and } Y \subseteq Y' \Rightarrow \langle X, Y'\rangle \in \mathcal{M}.$$

The next example shows that SM-forgetting results for some important classes of logic programs, including normal programs with and without constraints, are not expressible in disjunctive logic programs. Thus SM-forgetting results for disjunctive logic programs may be not expressible in disjunctive logic programs yet.

Example 3 Consider the following normal logic programs.

- Let Π_1 consists of

  $$p \leftarrow \mathit{not}\, q; \quad q \leftarrow \mathit{not}\, p; \quad r \leftarrow p; \quad r \leftarrow q.$$

  It has two stable models $\{p, r\}$ and $\{q, r\}$. We have

  $$\mathsf{Forget}_{\mathsf{SM}}(\Pi_1, \{q\}) \equiv_{\mathsf{HT}} \{p \leftarrow \mathit{not}\,\mathit{not}\, p; \quad r\},$$
  $$\mathsf{Forget}_{\mathsf{HT}}(\Pi_1, \{q\}) \equiv_{\mathsf{HT}} \{r \leftarrow \mathit{not}\, p; \quad r \leftarrow p\}.$$

  The stable models of $\mathsf{Forget}_{\mathsf{SM}}(\Pi_1, \{q\})$ are $\{r\}$ and $\{p, r\}$, that cannot form an antichain. Thus, there is no disjunctive logic program (on the signature $\{p, r\}$) that can capture $\mathsf{Forget}_{\mathsf{SM}}(\Pi_1, \{q\})$, even if $\mathsf{Forget}_{\mathsf{HT}}(\Pi_1, \{q\})$ is disjunctive.

- Let $\Pi_2 = \{\leftarrow p, q, r; \quad \leftarrow p, q, \mathit{not}\, r\}$. Its unique stable model is $\emptyset$. We have

  $$\mathsf{Forget}_{\mathsf{SM}}(\Pi_2, \{r\})$$
  $$\equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\Pi_2, \{r\})$$
  $$\equiv_{\mathsf{HT}} \{p \vee q \leftarrow \mathit{not}\,\mathit{not}\, p, \mathit{not}\,\mathit{not}\, q\},$$

which cannot be captured by a disjunctive logic program (on the signature $\{p, q\}$), even if $\mathsf{SM}(\Pi)_{\|V}$ forms an antichain. The reason is that $\langle\emptyset, \emptyset\rangle$ and $\langle\{p, q\}, \{p, q\}\rangle$ are HT-models of $\mathsf{Forget}_{\mathsf{SM}}(\Pi_2, \{r\})$, but $\langle\emptyset, \{p, q\}\rangle$ is not.

We present a sufficient and necessary condition for the disjunctive expressibility of SM-forgetting results.

Proposition 5 *Let φ be a formula and $V \subseteq \mathcal{A}$. There is a disjunctive logic program Π such that $\Pi \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\varphi, V)$ iff, for every HT-models $\langle X_1, Y_1\rangle, \langle Y_2, Y_2\rangle$ of φ such that $Y_1 \subseteq Y_2$, there exists an HT-model $\langle X_3, Y_3\rangle$ of φ such that $\langle X_3, Y_3\rangle \sim_V \langle X_1, Y_2\rangle$ and, no $M \in \mathsf{SM}(\varphi)$ with $M \sim_V Y_3$.*

For Horn logic programs, SM-forgetting has an interesting property, which implies that SM-forgetting results in Horn logic programs can be expressed in Horn logic programs and thus expressible in disjunctive logic programs.

Theorem 3 *Let Π be a Horn logic program and $V \subseteq \mathcal{A}$. Then $\mathsf{Forget}_{\mathsf{SM}}(\Pi, V) \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\Pi, V)$.*

Proof sketch: Firstly we can show that X is the least model of Π iff $X \setminus V$ is the least model of $\mathsf{Forget}_{\mathsf{HT}}(\Pi, V)$.

Secondly we denote:
$\mathcal{G} = \{\langle X, Y\rangle | X \subset Y \ \& \ Y \in \mathsf{SM}(\Pi)_{\|V}\}$,
$\mathcal{H} = \{\langle Y, Y\rangle | Y \in \mathsf{SM}(\mathsf{Forget}_{\mathsf{HT}}(\Pi, V)) \ \& \ Y \notin \mathsf{SM}(\Pi)_{\|V}\}$,
$\mathcal{N} = \mathsf{Mod}_{\mathsf{HT}}(\Pi)_{\dagger V}$.
It is evident that $\mathcal{H} = \emptyset$. We can further show $\mathcal{N} \cap \mathcal{G} = \emptyset$. ■

As HT-forgetting results of Horn logic programs are Horn expressible and HT-forgetting enjoys "modularity" (cf. Theorem 2 and (vi) of Proposition 3 in [Wang *et al.*, 2012] respectively). The following corollary follows by Theorem 3.

Corollary 4 *Let Π, Π' be Horn logic programs and $V \subseteq \mathcal{A}$.*

(i) $\mathsf{Forget}_{\mathsf{HT}}(\Pi, V)$ *is Horn expressible;*

(ii) $\mathsf{Forget}_{\mathsf{SM}}(\Pi \cup \Pi', V) \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V) \cup \Pi'$ *if* $\boldsymbol{IR}_{\mathsf{HT}}(\Pi', V)$.

3.3 Relation to other forms of forgetting

In this subsection, we first reveal a connection between SM-forgetting and HT-forgetting and then provide two results on relationships of SM-forgetting with the forgetting in propositional logic and a uniform interpolation for answer set programs respectively.

Proposition 6 *Let φ be a formula and $V \subseteq \mathcal{A}$. Then we have $\mathsf{Forget}_{\mathsf{SM}}(\varphi, V) \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\varphi, V)$ if and only if $\mathsf{SM}(\mathsf{Forget}_{\mathsf{HT}}(\varphi, V)) = \mathsf{SM}(\varphi)_{\|V}$.*

The notion of loop formulas plays an important role in ASP [Lin and Zhao, 2004]. As stable models of a logic program Π can be captured by models of its loop formulas $LF(\Pi)$ in propositional logic (cf. Theorem 2 of [Ferraris *et al.*, 2006]). Accordingly the following proposition partly connects SM-forgetting with the forgetting in propositional logic.

Proposition 7 *Let Π be a logic program and V a set of atoms. Then X is a stable model $\mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$ iff X is a model of $\mathsf{Forget}(LF(\Pi), V)$.*

Uniform interpolation. In monotonic logics, forgetting is closely related to uniform interpolation [Visser, 1996]. Gabbay *et al.* (2011) show that the class of disjunctive logic programs enjoys the uniform interpolation property with queries being disjunctions of literals. They first introduced a (non-monotonic) inference $\mid\!\sim$ between formulas in ASP:

$$\varphi \mid\!\sim \psi \quad \text{if} \quad M \models \psi \text{ for every } M \in \mathsf{SM}(\varphi). \tag{8}$$

A class of formulas in ASP has the *uniform interpolation* property iff for any formula φ and any set V of atoms, there is a formula ψ such that the following conditions hold:

(i) $var(\psi) \subseteq var(\varphi) \setminus V$, and

(ii) for any formula α with $var(\alpha) \cap V = \emptyset$,
$$\varphi \mid\!\sim \alpha \qquad \text{iff} \qquad \psi \mid\!\sim \alpha \tag{9}$$
where $var(\alpha)$ is the set of atoms occurring in α.

Gabbay *et al.* (2011) showed that the class of disjunctive logic programs enjoy the uniform interpolation property[2] and they raised an interesting open question: Given a signature $\mathcal{A}$, does $\mathcal{L}_{\mathcal{A}}$, the class of all formulas in ASP, possess the uniform interpolation property too?

This open question can be affirmatively answered.

Proposition 8 *Let $\mathcal{A}$ be a set of atoms. Then $\mathcal{L}_{\mathcal{A}}$, the class of all formulas in ASP, has the uniform interpolation property.*

3.4 Computation for SM-forgetting

According to Theorem 1 and the notion of countermodels[3] in here-and-there [Cabalar and Ferraris, 2007], we present an algorithm to compute SM-forgetting result – Algorithm 1, where the underlying signature $\mathcal{A}$ consists of the atoms occurring in Π and $\lambda_{X,Y}$ denotes the following formula:

$$\bigwedge X \wedge \bigwedge \neg\overline{Y} \supset \bigvee_{z \in (Y \setminus X)} z \vee \neg z. \tag{10}$$

In particular, $\lambda_{Y,Y} = \bigwedge Y \wedge \bigwedge \neg\overline{Y} \supset \bot$ since $\bigvee \emptyset$ is $\bot$.

Recall that $\lambda_{Y,Y}$ captures the countermodel $\langle Y, Y\rangle$ of Π, while $\lambda_{X,Y}$ captures the countermodel $\langle X, Y\rangle$ of Π where $X \subset Y$ and $\langle Y, Y\rangle$ is an HT-model of Π. The intuition of the algorithm is as follows:

(1) The **foreach** loop (Lines 3-5) ensures all stable models in $\mathsf{SM}(\Pi)$ are preserved when V is forgotten.

(2) The **foreach** loop (Lines 6-22) asserts no other stable models for Σ by Lines 17 and 20.

(3) The **foreach** loop (Lines 9-15) removes the countermodel $\langle X, Y\rangle$ that cannot be derived from Π.

Theorem 5 *Algorithm 1 outputs* $\mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$.

In the algorithm, since $\mathsf{SM}(\Pi)$ is possibly exponential in the size of $|\Pi|$ and the **foreach** loop (Line 9) considers all subsets of Y, it outputs an exponentially larger size logic program in the worst case.

Example 4 Let's consider Π_1 from Example 3 where $\mathcal{A} = \{p, q, r\}$. Let $V = \{q\}$. Note that $\mathcal{M} = \{\{p, r\}, \{q, r\}\}$ and $\mathcal{M}_{\|V} = \{\{p, r\}, \{r\}\}$. Now Σ is constructed as follows:

[2]The query α in (9) is restricted being a disjunction of literals.

[3]An HT-interpretation is a countermodel of a formula φ if it not an HT-model of φ.

Algorithm 1: SM-forgetting

```
input : A logic program $\Pi$ and a set $V$ of atoms
output: A result of SM-forgetting $V$ in $\Pi$
1  $\Sigma \leftarrow \emptyset$;
2  $\mathcal{M} \leftarrow \mathsf{SM}(\Pi)_{\|V}$;
3  foreach $Y \in \mathcal{M}$ do
4      foreach $X \subset Y$ do $\Sigma \leftarrow \Sigma \cup \{\lambda_{X,Y}\}$;
5  end
6  foreach $Y \notin \mathcal{M}$ do
7      if $\exists Y' \models \Pi$ s.t $Y' \sim_V Y$ then
8          if $\exists \langle X', Y'\rangle \models_{\mathsf{HT}} \Pi$ s.t $Y' \sim_V Y$ and
           $X' \setminus V \subset Y$ then
9              foreach $X \subset Y$ do
10                 if $\exists \langle H, T\rangle \models_{\mathsf{HT}} \Pi$ s.t $\langle H, T\rangle \sim_V \langle X, Y\rangle$
                   then
11                     Continued;
12                 else
13                     $\Sigma \leftarrow \Sigma \cup \{\lambda_{X,Y}\}$;
14                 end
15             end
16         else
17             $\Sigma \leftarrow \Sigma \cup \{\lambda_{Y,Y}\}$;
18         end
19     else
20         $\Sigma \leftarrow \Sigma \cup \{\lambda_{Y,Y}\}$;
21     end
22 end
23 return $\Sigma$;
```

(1) According to Lines 3-5 of Algorithm 1, Σ includes:

$$p \vee r \vee \neg p \vee \neg r, \qquad p \supset r \vee \neg r,$$
$$r \supset p \vee \neg p, \qquad \neg p \supset r \vee \neg r.$$

(2) For $\emptyset \notin \mathcal{M}$, the two sets $\emptyset$ and $\{q\}$ are not models of φ, according to Lines 6-22, Σ includes $\neg p \wedge \neg r \supset \bot$.

(3) For $\{p\} \notin \mathcal{M}$, the two sets $\{p\}$ and $\{p, q\}$ are not models of φ, according to Lines 6-22, Σ contains $p \wedge \neg r \supset \bot$.

One can further verify that $\Sigma \equiv_{\mathsf{HT}} (p \vee \neg p) \wedge r$.

3.5 Some complexity results

It is known that the credulous and skeptical reasoning problem of ASP is Σ_2^P-complete and Π_2^P-complete respectively, and the consistency problem (i.e., deciding if a formula has a stable model) is Σ_2^P-complete (cf. Theorem 10 of [Pearce *et al.*, 2009]). The following corollary follows.

Corollary 6 *Let Π be a (disjunctive) logic program, V a set of atoms and p an atom not in V. The following hold:*

(i) deciding if $\mathsf{SM}(\mathsf{Forget}_{\mathsf{SM}}(\Pi, V)) \neq \emptyset$ is Σ_2^P-complete.

(ii) deciding if $\mathsf{Forget}_{\mathsf{SM}}(\Pi, V) \models_s p$ is Π_2^P-complete,

(iii) deciding if $\mathsf{Forget}_{\mathsf{SM}}(\Pi, V) \models_c p$ is Σ_2^P-complete.

The next theorem shows that it is generally intractable to compute classical forgetting result of propositional Horn theories. It has its own interest and is used to prove the intractability of SM-forgetting.

Theorem 7 *Let Σ and Π be two Horn theories and $V \subseteq \mathcal{A}$. Deciding if $\Pi \equiv \mathsf{Forget}(\Sigma, V)$ is* co-NP-*complete.*

Proof sketch: We show the same hardness for deciding if $\Pi \models \mathsf{Forget}(\Sigma, V)$ by reduct from 3SAT. Let $\gamma = c_1 \wedge \ldots \wedge c_m$ be a 3CNF over atoms $x_1, \ldots, x_n$, where $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$. We introduce for each clause c_i a new atom y_i, for each atom x_j a new atom x_j' (which intuitively corresponds to $\neg x_j$), and a special atom z. The Horn theory $\Pi = \{\neg x_i \vee \neg x_i' | 1 \leq i \leq n\}$ and Σ contains Π and additional the following Horn clauses:

$$\neg z \vee y_1,$$
$$y_i \vee \neg l_{i,j}^* \vee \neg y_{i+1} \text{ for all } i = 1, \ldots, m-1, \text{ and } j = 1, 2, 3,$$
$$\neg y_m \vee \neg l_{m,j}^* \text{ for } j = 1, 2, 3$$

where $l^* = x$ if l is a positive literal x, and $l^* = x'$ if l is a negative literal $\neg x$. The remain is to prove that γ is satisfiable iff $\Pi \not\models \mathsf{Forget}(\Sigma, V)$ where $V = \{y_1, \ldots, y_m\}$. ■

As noted in the previous section that the algorithm of computing SM-forgetting result is quite expensive, this might not be avoided generally in terms of the following theorem.

Theorem 8 *Let Σ and Π be two logic programs, $V \subseteq \mathcal{A}$.*

(i) Deciding if $\Sigma \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$ is in Π_2^P.

(ii) Deciding if $\Sigma \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$ is co-NP-*completed if both Σ and Π are Horn logic programs.*

Proof sketch: (i) If $\Sigma \not\equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$ then there exists an HT-interpretation $\langle X, Y\rangle$ such that either (a) $\langle X, Y\rangle \models_{\mathsf{HT}} \Sigma$ and $\langle X, Y\rangle \not\models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$, or (b) $\langle X, Y\rangle \not\models_{\mathsf{HT}} \Sigma$ and $\langle X, Y\rangle \models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$. We consider the case (a). The case (b) is similar.

Note that checking if $\langle X, Y\rangle \models_{\mathsf{HT}} \Sigma$ is feasible in polynomial time of Σ. Moreover, $\langle X, Y\rangle \not\models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$ iff either (i.a) $\langle X, Y\rangle \not\models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\Pi, V)$, $X \subset Y$ and $Y \in \mathsf{SM}(\Pi)_{\|V}$, or (i.b) $X = Y$ and $Y \in \mathsf{SM}(\mathsf{Forget}_{\mathsf{HT}}(\Pi, V)) \setminus \mathsf{SM}(\Pi)_{\|V}$. The case (i.a) is feasible in polynomial by calling an NP-oracle. We consider the case (i.b). Note further that $Y \in \mathsf{SM}(\mathsf{Forget}_{\mathsf{HT}}(\Pi, V))$ iff $Y \models \mathsf{Forget}_{\mathsf{HT}}(\Pi, V)$, which is feasible in polynomial time of Π by calling an NP-oracle, and $\langle X, Y\rangle \not\models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\Pi, V)$ for every $X \subset Y$, whose complement is $\langle X, Y\rangle \models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{HT}}(\Pi, V)$ for some $X \subset Y$ iff there is $\langle X', Y'\rangle \models_{\mathsf{HT}} \Pi$ such that $Y \sim_V Y'$ and $X' \setminus V \subset Y$, which is feasible in polynomial time of Π by calling an NP-oracle as well.

(ii) The membership is proved by showing $\langle H, T\rangle \models_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$ iff the least model M of Π satisfies the condition $M_{\|V} \subseteq H \cap T$. The hardness is proved by showing $\Sigma \equiv_{\mathsf{HT}} \mathsf{Forget}_{\mathsf{SM}}(\Pi, V)$ iff $\Sigma \equiv \mathsf{Forget}(\Pi, V)$, according to Theorem 7. ■

4 Concluding Remarks and Future Work

In this paper, we presented a new semantic forgetting for general logic programs based on stable models semantics, called SM-forgetting. Different from the other forgetting for logic programs, it preserves skeptical and credulous consequence on unforgotten variables and (strong) equivalence. This provides a positive answer to Gabbay, Pearce and Valverde's open question (2011). Its properties, algorithm and computational complexities are extensively explored.

Though this forgetting is developed for answer set programs, we believe that the proposed criteria should somehow be also respected by a semantic forgetting in other nonmonotonic logic systems, which deserves our further efforts.

Acknowledgement

We thank reviewers for their helpful comments. This work is partially supported by .... Yisong Wang is also partially supported by NSFC grant 60963009, Open Funds of the State Key Laboratory of Computer Science of Chinese Academy of Science under grant SYSKF1106 and Stadholder foundation of Guizhou Province under grant (2012)62.

References

[Baral, 2003] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, New York, NY, 2003.

[Brewka *et al.*, 2011] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[Cabalar and Ferraris, 2007] Pedro Cabalar and Paolo Ferraris. Propositional theories are strongly equivalent to logic programs. *Theory and Practice of Logic Programming*, 7(6):745–759, 2007.

[Eiter and Wang, 2008] Thomas Eiter and Kewen Wang. Semantic forgetting in answer set programming. *Artificial Intelligence*, 172(14):1644–1672, 2008.

[Eiter *et al.*, 2005] Thomas Eiter, Hans Tompits, and Stefan Woltran. On solution correspondences in answer-set programming. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 97–102, Edinburgh, Scotland, UK, 2005. Professional Book Center.

[Ferraris *et al.*, 2006] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A generalization of the lin-zhao theorem. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):79–101, 2006.

[Ferraris, 2005] Paolo Ferraris. Answer sets for propositional theories. In *Logic Programming and Nonmonotonic Reasoning, 8th International Conference*, volume 3662 of *Lecture Notes in Computer Science*, pages 119–131, Diamante, Italy, 2005. Springer.

[Gabbay *et al.*, 2011] Dov M. Gabbay, David Pearce, and Agustín Valverde. Interpolable formulas in equilibrium logic and answer set programming. *Journal of Artificial Intelligence Research*, 42:917–943, 2011.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, Seattle, Washington, 1988. MIT Press.

[Konev *et al.*, 2012] Boris Konev, Michel Ludwig, Dirk Walther, and Frank Wolter. The logical difference for the lightweight description logic el. *Journal of Artificial Intelligence Research*, 44:633–708, 2012.

[Lang and Marquis, 2010] Jérôme Lang and Pierre Marquis. Reasoning under inconsistency: A forgetting-based approach. *Artificial Intelligence*, 174(12-13):799–823, 2010.

[Lang *et al.*, 2003] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Propositional independence: Formula-variable independence and forgetting. *Journal of Artifical Intelligence Research*, 18:391–443, 2003.

[Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):369–389, 1999.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[Lin and Reiter, 1994] Fangzhen Lin and Ray Reiter. Forget it! In *In Proceedings of the AAAI Fall Symposium on Relevance*, pages 154–159, 1994.

[Lin and Zhao, 2004] Fangzhen Lin and Yuting Zhao. ASSAT: computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, 157(1-2):115–137, 2004.

[Liu and Wen, 2011] Yongmei Liu and Ximing Wen. On the progression of knowledge in the situation calculus. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 976–982, Barcelona, Catalonia, Spain, 2011. IJCAI/AAAI.

[Pearce *et al.*, 2009] David Pearce, Hans Tompits, and Stefan Woltran. Characterising equilibrium logic and nested logic programs: Reductions and complexity. *Theory and Practice of Logic Programming*, 9(5):565–616, 2009.

[Pearce, 1996] David Pearce. A new logical characterisation of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming, NMELP'96*, volume 1216 of *Lecture Notes in Computer Science*, pages 57–70, Bad Honnef, Germany, 1996. Springer.

[Visser, 1996] Albert Visser. Uniform interpolation and layered bisimulation. In *Gödel'96*, pages 139–164, 1996.

[Wang *et al.*, 2010] Zhe Wang, Kewen Wang, Rodney W. Topor, and Jeff Z. Pan. Forgetting for knowledge bases in dl-lite. *Annuals of Mathematics and Artificial Intelligence*, 58(1-2):117–151, 2010.

[Wang *et al.*, 2012] Yisong Wang, Yan Zhang, Yi Zhou, and Mingyi Zhang. Forgetting in logic programs under strong equivalence. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference*, pages 643–647, Rome, Italy, 2012. AAAI Press.

[Wong, 2009] Ka-Shu Wong. *Forgetting in Logic Programs*. PhD thesis, The University of New South Wales, 2009.

[Zhang and Foo, 2006] Yan Zhang and Norman Y. Foo. Solving logic program conflict through strong and weak forgettings. *Artificial Intelligence*, 170(8-9):739–778, 2006.

[Zhang and Zhou, 2009] Yan Zhang and Yi Zhou. Knowledge forgetting: Properties and applications. *Artificial Intelligence*, 173(16-17):1525–1537, 2009.