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, QLD 4111, 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 unforbidden 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; 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 \( \Pi_1 \) and \( \Pi_2 \), they are equivalent if they have the same stable models; they are strongly equivalent if \( \Pi_1 \cup \Pi_2 \) and \( \Pi_2 \cup \Pi_1 \) have the same stable models for every logic program \( \Pi \). 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 \( \Pi \) as a logical theory \( \Pi \), the answer sets of \( \Pi \) exactly correspond to the equilibria of \( \Pi \) in equilibrium logic; two logic programs \( \Pi \) and \( \Pi' \) are strongly equivalent whenever \( \hat{\Pi} \) and \( \hat{\Pi}' \) are equivalent in the monotonic logic here-and-there (HT) [Lifschitz et al., 2001]. This naturally defines the entailment relationship between logic programs, viz. \( \Pi \) entails \( \Pi' \) if \( \hat{\Pi} \) entails \( \hat{\Pi}' \) 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 \( L \) be an ASP language on a signature \( A \), \( \Pi \) a logic program in \( L, V \subseteq A \) and \( f(\Pi, V) \) a result of forgetting about \( V \) in \( \Pi \).

(E) Existence: \( f(\Pi, V) \) is expressible in \( L \).

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

(W) Weakening: \( \Pi \) entails \( f(\Pi, V) \).

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

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

(SE) Strong Equivalence: If \( \Pi \) and \( \Pi' \) are strongly equivalent, then \( f(\Pi, V) \) and \( f(\Pi', V) \) are strongly equivalent.

(CP) Consequence Persistence: \( f(\Pi, V) \) has an answer set \( X' \) whenever \( \Pi \) 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 \( \Pi \) if \( p \) belongs to one (resp. all) answer set(s) of \( \Pi \).
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 comes out: 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

\[ \mathcal{F} = \{ (E), (IR), (CP), (PP), (SE) \} \]

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 \( \Pi \) consisting of

\[ \text{plum} \leftarrow \text{banana}; \quad \text{apple} \lor \neg \text{pear}; \quad \neg \text{pear} \leftarrow \neg \text{not apple}. \]

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

In terms of forgetting, if \text{pear} is forgotten in \( \Pi \), we obtain the following results under the semantic forgetting, \( F_W \), \( F_S \) and the HT-forgetting, respectively:

\[ \{ \text{apple} \leftarrow \neg \text{not apple} \}, \quad \{ \text{plum} \leftarrow \text{banana} \}, \quad \{ \text{plum} \leftarrow \text{banana}; \text{apple} \}, \quad \{ \text{plum} \leftarrow \text{banana} \}. \]

One can see that the information “\text{plum} \leftarrow \text{banana}” is lost in the semantic forgetting; in terms of \( F_W \) and HT-forgetting, the only diet scheme is \{\}; it is \{\text{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 fulfills 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 \( \mathcal{L}_A \) on a finite set \( A \) of propositional variables. \( A \) is also referred to as the signature of \( \mathcal{L}_A \). The formulas of \( \mathcal{L}_A \) are inductively constructed using connectives \( \bot, \land, \lor \) and \( \lnot \) as the following:

\[ \varphi ::= \bot \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \]

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

Let \( S \) be a finite set of formulas. We denote \( \bigwedge S \) (resp. \( \bigvee S \)) the disjunction (resp. conjunction) of all formulas in \( S \). Particularly, \( \bigwedge \emptyset \) (resp. \( \bigvee \emptyset \)) 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 \( H \subseteq T \subseteq A \). The satisfiability relation in HT, written \( \models_{HT} \), is recursively defined in the following.

\[ \langle H, T \rangle \models_{HT} p \quad \text{if} \quad p \in H \land \langle H, T \rangle \models_{HT} T \]

\[ \langle H, T \rangle \models_{HT} \varphi \lor \psi \quad \text{if} \quad \langle H, T \rangle \models_{HT} \varphi \lor \langle H, T \rangle \models_{HT} \psi \]

\[ \langle H, T \rangle \models_{HT} \varphi \land \psi \quad \text{if} \quad \langle H, T \rangle \models_{HT} \varphi \land \langle H, T \rangle \models_{HT} \psi \]

\[ \langle H, T \rangle \models_{HT} \varphi \lor \psi \quad \text{if} \quad \text{both} (i) T \models \varphi \lor \psi, \text{and} \quad \text{(ii)} \langle H, T \rangle \models_{HT} \varphi \lor \psi \]

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

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

A formula of the following form is called a rule:

\[ \bigwedge \neg D \land \bigwedge \neg C \land B \lor A \]
where $A = \{a_1, \ldots, a_t\}$, $B = \{b_1, \ldots, b_k\}$, $C = \{c_1, \ldots, c_m\}$ and $D = \{d_1, \ldots, d_n\}$. It is also identified as

$$\tilde{v} = \{a_1, \ldots, a_t \leftarrow b_1, \ldots, b_k, c_1, \ldots, c_m, \neg d_1, \ldots, \neg d_n\},$$

which is called a nested expression in [Lifschitz et al., 1999].

A rule $r$ of the 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 $\Pi$ 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 $\hat{r}$ of the form (2), it is well-known that a set $T$ of atoms is a stable model/answer set of a logic program $\Pi$ iff $(T, \Pi)$ is an equilibrium model of $\Pi$, where $\Pi = \{\hat{r} \mid r \in \Pi\}$ [Pearce, 1996; Ferraris, 2005].

Let $\Pi$ be a logic program. By $\text{SM}(\Pi)$ we denote the set of all stable models of $\Pi$. $\Pi$ is consistent if $\text{SM}(\Pi) \neq \emptyset$. If $\Pi$ is disjunctive then $\text{SM}(\Pi)$ forms an antichain, i.e., no two elements of $\text{SM}(\Pi)$ are comparable under set inclusion.

An atom $p$ is a skeptical (resp. credulous) consequence of $\Pi$, denoted by $\Pi \models \exists p$ (resp. $\Pi \models \forall p$), if $p \in \bigcap \text{SM}(\Pi)$ (resp. $p \in \bigcup \text{SM}(\Pi)$). A logic programs $\Pi'$ is equivalent to $\Pi$, written $\Pi' \equiv \Pi$, if $\text{SM}(\Pi') = \text{SM}(\Pi)$; $\Pi'$ is strongly equivalent to $\Pi$ if $\text{SM}(\Pi \cup \Pi'') = \text{SM}(\Pi' \cup \Pi'')$ for every logic program $\Pi''$. 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 every 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 $A \setminus M$. 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 $V \subseteq A$ 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 $X_V$, 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 $\langle H, T \rangle_V$, 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 $\bigcup_{\mathcal{M}}^\mathcal{M}_V$.

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

**Definition 1 (forgetting)** Let $\varphi$ be a formula and $V \subseteq A$. A formula $\psi$ is a result of forgetting $V$ in $\varphi$ if and only if $\text{Mod}(\psi) = \text{Mod}(\varphi)_V$.

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

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

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

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

### 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 $V \subseteq A$ and $X$ an interpretation. The V-exclusion of $X$, written $X\parallel V$, is the set $X \setminus V$. The V-exclusion of a collection $\mathcal{M}$ of interpretations is $\{X\parallel V \mid X \in \mathcal{M}\}$, denoted by $\mathcal{M}\parallel V$.

**Definition 3 (SM-forgetting)** Let $\varphi$ be a formula and $V \subseteq A$. A formula $\psi$ is a result of SM-forgetting $V$ in $\varphi$ if and only if $\text{Mod}_{HT}(\psi) = \text{Mod}_{HT}(\varphi)\parallel V$.

**Example 1** For the logic program $\Pi$ in Section 1, the following logic program is a result of SM-forgetting $\text{pear}$ in $\Pi$:

$$\{\text{plum} \leftarrow \text{banana}; \text{apple} \leftarrow \neg \text{not apple}\}.$$

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

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

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

**Proof sketch:** We denote that

$$G = \{(X, Y) \mid X \subset Y \land Y \in \text{SM}(\varphi)_V\},$$

$$\mathcal{H} = \{(Y, Y') \mid Y \in \text{SM}(\text{Forget}_{HT}(\varphi, V)) \setminus \text{SM}(\varphi)_V\},$$

$$\mathcal{N} = \text{Mod}_{HT}(\varphi)_V \setminus (G \cup \mathcal{H}).$$

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

$$\langle X, Y \rangle \in \mathcal{M} \text{ implies } \langle Y, Y \rangle \in \mathcal{M}.$$  \hspace{1cm} (7)

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

Recall that $\text{Forget}_{HT}(\varphi, V) = \text{HT}\{\{\varphi\} = \text{HT}\xi \land \text{IR}_{HT}(\xi, V)\}$ (cf. Theorem 3 of [Wang et al., 2012]). By the construction of $\text{Forget}_{SM}(\varphi, V)$ 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 $\varphi \models_{HT} \psi$ implies ForgetSM$(\varphi, V) \models_{HT} \text{For} \text{get}_{SM}(\psi, V)$.

Proposition 2 Let $\varphi, \psi$ be two formulas and $V \subseteq A$. Then

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

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

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

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 $\psi = q \land (\neg p \supset p)$. It is not difficult to see that, on the signature $\{p,q\}$, $\text{Mod}_{HT}(\psi) = \{\{q\}, \{p,q\}\}$ and then $\psi$ has no stable model. One can verify that $\text{Forget}_{HT}(\psi, \{q\}) \models_{HT} \neg p \supset p$, $\text{Forget}_{HT}(\psi, \{p\}) \models_{HT} q$.
- Let $\varphi = (p \lor \neg p \lor q \lor \neg q) \land (p \supset q \lor \neg q) \land (q \supset p \lor \neg p)$. One can check that $\text{SM}(\varphi) = \emptyset$, $\{p,q\}$, and $\text{Forget}_{HT}(\varphi, \{p\}) \models_{HT} q \lor \neg q$.

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

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

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 $\varphi$ be a formula and $V \subseteq A$. Then we have $\varphi \models_{HT} \text{Forget}_{SM}(\varphi, V)$ iff, for every $Y \subseteq A$.

- $\langle X, Y \rangle \models_{HT} \varphi$ for every $X \subset Y$ if $Y \in \text{SM}(\varphi)_{V}$.
- $Y \models \varphi$ implies $\langle X, Y \rangle \models_{HT} \varphi$ 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 $\varphi$ be a formula, $V_1, V_2$ two sets of atoms. $\text{Forget}_{SM}(\varphi, V_1 \cup V_2) \equiv_{HT} \text{Forget}_{SM}(\text{Forget}_{SM}(\varphi, V_1), V_2)$.

Proof sketch: Let’s denote $\psi := \text{Forget}_{SM}(\varphi, V_1)$ and $\psi^* := \text{Forget}_{SM}(\varphi, V_1)$, $\psi^* := \text{Forget}_{SM}(\varphi, V_1)$. $\psi := \text{Forget}_{SM}(\varphi, V_1)$, $\psi^* := \text{Forget}_{SM}(\varphi, V_1)$. We can show that, if $Y \notin \text{SM}(\varphi)_{V_1}$ then

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

The next example shows that SM-forgetting results for some important classes of logic programs, including disjunctive logic programs and Horn logic programs.

Example 3 Consider the following normal logic programs.

- Let $\Pi_1$ consists of
  
  $p \not\models q$; $q \not\models p$; $r \not\models p$; $r \not\models q$.

  It has two stable models $\{p, q\}$ and $\{p, r\}$. We have
  
  $\text{Forget}_{SM}(\Pi_1, \{q\}) \equiv_{HT} \{p \not\models not p; r \not\models p\}$,
  
  $\text{Forget}_{HT}(\Pi_1, \{q\}) \equiv_{HT} \{r \not\models not p; r \not\models p\}$.

  The stable models of $\text{Forget}_{SM}(\Pi_1, \{q\})$ 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 $\text{Forget}_{SM}(\Pi_1, \{q\})$, even if $\text{Forget}_{HT}(\Pi_1, \{q\})$ is disjunctive.

- Let $\Pi_2 = \{\leftarrow p, q, r; \leftarrow p, q, not r\}$. Its unique stable model is $\emptyset$. We have
  
  $\text{Forget}_{SM}(\Pi_2, \{r\}) \equiv_{HT} \text{Forget}_{HT}(\Pi_2, \{r\})$,
  
  $\equiv_{HT} \{p \lor q \not\models not p, not q\}$.
Theorem 3 Thus expressible in disjunctive logic programs.

Proposition 5 Let \( \varphi \) be a formula and \( V \subseteq A \). There is a disjunctive logic program \( \Pi \) such that \( \Pi \equiv_{HT} \text{For}

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

Secondly we denote:

\[ G = \{ (X, Y) | X \subseteq Y \land Y \in SM(\Pi) \} \]

\[ \mathcal{H} = \{ (Y, Y) | Y \in SM(\text{Forget}_{HT}(\Pi, V)) \land Y \notin SM(\Pi) \} \]

\[ \mathcal{N} = \text{Mod}_{HT}(\Pi, V) \].

It is evident that \( \mathcal{H} = \emptyset \). We can further show \( \mathcal{N} \cap G = \emptyset \).)

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 \( \Pi, \Pi' \) be Horn logic programs and \( V \subseteq A \).

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

(ii) \( \text{Forget}_{HT}(\Pi \cup \Pi', V) \equiv_{HT} \text{Forget}_{SM}(\Pi, V) \cup \Pi' \) if \( I_{HT}(\Pi', V) \).

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 \( \varphi \) be a formula and \( V \subseteq A \). Then we have \( \text{Forget}_{SM}(\varphi, V) = I_{HT}(\varphi, V) \) and only if \( SM(\text{Forget}_{HT}(\varphi, V)) \).

The notion of loop formulas plays an important role in ASP [Lin and Zhao, 2004]. As stable models of a logic program \( \Pi \) 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 \( \Pi \) be a logic program and \( V \) a set of atoms. Then \( X \) is a stable model \( \text{Forget}_{SM}(\Pi, V) \) iff \( X \) is a model of \( \text{Forget}(LF(\Pi), V) \).

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 \( \sim \) between formulas in ASP:

\[ \varphi \sim \psi \text{ if } M \models \psi \text{ for every } M \in SM(\varphi). \] (8)

A class of formulas in ASP has the uniform interpolation property iff for any formula \( \varphi \) and any set \( V \) of atoms, there is a formula \( \psi \) such that the following conditions hold:

(i) \( \varphi(\varphi) \subseteq \varphi(\varphi) \setminus V \), and

(ii) for any formula \( \alpha \) with \( \varphi(\alpha) \cap V = \emptyset \),

\[ \varphi(\alpha) \models \varphi(\alpha) \iff \psi \models \psi \] (9)

where \( \varphi(\alpha) \) is the set of atoms occurring in \( \alpha \).

Gabbay et al. (2011) showed that the class of disjunctive logic programs enjoy the uniform interpolation property and they raised an interesting open question: Given a signature \( A \), does \( L_A \) the class of all formulas in ASP, possess the uniform interpolation property too?

This open question can be affirmatively answered.

Proposition 8 Let \( A \) be a set of atoms. Then \( L_A \), 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 in here-and-there [Cabalar and Ferraris, 2007], we present an algorithm to compute SM-forgetting result – Algorithm 1, where the underlying signature \( A \) consists of the atoms occurring in \( \Pi \) and \( \lambda_{X,Y} \) denotes the following formula:

\[ \bigwedge X \land \bigwedge \neg Y \supset \bigvee z \lor \neg z. \] (10)

In particular, \( \lambda_{Y,Y} = \bigwedge Y \land \bigwedge \neg Y \supset \bot \) since \( \bigvee \emptyset \) is \( \bot \).

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

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

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

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

Theorem 5 Algorithm 1 outputs \( \text{Forget}_{SM}(\Pi, V) \). In the algorithm, since \( SM(\Pi) \) 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 \( \Pi_1 \) from Example 3 where \( A = \{ p, q, r \} \). Let \( V = \{ q \} \). Note that \( M = \{ \{ p, r \}, \{ q, r \} \} \) and \( \mathcal{M}_{\Pi_1} = \{ \{ p, r \}, \{ r \} \} \). Now \( \Sigma \) is constructed as follows:

\(^2\)The query \( \alpha \) in (9) is restricted being a disjunction of literals.

\(^3\)An HT-interpretation is a countermodel of a formula \( \varphi \) if it not an HT-model of \( \varphi \).
Corollary 6 Let $\Pi$ be a (disjunctive) logic program, $V$ a set of atoms and $p$ an atom not in $V$. The following hold:

(i) deciding if $\text{SM}(\text{Forget}_{\Sigma}(\Pi, V)) \neq \emptyset$ is $\Sigma^P_{2}$-complete.

(ii) deciding if $\text{Forget}_{\Sigma}(\Pi, V) \models p$ is $\Pi^P_{2}$-complete.

(iii) deciding if $\text{Forget}_{\Sigma}(\Pi, V) \models_{c} p$ is $\Sigma^P_{2}$-complete.

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 7 Let $\Sigma$ and $\Pi$ be two logic programs, $V \subseteq \Sigma$.

(i) Deciding if $\Sigma \equiv_{HT} \text{Forget}_{\Sigma}(\Pi, V)$ is $\Pi^P_{2}$-complete.

(ii) Deciding if $\Sigma \equiv_{HT} \text{Forget}_{\Sigma}(\Pi, V)$ is co-NP-complete if both $\Sigma$ and $\Pi$ are Horn logic programs.

Proof sketch: (i) Membership: If $\Sigma \neq_{HT} \text{Forget}_{\Sigma}(\Pi, V)$ then there exists an HT-interpretation $(X, Y)$ such that either (a) $(X, Y) \models_{HT} \Sigma$ and $(X, Y) \neq_{HT} \text{Forget}_{\Sigma}(\Pi, V)$, or (b) $(X, Y) \neq_{HT} \Sigma$ and $(X, Y) \models_{HT} \text{Forget}_{\Sigma}(\Pi, V)$. We consider the case (a). The case (b) is similar.

Note that checking if $(X, Y) \models_{HT} \Sigma$ is feasible in polynomial time of $\Sigma$. Moreover, $(X, Y) \neq_{HT} \text{Forget}_{\Sigma}(\Pi, V)$ iff either (i.a) $(X, Y) \neq_{HT} \text{Forget}_{HT}(V, Y, \Pi)$ and $X \subseteq Y \subseteq \Pi$ or (i.b) $X = Y$ and $Y \in \text{SM}(\text{Forget}_{HT}(\Pi, V)) \setminus \text{SM}(\Pi)$. The case (i.a) is feasible in polynomial by calling an NP-oracle. We consider the case (i.b). Note further that $Y \in \text{SM}(\text{Forget}_{HT}(\Pi, V))$ if $Y \models_{HT} \text{Forget}_{HT}(\Pi, V)$, which is feasible in polynomial time of $\Pi$ by calling an NP-oracle, and $(X, Y) \neq_{HT} \text{Forget}_{HT}(\Pi, V)$ for every $X \subseteq Y$, whose complement is $(X, Y) \models_{HT} \text{Forget}_{HT}(\Pi, V)$ for some $X \subseteq Y$ iff there is $(X', Y') \neq_{HT} \Pi$ such that $Y' \sim_{V} X'$ and $X' \setminus V \subseteq Y$, which is feasible in polynomial time of $\Pi$ by calling an NP-oracle as well.

Note that $\Sigma \models_{HT} \text{Forget}_{\Sigma}(\Pi, V)$ iff (a) $\Sigma \models_{HT} \text{Forget}_{HT}(\Pi, V)$, (b) $\text{SM}(\Sigma) = \text{SM}(\Pi) \cup V$ and (c) there is no $\Sigma'$ such that $\text{Mod}_{HT}(\Sigma') \subseteq \text{Mod}_{HT}(\Sigma)$ and $\Sigma'$ satisfies the conditions (a) and (b) in the place of $\Sigma$. Its hardness follows from the fact that the condition (a) is $\Pi^P_{2}$-complete (cf. (ii) of Theorem 5 in [Wang et al., 2012]).

(ii) The membership is proved by showing $(H, T) \models_{HT} \text{Forget}_{\Sigma}(\Pi, V)$ iff the least model $M$ of $\Pi$ satisfies the condition $M \cup V \subseteq H \cap T$. Its hardness follows from the fact that $\Sigma \equiv_{HT} \text{Forget}_{\Sigma}(\Pi, V)$ iff $\Sigma \equiv \text{Forget}(\Pi, V)$ which is co-NP-complete.

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 was partially supported by the Australia Research Council (ARC) grants DP1093652 and DP110101042. Yisong Wang was also partially supported by NSFC grant 60963009. Open Funds of the State Key Laboratory of Computer Science of Chinese Academy of Science grant SYSKF1106 and Stadholder Fund of Guizhou Province grant (2012)62.
References


