Journal of Artificial Intelligence Research 50 (2014) 31-70 Submitted 08/13; published 05/14

Knowledge Forgetting in Answer Set Programming

Yisong Wang CSC.YSWANG@GZU.EDU.CN
Department of Computer Science,
Guizhou University, Guiyang, China

Yan Zhang YAN@SCEM.UWS.EDU.AU
Yi Zhou YZHOU@SCEM.UWS.EDU.AU
Artificial Intelligence Research Group,
University of Western Sydney, Australia

Mingyi Zhang ZHANGMINGYI045@GMAIL.COM
Guizhou Academy of Sciences, Guiyang, China

Abstract

The ability of discarding or hiding irrelevant information has been recognized as an important feature for knowledge based systems, including answer set programming. The notion of strong equivalence in answer set programming plays an important role for different problems as it gives rise to a substitution principle and amounts to knowledge equivalence of logic programs. In this paper, we uniformly propose a semantic knowledge forgetting, called HT- and FLP-*forgetting*, for logic programs under stable model and FLP-stable model semantics, respectively. Our proposed knowledge forgetting discards exactly the knowledge of a logic program which is relevant to forgotten variables. Thus it preserves strong equivalence in the sense that strongly equivalent logic programs will remain strongly equivalent after forgetting the same variables. We show that this semantic forgetting result is always expressible; and we prove a representation theorem stating that the HT- and FLP-forgetting can be precisely characterized by Zhang-Zhou's four forgetting postulates under the HT- and FLP-model semantics, respectively. We also reveal underlying connections between the proposed forgetting and the forgetting of propositional logic, and provide complexity results for decision problems in relation to the forgetting. An application of the proposed forgetting is also considered in a conflict solving scenario.

1. Introduction

Motivated by Lin and Reiter's seminal work (Lin & Reiter, 1994), the notion of forgetting in propositional and first-order logics – distilling from a knowledge base only the part that is relevant to a subset of the alphabet – has attracted extensive interests in the KR community, (e.g., see Lang & Marquis, 2010; Zhou & Zhang, 2011). In recent years, researchers have developed forgetting notions and theories in other non-classical logic systems from various perspectives, such as forgetting in description logics (Kontchakov, Wolter, & Zakharyaschev, 2008; Wang, Wang, Topor, & Pan, 2010; Lutz & Wolter, 2011; Packer, Gibbins, & Jennings, 2011), forgetting in logic programs (Zhang & Foo, 2006; Eiter & Wang, 2008; Wong, 2009; Wang, Wang, & Zhang, 2013), and forgetting in modal logic (Zhang & Zhou, 2009; Su, Sattar, Lv, & Zhang, 2009; van Ditmarsch, Herzig, Lang, & Marquis, 2009; Liu & Wen, 2011). As a logical notion, forgetting has also been studied under some different terms such as variable elimination (Lang, Liberatore, & Marquis, 2003), irrelevance, independence, irredundancy, novelty, or separability (Bobrow, Subramanian, Greiner, &

Pearl, 1997). It has been shown that in the study of modeling agents' behaviors, forgetting plays an important role in conflict resolution (Zhang & Foo, 2006; Lang & Marquis, 2010).

In propositional logic, the result of forgetting an atom p from a formula φ, written $\mathsf{Forget}(\varphi, \{p\})$, is the formula $\varphi[p/\bot] \vee \varphi[p/\top]$, where $\varphi[p/\bot]$ and $\varphi[p/\top]$ is the formula obtained from φ by replacing each occurrence of atom p with $\bot$ (false) and $\top$ (true) respectively. Forgetting a set of atoms from a formula φ is defined as $\mathsf{Forget}(\varphi, V \cup \{p\}) = \mathsf{Forget}(\mathsf{Forget}(\varphi, \{p\}), V)$ (Lin, 2001). It is easy to see that the forgetting preserves logical equivalence. That is, logically equivalent formulas (theories) will remain logically equivalent after forgetting the same atoms. It is well known that, if ψ does not mention any atoms from V then

$$\varphi \models \psi \text{ iff } \mathsf{Forget}(\varphi, V) \models \psi.$$

In this sense the forgetting in propositional logic, called *propositional forgetting*, is a knowledge forgetting since $\mathsf{Forget}(\varphi, V)$ exactly contains the "logical content" of φ that is irrelevant to V.

For logic programs under stable model/answer set semantics (Gelfond & Lifschitz, 1988), the issue of logical equivalence is rather complicated due to its different notions of "equivalence": (weak) equivalence and strong equivalence. Two logic programs Π_1 and Π_2 are *(weakly) equivalent* if and only if Π_1 and Π_2 have the same stable models; Π_1 and Π_2 are *strongly equivalent* if and only if $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ are equivalent for every logic program Π. It is well known that strong equivalence is an important concept in answer set programming (ASP), because it amounts to *knowledge equivalence* which captures the *logical content* of a logic program (Osorio & Zacarias, 2004; Osorio & Cuevas, 2007; Delgrande, Schaub, Tompits, & Woltran, 2013), and can be used for simplifying logic programs where two strongly equivalent rules may be interchangeable without affecting the original logic programs' stable models (Pearce, Tompits, & Woltran, 2001; Ferraris, Lee, & Lifschitz, 2011; Lin & Chen, 2007; Lin & Zhou, 2011). The strong equivalence can be characterized in the logic here-and-there (HT), viz, two logic programs are strongly equivalent if and only if they have the same HT-models (Lifschitz, Pearce, & Valverde, 2001). For instance, a rule of the following form "$p \leftarrow p \wedge \varphi$" has the same HT-models as that of $\top$ (tautology), where φ can be an arbitrary formula. Thus it can be safely removed from every logic programs without changing their stable models.

Besides the stable model/answer set semantics of logic programs (Gelfond & Lifschitz, 1988), FLP-stable model semantics also steadily gains its importance (Faber, Pfeifer, & Leone, 2011; Truszczynski, 2010). The notion of strong equivalence is similarly generalized to logic programs under FLP-stable models semantics: two theories Π_1 and Π_2 are *strongly FLP-equivalent* if and only if $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ have the same FLP-stable models for every logic program Π. It is shown that this strong equivalence can be characterized in terms of FLP-models, viz, two logic programs are strongly FLP-equivalent if and only if they have the same FLP-models (Truszczynski, 2010).

When we develop the notion of forgetting in logic programs, preserving strong equivalence is important, like that the propositional forgetting preserves equivalence of propositional logic. Consider that two agents need to achieve an agreement for a certain goal, where each agent's knowledge base is represented by a logic program. Suppose that there are two consistent[1] logic programs, but their combination is inconsistent. To achieve a consistent combination, one may forget some atoms from each of the logic programs, so that the combination of their forgetting results is consistent. Then forgetting may be effectively used to solve the conflict between the two agents' knowledge

1. A logic program is *consistent* if it has some stable models.

bases (Zhang & Foo, 2006; Eiter & Wang, 2008; Lang & Marquis, 2010). For the purpose of simplicity, on the other hand, agents may also replace their knowledge bases with strongly equivalent but syntactically simpler ones.

Let us consider a simple Yale Shooting scenario where the logic program Π consisting of the following rules:[2]

$$shoot \leftarrow not\, aux; \qquad aux \leftarrow not\, shoot; \qquad \leftarrow aux, shoot.$$

Here aux is used to generate possible occurrences of action $shoot$. One can be interested in which logic program represents the same knowledge as that of Π when the auxiliary atom aux is ignored. This intuitively results in a logic program Π' consisting of the rule[3]:

$$shoot \leftarrow not\, not\, shoot,$$

which captures exactly the knowledge of Π that is irrelevant to aux. We will see that Π' can be obtained from Π by HT-forgetting aux (cf. Example 5 with other atom names), while it cannot be obtained in terms of previous forgetting approaches in logic programming (cf. Example 11).

It turns out that preserving strong equivalence in forgetting is challenging. There have been several attempts to define the notion of forgetting in logic programs, but none of these approaches is fully satisfactory. Zhang and Foo (2006) first defined syntax oriented weak and strong forgetting notions for normal logic programs. But these forgetting notions preserve neither (weak) equivalence nor strong equivalence. Eiter and Wang (2008) then proposed a semantic forgetting for consistent disjunctive logic programs, which preserves equivalence but not strong equivalence. They specifically indicated the importance of preserving strong equivalence in logic programming forgetting and raised this issue as a future work. Wong (2009) proposed two forgetting operators for disjunctive logic programs. Although the two operators indeed preserve strong equivalence, it may lose the intuition of weakening under various circumstances (see Section 5 for details). A recently proposed forgetting for logic programs may introduce extra knowledge (cf., see Wang et al., 2013, Ex. 2). Thus it is not a knowledge forgetting.

Together with preserving strong equivalence, expressiveness is another desired criterion for logic programming forgetting. Ideally we would expect that the result of forgetting some atoms from a logic program is still expressible by a logic program. This is particularly necessary when we view agents' knowledge bases as logic programs and forgetting is employed as a means of conflict solving among these agents' knowledge bases (Zhang & Foo, 2006). While previous logic programming forgetting approaches all meet this criterion, as we will see in this paper, once we consider forgetting in arbitrary logic programs, retaining expressibility is challenging objective to achieve for a semantic forgetting notion.

Finally, we believe that as a way of weakening, knowledge forgetting in logic programs should obey some common intuitions shared by forgetting in classical logics. For instance, forgetting something from a logic program should lead to a weaker program in certain sense. On the other hand, such weakening should only be associated to the relevant information that has been forgotten. For this purpose, Zhang and Zhou (2009) proposed four forgetting postulates to formalize these common intuitions and showed that forgetting in propositional logic and modal logic S5 can be precisely captured by these postulates. Surprisingly, none of previous forgetting notions in logic

2. This is due to one of the anonymous reviewers.
3. The rule is strongly equivalent to the choice rule "$0\{shoot\}1$" but it is not a normal rule.

programs actually satisfies Zhang-Zhou's postulates. In this sense these previous forgetting notions for logic programs are not knowledge forgetting operators.

In summary, we consider the following criteria that a knowledge forgetting notion in logic programs should meet:

- Expressibility. The result of forgetting in an arbitrary logic program should also be expressible via a logic program;
- Preserving strong equivalence. Two strongly equivalent logic programs should remain strongly equivalent after forgetting the same variables;
- Satisfying common intuitions of forgetting. Preferably, forgetting in logic programs should be semantically characterized by Zhang-Zhou's four forgetting postulates.

In this paper we present a comprehensive study on knowledge forgetting in the context of arbitrary logic programs (propositional theories) under stable model and FLP-stable models semantics, called HT- and FLP-*forgetting* respectively. We show that the HT- and FLP-forgetting meet all above criteria, and hence have primary advantages when compared to previous logic program forgetting notions.

The main contributions of the paper may be summarized as follows, where $\star \in \{\text{HT}, \text{FLP}\}$,

- As a starting point, we investigate the model theoretical characterization for strong equivalence of logic programs under stable model and FLP-stable model semantics, and explore their strong equivalence by the equivalence in propositional logic.
- We propose a semantic $\star$-forgetting for logic programs under $\star$-stable model semantics respectively. Here HT-stable model means stable model. The $\star$-forgetting result is always expressible via a logic program and it preserves strong equivalence under stable model and FLP-stable model semantics.
- We investigate semantic properties of the $\star$-forgetting, and show that the $\star$-forgetting satisfies Zhang-Zhou's four postulates under the $\star$-model respectively. In particular, the forgetting result consists of the logical content that is irrelevant to forgotten atoms.
- We establish the underlying connections between $\star$-forgetting and propositional forgetting, based on which we provide complexity results for some decision problems in relation to $\star$-forgetting. In particular, we show that resulting checking – deciding if a logic program is a result of $\star$-forgetting a set of atoms from a logic program – is Π_2^P-complete, while the related inference problem in terms of $\star$-forgetting varies from co-NP-complete to Π_2^P-complete.

  The theoretical negative results confirm that it is not a easy task to simplify logic programs by forgetting. But fortunately, this kind of simplification can be computed offline in general. For instance, a problem domain description involves a lot of auxiliary propositional variables. One can firstly simplify the description by forgetting (part of) the auxiliary propositional variables, like a kind of compilation (Lang et al., 2003).
- Finally we consider an application of knowledge forgetting in the solving of conflicts in the context of logic programming.

The rest of the paper is organized as follows. Section 2 briefly reviews necessary concepts and notions of answer set programming. Section 3 presents the characterizations for strong equivalence of logic programs. We firstly present a uniform definition of the knowledge forgetting for logic programs in section 4, and then explore their expressibility, forgetting postulates, relationship with propositional forgetting, computational complexity and an application of knowledge forgetting in conflict solving. Section 5 discusses other forgetting approaches in logic programs, and finally, Section 6 concludes the paper with some remarks. All the proofs in the paper are deferred to Appendix for clarity.

This paper is the revised and extended version of a paper which appeared in Proceedings of KR 2012 (Wang, Zhang, Zhou, & Zhang, 2012).

2. Answer Set Programming

In this section we briefly recall the basic notions of logic programming under stable model semantics, including its syntax, reduction, stable model (Ferraris, 2005) and FLP-stable models (Truszczynski, 2010) and strong equivalence (Lifschitz et al., 2001; Truszczynski, 2010). In the paper a "stable model" is called an HT-stable model for convenience, and we assume $\star \in \{\text{HT}, \text{FLP}\}$.

We assume a propositional language $\mathcal{L}_\mathcal{A}$ over the finite set $\mathcal{A}$ of propositional atoms, which is called the *signature* of the language $\mathcal{L}_\mathcal{A}$.

2.1 Syntax

The *formulas* of $\mathcal{L}_\mathcal{A}$ are built from the signature[4] $\mathcal{A}$ and the 0-place connective $\bot$ ("*false*") using the binary connectives $\wedge, \vee$ and $\supset$ as follows:

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

where $p \in \mathcal{A}$. $\top$ ("*true*") is the shorthand of $\bot \supset \bot$, $\neg\varphi$ for $\varphi \supset \bot$, and $\psi \leftrightarrow \phi$ for $(\psi \supset \phi) \wedge (\phi \supset \psi)$. A *theory* is a set of formulas.

An *interpretation* is a set I of atoms from $\mathcal{A}$, where each atom of $\mathcal{A}$ is viewed to be true if it is in I, and false otherwise. In propositional logic, the notions of *model* and *satisfaction* relation $\models$ are defined as usual. In the following we denote $\mathcal{A} \setminus X$ by $\overline{X}$ for $X \subseteq \mathcal{A}$, $\mathsf{Mod}(\varphi)$ for $\{M | M \models \varphi\}$, $\varphi \equiv \psi$ for $\mathsf{Mod}(\varphi) = \mathsf{Mod}(\psi)$ (i.e. φ is *equivalent* to ψ) and $\overline{\mathcal{M}}$ for $\{I \subseteq \mathcal{A} | I \notin \mathcal{M}\}$ where $\mathcal{M} \subseteq 2^\mathcal{A}$. A formula φ is *irrelevant to* a set V of atoms, written $\mathsf{IR}(\varphi, V)$, if there exists a formula ψ mentioning no atoms from V such that $\varphi \equiv \psi$.

For convenience, we also define the following notations. Let S be a finite set of formulas. We denote $\bigvee S$ (resp. $\bigwedge S$) the disjunction (resp. conjunction) of all formulas in S, where $\bigvee \emptyset$ denotes $\bot$ and $\bigwedge \emptyset$ denotes $\top$, and $|S|$ the cardinality of S. Similarly by $\neg S$ (resp. $\neg\neg S$) we mean $\{\neg\phi \mid \phi \in S\}$ (resp. $\{\neg\neg\phi \mid \phi \in S\}$).

2.2 Reduct and Stable Models

Let φ be a formula and $X \subseteq \mathcal{A}$. The $\star$-*reduct* of φ w.r.t. X, written $\mathsf{Red}_\star(\varphi, X)$, is recursively and uniformly defined as follows:

4. In the rest of this paper, whenever there is no confusion, we may not explicitly mention the signature when we talk about formulas of $\mathcal{L}_\mathcal{A}$.

($\star$-R1) $\mathsf{Red}_\star(\bot, X) = \bot$;

($\star$-R2) $\mathsf{Red}_\star(p, X) = p$ if $X \models p$, and $\bot$ otherwise;

($\star$-R3) $\mathsf{Red}_\star(\varphi_1 \circ \varphi_2, X) = \mathsf{Red}_\star(\varphi_1, X) \circ \mathsf{Red}_\star(\varphi_2, X)$ if $X \models \varphi_1 \circ \varphi_2$ where $\circ \in \{\wedge, \vee\}$, and $\bot$ otherwise;

(HT-R4) $\mathsf{Red}_{\text{HT}}(\varphi_1 \supset \varphi_2, X) = \mathsf{Red}_{\text{HT}}(\varphi_1, X) \supset \mathsf{Red}_{\text{HT}}(\varphi_2, X)$ if $X \models \varphi_1 \supset \varphi_2$, and $\bot$ otherwise;

(FLP-R4) $\mathsf{Red}_{\text{FLP}}(\varphi_1 \supset \varphi_2, X) = \begin{cases} \varphi_1 \supset \mathsf{Red}_{\text{FLP}}(\varphi_2, X), & \text{if } X \models \varphi_1 \wedge \varphi_2; \\ \top, & \text{if } X \not\models \varphi_1; \\ \bot, & \text{otherwise (i.e. } X \not\models \varphi_1 \supset \varphi_2). \end{cases}$

Definition 1 *A set $X \subseteq \mathcal{A}$ is a $\star$-*stable model *of a formula φ if X is a minimal (under set inclusion) model of* $\mathsf{Red}_\star(\varphi, X)$. *We denote the set of $\star$-stable models of φ by* $\text{SM}_\star(\varphi)$.

Please note that, traditionally, the HT-reduct is named "reduct"; $\mathsf{Red}_{\text{HT}}(\varphi, X)$ is written as "φ^X"; HT-stable model is called "stable model" (Ferraris, 2005); and $\mathsf{Red}_{\text{FLP}}(\varphi, X)$ is written as "$\varphi^{\underline{X}}$" (Truszczynski, 2010).

It is known that, HT-stable models and FLP-stable models are not comparable in the sense that some HT-stable models are not FLP-stable models, and some FLP-stable models are not HT-stable models (cf., see Truszczynski, 2010, Exs. 1, 2, 4 and 5).

Example 1 Let us consider the following formulas:

- Let $\varphi = p \vee \neg p \supset p$. We have that

  $$\mathsf{Red}_{\text{HT}}(\varphi, \emptyset) \equiv \bot, \mathsf{Red}_{\text{HT}}(\varphi, \{p\}) \equiv \top, \mathsf{Red}_{\text{FLP}}(\varphi, \emptyset) \equiv \bot, \mathsf{Red}_{\text{HT}}(\varphi, \{p\}) \equiv p.$$

  Thus $\text{SM}_{\text{HT}}(\varphi) = \emptyset$, while $\text{SM}_{\text{FLP}}(\varphi) = \{\{p\}\}$.

- Let $\varphi_1 = p \vee \neg p$ and $\varphi_2 = \neg\neg p \supset p$. We have the following:

  $\mathsf{Red}_{\text{HT}}(\varphi_i, \emptyset) \equiv \top$ and $\mathsf{Red}_{\text{HT}}(\varphi_i, \{p\}) \equiv p$, for $i = 1, 2$,
  $\mathsf{Red}_{\text{FLP}}(\varphi_1, \emptyset) \equiv \top, \mathsf{Red}_{\text{FLP}}(\varphi_1, \{p\}) \equiv p, \mathsf{Red}_{\text{FLP}}(\varphi_2, \emptyset) \equiv \top, \mathsf{Red}_{\text{FLP}}(\varphi_2, \{p\}) \equiv \top$.

  Thus, while $\text{SM}_{\text{FLP}}(\varphi_1) = \text{SM}_{\text{HT}}(\varphi_1) = \{\emptyset, \{p\}\}$, $\text{SM}_{\text{FLP}}(\varphi_2) = \{\emptyset\}$.

Definition 2 *Two formulas φ_1 and φ_2 are* $\star$-SM-equivalent (under $\star$-stable model semantics)*, written $\varphi_1 \equiv_\star^{\text{SM}} \varphi_2$, iff they have the same $\star$-stable models.*

Here the notion of HT-SM-equivalence is indeed the notion of equivalence in logic programs under stable model semantics (cf., see Lifschitz et al., 2001, Thm. 1).

2.3 Strong Equivalence and Knowledge of Logic Programs

Unlike the equivalence in propositional logic, the equivalence of logic programs does not allow equivalent replacement i.e., $\varphi \wedge \varphi_1$ and $\varphi \wedge \varphi_2$ may have different stable models, even though φ_1 and φ_2 are equivalent.

Example 2 Let $\varphi_1 = p \supset q$ and $\varphi_2 = p \supset p$. As $\text{SM}_\star(\varphi_1) = \text{SM}_\star(\varphi_2) = \{\emptyset\}$, φ_1 and φ_2 are $\star$-SM-equivalent; however, $p \wedge \varphi_1$ has a $\star$-stable model $\{p, q\}$ while the unique $\star$-stable model of $p \wedge \varphi_2$ is $\{p\}$. Thus it does not allow replacing φ_1 by φ_2 in $p \wedge \varphi_1$. It also indicates that φ_1 has different "knowledge" from φ_2 under the $\star$-stable model semantics.

This motivates the notion of strong equivalence.

Definition 3 *Two formulas φ_1 and φ_2 are* strongly $\star$-equivalent (under $\star$-stable model semantics) *iff $\varphi \wedge \varphi_1 \equiv_\star^{\text{SM}} \varphi \wedge \varphi_2$ for every formula φ. In the case φ_1 and φ_2 are strongly $\star$-equivalent, they are* $\star$-knowledge equivalent.

It is known that the notion of strong $\star$-equivalence can be captured in terms of $\star$-models, where a $\star$-interpretation is a pair $\langle X, Y\rangle$ such that $X \subseteq Y \subseteq \mathcal{A}$. The *$\star$-satisfiability* (thus *$\star$-models*), denoted by $\models_\star$, is recursively defined as follows:

($\star$-S1) $\langle X, Y\rangle \not\models_\star \bot$;

($\star$-S2) $\langle X, Y\rangle \models_\star p$ if $p \in X$;

($\star$-S3) $\langle X, Y\rangle \models_\star \varphi_1 \vee \varphi_2$ if $\langle X, Y\rangle \models_\star \varphi_1$ or $\langle X, Y\rangle \models_\star \varphi_2$;

($\star$-S4) $\langle X, Y\rangle \models_\star \varphi_1 \wedge \varphi_2$ if $\langle X, Y\rangle \models_\star \varphi_1$ and $\langle X, Y\rangle \models_\star \varphi_2$;

(HT-S5) $\langle X, Y\rangle \models_{\text{HT}} \varphi_1 \supset \varphi_2$ if $Y \models \varphi_1 \supset \varphi_2$; and $\langle X, Y\rangle \models_{\text{HT}} \varphi_1$ implies $\langle X, Y\rangle \models_{\text{HT}} \varphi_2$;

(FLP-S5) $\langle X, Y\rangle \models_{\text{FLP}} \varphi_1 \supset \varphi_2$ if $Y \models \varphi_1 \supset \varphi_2$; and $Y \not\models \varphi_1$ or $X \not\models \varphi_1$ or $\langle X, Y\rangle \models_{\text{FLP}} \varphi_2$.

By $\mathsf{Mod}_\star(\varphi)$ we denote the set of all $\star$-models of formula φ. Please note here that, $\star$ can be either HT or FLP. In particular, $\mathsf{Mod}_{\text{HT}}(\varphi)$ (resp. $\mathsf{Mod}_{\text{FLP}}(\varphi)$) denotes the set of all HT-models (resp. FLP-models) of φ. For the formulas φ_1 and φ_2 in Example 2, one can check that none of $\langle\emptyset, \{p\}\rangle$, $\langle\{p\}, \{p\}\rangle$ or $\langle\{p\}, \{p, q\}\rangle$ is a $\star$-model of φ_1, while every $\star$-interpretation is a $\star$-model of φ_2.

Definition 4 *A formula ψ is a* logical $\star$-consequence *of a formula φ, written $\varphi \models_\star \psi$, iff* $\mathsf{Mod}_\star(\varphi) \subseteq \mathsf{Mod}_\star(\psi)$*; two formulas φ and ψ are* $\star$-equivalent (under $\star$-model semantics)*, written $\varphi \equiv_\star \psi$, iff* $\mathsf{Mod}_\star(\varphi) = \mathsf{Mod}_\star(\psi)$.

In the following proposition, item (i) is proved by Lifschitz, Tang, and Turner (cf., see Lifschitz et al., 1999, (iii) of Prop. 6).

Proposition 1 *Let A, B, C, D be set of atoms. We have the following*

(i) $\bigwedge(A \cup \neg B) \supset \bigvee(D \cup \neg C) \equiv_{\text{HT}} \bigwedge(A \cup \neg B \cup \neg\neg C) \supset \bigvee D$.

(ii) $\bigwedge(A \cup \neg B) \supset \bigvee(D \cup \neg C) \models_{\text{FLP}} \bigwedge(A \cup \neg B \cup \neg\neg C) \supset \bigvee D$.

Please note here that the inverse of (ii) does not generally hold. For instance, $\neg\neg p \supset p \equiv_{\text{FLP}} \top$ while $\langle \emptyset, \{p\}\rangle \not\models_{\text{FLP}} p \vee \neg p$.

Given two formulas φ_1 and φ_2, it is known that φ_1 and φ_2 are strongly HT-equivalent under HT-stable model semantics if and only if they are HT-equivalent, viz. $\varphi_1 \equiv_{\text{HT}} \varphi_2$; φ_1 and φ_2 are strongly FLP-equivalent under FLP-stable model semantics if and only if they are FLP-equivalent, viz. $\varphi_1 \equiv_{\text{FLP}} \varphi_2$ (cf., see Truszczynski, 2010, Thm. 7). It is commonly recognized that strong equivalence amounts to *knowledge equivalence* of formulas. That is, strong $\star$-equivalence captures the logical content of a formula under $\star$-stable model semantics (Osorio & Zacarias, 2004; Osorio & Cuevas, 2007; Delgrande et al., 2013). Now we formally define the knowledge of logic programs.

Definition 5 *The* $\star$-knowledge *of a formula* φ *under* $\star$*-stable model semantics, written* $Cn_\star(\varphi)$*, consists of the logical* $\star$*-consequence of* φ*, viz.* $Cn_\star(\varphi) = \{\psi \mid \varphi \models_\star \psi\}$.

The $\star$-knowledge of a formula stands for the $\star$*-logical content* of the formula. For instance, $Cn_{\text{HT}}(\top) = Cn_{\text{HT}}(p \supset p) \subset Cn_{\text{HT}}(p \supset q)$.

Recall that, under $\star$-model semantics, every formula can be transformed into a conjunction of formulas in the following normal form:

$$\bigwedge(B \cup \neg C) \supset \bigvee(A \cup \neg D) \tag{2}$$

where A, B, C, D are sets of atoms (cf., for $\star = $ HT, see Cabalar & Ferraris, 2007, Thm. 2; Truszczynski, 2010, Thm. 9 for $\star = $ FLP). That is, for every formula φ, there is a conjunction of formulas in the form (2) which is strongly $\star$-equivalent to φ.

A formula of the form (2) is called a *rule*, which is also generally written as

$$a_1; \ldots; a_l; not\, d_1; \ldots; not\, d_n \leftarrow b_1, \ldots, b_k, not\, c_1, \ldots, not\, c_m \tag{3}$$

where $A = \{a_i | 1 \leq i \leq l\}, B = \{b_i | 1 \leq i \leq k\}, C = \{c_i | 1 \leq i \leq m\}$ and $D = \{d_i | 1 \leq i \leq n\}$. A *logic program* is a finite set of rules. Let r be a rule of the form (2). It is said to be

- *disjunctive* if $D = \emptyset$;
- *positive* if $C = D = \emptyset$;
- *normal* if $|A| \leq 1$ and $D = \emptyset$; and
- *Horn* if $|A| \leq 1$ and $C = D = \emptyset$.

A logic program is *disjunctive* (resp. *positive, normal*, and *Horn*) iff it consists of disjunctive (resp. positive, normal, Horn) rules. A logic program is $\star$*-consistent (under* $\star$*-stable model semantics)* if it has at least one $\star$-stable model.

It is known that every logic program has the same HT-models and FLP-models (cf., see Truszczynski, 2010, Prop. 8).

Proposition 2 *Every logic program has the same* HT- *and* FLP-*models.*

3. Characterizations of Knowledge Equivalence

In the section, from the perspective of $\star$-models, we consider the characterization for knowledge equivalence of various logic programs firstly, and relate the knowledge equivalence to the equivalence of propositional logic secondly.

3.1 Model Theoretical Characterization

We firstly recall some basic properties of the $\star$-satisfiability (Ferraris & Lifschitz, 2005; Ferraris, 2005; Truszczynski, 2010).

Proposition 3 *Let φ be a formula and $X \subseteq Y \subseteq \mathcal{A}$.*

(i) If $\langle X, Y\rangle \models_\star \varphi$ then $\langle Y, Y\rangle \models_\star \varphi$ (i.e., $Y \models \varphi$).

(ii) $\langle X, Y\rangle \models_\star \neg\varphi$ iff $Y \models \neg\varphi$.

(iii) $\langle X, Y\rangle \models_\star \varphi$ iff $X \models \mathsf{Red}_\star(\varphi, Y)$.

A collection $\mathcal{M}$ of $\star$-interpretations is $\star$-*expressible* whenever there exists a formula φ such that $\mathsf{Mod}_\star(\varphi) = \mathcal{M}$. A collection $\mathcal{M}$ of $\star$-interpretations may be not $\star$-expressible. For instance, there is no formula whose $\star$-models are the ones in $\mathcal{M} = \{\langle\emptyset, \{p\}\rangle\}$. The reason is that if there is a formula φ such that $\mathsf{Mod}_\star(\varphi) = \mathcal{M}$ then we have $\langle\{p\}, \{p\}\rangle \models_\star \varphi$ by (i) of Proposition 3. This requires $\langle\{p\}, \{p\}\rangle$ belonging to $\mathsf{Mod}_\star(\varphi)$, a contradiction.

Given a formula φ and $X \subset Y \subseteq \mathcal{A}$, $\langle X, Y\rangle$ is a $\star$-*countermodel* of φ if $\langle X, Y\rangle \not\models_\star \varphi$ and $\langle Y, Y\rangle \models_\star \varphi$; $\langle Y, Y\rangle$ is a $\star$-*countermodel* of φ if $\langle Y, Y\rangle \not\models_\star \varphi$. Let $X \subset Y \subseteq \mathcal{A}$, we define the following formulas:

$$\lambda_{\text{HT}}(X, Y) = \bigwedge(X \cup \neg\overline{Y}) \supset \bigvee((Y \setminus X) \cup \neg(Y \setminus X)), \tag{4}$$

$$\lambda_{\text{FLP}}(X, Y) = \bigwedge(X \cup \neg\overline{Y}) \supset \bigvee(\overline{X} \cup \neg Y), \tag{5}$$

$$\lambda(Y, Y) = \bigwedge(Y \cup \neg\overline{Y}) \supset \bot, \tag{6}$$

$$\xi(X, Y) = \bigwedge(X \cup \neg\overline{Y}) \supset \bigvee(Y \setminus X). \tag{7}$$

Here $\lambda_\star(X, Y)$ and $\lambda(Y, Y)$ is to capture the $\star$-countermodel $\langle X, Y\rangle$ and $\langle Y, Y\rangle$ respectively.

The following lemma shows that the $\star$-countermodel can be captured by a formula (cf., for $\star = $ HT, see Cabalar & Ferraris, 2007, Prop. 1; Truszczynski, 2010, Props. 5 and 6 for $\star = $ FLP).

Lemma 1 *Let $X \subset Y \subseteq \mathcal{A}$ and $U \subseteq V \subseteq \mathcal{A}$.*

(i) $\langle U, V\rangle$ is a $\star$-countermodel of $\lambda_\star(X, Y)$ iff $U = X$ and $V = Y$.

(ii) $\langle U, V\rangle$ is a $\star$-countermodel of $\lambda(Y, Y)$ iff $V = Y$.

Proposition 4 *A collection $\mathcal{M}$ of $\star$-interpretations is $\star$-expressible iff*

$$\langle X, Y\rangle \in \mathcal{M} \text{ implies } \langle Y, Y\rangle \in \mathcal{M}. \tag{8}$$

Actually, if $\mathcal{M}$ satisfy condition (8) then the following logic program

$$\Pi_\star = \{\lambda_\star(X, Y) | \langle X, Y\rangle \notin \mathcal{M} \text{ and } \langle Y, Y\rangle \in \mathcal{M}\} \cup \{\lambda(Y, Y) | \langle Y, Y\rangle \notin \mathcal{M}\}$$

captures $\mathcal{M}$ in the sense that $\mathsf{Mod}_\star(\Pi_\star) = \mathcal{M}$.

Note that Wong (2009) presented a model-theoretical characterization for the HT-models of disjunctive logic programs (cf., see Wong, 2009, Thm. 2.7). Formally speaking, a collection $\mathcal{M}$ of HT-interpretations is *disjunctively* HT-*expressible*, i.e., there is a disjunctive logic program Π such that $\mathsf{Mod}_{\mathrm{HT}}(\Pi) = \mathcal{M}$, iff the condition (8) and the following one hold:

$$\text{if } \langle X, Y\rangle \in \mathcal{M}, Y \subseteq Y' \text{ and } \langle Y', Y'\rangle \in \mathcal{M} \text{ then } \langle X, Y'\rangle \in \mathcal{M}. \tag{9}$$

Together with Proposition 2, we have

Corollary 1 *A collection $\mathcal{M}$ of* FLP-*interpretations is disjunctively* FLP-*expressible iff the conditions (8) and (9) hold.*

Actually, if $\mathcal{M}$ satisfies the conditions (8) and (9) then the following disjunctive logic program captures $\mathcal{M}$.

$$\Pi = \{\xi(X, Y) | \langle X, Y\rangle \notin \mathcal{M} \text{ and } \langle Y, Y\rangle \in \mathcal{M}\} \cup \{\lambda(Y, Y) | \langle Y, Y\rangle \notin \mathcal{M}\}.$$

Lemma 2 *Let A, B be two sets of atoms, and $X \subseteq Y \subseteq \mathcal{A}$. $\langle X, Y\rangle \models_\star \bigwedge B \supset \bigvee A$ iff $X \models \bigwedge B \supset \bigvee A$ and $Y \models \bigwedge B \supset \bigvee A$.*

Proposition 5 *A set $\mathcal{M}$ of $\star$-interpretations is* positively $\star$-expressible*, i.e., there is a positive logic program Π s.t $\mathsf{Mod}_\star(\Pi) = \mathcal{M}$, iff $\mathcal{M}$ satisfies the criteria:*

$$\langle X, Y\rangle \in \mathcal{M} \textit{ iff } X \subseteq Y, \langle X, X\rangle \in \mathcal{M} \textit{ and } \langle Y, Y\rangle \in \mathcal{M}. \tag{10}$$

As a matter of fact, in the case $\mathcal{M}$ satisfies the condition (10), the positive logic program $\Pi = \{\bigwedge X \supset \bigvee \overline{X} | \langle X, X\rangle \notin \mathcal{M}\}$ captures $\mathcal{M}$.

Corollary 2 *Two positive logic programs are strongly $\star$-equivalent if and only if they are equivalent in propositional logic.*

Eiter, Fink, Tompits, and Woltran (2004) have showed that a disjunctive logic program Π is strongly equivalent to a normal logic program if and only if Π is closed under here-intersection, i.e., for every pair of HT-models $\langle X, Y\rangle$ and $\langle X', Y\rangle$ of Π, $\langle X \cap X', Y\rangle$ is also an HT-model of Π (cf., see Eiter et al., 2004, Thms. 1 and 2). In terms of the characterization of disjunctive logic programs and Proposition 2, we obtain a $\star$-model characterization for normal logic programs as follows.

Corollary 3 *A set $\mathcal{M}$ of $\star$-interpretations is* normally $\star$-expressible*, i.e., there is a normal logic program Π such that $\mathsf{Mod}_\star(\Pi) = \mathcal{M}$, iff $\mathcal{M}$ satisfies, in addition to (8) and (9), the following criteria:*

$$\textit{if } \langle X, Y\rangle \in \mathcal{M} \textit{ and } \langle X', Y\rangle \in \mathcal{M} \textit{ then } \langle X \cap X', Y\rangle \in \mathcal{M}. \tag{11}$$

Proposition 6 *A collection $\mathcal{M}$ of $\star$-interpretations is* Horn $\star$-expressible*, i.e., there is a Horn logic program Π such that $\mathsf{Mod}_\star(\Pi) = \mathcal{M}$, iff $\mathcal{M}$ satisfies, in addition to (10), the following criteria:*

$$\langle X, Y\rangle \in \mathcal{M} \textit{ and } \langle H, T\rangle \in \mathcal{M} \Rightarrow \langle X \cap H, Y \cap T\rangle \in \mathcal{M}. \tag{12}$$

3.2 Relating Knowledge Equivalence to Propositional Logic

It is proved that strong equivalence of logic programs under stable model semantics can be related to the equivalence in propositional logic (Pearce et al., 2001; Lin, 2002). This holds for the strong FLP-equivalence of logic programs as we will show in the following.

Firstly, we extend the language $\mathcal{L}_\mathcal{A}$ to $\mathcal{L}_{\mathcal{A}\cup\mathcal{A}'}$ where $\mathcal{A}' = \{p' | p \in \mathcal{A}\}$ and p's are fresh atoms. For each expression α of $\mathcal{L}_\mathcal{A}$, by α' we denote the result obtained from α by replacing each atom p from $\mathcal{A}$ by the corresponding atom p' in $\mathcal{A}'$. In the following we denote

$$\Delta(\mathcal{A}) = \{p \supset p' \mid p \in \mathcal{A}\}. \tag{13}$$

Please note that, for each model M of $\Delta(\mathcal{A})$, M has a splitting $M_\mathcal{A}$ and $M_{\mathcal{A}'}$ where $M_\mathcal{A} = M \cap \mathcal{A}$ and $M_{\mathcal{A}'} = M \cap \mathcal{A}'$ and, for every $p \in M_\mathcal{A}$, the atom p' of $\mathcal{A}'$ belongs to $M_{\mathcal{A}'}$. For $M \subseteq \mathcal{A}'$ we denote by M^* the set $\{p \in \mathcal{A} | p' \in M\}$.

Definition 6 *$\tau_{\mathrm{HT}}[.]$ and $\tau_{\mathrm{FLP}}[.]$ are recursively defined as follows:*

(T1) $\tau_\star[\bot] = \bot$;

(T2) $\tau_\star[p] = p$;

(T3) $\tau_\star[\varphi_1 \circ \varphi_2] = \tau_\star[\varphi_1] \circ \tau_\star[\varphi_2]$ *where* $\circ \in \{\wedge, \vee\}$;

(HT-*T4)* $\tau_{\mathrm{HT}}[\varphi_1 \supset \varphi_2] = (\varphi_1' \supset \varphi_2') \wedge (\tau_{\mathrm{HT}}[\varphi_1] \supset \tau_{\mathrm{HT}}[\varphi_2])$;

(FLP-*T4)* $\tau_{\mathrm{FLP}}[\varphi_1 \supset \varphi_2] = (\varphi_1' \supset \varphi_2') \wedge (\varphi_1 \wedge \varphi_1' \supset \tau_{\mathrm{FLP}}[\varphi_2])$.

Please note that the translation τ_{HT} is same to the translation τ defined by Pearce, Tompits, and Woltran (2001). One can verify that $\tau_{\mathrm{HT}}[\neg\varphi] = \neg\varphi' \wedge \neg\tau_{\mathrm{HT}}[\varphi]$, while $\tau_{\mathrm{FLP}}[\neg\varphi] = \neg\varphi'$. Given a theory Σ of $\mathcal{L}_\mathcal{A}$, we define $\tau_\star[\Sigma] = \{\tau_\star[\varphi] \mid \varphi \in \Sigma\}$. It is evident that $\tau_\star[\Sigma]$ is in linear size of Σ.

Example 3 Let $\varphi = p \vee \neg p \supset p$. We have that

$$\tau_{\mathrm{HT}}[\varphi] = ((p' \vee \neg p') \supset p') \wedge ((p \vee \neg p \wedge \neg p') \supset p) \equiv p',$$
$$\tau_{\mathrm{FLP}}[\varphi] = ((p' \vee \neg p') \supset p') \wedge ((p \vee \neg p) \wedge (p' \vee \neg p') \supset p) \equiv p' \wedge p.$$

The unique FLP-model (over the signature $\{p\}$) of φ is $\langle\{p\}, \{p\}\rangle$. However, φ has two HT-models $\langle\emptyset, \{p\}\rangle$ and $\langle\{p\}, \{p\}\rangle$. Over the signature $\{p, p'\}$, one can easily check that $\{\tau_{\mathrm{HT}}[\varphi]\} \cup \Delta(\mathcal{A})$ has two models $\{p, p'\}$ and $\{p'\}$, while $\{\tau_{\mathrm{FLP}}[\varphi]\} \cup \Delta(\mathcal{A})$ has a unique model $\{p, p'\}$.

Proposition 7 *Let $\varphi = \bigwedge(B \cup \neg C) \supset \bigvee(A \cup \neg D)$, where A, B, C, D are subsets of $\mathcal{A}$. Then we have $\Delta(\mathcal{A}) \models \tau_{\mathrm{FLP}}[\varphi] \leftrightarrow \tau_{\mathrm{HT}}[\varphi]$.*

The following proposition connects the $\star$-equivalence with the equivalence in classical propositional logic (cf., for $\star = \mathrm{HT}$, see Pearce et al., 2001, Lem. 2).

Proposition 8 *Let φ be a formula of $\mathcal{L}_\mathcal{A}$ and $X \subseteq Y \subseteq \mathcal{A}$. Then $\langle X, Y\rangle$ is a $\star$-model of φ iff $X \cup Y'$ is a model of $\Delta(\mathcal{A}) \cup \{\tau_\star[\varphi]\}$.*

The following theorem shows that the strong $\star$-equivalence of logic programs under $\star$-stable model semantics can be reduced to the equivalence in propositional logic (cf., for $\star = \text{HT}$, see Ferraris et al., 2011, Thm. 9; or Lin & Zhou, 2011, (5) of Thm. 6).

Theorem 4 *Two formulas φ and ψ have the same $\star$-models (over $\mathcal{A}$) iff $\Delta(\mathcal{A}) \cup \{\tau_\star[\varphi]\}$ and $\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}$ have the same models (over $\mathcal{A} \cup \mathcal{A}'$).*

Based on the theorem, we obtain the following complexity result (cf., for $\star = \text{HT}$, see Pearce, Tompits, & Woltran, 2009, Thms. 8 and 11).

Proposition 9 *(i) The problem of deciding if a formula is $\star$-satisfiable is* NP-*complete.*
(ii) The problem of deciding if two formulas are $\star$-equivalent is co-NP-*complete.*

4. Knowledge Forgetting in Logic Programs

As mentioned in the introduction, we concentrate on the knowledge forgetting of logic programs under stable model semantics. It is formally stated as following:

Definition 7 (Knowledge forgetting) *Let Π be a logic program and $V \subseteq \mathcal{A}$. A logic program Σ is a* result of $\star$-knowledge forgetting V from Π, *if and only if Σ consists of the $\star$-knowledge of Π that mentions no atom from V.*

We will show that such a knowledge forgetting result always exists and it is unique up to strong equivalence (cf. Theorem 6) after a semantic $\star$-forgetting is defined and explored.

Let V, X, Y be sets of atoms. The set Y is *V-bisimilar* to X, written $Y \sim_V X$, if $Y \setminus V = X \setminus V$. It intuitively states that the interpretations X and Y agree with each other on those atoms not in V. Two $\star$-interpretations $\langle H, T\rangle$ and $\langle X, Y\rangle$ are *V-bisimilar*, written $\langle H, T\rangle \sim_V \langle X, Y\rangle$, if $H \sim_V X$ and $T \sim_V Y$. Now, we are in the position to define the semantic knowledge forgetting in terms of bisimulation.

Definition 8 (Semantic knowledge forgetting) *Let φ be a formula and $V \subseteq \mathcal{A}$. A formula ψ is a* result of (semantic) $\star$-forgetting V from φ *whenever, for every $\star$-interpretation M,*

$$M \in \mathsf{Mod}_\star(\psi) \textit{ iff } \exists M' \in \mathsf{Mod}_\star(\varphi) \; s.t \; M \sim_V M'. \tag{14}$$

According the definition, one can see that the $\star$-models of ψ can somehow exactly constructed from those of ψ. This motivates us to define the following notion of extension.

Let V, X, Y be sets of atoms. The *V-extension* of X, denoted by $X_{\dagger V}$, is the collection of interpretations that are V-bisimilar to X. The *V-extension* of a $\star$-interpretation $\langle H, T\rangle$, denoted by $\langle H, T\rangle_{\dagger V}$, is the collection of $\star$-interpretations that are V-similar to $\langle H, T\rangle$. For instance, let $\langle H, T\rangle = \langle\{p, q\}, \{p, q\}\rangle$ and $V = \{q, r\}$. Then $\langle H, T\rangle_{\dagger V}$ contains $\langle\{p\}, \{p\}\rangle$, $\langle\{p\}, \{p, q\}\rangle$, $\langle\{p\}, \{p, q, r\}\rangle$, $\langle\{p, q, r\}, \{p, q, r\}\rangle$ and so on. Intuitively speaking, the V-extension of an interpretation M is the collection of interpretations formed from M by freely adding or removing some atoms in V. The *V-extension* of a collection $\mathcal{M}$ of ($\star$-)interpretations, written $\mathcal{M}_{\dagger V}$, is the collection $\bigcup_{\beta \in \mathcal{M}} \beta_{\dagger V}$.

In classical propositional logic if $\mathcal{M}$ corresponds to a formula φ, i.e. $\mathcal{M} = \mathsf{Mod}(\varphi)$, then $\mathcal{M}_{\dagger V}$ corresponds to a formula whose truth value has nothing to do with the atoms in V. The intended meaning in the case of $\star$-models is similar when $\mathcal{M}_{\dagger V}$ corresponds to a formula under $\star$-model semantics that is relevant to only the atoms not in V. In other words, suppose $\mathcal{M}_{\dagger V} = \mathsf{Mod}_\star(\varphi)$. If $\langle X, Y\rangle \models_\star \varphi$ then $\langle H, T\rangle \models_\star \varphi$ where H (resp. T) is obtained from X (resp. Y) by freely adding or removing any atoms in V whenever $H \subseteq T$. The following lemma shows an equivalent condition for the semantic $\star$-knowledge forgetting.

Lemma 3 *Let φ be a formula and $V \subseteq \mathcal{A}$. A formula ψ is a* result of $\star$-forgetting V from φ, *iff the following condition holds:*

$$\mathsf{Mod}_\star(\psi) = \mathsf{Mod}_\star(\varphi)_{\dagger V}. \tag{15}$$

This condition of $\star$-forgetting is a generalization of the forgetting in propositional logic (Lin & Reiter, 1994) in terms of the following corollary.

Corollary 5 *A formula ψ is a result of forgetting a set V of atoms in a formula φ iff* $\mathsf{Mod}(\psi) = \mathsf{Mod}(\varphi)_{\dagger V}$*, where* $\mathsf{Mod}(.)$ *refers to classical propositional logic.*

A syntactic counterpart of the forgetting in propositional logic is defined as follows (Lin, 2001; Lang et al., 2003):

$$\mathsf{Forget}(\varphi, \{p\}) = \varphi[p/\bot] \vee \varphi[p/\top],$$

$$\mathsf{Forget}(\varphi, V \cup \{p\}) = \mathsf{Forget}(\mathsf{Forget}(\varphi, \{p\}), V)$$

where $\varphi[p/\top]$ (resp. $\varphi[p/\bot]$) is the formula obtained from φ by replacing every occurrence of the atom p with $\top$ (resp. $\bot$).

As $\star$-interpretations are related to the given signature $\mathcal{A}$, in what follows, we shall assume that the signature of a formula/theory is implicitly given by the atoms occurring in the formula/theory, unless explicitly stated otherwise. The example below illustrates how $\star$-forgetting results can be computed.

Example 4 Let φ be the following formula

$$(p \supset q) \wedge (q \supset p) \wedge (\neg p \supset \bot) \wedge (\neg q \supset \bot).$$

Over the signature $\{p, q\}$, we have $\mathsf{Mod}_\star(\varphi) = \{\langle \emptyset, \{p, q\}\rangle, \langle \{p, q\}, \{p, q\}\rangle\}$. Please note here that $\star$ can be either HT or FLP. Then from Definition 8, we can verify that $\mathsf{Mod}_\star(\varphi)_{\dagger\{p\}} = \{\langle \emptyset, \{q\}\rangle, \langle \{q\}, \{q\}\rangle\}_{\dagger\{p\}}$. It corresponds to the formula $\psi = (p \wedge \neg q \supset \bot) \wedge (\neg p \wedge \neg q) \supset \bot$ under the $\star$-model semantics by Proposition 4. As a matter of fact, we have $\psi \equiv_\star \neg q \supset \bot \equiv_\star \neg\neg q$.

Note that $\mathsf{Forget}(\varphi, \{p\}) = \varphi[p/\top] \vee \varphi[p/\bot] \equiv q$ and $\neg\neg q \not\equiv_\star q$. It shows that, unlike the syntactic counterpart of the forgetting in classical propositional logic, the $\star$-forgetting results cannot be computed via $\varphi[p/\top] \vee \varphi[p/\bot]$ as $\mathsf{Mod}_\star(\neg\neg q) = \{\langle \emptyset, \{q\}\rangle, \langle \{q\}, \{q\}\rangle\}$, while $\mathsf{Mod}_\star(q) = \{\langle \{q\}, \{q\}\rangle\}$ (over the signature $\{q\}$). □

4.1 Expressibility

Please note that Definition 8 does not guarantee the existence of the forgetting results, however the next theorem shows that the $\star$-forgetting result always exists. It also implies that the $\star$-forgetting result is unique (up to strong $\star$-equivalence).

Theorem 6 (Expressibility theorem) *Let φ be a formula and V a set of atoms. There exists a formula ψ such that* $\mathsf{Mod}_\star(\psi) = \mathsf{Mod}_\star(\varphi)_{\dagger V}$.

Here, the uniqueness up to strong $\star$-equivalence of the $\star$-forgetting result follows from the fact that, if a formula ψ' is a result $\star$-forgetting V from φ as well then $\mathsf{Mod}_\star(\psi') = \mathsf{Mod}_\star(\varphi)_{\dagger V} = \mathsf{Mod}_\star(\psi)$, which shows that ψ and ψ' are strongly $\star$-equivalent under the $\star$-stable model semantics.

Based on the expressibility result and by abusing the denotation, we denote the forgetting result by $\mathsf{Forget}_\star(\varphi, V)$:

Definition 9 *Let φ be a formula and $V \subseteq \mathcal{A}$.* $\mathsf{Forget}_\star(\varphi, V)$ *is a formula ψ s.t* $\mathsf{Mod}_\star(\psi) = \mathsf{Mod}_\star(\varphi)_{\dagger V}$, *i.e.,* $\mathsf{Forget}_\star(\varphi, V)$ *is a result of $\star$-forgetting V from φ.*

In this sense $\mathsf{Forget}_\star$ is an operator which maps a formula and a set of atoms to a formula. According to Definition 8 and the expressibility theorem, the following corollary easily follows.

Corollary 7 *Let ψ, φ be formulas, V, V_1 and V_2 be sets of atoms.*

(i) $\mathsf{Forget}_\star(\mathsf{Forget}_\star(\varphi, V_1), V_2) \equiv_\star \mathsf{Forget}_\star(\mathsf{Forget}_\star(\varphi, V_2), V_1)$.

(ii) If $\psi \equiv_\star \varphi$ then $\mathsf{Forget}_\star(\psi, V) \equiv_\star \mathsf{Forget}_\star(\varphi, V)$.

It firstly states that $\star$-forgetting is independent of the order of forgotten atoms, and secondly, the $\star$-forgetting preserves strong $\star$-equivalence of logic programs under $\star$-stable model semantics.

To further investigate the properties of the forgetting, we introduce a notion of irrelevance under $\star$-model semantics.

Definition 10 *A formula ψ is $\star$-*irrelevant *to a set V of atoms, denoted as* $\mathsf{IR}_\star(\psi, V)$, *if there exists a formula ϕ mentioning no atoms from V and $\psi \equiv_\star \phi$.*

Some basic properties on $\star$-forgetting are presented below.

Proposition 10 *Let ψ and φ be two formulas and V a set of atoms.*

(i) $\mathsf{IR}_\star(\mathsf{Forget}_\star(\psi, V), V)$.

(ii) ψ has a $\star$-model iff $\mathsf{Forget}_\star(\psi, V)$ *has.*

(iii) $\psi \models_\star \mathsf{Forget}_\star(\psi, V)$.

(iv) If $\psi \models_\star \varphi$ then $\mathsf{Forget}_\star(\psi, V) \models_\star \mathsf{Forget}_\star(\varphi, V)$.

(v) $\mathsf{Forget}_\star(\psi \vee \varphi, V) \equiv_\star \mathsf{Forget}_\star(\psi, V) \vee \mathsf{Forget}_\star(\varphi, V)$.

(vi) $\mathsf{Forget}_\star(\psi \wedge \varphi, V) \models_\star \mathsf{Forget}_\star(\psi, V) \wedge \mathsf{Forget}_\star(\varphi, V)$.

(vii) $\mathsf{Forget}_\star(\psi \wedge \varphi, V) \equiv_\star \mathsf{Forget}_\star(\psi, V) \wedge \varphi$ *if* $\mathsf{IR}_\star(\varphi, V)$.

Intuitively, (i) of the Proposition says that the $\star$-forgetting result is irrelevant to atoms in V, i.e., those forgotten atoms. In this sense, the signature of $\star$-forgetting result can be constrained to $\mathcal{A} \setminus V$. The intended meaning of the others can be easily read out. E.g., item (iii) says that this forgetting is a kind of weakening, while item (v) shows that the forgetting has a distributive property for disjunction.

As mentioned earlier, disjunctive programs, positive programs, normal logic programs and Horn programs are four types of special cases of (arbitrary) logic programs under our setting. Then it is interesting to consider whether the expressibility result also holds for each of these special programs. For instance, we would like to know whether the result of $\star$-forgetting in a disjunctive (positive, normal, and Horn) logic program is still expressible by a disjunctive (resp. positive, normal, and Horn) logic program.

As indicated by the following two examples, HT- and FLP-forgetting in disjunctive, positive and normal logic programs is possibly not expressible in either disjunctive or positive logic programs. For simplicity, we identify a singleton set $\{\alpha\}$ as α when it is clear from its context, and thus we denote $\mathsf{Forget}_\star(\psi, \{p\})$ as $\mathsf{Forget}_\star(\psi, p)$, and $\mathsf{IR}_\star(\psi, \{p\})$ as $\mathsf{IR}_\star(\psi, p)$, and $\mathcal{M}_{\dagger\{p\}}$ as $\mathcal{M}_{\dagger p}$ etc..

Example 5 Consider the following normal logic program Π over signature $\{p, q\}$:

$$(\neg p \supset q) \wedge (\neg q \supset p) \wedge (p \wedge q \supset \bot).$$

We have that $\mathsf{Mod}_\star(\Pi) = \{\langle\{p\}, \{p\}\rangle, \langle\{q\}, \{q\}\rangle\}$ and

$$\mathsf{Mod}_\star(\Pi)_{\dagger p} = \{\langle\emptyset, \emptyset\rangle, \langle\{q\}, \{q\}\rangle\}_{\dagger\{p\}}.$$

Here $\langle\{p\}, \{p\}\rangle_{\dagger\{p\}} = \langle\emptyset, \emptyset\rangle_{\dagger\{p\}}$. It implies that $\mathsf{Forget}_\star(\Pi, p) \equiv_\star q \vee \neg q$. It can be easily seen that $q \vee \neg q$ cannot be expressed as a disjunctive logic program because $\mathsf{Mod}_\star(\Pi)_{\dagger p}$ does not satisfy (9). Hence $\mathsf{Forget}_\star(\Pi, p)$ cannot be expressed by a normal logic program.

Please note that $\neg\neg q \supset q \equiv_{\text{HT}} q \vee \neg q$. Thus $\neg\neg q \supset q$ is also a result of HT-forgetting p from Π. However, $\neg\neg q \supset q$ is not a result of FLP-forgetting p from Π as $\neg\neg q \supset q \equiv_{\text{FLP}} \top \not\equiv_{\text{FLP}} q \vee \neg q$. □

Example 6 Let Π be a positive logic program over signature $\{p, q, r\}$ as follows:

$$(p \vee q \vee r) \wedge (p \wedge q \supset r) \wedge (p \wedge r \supset q) \wedge (q \wedge r \supset p).$$

It is not difficult to verify that, over the signature $\{p, r\}$, $\mathsf{Mod}_\star(\Pi)_{\dagger\{q\}}$ consists of

$$\langle\emptyset, \emptyset\rangle, \langle\emptyset, \{p, r\}\rangle, \langle\{p\}, \{p\}\rangle, \langle\{p\}, \{p, r\}\rangle, \langle\{r\}, \{r\}\rangle, \langle\{r\}, \{p, r\}\rangle, \langle\{p, r\}, \{p, r\}\rangle.$$

Clearly it does not satisfy the condition (9). Hence it can not captured by a disjunctive logic program. As a matter of fact, we have the following

$$\mathsf{Forget}_{\text{HT}}(\Pi, q) \equiv_{\text{HT}} \lambda_{\text{HT}}(\emptyset, \{p\}) \wedge \lambda_{\text{HT}}(\emptyset, \{r\}) = (\neg r \supset p \vee \neg p) \wedge (\neg p \supset r \vee \neg r),$$
$$\mathsf{Forget}_{\text{FLP}}(\Pi, q) \equiv_{\text{FLP}} \lambda_{\text{FLP}}(\emptyset, \{p\}) \wedge \lambda_{\text{FLP}}(\emptyset, \{r\}) = (\neg r \supset p \vee r \vee \neg p) \wedge (\neg p \supset p \vee r \vee \neg r)$$

in terms of Proposition 4. Interestingly, this example also shows that, though a logic program may have the same HT-models as FLP-models, its HT-forgetting result may be different from its FLP-forgetting result. □

The HT- and FLP-forgetting in Horn logic programs is of special interest, because unlike disjunctive, positive and normal logic programs, the result of HT- and FLP-forgetting result in a Horn logic program is always expressible by a Horn logic program, as we show below.

Theorem 8 (Horn expressibility) *Let* Π *be a Horn logic program and* $V \subseteq \mathcal{A}$. *There is a Horn logic program* Π' *such that* $\mathsf{Forget}_\star(\Pi, V) \equiv_\star \Pi'$.

Having obtained the model-theoretical characterization of the classes of disjunctive and normal logic programs respectively, we can easily derive a sufficient and necessary condition for HT- and FLP-forgetting results to remain in the same class, i.e., the result of HT- and FLP-forgetting a set of atoms in a disjunctive (resp. normal) logic program is a disjunctive (resp. normal) logic program.

Proposition 11 *Let* Π *be a disjunctive logic program,* $V \subseteq \mathcal{A}$. *We have that* $\mathsf{Forget}_\star(\Pi, V)$ *is expressible in disjunctive logic programs if and only if,*

$$\langle H_1, T_1\rangle \models_\star \Pi, \langle T_2, T_2\rangle \models_\star \Pi \textit{ and } T_1 \subseteq T_2 \Rightarrow \exists \langle H_3, T_3\rangle \models_\star \Pi \textit{ such that } \langle H_3, T_3\rangle \sim_V \langle H_1, T_2\rangle.$$

Proposition 12 *Let* Π *be a normal logic program,* $V \subseteq \mathcal{A}$. *Then* $\mathsf{Forget}_\star(\Pi, V)$ *is expressible in normal logic programs if and only if, in addition to condition (16), the following condition holds,*

$$\begin{aligned} &\langle H_1, T_1\rangle \models_\star \Pi, \langle H_2, T_2\rangle \models_\star \Pi \textit{ and } T_1 \sim_V T_2 \\ \Rightarrow\ &\exists \langle H_3, T_3\rangle \models_\star \Pi \textit{ such that } H_3 \sim_V H_1 \cap H_2 \textit{ and } (T_3 \sim_V T_1 \textit{ or } T_3 \sim_V T_2). \end{aligned} \qquad (16)$$

4.2 Forgetting Postulates

Zhang and Zhou (2009) proposed four forgetting postulates in their work of knowledge forgetting, and showed that their knowledge forgetting can be precisely characterized by the four postulates. They further argued that these postulates should be viewed as a general semantic characterization for knowledge forgetting in other logics. Indeed, the classical propositional forgetting can be also characterized by these postulates. In terms of forgetting in logic programs, as we addressed in the introduction, imposing these postulates is not feasible for existing approaches. In the following, we show that $\star$-forgetting is exactly captured by these postulates, which we think is one major advantage over other logic program forgetting approaches.

The notion of forgetting is closely related to that of uniform interpolation property (Visser, 1996; Goranko & Otto, 2007), for instance, the forgetting in description logics (Lutz & Wolter, 2011) and the semantic forgetting in logic programs (Gabbay, Pearce, & Valverde, 2011). The following corollary follows from Theorem 6, which actually implies the *uniform interpolation property* of the logics under $\star$-model semantics. Namely, for any formulas φ and ψ with $\psi \models_\star \varphi$, there exists a formula ξ such that $\psi \models_\star \xi$, $\xi \models_\star \varphi$ and ξ contains only the atoms occurring in both ψ and φ. The formula ξ is called a *uniform interpolant* of ψ and φ. This is stated as:

Corollary 9 *Let* ψ *and* φ *be two formulas,* V *a set of atoms and* $\mathsf{IR}_\star(\varphi, V)$.

$$\psi \models_\star \varphi \qquad \textit{iff} \qquad \mathsf{Forget}_\star(\psi, V) \models_\star \varphi.$$

Let ψ and φ be two formulas and V a set of atoms. The following are Zhang-Zhou's four postulates for logic programs under $\star$-model semantics.

(W) Weakening: $\psi \models_\star \varphi$.

(PP) Positive persistence: if $\mathsf{IR}_\star(\xi, V)$ and $\psi \models_\star \xi$ then $\varphi \models_\star \xi$.

(NP) Negative persistence: if $\mathsf{IR}_\star(\xi, V)$ and $\psi \not\models_\star \xi$ then $\varphi \not\models_\star \xi$.

(IR) Irrelevance: $\mathsf{IR}_\star(\varphi, V)$.

By specifying $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$, (W), (PP), (NP) and (IR) are called *postulates for knowledge forgetting* in logic programs under $\star$-stable model semantics. Viz, φ is a result of $\star$-forgetting V in ψ. Based on the uniform interpolation property (cf. Corollary 9), we can show the following representation theorem.

Theorem 10 (Representation theorem) *Let ψ and φ be two formulas and V a set of atoms. Then the following statements are equivalent:*

(i) $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$.

(ii) $\varphi \equiv_\star \{\varphi' \mid \psi \models_\star \varphi'$ *and* $\mathsf{IR}_\star(\varphi', V)\}$.

(iii) Postulates (W), (PP), (NP) and (IR) hold.

This theorem justifies that the knowledge forgetting (cf. Definition 7) exists and is unique up to strong equivalence.

An obvious consequence follows from the representation theorem is that

$$\mathsf{Forget}_\star(\varphi, V) \equiv_\star \{\psi \mid \varphi \models_\star \psi \text{ and } \mathsf{IR}_\star(\psi, V)\}.$$

It says that the result of $\star$-forgetting V from φ consists of the $\star$-logical consequence of φ that is $\star$-irrelevant to V. For this reason the forgetting is a knowledge forgetting of logic programs under stable models semantics. As we have mentioned in the introduction that none of the other forgetting approaches in logic programs is a knowledge forgetting since it does not satisfy some of the postulates (see Section 5 for details).

One should note that the representation theorem is applicable for the forgetting in classical propositional logic, viz, $\mathsf{Forget}(\varphi, V) \equiv \{\psi \mid \varphi \models \psi \text{ and } \mathsf{IR}(\psi, V)\}$.

4.3 Relating to Propositional Forgetting

It has been shown that strong equivalence of logic programs may be related to the equivalence of propositional logic (Pearce et al., 2001; Lin, 2002). As the $\star$-forgetting preserves strong equivalence of logic programs under $\star$-stable model semantics, it is worth exploring further connections between $\star$-forgetting and the forgetting in propositional logic. In this section, we undertake an in-depth investigation on this aspect.

We first provide a direct connection between $\star$-forgetting and propositional forgetting via the following proposition.

Proposition 13 *Let φ, φ', ψ be formulas and $V \subseteq \mathcal{A}$ such that $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$ and $\varphi' \equiv \mathsf{Forget}(\psi, V)$. Then*

(i) $\varphi \equiv \varphi'$.

(ii) $\varphi' \models_\star \varphi$.

The result (i) in Proposition 13 simply says that the result of $\star$-forgetting and classical propositional forgetting are equivalent in classical propositional logic. Thus the forgetting in classic propositional logic can be computed by a $\star$-forgetting in logic programs. However as we have seen in Example 4, $\mathsf{Forget}_\star(\psi, V)$ is possibly not $\star$-equivalent to $\mathsf{Forget}(\psi, V)$. The reverse of (ii) does not hold generally. For instance, $\mathsf{Forget}_\star(\neg\neg p, q) \equiv_\star \neg\neg p$, while $\mathsf{Forget}(\neg\neg p, q) \equiv p$, and evidently $\neg\neg p \not\models_\star p$. From this result and Theorem 8, we immediately have the following corollary.

Corollary 11 *Let Π be a Horn logic program and V a set of atoms. Then* $\mathsf{Forget}(\Pi, V)$ *is expressible by a Horn logic program.*

The following result states that, for Horn logic programs, $\star$-forgetting and the forgetting of propositional logic are strongly $\star$-equivalent. Thus it provides a method of computing $\star$-forgetting results of Horn logic programs through the propositional forgetting.

Proposition 14 *Let Π and Π' be two Horn logic programs, and V a set of atoms such that $\Pi' \equiv$* $\mathsf{Forget}(\Pi, V)$*. Then $\Pi' \equiv_\star$* $\mathsf{Forget}_\star(\Pi, V)$.

The following proposition states that the $\star$-forgetting of double negative formulas is closely connected with the classical propositional forgetting, which will be used to prove some complexity results later.

Proposition 15 *Let ψ and φ be two formulas and V a set of atoms.*

(i) $\varphi \equiv \mathsf{Forget}(\psi, V)$ *iff* $\neg\neg\varphi \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$.

(ii) $\mathsf{Forget}(\varphi, V) \equiv \mathsf{Forget}(\psi, V)$ *iff* $\mathsf{Forget}_\star(\neg\neg\varphi, V) \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$.

As it is known that the strong equivalence of logic programs is closed related to the equivalence in propositional logic by translating logic programs into propositional theories (Pearce et al., 2001; Lin, 2002). This motivates us to investigate the connection between the forgettings in the view of the translations. Now our main result of this section is stated as follows.

Theorem 12 ($\star$-forgetting vs propositional forgetting) *Let ψ and φ be two formulas of $\mathcal{L}_\mathcal{A}$ and $V \subseteq \mathcal{A}$. Then*

$$\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V) \textit{ iff } \Delta(\mathcal{A}) \models \tau_\star[\varphi] \leftrightarrow \mathsf{Forget}(\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}, V \cup V').$$

By Theorem 12, we know that to check whether a formula φ is a result of $\star$-forgetting a set V of atoms from a formula ψ, it is equivalent to check whether $\tau_\star[\varphi]$ is classically equivalent to $\mathsf{Forget}(\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}, V \cup V')$ under the theory $\Delta(\mathcal{A})$. The following example shows an application of this theorem.

Example 7 [Example 5 continued] Recall that Π is the following formula:

$$(\neg p \supset q) \wedge (\neg q \supset p) \wedge (p \wedge q \supset \bot)$$

and $\mathsf{Forget}_\star(\Pi, p) \equiv_\star q \vee \neg q$. Over the signature $\{p, q\}$, $\Delta(\mathcal{A}) = (p \supset p') \wedge (q \supset q')$ and, the program translation yields:

$$\tau_\star(\Pi) \equiv (\neg p' \supset q) \wedge (\neg p' \supset q') \wedge (\neg q' \supset p) \wedge (\neg q' \supset p') \wedge (\neg p' \vee \neg q').$$

Now we have that $\mathsf{Forget}(\tau_\star[\Pi] \cup \Delta(\mathcal{A}), \{p, p'\})$ is equivalent to:

$$(\neg q \wedge \neg q') \vee (q \wedge q'), \quad \text{i.e.} \quad (q' \supset q) \wedge (q \supset q')$$

which is equivalent to $\neg q' \vee q$ under the theory $\Delta(\{q\}) = \{q \supset q'\}$. One can further check that $\tau_\star[\neg q \vee q] = \neg q' \wedge \neg q \vee q \equiv \neg q' \vee q$ (under the theory $\Delta(\{q\})$). Thus the formula $\neg q \vee q$ is a result of $\star$-forgetting p from Π by Theorem 12. $\Box$

The following example further shows that $\Delta(\mathcal{A})$ occurring in $\mathsf{Forget}_\star(\{\tau[\psi]\} \cup \Delta(\mathcal{A}), V \cup V')$ is necessary for Theorem 12.

Example 8 [Continued from Example 6] Recall that $\mathcal{A} = \{p, q, r\}$, $\Delta(\mathcal{A}) = \{p \supset p', q \supset q', r \supset r'\}$ and Π consists of

$$(p \vee q \vee r) \wedge (p \wedge q \supset r) \wedge (p \wedge r \supset q) \wedge (q \wedge r \supset p).$$

We have that,

$$\tau_{\text{HT}}[\Pi] \equiv \Pi \wedge \Sigma,$$
$$\Delta(\mathcal{A}) \models \tau_\star[\Pi] \leftrightarrow \Pi \wedge \Sigma,$$
$$\tau_{\text{FLP}}[\Pi] \equiv (p \vee q \vee r) \wedge (p \wedge q \wedge p' \wedge q' \supset r) \wedge (p \wedge r \wedge p' \wedge r' \supset q) \wedge (q \wedge r \wedge q' \wedge r' \supset p) \wedge \Sigma$$

where $\Sigma = (p' \wedge q' \supset r') \wedge (p' \wedge r' \supset q') \wedge (q' \wedge r' \supset p')$.

One can check that

$$\mathsf{Forget}(\tau_{\text{HT}}[\Pi], \{q, q'\}) \equiv \top,$$
$$\Delta(\mathcal{A}) \models \mathsf{Forget}(\tau_{\text{FLP}}[\Pi], \{q, q'\}) \leftrightarrow \top.$$

Recall that the formula $\varphi_1 = (\neg r \supset p \vee \neg p) \wedge (\neg p \supset r \vee \neg r)$ is a result of HT-forgetting q from Π; and $\varphi_2 = (\neg r \supset p \vee r \vee \neg p) \wedge (\neg p \supset p \vee r \vee \neg r)$ is a result of FLP-forgetting q from Π. We have that

$$\tau_{\text{HT}}[\varphi_1] \equiv \varphi_1' \wedge (\neg r \wedge \neg r' \supset p \vee \neg p \wedge \neg p') \wedge (\neg p \wedge \neg p' \supset r \vee \neg r \wedge \neg r'),$$
$$\tau_{\text{FLP}}[\varphi_2] \equiv \varphi_2' \wedge (\neg r \wedge \neg r' \supset p \vee r \vee \neg p') \wedge (\neg p \wedge \neg p' \supset p \vee r \vee \neg r').$$

Under the theory $\Delta(\mathcal{A})$, we have

$$\Delta(\mathcal{A}) \models \tau_{\text{HT}}[\varphi_1] \leftrightarrow (p' \supset p \vee r') \wedge (r' \supset r \vee p'),$$
$$\Delta(\mathcal{A}) \models \tau_{\text{HT}}[\varphi_1] \leftrightarrow (p' \supset p \vee r') \wedge (r' \supset r \vee p').$$

One can verify further that the model $\{p'\}$ of $\Delta(\mathcal{A})$ is not a model of $\tau_{\text{HT}}[\varphi_1]$, nor it is a model of $\tau_{\text{FLP}}[\varphi_2]$, i.e. $\Delta(\mathcal{A}) \not\models \tau_{\text{HT}}[\varphi_1] \leftrightarrow \top$ and $\Delta(\mathcal{A}) \not\models \tau_{\text{FLP}}[\varphi_2] \leftrightarrow \top$. Actually, we have that,

$$\Delta(\mathcal{A}) \models \mathsf{Forget}(\{\tau_\star[\Pi]\} \cup \Delta(\mathcal{A}), \{q, q'\}) \leftrightarrow ((p' \leftrightarrow r') \vee (p \leftrightarrow \neg r) \wedge \neg(p' \wedge r')).$$

One can check further that

$$\Delta(\mathcal{A}) \models (p' \supset p \vee r') \wedge (r' \supset r \vee p') \leftrightarrow ((p' \leftrightarrow r') \vee (p \leftrightarrow \neg r) \wedge \neg(p' \wedge r')),$$

which shows that φ_1 (resp. φ_2) is a result of HT-forgetting (resp. FLP-forgetting) q from Π. $\Box$

The following result states that we can reduce checking whether the $\star$-forgetting results of two formulas are strongly $\star$-equivalent to checking whether the propositional forgetting results of corresponding two formulas are equivalent.

Proposition 16 *Let ψ and φ be two formulas of $\mathcal{L}_{\mathcal{A}}$ and V a set of atoms. Then* $\mathsf{Forget}_\star(\psi, V) \equiv_\star \mathsf{Forget}_\star(\varphi, V)$ *iff the following condition holds:*

$$\mathsf{Forget}(\{\tau_\star[\psi]\} \cup \Delta(\mathcal{A}), V \cup V') \equiv \mathsf{Forget}(\{\tau_\star[\varphi]\} \cup \Delta(\mathcal{A}), V \cup V').$$

4.4 Computation and Complexity

Theorem 6 and Propositions 4 and 10 imply a naive approach to compute $\star$-forgetting results. Formally speaking, given a formula ψ over a signature $\mathcal{A}$ and a set V of atoms, $\mathsf{Forget}_\star(\psi, V)$ can be computed as follows:

(Step 1) Evaluating all $\star$-models of ψ, denoted by $\mathcal{M}$.

(Step 2) Restrict $\mathcal{M}$ to $\mathcal{A} \setminus V$, denoted by $\mathcal{M}_{|V}$, i.e.

$$\mathcal{M}_{|V} = \{\langle H \setminus V, T \setminus V\rangle | \langle H, T\rangle \in \mathcal{M}\}.$$

(Step 3) Enumerating the following formulas (over the signature $\mathcal{A} \setminus V$) from $\mathcal{M}_{|V}$:

- $\lambda_\star(X, Y)$ if $\langle X, Y\rangle \notin \mathcal{M}_{|V}$ but $\langle Y, Y\rangle \in \mathcal{M}_{|V}$,
- $\lambda(Y, Y)$ if $\langle Y, Y\rangle \notin \mathcal{M}_{|V}$.

(Step 4) Finally, conjunct all the constructed formulas, denoted by φ.

Corollary 13 *Let ψ, V and φ be given as above. Then $\varphi \equiv_\star$* $\mathsf{Forget}_\star(\psi, V)$.

Alternatively, in terms of Theorem 10, we can compute $\mathsf{Forget}_\star(\psi, V)$ by enumerating the $\star$-consequences of ψ that are $\star$-irrelevant to V. As there exist sound and complete axiomatic systems for the HT-logic (Jongh & Hendriks, 2003), checking HT-consequence relation is axiomatically doable. Though a sound and complete axiomatic system for FLP-logic is recently unknown, we still can enumerate all the formulas of form (2) over the signature $\mathcal{A} \setminus V$ and check if they are FLP-consequence of ψ. Nevertheless, it is also observed that from a computational viewpoint, like the propositional forgetting, each of the above two approaches would be expensive. This appears to be inevitable in terms of the following complexity results, unless the complexity hierarchy collapses.

Theorem 14 *Let ψ and φ be two formulas and V a set of atoms.*

(i) The problem of deciding if $\psi \equiv_\star$ $\mathsf{Forget}_\star(\psi, V)$ is co-NP*-complete.*

(ii) The problem of deciding if $\mathsf{Forget}_\star(\varphi, V) \equiv_\star \mathsf{Forget}_\star(\psi, V)$ *is Π_2^P-complete.*

(iii) The problem of deciding if $\varphi \equiv_\star$ $\mathsf{Forget}_\star(\psi, V)$ *is Π_2^P-complete.*

According to our representation theorem (i.e. Theorem 10), the result (i) in Theorem 14 means that checking if ψ is $\star$-irrelevant to V, i.e. $\mathsf{IR}_\star(\psi, V)$, is intractable. The result (ii) of Theorem 14, on the other hand, presents the complexity of $\star$-forgetting equivalence checking, i.e., if two formulas are strongly $\star$-equivalent when they are restricted to a common signatures. The last result (iii) of Theorem 14 states that checking if a formula is a result of $\star$-forgetting is generally difficult.

Proposition 17 *Let ψ and φ be two formulas and V a set of atoms.*

(i) The problem of deciding whether $\psi \models_\star \mathsf{Forget}_\star(\varphi, V)$ is Π_2^P-complete.

(ii) The problem of deciding whether $\mathsf{Forget}_\star(\psi, V) \models_\star \varphi$ is co-NP-*complete.*

Theorem 14 and Proposition 17 tell us that for $\star$-forgetting, in general the complexity of resulting checking and inference problems is located at the same level of the complexity polynomial hierarchy as the propositional forgetting.

4.5 Conflict Solving Based on Knowledge Forgetting

In the following, we consider the application of the proposed forgetting in conflict solving for logic program contexts, that represent a knowledge system consisting of knowledge bases of multiple agents.

Definition 11 A logic program context *is an n-ary tuple $\Omega = (\Pi_1, \dots, \Pi_n)$ where Π_i is a consistent logic program. Ω is* $\star$-conflict-free *if $\Pi_1 \cup \dots \cup \Pi_n$ is consistent under $\star$-stable model semantics.*

Definition 12 *Let $\Omega = (\Pi_1, \dots, \Pi_n)$ be a logic program context. A* $\star$-solution *of Ω is a minimal subset S of $\mathcal{A}$ such that $(\mathsf{Forget}_\star(\Pi_1, S), \dots, \mathsf{Forget}_\star(\Pi_n, S))$ is $\star$-conflict-free, where $\mathcal{A}$ is the underlying signature.*

It is obvious that $\emptyset$ is a $\star$-solution of $\star$-conflict-free logic program context Ω.

We consider the following simplified Zhang and Foo's conflict solving scenario (cf., see Zhang & Foo, 2006, Ex. 6).

Example 9 A couple John and Mary are discussing their family investment plan. There are four different shares *shareA, shareB, shareC* and *shareD*, where *shareA* and *shareB* are of high risk but also have high return; *shareC* and *shareD* are of low risk and may be suitable for a long term investment. John's and Mary's investment preference over these shares are encoded as the following logic programs Π_J and Π_m respectively:

Π_J :	Π_M :
r_1 : $sA \leftarrow not\, sB$,	r_1' : $sC \leftarrow$,
r_2 : $sC \leftarrow not\, sD$,	r_2' : $sD \leftarrow$,
r_3 : $sD \leftarrow not\, sC$,	r_3' : $sB \leftarrow not\, sA, not\, sC$,
r_4 : $\leftarrow sC, sD$,	r_4' : $\leftarrow sA, sB$,

where *s#* stands for *share#*. The intuitive meaning of these rules can be easily read out. E.g. rule r_1 says that John wants to buy *shareA* if he don't buy *shareB*, while rules r_2, r_3 and r_4 mean that John wants to buy *shareC* or *shareD*, but not both of them.

As one can see that $\Pi_J \cup \Pi_M$ has no $\star$-stable model due to the confliction between rule r_4 and r_1', r_2', the logic program context $\Omega = (\Pi_J, \Pi_M)$ is not $\star$-conflict-free.

For $S = \{sD\}$, we have the following

$$\begin{aligned}
&\mathsf{Forget}_{\mathrm{HT}}(\Pi_J, S) \equiv_{\mathrm{HT}} \{sA \leftarrow not\, sB, \qquad sC; not\, sC \leftarrow\},\\
&\mathsf{Forget}_{\mathrm{HT}}(\Pi_M, S) \equiv_{\mathrm{HT}} \{sC \leftarrow, \qquad sB \leftarrow not\, sA, not\, sC, \qquad \leftarrow sA, sB\}.
\end{aligned}$$

One can check that $\mathsf{Forget}_{\mathrm{HT}}(\Pi_J, S) \cup \mathsf{Forget}_{\mathrm{HT}}(\Pi_M, S)$ has a unique HT-stable model $\{sA, sC\}$. Thus S is an HT-solution of Ω. It can be said that John and Mary may have an agreement on their investment plan about shares *shareA*, *shareB* and *shareC* if they agree to give up the belief (knowledge) about *shareD*. It results in an investment to shares *shareA* and *shareC*, but not to *shareB*.

One can further check that, under the FLP-stable model semantics, if John and Mary can give up the belief about *shareD* then it results in the same investment plan to shares *shareA* and *shareC*, but not to share *shareB*. The reason is that $\mathsf{Forget}_{\mathrm{FLP}}(\Pi_J, S) \cup \mathsf{Forget}_{\mathrm{FLP}}(\Pi_M, S)$ has a unique FLP-stable model $\{sA, sC\}$.

5. Related Work

In this section we compare the $\star$-forgetting with weak and strong forgetting (Zhang & Foo, 2006), semantic forgetting (Eiter & Wang, 2008) and the forgetting operators F_{S} and F_{W} (Wong, 2009).

5.1 Weak and Strong Forgetting

Let Π be a normal logic program and p a propositional atom. The *reduction* of Π with respect to p, denoted by $\mathsf{Red}(\Pi, \{p\})$, is the normal logic program obtained from Π by

(1) for each rule r of Π with $p \in Head(r)$, if there is a rule r' in Π such that $p \in Body^+(r')$, then replacing r' with
$$Head(r') \leftarrow Body(r), Body(r') \setminus \{p\}.$$

(2) if there is such a rule r' in Π and it has been replaced by a new rule in the previous step, then removing the rule r from the remaining normal logic program.

Let X be a set of propositional atoms. Then the *reduction* of Π with respect to X is inductively defined as follows:

$$\begin{aligned}
\mathsf{Red}(\Pi, \emptyset) &= \Pi,\\
\mathsf{Red}(\Pi, X \cup \{p\}) &= \mathsf{Red}(\mathsf{Red}(\Pi, \{p\}), X).
\end{aligned}$$

The *strong forgetting* p in a normal logic program Π is the normal logic program $\mathsf{SForget}(\Pi, \{p\})$ obtained from $\mathsf{Red}(\Pi, \{p\})$ by removing each rule r if either r is valid [5] or $p \in Head(r) \cup Body^+(r) \cup Body^-(r)$. The *weak forgetting* p in Π is the normal logic program $\mathsf{WForget}(\Pi, \{p\})$ obtained from $\mathsf{Red}(\Pi, \{p\})$ by firstly removing each rule r if either r is valid, or $p \in Head(r) \cup Body^+(r)$ and then removing "$not\, p$" from the remaining rules.

5. A rule r is *valid* if $Head(r) \cap Body^+(r) \neq \emptyset$ or $Body^+(r) \cap Body^-(r) \neq \emptyset$.

Let X be a set of atoms. The strong (and weak) forgetting X in Π is recursively defined as

$$\begin{aligned}
&\mathsf{SForget}(\Pi, \emptyset) = \Pi; \qquad \mathsf{WForget}(\Pi, \emptyset) = \Pi;\\
&\mathsf{SForget}(\Pi, X \cup \{p\}) = \mathsf{SForget}(\mathsf{SForget}(\Pi, \{p\}), X);\\
&\mathsf{WForget}(\Pi, X \cup \{p\}) = \mathsf{WForget}(\mathsf{WForget}(\Pi, \{p\}), X).
\end{aligned}$$

It is known that the two forgetting operators are independent of the ordering of forgotten atoms in the sense of strong HT-equivalence of logic programs under HT-stable model semantics (cf., see Zhang & Foo, 2006, Prop. 2).

Example 10 Consider the below two normal logic programs:

$$\begin{aligned}
\Pi &= \{p \leftarrow q, \qquad q \leftarrow p, \qquad r \leftarrow \mathit{not}\, p\},\\
\Sigma &= \{p \leftarrow q, \qquad q \leftarrow p, \qquad r \leftarrow \mathit{not}\, q\}.
\end{aligned}$$

One can check that Π and Σ are strongly equivalent. We have that

$$\begin{aligned}
&\mathsf{SForget}(\Pi, \{p\}) = \emptyset, \qquad \mathsf{WForget}(\Pi, \{p\}) = \{r \leftarrow\},\\
&\mathsf{SForget}(\Sigma, \{p\}) = \mathsf{WForget}(\Sigma, \{p\}) = \{r \leftarrow \mathit{not}\, q\}.
\end{aligned}$$

The example shows that neither weak forgetting preserves strong equivalence, nor is strong forgetting. One can further verify that $\Pi \models_\star \neg q \wedge \neg r \supset \bot$ and $\Pi \not\models_\star r$ for $\star \in \{\mathrm{HT}, \mathrm{FLP}\}$. Thus the strong forgetting does not satisfy "positive persistence", and the weak forgetting does not satisfy "weakening" and "negative persistence". Actually, for HT- and FLP-forgetting, we have the following

$$\begin{aligned}
&\mathsf{Forget}_{\mathrm{HT}}(\Pi, p) \equiv_{\mathrm{HT}} \mathsf{Forget}_{\mathrm{HT}}(\Sigma, p) \equiv_{\mathrm{HT}} \{\neg q \wedge \neg r \supset \bot\},\\
&\mathsf{Forget}_{\mathrm{FLP}}(\Pi, p) \equiv_{\mathrm{FLP}} \mathsf{Forget}_{\mathrm{FLP}}(\Sigma, p) \equiv_{\mathrm{FLP}} \{\neg q \wedge \neg r \supset \bot\}.
\end{aligned}$$

Here $\Pi \equiv_{\mathrm{FLP}} \Sigma$ follows from the fact that $\Pi \equiv_{\mathrm{HT}} \Sigma$ and Proposition 2. □

5.2 Semantic Forgetting

Having addressed certain issues of weak and strong forgetting, Eiter and Wang (2008) proposed a semantic forgetting for consistent disjunctive logic programs. Formally speaking, let Π be a consistent disjunctive logic program and p an atom. A set M of atoms is a *p-stable model* of Π iff M is a stable model of Π and there is no stable model M^* of Π such that $M^* \setminus \{p\} \subset M \setminus \{p\}$. A disjunctive logic program Π' represents the result of forgetting about p in Π, if

- Π' does not mention the atom p, and
- a set M' of atoms is a stable model of Π' iff Π has a p-stable model M such that $M' \sim_p M$.

In terms of the above definition, such forgetting results are not unique under strong equivalence. This means, their forgetting does not preserve strong equivalence. To compute the result of forgetting an atom in a consistent disjunctive logic program, they proposed three algorithms forget_1, forget_2 and forget_3 (Eiter & Wang, 2008). The example below further demonstrates the difference between this semantic forgetting and the $\star$-forgetting.

Example 11 Let $\Pi = \{p \leftarrow q\}$ be a program over signature $\mathcal{A} = \{p, q, r\}$. Although program Π has nothing to do with the atom r, we have that $\mathsf{forget}_i(\Pi, r) = \emptyset$ ($i = 1, 2, 3$), which seems not intuitive as it loses some information irrelevant to what we want to forget. However $\mathsf{Forget}_\star(\Pi, r) \equiv_\star \Pi$. $\square$

This example also shows that the semantic forgetting does not satisfy "positive persistence" postulate as $\Pi \models_\star q \supset p$, which is lost in the semantic forgetting result $\mathsf{forget}_i(\Pi, r)$ for $i = 1, 2, 3$.

5.3 Forgetting Operators $\mathsf{F_S}$ and $\mathsf{F_W}$

Wong (2009) developed his forgetting for disjunctive logic programs. Differently from the work of Zhang and Foo (2006), and Eiter and Wang (2008), Wong's forgetting is defined based on the HT-logic. In this sense, his approach probably shares a common logic ground with HT-forgetting. Wong also defined two forgetting operators $\mathsf{F_S}$ and $\mathsf{F_W}$, which correspond to two series of program transformations. See Appendix D for the detailed definitions.

The interesting feature of Wong's forgetting is that it preserves strong equivalence. However, a major issue with this forgetting is that: on one hand, the forgetting $\mathsf{F_S}$ may cause unnecessary information loss; on the other hand, the forgetting $\mathsf{F_W}$ may also introduce extra information that one does not want, as illustrated by the following example.

Example 12 Let us consider the normal logic program Π consisting of:

$$a \leftarrow x, \qquad y \leftarrow a, not\, z, \qquad q \leftarrow not\, p, \qquad p \leftarrow not\, q, \qquad \leftarrow p, q.$$

Then we have:

$$\begin{aligned}
&\mathsf{F_S}(\Pi, \{a, p\}) \equiv_{\text{HT}} \{y \leftarrow x, not\, z\},\\
&\mathsf{F_W}(\Pi, \{a, p\}) \equiv_{\text{HT}} \{y \leftarrow x, not\, z, \qquad \leftarrow x, \qquad q \leftarrow\},\\
&\mathsf{Forget}_{\text{HT}}(\Pi, \{a, p\}) \equiv_{\text{HT}} \{y \leftarrow x, not\, z, \qquad q \leftarrow not\, not\, q\},\\
&\mathsf{Forget}_{\text{FLP}}(\Pi, \{a, p\}) \equiv_{\text{FLP}} \{y \leftarrow x, not\, z, \qquad q \leftarrow not\, not\, q\}.
\end{aligned}$$

Since $\Pi \models_{\text{HT}} \{q \leftarrow not\, not\, q\}$, which is irrelevant to atoms a and p, it seems to us that forgetting $\{a, p\}$ from Π should not affect this fact. But $\mathsf{F_S}(\Pi, \{a, p\}) \not\models_{\text{HT}} \{q \leftarrow not\, not\, q\}$. In this sense, we see that $\mathsf{F_S}$ has lost some information that we wish to keep. This shows that the operator $\mathsf{F_S}$ does not satisfy "positive persistence" postulate.

On the other hand, from the fact that $\Pi \not\models_{\text{HT}} q$ but $\mathsf{F_W}(\Pi, \{a, p\}) \models_{\text{HT}} q$, it appears that $\mathsf{F_W}$ may introduce unnecessary information, which indeed conflicts our intuition of program weakening via forgetting, i.e., it does not satisfy the "weakening" postulate. $\square$

As we mentioned in the introduction, the following example confirms that an expected result can not be obtained from either one of the above three forgetting approaches.

Example 13 [Continued from Example 5] For the normal logic program Π:

$$(\neg p \supset q) \wedge (\neg q \supset p) \wedge (p \wedge q \supset \bot),$$

we have the following:

$$\mathsf{SForget}(\Pi, \{p\}) = \mathsf{forget}_1(\Pi, \{p\}) = \mathsf{F}_\mathsf{S}(\Pi, \{p\}) = \emptyset,$$
$$\mathsf{WForget}(\Pi, \{p\}) = \mathsf{F}_\mathsf{W}(\Pi, \{p\}) = \{q\}.$$

Here, the expected logic program that represents the same information of Π when the auxiliary atom p is ignored should be $\neg\neg q \supset q$. $\square$

6. Concluding Remarks

In this paper two semantic knowledge forgetting approaches, called HT- and FLP-forgetting respectively, were proposed for logic programs under stable model and FLP-stable model semantics respectively. These knowledge forgetting results can be captured by the corresponding logical consequence of forgotten logic programs that are irrelevant to forgotten atoms. It consequently preserves strong equivalence of logic programs under HT- and FLP-stable model semantics respectively. This is a major advantage when compared to other existing forgetting approaches in logic programming.

As a starting point, we investigated the model theoretical characterization of logic programs under HT- and FLP-stable model semantics, and studied their respective strong equivalence problems using classical propositional logic equivalence. Many properties of forgetting have been explored, such as existence of forgetting results, a representation theorem, and the complexity of some decision problems related to these forgettings. We also considered an application of knowledge forgetting in conflict solving.

Although we have presented abstract approaches to computing the forgetting results and we showed the underlying difficulties of the computation, it is valuable to study practical algorithms for different subclasses of logic programs. Another challenging future work is to extend the knowledge forgetting to other nonmonotonic systems, and in particular first-order logic programs (Ferraris et al., 2011). As we have mentioned in the introduction that forgetting can be effectively used to solve some confliction, e.g. the strong and weak forgetting (Zhang & Foo, 2006) and the propositional forgetting (Lang & Marquis, 2010), such an application of knowledge forgetting deserves further studying.

As what we concentrate upon in this paper is knowledge forgetting in logic programs, which is based on the notion of strong equivalence, an interesting work is to consider forgetting under the stable model semantics of logic programs along the work (Wang et al., 2013). Last but not least, logic programs under supported model semantics enjoys some similar properties as that of logic programs under HT- and FLP-stable models semantics (Truszczynski, 2010), we will consider the knowledge forgetting for logic programs under the supported model semantics in another paper.

Acknowledgments

We thank Mirek Truszczynski for encouraging us to consider knowledge forgetting for logic programs under the FLP-stable model semantics. We thank the anonymous reviewers for their insightful comments, and Robin Bianchi for his help on formatting the paper. Yisong Wang is partially supported by the National Natural Science Foundation of China grant 61370161 and Stadholder Foundation of Guizhou Province under grant (2012)62.

Appendix A. Proofs for Section 2

Proposition 1 *Let A, B, C, D be set of atoms. We have the following*

(i) $\bigwedge(A \cup \neg B) \supset \bigvee(D \cup \neg C) \equiv_{\text{HT}} \bigwedge(A \cup \neg B \cup \neg\neg C) \supset \bigvee D$.

(ii) $\bigwedge(A \cup \neg B) \supset \bigvee(D \cup \neg C) \models_{\text{FLP}} \bigwedge(A \cup \neg B \cup \neg\neg C) \supset \bigvee D$.

Proof: (ii) Suppose $\langle X, Y\rangle$ is an FLP-model of $\bigwedge(A \cup \neg B) \supset \bigvee(D \cup \neg C)$ but not an FLP-model of $\bigwedge(A \cup \neg B \cup \neg\neg C) \supset \bigvee D$. It follows that the following conditions hold:

(a) $X \models \bigwedge(A \cup \neg B \cup \neg\neg C)$, which implies $X \models \bigwedge(A \cup \neg B)$.

(b) $Y \models \bigwedge(A \cup \neg B \cup \neg\neg C)$, which implies $Y \models \bigwedge(A \cup \neg B) \wedge \bigwedge C$, and

(c) $\langle X, Y\rangle \not\models_{\text{FLP}} \bigvee D$, i.e. $X \not\models \bigvee D$.

The conditions (a) and (b) show that $\langle X, Y\rangle \models_{\text{FLP}} \bigvee(D \cup \neg C)$, i.e. $X \models \bigvee D$ or $Y \models \bigvee \neg C$. Together with the conditions (b) and (c), a contradiction follows. $\Box$

Appendix B. Proofs for Section 3

Proposition 4 *A collection $\mathcal{M}$ of $\star$-interpretations is $\star$-expressible iff*

$$\langle X, Y\rangle \in \mathcal{M} \textit{ implies } \langle Y, Y\rangle \in \mathcal{M}. \tag{17}$$

Actually, if $\mathcal{M}$ satisfy condition (17) then the following logic program

$$\Pi_\star = \{\lambda_\star(X, Y) | \langle X, Y\rangle \notin \mathcal{M} \textit{ and } \langle Y, Y\rangle \in \mathcal{M}\} \cup \{\lambda(Y, Y) | \langle Y, Y\rangle \notin \mathcal{M}\}$$

captures $\mathcal{M}$ in the sense that $\mathsf{Mod}_\star(\Pi_\star) = \mathcal{M}$.

Proof: The direction from left to right follows from (i) of Proposition 3. We prove the other direction. Let $\Pi_\star$ be the propositional theory consisting of, for every $X \subset Y \subseteq \mathcal{A}$,

- $\lambda_\star(X, Y)$ if $\langle X, Y\rangle \notin \mathcal{M}$ and $\langle Y, Y\rangle \in \mathcal{M}$, and
- $\lambda(Y, Y)$ if $\langle Y, Y\rangle \notin \mathcal{M}$.

By Lemma 1, $\mathsf{Mod}_\star(\Pi_\star) = \mathcal{M}$. $\Box$

Lemma 2 *Let A, B be two sets of atoms, and $X \subseteq Y \subseteq \mathcal{A}$. $\langle X, Y\rangle \models_\star \bigwedge B \supset \bigvee A$ iff $X \models \bigwedge B \supset \bigvee A$ and $Y \models \bigwedge B \supset \bigvee A$.*

Proof: According to (iii) of Proposition 3 and Proposition 2, it is sufficient to show that, for the case $\star = \text{HT}$,

$$X \models (\bigwedge B \supset \bigvee A)^Y \text{ iff } X \models \bigwedge B \supset \bigvee A \text{ and } Y \models \bigwedge B \supset \bigvee A.$$

($\Rightarrow$) Note that $Y \models \bigwedge B \supset \bigvee A$ and $X \models (\bigwedge B)^Y$ implies $X \models (\bigvee A)^Y$. Suppose $X \not\models \bigwedge B \supset \bigvee A$, i.e. $B \subseteq X$ and $A \cap X = \emptyset$. It follows that $Y \models \bigwedge B$ due to $B \subseteq Y$, and then

$Y \models \bigvee A$, i.e. $A \cap Y \neq \emptyset$. Thus we have $X \models (\bigwedge B)^Y$ since $(\bigwedge B)^Y = \bigwedge B$. By $X \models (\bigvee A)^Y$ i.e. $X \models \bigvee A$, we have $X \cap A \neq \emptyset$, a contradiction.

($\Leftarrow$) We need only to show $X \models (\bigwedge B)^Y \supset (\bigvee A)^Y$ since $Y \models \bigwedge B \supset \bigvee A$. Suppose $X \models (\bigwedge B)^Y$ and $X \not\models (\bigvee A)^Y$. The former implies $B \subseteq X \subseteq Y$, thus $X \cap A \neq \emptyset$ by $X \models \bigwedge B \supset \bigvee A$. The latter implies $X \cap (A \cap Y) = \emptyset$, which means $X \cap A = \emptyset$ since $X \subseteq Y$, a contradiction. $\square$

Proposition 5 *A set $\mathcal{M}$ of $\star$-interpretations is* positively $\star$-expressible*, i.e., there is a positive logic program Π s.t* $\mathsf{Mod}_\star(\Pi) = \mathcal{M}$*, iff $\mathcal{M}$ satisfies the criteria:*

$$\langle X, Y\rangle \in \mathcal{M} \textit{ iff } X \subseteq Y, \langle X, X\rangle \in \mathcal{M} \textit{ and } \langle Y, Y\rangle \in \mathcal{M}. \tag{18}$$

Actually, if $\mathcal{M}$ satisfy condition (18) then the following logic program

$$\Pi_\star = \{\bigwedge X \supset \bigvee \overline{X} | \langle X, X\rangle \notin \mathcal{M}\}$$

captures $\mathcal{M}$ in the sense that $\mathsf{Mod}_\star(\Pi_\star) = \mathcal{M}$.

Proof: It suffices to prove the case $\star = \textsc{ht}$ by Proposition 2.

($\Rightarrow$) Let Π be a positive logic program whose HT-models are exact the ones in $\mathcal{M}$. For every HT-interpretation $\langle X, Y\rangle$, by Lemma 2, $\langle X, Y\rangle \models_{\text{HT}} \Pi$ iff $X \subseteq Y$, $X \models \Pi$ i.e. $\langle X, X\rangle \models_{\text{HT}} \Pi$, and $\langle Y, Y\rangle \models_{\text{HT}} \Pi$ i.e. $Y \models \Pi$ since every rule of Π is positive. The condition (18) follows.

($\Leftarrow$) Let $\mathcal{N} = \{X \subseteq \mathcal{A} | \langle X, X\rangle \in \mathcal{M}\}$. We construct the propositional theory Π consisting of

$$\bigwedge X \supset \bigvee \overline{X}$$

for every $X \in \overline{\mathcal{N}} (= 2^{\mathcal{A}} \setminus \mathcal{N})$.

Firstly we show $\mathsf{Mod}(\Pi) = \mathcal{N}$. Suppose $X \models \Pi$ and $X \notin \mathcal{N}$. We have that $X \in \overline{\mathcal{N}}$. It follows that $X \not\models \Pi$ as $\bigwedge X \supset \bigvee \overline{X}$ belongs to Π. On the other hand, suppose $X \in \mathcal{N}$ and $X \not\models \Pi$. It follows that there exists $X' \in \overline{\mathcal{N}}$ such that $X \not\models \bigwedge X' \supset \bigvee \overline{X'}$, i.e., $X' \subseteq X$ and $X \cap \overline{X'} = \emptyset$, from which we have $X = X'$ thus $X \in \mathcal{N}$, a contradiction.

Secondly we show $\mathsf{Mod}_{\text{HT}}(\Pi) = \mathcal{M}$. On the one hand, let $\langle X, Y\rangle \models_{\text{HT}} \Pi$. We have that $X \models \Pi$ and $Y \models \Pi$ by Lemma 2. It follows $X, Y \in \mathcal{N}$, which implies $\langle X, X\rangle \in \mathcal{M}$ and $\langle Y, Y\rangle \in \mathcal{M}$. Thus $\langle X, Y\rangle \in \mathcal{M}$ by (18). On the other hand, let $\langle X, Y\rangle \in \mathcal{M}$. In terms of (18), we have $\langle X, X\rangle \in \mathcal{M}$ and $\langle Y, Y\rangle \in \mathcal{M}$. Thus $X \in \mathcal{N}$ and $Y \in \mathcal{N}$, i.e. $X \models \Pi$ and $Y \models \Pi$. Thus $\langle X, Y\rangle \models_{\text{HT}} \Pi$ by Lemma 2. $\square$

Proposition 6 *A collection $\mathcal{M}$ of $\star$-interpretations is* Horn $\star$-expressible*, i.e., there is a Horn logic program Π such that* $\mathsf{Mod}_\star(\Pi) = \mathcal{M}$*, iff $\mathcal{M}$ satisfies, in addition to (10), the following criteria:*

$$\langle X, Y\rangle \in \mathcal{M} \textit{ and } \langle H, T\rangle \in \mathcal{M} \Rightarrow \langle X \cap H, Y \cap T\rangle \in \mathcal{M}. \tag{19}$$

Proof: It suffices to prove the case $\star = \textsc{ht}$ by Proposition 2.

($\Rightarrow$) Suppose Π is a Horn logic program such that $\mathsf{Mod}_{\text{HT}}(\Pi) = \mathcal{M}$. By Proposition 5, $\mathsf{Mod}_{\text{HT}}(\Pi)$ satisfies (18). Suppose $\langle X, Y\rangle$ and $\langle H, T\rangle$ are two HT-models of Π. It follows that X, Y, H and T are models of Π by Lemma 2. Thus $X \cap H \models \Pi$ and $Y \cap T \models \Pi$, by which $\langle X \cap H, Y \cap T\rangle \models \Pi$ due to $X \cap H \subseteq Y \cap T$.

($\Leftarrow$) Let $\mathcal{N}$ and Π be the ones defined in the proof of Proposition 5. If $X, Y \in \mathcal{N}$ then $X \cap Y \in \mathcal{N}$ according to (19). It follows that there exists a Horn logic program (a set of Horn clauses) whose models are exactly the ones in $\mathcal{N}$. As a matter of fact, the Horn program Π' can be constructed from Π by replacing each $\bigwedge X \supset \bigvee Y$ with

$$\bigwedge X \supset p_1, \ldots, \bigwedge X \supset p_k \tag{20}$$

where $X \cap Y = \emptyset$ and $\bigcap\{Y' \setminus X | X \subseteq Y' \text{ and } Y' \in \mathcal{N}\} = \{p_1, \ldots, p_k\}$.

We firstly show $\Pi \equiv \Pi'$ by proving

$$\Pi \models \left(\bigwedge X \supset \bigvee Y\right) \equiv \left(\bigwedge X \supset \bigwedge_{1 \le i \le k} p_i\right)$$

where p_i $(1 \le i \le k)$ are defined in (20). The direction from right to left is trivial as $\bigwedge X \supset \bigvee Y$ belongs to Π. Let us consider the other direction. Suppose $H \models \Pi$, H is a model of $\bigwedge X \supset \bigvee Y$ and $H \not\models \bigwedge X \supset p_i$ for some i $(1 \le i \le k)$. We have that $X \subseteq H$ and $H \cap Y \neq \emptyset$. It follows that H is some element of $\{Y' \setminus X | X \subseteq Y' \text{ and } Y' \in \mathcal{N}\}$ and then $\{p_1, \ldots, p_k\} \subseteq H$. It is a contradiction.

Finally $\mathsf{Mod}_{\text{HT}}(\Pi') = \mathcal{M}$ follows from $\mathsf{Mod}_{\text{HT}}(\Pi) = \mathcal{M}$ and Proposition 5. $\Box$

Proposition 7 *Let $\varphi = \bigwedge(B \cup \neg C) \supset \bigvee(A \cup \neg D)$, where A, B, C, D are subsets of $\mathcal{A}$. Then we have $\Delta(\mathcal{A}) \models \tau_{\text{FLP}}[\varphi] \leftrightarrow \tau_{\text{HT}}[\varphi]$.*

Proof: Note that $\tau_{\text{HT}}[\neg p] = \neg p \wedge \neg p'$ and $\tau_{\text{FLP}}[\neg p] = \neg p'$. We have

$$\tau_{\text{HT}}[\varphi] = \varphi' \wedge \left(\bigwedge B \wedge \bigwedge_{c \in C} (\neg c \wedge \neg c') \supset \bigvee A \vee \bigvee_{d \in D} (\neg d \wedge \neg d')\right),$$
$$\tau_{\text{FLP}}[\varphi] = \varphi' \wedge \left(\bigwedge (B \cup \neg C \cup B' \cup \neg C') \supset \bigvee (A \cup \neg D')\right).$$

Since $\Delta(\mathcal{A}) \models \neg p \wedge \neg p' \leftrightarrow \neg p'$, we have that

$$\Delta(\mathcal{A}) \models \tau_{\text{HT}}[\varphi] \leftrightarrow \varphi' \wedge \left(\bigwedge (B \cup \neg C') \supset \bigvee A \vee \bigvee_{d \in D} \neg d'\right),$$
$$\Delta(\mathcal{A}) \models \tau_{\text{FLP}}[\varphi] \leftrightarrow \varphi' \wedge \left(\bigwedge (B \cup \neg C') \supset \bigvee (A \cup \neg D')\right).$$

It completes the proof. $\Box$

Proposition 8 *Let φ be a formula of $\mathcal{L}_{\mathcal{A}}$ and $X \subseteq Y \subseteq \mathcal{A}$. $\langle X, Y \rangle$ is a $\star$-model of φ iff $X \cup Y'$ is a model of $\Delta(\mathcal{A}) \cup \{\tau_\star[\varphi]\}$.*

Proof: We prove the case $\star = \text{FLP}$ by induction on the structures of φ. Let $X \subseteq Y \subseteq \mathcal{A}$.

- $\varphi = p$ or $\varphi = \bot$. It is trivial for $\varphi = \bot$. On the other hand, $\langle X, Y \rangle \models_{\text{FLP}} p$ iff $X \models p$ iff $X \cup Y' \models p$.

- $\varphi = \varphi_1 \circ \varphi_2$ where $\circ \in \{\wedge, \vee\}$. It follows from the inductive assumption.
- $\varphi = \varphi_1 \supset \varphi_2$. We have $\tau_{\text{FLP}}[\varphi_1 \supset \varphi_2] = (\varphi_1' \supset \varphi_2') \wedge (\varphi_1 \wedge \varphi_1' \supset \tau_{\text{FLP}}[\varphi_2])$. Recall that $\langle X, Y\rangle \models_{\text{FLP}} \varphi_1 \supset \varphi_2$ iff
  - $Y \models (\varphi_1 \supset \varphi_2)$ and,
  - either (a) $X \not\models \varphi_1$, or (b) $Y \not\models \varphi_1$, or (c) $\langle X, Y\rangle \models_{\text{FLP}} \varphi_2$.

  Note that

  - $Y \models (\varphi_1 \supset \varphi_2)$ iff $Y' \models \varphi_1' \supset \varphi_2'$ iff $X \cup Y' \models \varphi_1' \supset \varphi_2'$, and
  - (a) $X \not\models \varphi_1$ iff $X \cup Y' \not\models \varphi_1$, (b) $Y \not\models \varphi_1$ iff $Y' \not\models \varphi_1'$ iff $X \cup Y' \not\models \varphi_1'$, and (c) $\langle X, Y\rangle \models_{\text{FLP}} \varphi_2$ iff $X \cup Y' \models \tau_{\text{FLP}}[\varphi_2]$ by the inductive assumption.

  It follows that $\langle X, Y\rangle \models_{\text{FLP}} \varphi_1 \supset \varphi_2$ iff $X \cup Y' \models \tau_{\text{FLP}}[\varphi_1 \supset \varphi_2]$.

This completes the proof. □

Theorem 4 *Two formulas φ and ψ have the same $\star$-models (over $\mathcal{L}_\mathcal{A}$) iff $\Delta(\mathcal{A}) \cup \{\tau_\star[\varphi]\}$ and $\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}$ have the same models (over $\mathcal{L}_{\mathcal{A}\cup\mathcal{A}'}$).*

Proof: We prove the case $\star = \text{FLP}$.
($\Rightarrow$) $M \models \Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\varphi]\}$
iff $M_\mathcal{A} \cup M_{\mathcal{A}'} \models \Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\varphi]\}$
iff $\langle M_\mathcal{A}, M^*_{\mathcal{A}'}\rangle \models_{\text{FLP}} \varphi$ by Proposition 8, here $M^*_{\mathcal{A}'} = \{p | p' \in M_{\mathcal{A}'}\}$
iff $\langle M_\mathcal{A}, M^*_{\mathcal{A}'}\rangle \models_{\text{FLP}} \psi$ since $\varphi \equiv_{\text{FLP}} \psi$
iff $M_\mathcal{A} \cup M_{\mathcal{A}'} \models \Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\psi]\}$ by Proposition 8
iff $M \models \Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\psi]\}$.
($\Leftarrow$) $\langle X, Y\rangle \models_{\text{FLP}} \varphi$
iff $X \cup Y' \models \Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\varphi]\}$ by Proposition 8, here $Y' = \{p' | p \in Y\}$
iff $X \cup Y' \models \Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\psi]\}$ since $\Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\psi]\} \equiv \Delta(\mathcal{A}) \cup \{\tau_{\text{FLP}}[\varphi]\}$
iff $\langle X, Y\rangle \models_{\text{FLP}} \psi$ by Proposition 8. □

Proposition 9 *(i) The problem of deciding if a formula is $\star$-satisfiable is* NP*-complete.*
(ii) The problem of deciding if two formulas are $\star$-equivalent is co-NP*-complete.*

Proof: (i) Membership. If a formula φ is FLP-satisfiable then there exists an FLP-interpretation $\langle H, T\rangle$ such that $\langle H, T\rangle \models_{\text{FLP}} \varphi$. It is feasible to guess such an FLP-interpretation and check the condition $\langle H, T\rangle \models_{\text{FLP}} \varphi$. Thus the problem is in NP.

Hardness. It follows from the fact that $\neg\varphi$ is FLP-satisfiable iff $\neg\varphi$ is satisfiable, which is NP-hard, by (ii) of Proposition 3. This shows that the problem is NP-hard.

(ii) Membership. If $\varphi \not\equiv_{\text{FLP}} \psi$ then there exists $\langle H, T\rangle$ such that, either

(a) $\langle H, T\rangle \models_{\text{FLP}} \varphi$ and $\langle H, T\rangle \not\models_{\text{FLP}} \psi$, or

(b) $\langle H, T\rangle \not\models_{\text{FLP}} \varphi$ and $\langle H, T\rangle \models_{\text{FLP}} \psi$.

To guess such an FLP-interpretation $\langle H, T\rangle$ and to check the conditions (a) and (b) are feasible in polynomial time in the size of φ and ψ. Thus the problem in co-NP.

Hardness. We have that $\neg\varphi \equiv_{\text{FLP}} \bot$
iff $\neg\varphi$ has no FLP-model
iff $\neg\varphi$ has no model by (ii) of Proposition 3
iff φ is valid, which is co-NP-hard. Thus the problem is co-NP-hard. □

Appendix C. Proofs for Section 4

Lemma 3 *Let φ be a formula and $V \subseteq \mathcal{A}$. A formula ψ is a* result of $\star$-forgetting V from φ*, iff the following condition holds:*

$$\mathsf{Mod}_\star(\psi) = \mathsf{Mod}_\star(\varphi)_{\dagger V}.$$

Proof: ψ is a result of $\star$-knowledge forgetting V from φ
iff, for every $\star$-interpretation M, $M \models_\star \psi$ iff there exists $M' \models_\star \varphi$ s.t. $M \sim_V M'$
iff $\mathsf{Mod}_\star(\psi) = \{M \text{ is an } \star\text{-interpretation} \mid \exists M' \models_\star \varphi \textit{ and } M \sim_V M'\}$
iff $\mathsf{Mod}_\star(\psi) = \mathsf{Mod}(\varphi)_{\dagger V}$. □

Lemma 4 *Let X, Y, H, T and V be subsets of $\mathcal{A}$.*

(i) If $X \sim_V H$ and $Y \sim_V T$ then $X \cap Y \sim_V H \cap T$ and $X \cup Y \sim_V H \cup T$.

(ii) If $X \sim_V H$ and $Y' \sim_{V'} T'$ then $H \cup T' \sim_{V \cup V'} X \cup Y'$.

Proof: (i) Note that $(X \cap Y) \setminus V$
=$(X \setminus V) \cap (Y \setminus V)$
=$(H \setminus V) \cap (T \setminus V)$ due to $X \sim_V H$ and $Y \sim_V T$
=$(H \cap T) \setminus V$.

Thus $X \cap Y \sim_V T \cap T$. We can similarly prove $X \cup Y \sim_V H \cup T$.

(ii) Please note that $Y' = \{p' | p \in Y\}, V' = \{p' | p \in V\}$ and $T' = \{p' | p \in V\}$. We have that $(H \cup T') \setminus (V \cup V')$.
= $(H \setminus (V \cup V')) \cup (T' \setminus (V \cup V'))$
= $(H \setminus V) \cup (T' \setminus V')$ since $H \cap V' = \emptyset$ and $T' \cap V = \emptyset$
= $(X \setminus V) \cup (Y' \setminus V')$ since $H \sim_V H$ and $T' \sim_{V'} Y'$
= $(X \setminus (V \cup V')) \cup (Y' \setminus (V \cup V'))$ since $X \cap V' = \emptyset$ and $Y' \cap V = \emptyset$
= $(X \cup Y') \setminus (V \cup V')$.
It follows that $H \cup T' \sim_{V \cup V'} X \cup Y'$. □

Theorem 6 (Expressibility theorem) *Let φ be a formula and V a set of atoms. There exists a formula ψ such that $\mathsf{Mod}_\star(\psi) = \mathsf{Mod}_\star(\varphi)_{\dagger V}$.*

Proof: For every $\langle X, Y\rangle \in \mathsf{Mod}_\star(\varphi)_{\dagger V}$, there exists $\langle H, T\rangle \models_\star \varphi$ such that $\langle H, T\rangle \sim_V \langle X, Y\rangle$, i.e. $X \sim_V H$ and $Y \sim_V T$. By (i) of Proposition 3, $\langle T, T\rangle \models_\star \varphi$. Thus $\langle Y, Y\rangle \in \mathsf{Mod}_\star(\varphi)_{\dagger V}$ due to $\langle Y, Y\rangle \sim_V \langle T, T\rangle$. It follows that the collection $\mathsf{Mod}_\star(\varphi)_{\dagger V}$ satisfies the condition (8), then there is a formula ψ such that $\mathsf{Mod}_\star(\psi) = \mathsf{Mod}_\star(\varphi)_{\dagger V}$ by Proposition 4. □

Lemma 5 *A formula φ is $\star$-irrelevant to a set V of atoms iff $\langle H, T\rangle \models_\star \varphi$ implies $\langle X, Y\rangle \models_\star \varphi$ for every two $\star$-interpretations $\langle X, Y\rangle$ and $\langle H, T\rangle$ with $\langle X, Y\rangle \sim_V \langle H, T\rangle$*

Proof: φ is $\star$-irrelevant to V
iff there exists a formula ψ mentioning no atoms in V such that $\varphi \equiv_\star \psi$
iff there exists a formula ψ mentioning no atoms in V s.t $\mathsf{Mod}_\star(\varphi) = \mathsf{Mod}_\star(\psi)$
iff $\mathsf{Mod}_\star(\varphi) = \{\langle X, Y\rangle | X \subseteq Y$ and $\exists \langle H, T\rangle \sim_V \langle X, Y\rangle$ s.t $\langle H, T\rangle \models_\star \varphi\}$
iff $\langle H, T\rangle \models_\star \varphi$ implies $\langle X, Y\rangle \models_\star \varphi$ for every two $\star$-interpretations $\langle X, Y\rangle$ and $\langle H, T\rangle$ such that $\langle X, Y\rangle \sim_V \langle H, T\rangle$. □

Proposition 10 *Let ψ and φ be two formulas and V a set of atoms.*

(i) $\mathsf{IR}_\star(\mathsf{Forget}_\star(\psi, V), V)$.

(ii) *ψ has a $\star$-model iff* $\mathsf{Forget}_\star(\psi, V)$ *has.*

(iii) $\psi \models_\star \mathsf{Forget}_\star(\psi, V)$.

(iv) *If $\psi \models_\star \varphi$ then* $\mathsf{Forget}_\star(\psi, V) \models_\star \mathsf{Forget}_\star(\varphi, V)$.

(v) $\mathsf{Forget}_\star(\psi \vee \varphi, V) \equiv_\star \mathsf{Forget}_\star(\psi, V) \vee \mathsf{Forget}_\star(\varphi, V)$.

(vi) $\mathsf{Forget}_\star(\psi \wedge \varphi, V) \models_\star \mathsf{Forget}_\star(\psi, V) \wedge \mathsf{Forget}_\star(\varphi, V)$.

(vii) $\mathsf{Forget}_\star(\psi \wedge \varphi, V) \equiv_\star \mathsf{Forget}_\star(\psi, V) \wedge \varphi$ *if* $\mathsf{IR}_\star(\varphi, V)$.

Proof: (i) It immediately follows from Lemma 5.

(ii) It is evident that $\mathsf{Mod}_\star(\psi) \neq \emptyset$ iff $\mathsf{Mod}_\star(\psi)_{\dagger V} \neq \emptyset$ by Definition 8.

(iii) It is easy to see that $\mathsf{Mod}_\star(\psi) \subseteq \mathsf{Mod}_\star(\psi)_{\dagger V}$ by Definition 8.

(iv) Let $\psi \models_\star \varphi$, and $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$, i.e. $\langle H, T\rangle \in \mathsf{Mod}_\star(\psi)_{\dagger V}$. In terms of Definition 8, there exists $\langle H', T'\rangle \models_\star \psi$ such that $\langle H, T\rangle \sim_V \langle H', T'\rangle$. It implies that $\langle H', T'\rangle \models_\star \varphi$ since $\psi \models_\star \varphi$. Thus $\langle H, T\rangle \in \mathsf{Mod}_\star(\varphi)_{\dagger V}$, i.e. $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\varphi, V)$.

(v) $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi \vee \varphi, V)$
iff $\langle H, T\rangle \in \mathsf{Mod}_\star(\psi \vee \varphi)_{\dagger V}$
iff $\exists \langle H', T'\rangle \models_\star \psi \vee \varphi$ such that $\langle H, T\rangle \sim_V \langle H', T'\rangle$
iff $\exists \langle H', T'\rangle$ such that $\langle H, T\rangle \sim_V \langle H', T'\rangle$ and, either $\langle H', T'\rangle \models_\star \psi$ or $\langle H', T'\rangle \models_\star \varphi$
iff $\langle H, T\rangle \in \mathsf{Mod}_\star(\psi)_{\dagger V}$ or $\langle H, T\rangle \in \mathsf{Mod}_\star(\varphi)_{\dagger V}$
iff $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ or $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$
iff $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V) \vee \mathsf{Forget}_\star(\psi, V)$.

(vi) $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi \wedge \varphi, V)$
$\Rightarrow \langle H, T\rangle \in \mathsf{Mod}_\star(\psi \wedge \varphi)_{\dagger V}$
$\Rightarrow \exists \langle H', T'\rangle \models_\star \psi \wedge \varphi$ such that $\langle H, T\rangle \sim_V \langle H', T'\rangle$
$\Rightarrow \exists \langle H', T'\rangle$ such that. $\langle H, T\rangle \sim_V \langle H', T'\rangle$, $\langle H', T'\rangle \models_\star \psi$ and $\langle H', T'\rangle \models_\star \varphi$
$\Rightarrow \langle H, T\rangle \in \mathsf{Mod}_\star(\psi)_{\dagger V}$ and $\langle H, T\rangle \in \mathsf{Mod}_\star(\varphi)_{\dagger V}$
$\Rightarrow \langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ and $\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\varphi, V)$
$\Rightarrow \langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V) \wedge \mathsf{Forget}_\star(\varphi, V)$.

(vii) The direction from left to right follows from (vi) and the fact $\mathsf{IR}(\varphi, V)$, i.e. $\mathsf{Forget}_\star(\varphi, V) \equiv_\star \varphi$. Let us consider the other direction.

$\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V) \wedge \varphi$
$\Rightarrow \langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ and $\langle H, T\rangle \models_\star \varphi$
$\Rightarrow \exists \langle H', T'\rangle \models_\star \psi$ such that $\langle H, T\rangle \sim_V \langle H', T'\rangle$, and $\langle H, T\rangle \models_\star \varphi$

$\Rightarrow \exists\langle H,T\rangle \sim_V \langle H',T'\rangle$ such that $\langle H',T'\rangle \models_\star \psi \wedge \varphi$ by $\mathsf{IR}(\varphi, V)$ and Lemma 5
$\Rightarrow \langle H,T\rangle \in \mathsf{Mod}_\star(\psi \wedge \varphi)_{\dagger V}$
$\Rightarrow \langle H,T\rangle \models_\star \mathsf{Forget}_\star(\psi \wedge \varphi, V)$. □

Theorem 8 (Horn expressibility) *Let Π be a Horn logic program and $V \subseteq \mathcal{A}$. There is a Horn logic program Π' such that $\mathsf{Forget}_\star(\Pi, V) \equiv_\star \Pi'$.*

Proof: In terms of Proposition 2, it suffices to prove for $\star = \textsc{ht}$. Let $\mathcal{M} = \mathsf{Mod}_{\textsc{ht}}(\Pi)_{\dagger V}$. By Proposition 6, it is sufficient to show that $\mathcal{M}$ satisfies conditions (5) and (12).

We first prove that $\mathcal{M}$ satisfies (5). For each HT-interpretation $\langle X,Y\rangle \in \mathcal{M}$, we have that $X \subseteq Y$, and there exists $\langle H,T\rangle \in \mathsf{Mod}_{\textsc{ht}}(\Pi)$ such that $\langle X,Y\rangle \sim_V \langle H,T\rangle$. Note that Π is positive, which shows that $\langle H,H\rangle$ and $\langle T,T\rangle$ are HT-models of Π by Lemma 2. Thus $\langle X,X\rangle \in \mathcal{M}$ and $\langle Y,Y\rangle \in \mathcal{M}$ due to $X \sim_V H$ and $T \sim_V Y$. On the other hand, suppose $\langle X,X\rangle \in \mathcal{M}$, $\langle Y,Y\rangle \in \mathcal{M}$ and $X \subseteq Y$. There exist two HT-models $\langle H',T'\rangle$ and $\langle H'',T''\rangle$ of Π such that $\langle H',T'\rangle \sim_V \langle X,X\rangle$ and $\langle H'',T''\rangle \sim_V \langle Y,Y\rangle$. By Lemma 2, we have $H' \models \Pi$, $T' \models \Pi$, $H'' \models \Pi$ and $T'' \models \Pi$. Since models of Horn theories are closed under set intersection (Alfred, 1951), $H' \cap H'' \models \Pi$. By Lemma 2 again, we have $\langle H' \cap H'', T''\rangle \models_{\textsc{ht}} \Pi$. By Lemma 4, $H' \cap H'' \sim_V X \cap Y (= X)$. Thus $\langle H' \cap H'', T''\rangle \sim_V \langle X,Y\rangle$. It follows $\langle X,Y\rangle \in \mathcal{M}$.

Now we show that $\mathcal{M}$ satisfies (12). Suppose $\langle X,Y\rangle$ and $\langle H,T\rangle$ are two HT-interpretations in $\mathcal{M}$. It follows that there are two HT-models $\langle X',Y'\rangle$ and $\langle H',T'\rangle$ of Π such that $\langle X',Y'\rangle \sim_V \langle X,Y\rangle$ and $\langle H',T'\rangle \sim_V \langle H,T\rangle$. Since Π is Horn, we have that $\langle H' \cap X', T' \cap Y'\rangle \models_{\textsc{ht}} \Pi$ by Proposition 6. By Lemma 4, we have $H' \cap X' \sim_V H \cap X$ and $Y' \cap T' \sim_V Y \cap T$. It implies $\langle H' \cap X', T' \cap Y'\rangle \sim_V \langle X \cap H, Y \cap T\rangle$, thus $\langle X \cap H, Y \cap T\rangle \in \mathcal{M}$. □

Proposition 11 *Let Π be a disjunctive logic program, $V \subseteq \mathcal{A}$. We have that $\mathsf{Forget}_\star(\Pi, V)$ is expressible in disjunctive logic programs if and only if,*

$$\langle H_1,T_1\rangle \models_\star \Pi, \langle T_2,T_2\rangle \models_\star \Pi \textit{ and } T_1 \subseteq T_2 \Rightarrow \exists\langle H_3,T_3\rangle \models_\star \Pi \textit{ such that } \langle H_3,T_3\rangle \sim_V \langle H_1,T_2\rangle.$$

Proof: By Proposition 2, it suffices to prove $\star = \textsc{ht}$. Let $\Pi' \equiv_{\textsc{ht}} \mathsf{Forget}_{\textsc{ht}}(\Pi, V)$. The direction from left to right is obvious. We show the other direction.

Suppose that Π' is not expressible in disjunctive logic programs. There exists $\langle X,Y\rangle \models_{\textsc{ht}} \Pi'$, $Y \subseteq Y'$ and $\langle Y',Y'\rangle \models_{\textsc{ht}} \Pi'$ such that $\langle X,Y'\rangle \not\models_{\textsc{ht}} \Pi'$. It follows that, for each $\langle H_1,T_1\rangle \models_{\textsc{ht}} \Pi$ and $\langle T_2,T_2\rangle \models_{\textsc{ht}} \Pi$ such that $\langle H_1,T_1\rangle \sim_V \langle X,Y\rangle$, $T_2 \sim_V Y'$ and $T_1 \subseteq T_2$, there exists no $\langle H_3,T_3\rangle \models_{\textsc{ht}} \Pi$ such that $\langle H_3,T_3\rangle \sim_V \langle H_1,T_2\rangle$, viz. $\langle H_3,T_3\rangle \sim_V \langle X,Y'\rangle$ by $\langle X,Y'\rangle \sim_V \langle H_1,T_2\rangle$, a contradiction. □

Proposition 12 *Let Π be a normal logic program, $V \subseteq \mathcal{A}$. Then $\mathsf{Forget}_\star(\Pi, V)$ is expressible in normal logic programs if and only if, in addition to condition (21), the following condition holds,*

$$\begin{aligned} &\langle H_1,T_1\rangle \models_\star \Pi, \langle H_2,T_2\rangle \models_\star \Pi \textit{ and } T_1 \sim_V T_2 \\ \Rightarrow\ &\exists\langle H_3,T_3\rangle \models_\star \Pi \textit{ such that } H_3 \sim_V H_1 \cap H_2 \textit{ and } (T_3 \sim_V T_1 \textit{ or } T_3 \sim_V T_2). \end{aligned} \tag{21}$$

Proof: By Proposition 2, it suffices to prove $\star = \textsc{ht}$. Let $\Pi' \equiv_{\textsc{ht}} \mathsf{Forget}_{\textsc{ht}}(\Pi, V)$. The direction from left to right is easy. We consider the other direction in what follows.

In terms of Proposition 11 and Corollary 3, it is sufficient to show that, for each $\langle X,Y\rangle \models_{\textsc{ht}} \Pi'$ and $\langle X',Y\rangle \models_{\textsc{ht}} \Pi'$, $\langle X \cap X',Y\rangle \models_{\textsc{ht}} \Pi'$ according to Corollary 3. Suppose that $\langle X,Y\rangle$ and

$\langle X', Y\rangle$ are two HT-models of Π'. There are two HT-models $\langle H_1, T_1\rangle$ and $\langle H_2, T_2\rangle$ of Π such that $\langle X, Y\rangle \sim_V \langle H_1, T_1\rangle$ and $\langle X', Y\rangle \sim_V \langle H_2, T_2\rangle$. It follows that $T_1 \sim_V T_2$ and, by condition (21), there exists an HT-model $\langle H_3, T_3\rangle$ of Π satisfying either $\langle H_3, T_3\rangle \sim_V \langle H_1 \cap H_2, T_1\rangle$ or $\langle H_3, T_3\rangle \sim_V \langle H_1 \cap H_2, T_2\rangle$, which shows $\langle H_3, T_3\rangle \sim_V \langle X \cap X', Y\rangle$, hence $\langle X \cap X', Y\rangle \models_{\mathrm{HT}} \Pi'$. $\Box$

Theorem 10 (Representation theorem) *Let ψ and φ be two formulas and V a set of atoms. Then the following statements are equivalent:*

(i) $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$.

(ii) $\varphi \equiv_\star \{\varphi' \mid \psi \models_\star \varphi'$ *and* $\mathsf{IR}_\star(\varphi', V)\}$.

(iii) Postulates (W), (PP), (NP) *and* (IR) *hold.*

Proof: Let $\Sigma_\star = \{\xi \mid \psi \models_\star \xi$ and $\mathsf{IR}_\star(\xi, V)\}$. It is evident that $\mathsf{IR}_\star(\Sigma_\star, V)$.

The equivalence between (i) and (ii) follows from Corollary 9. (ii) obviously implies (iii). It suffices to show (iii) $\Rightarrow$ (ii).

By Positive Persistence, we have $\varphi \models_\star \xi$ for each $\xi \in \Sigma_\star$, from which follows $\mathsf{Mod}_\star(\varphi) \subseteq \mathsf{Mod}_\star(\Sigma_\star)$. On the other hand, by (W) $\psi \models_\star \varphi$ and (IR) $\mathsf{IR}_\star(\varphi, V)$, it follows $\varphi \in \Sigma_\star$. Thus $\mathsf{Mod}_\star(\Sigma_\star) \subseteq \mathsf{Mod}_\star(\varphi)$. Thus $\varphi \equiv_\star \Sigma_\star$. $\Box$

Proposition 13 *Let φ, φ', ψ be formulas and $V \subseteq \mathcal{A}$ such that $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$ and $\varphi' \equiv \mathsf{Forget}(\psi, V)$. Then*

(i) $\varphi \equiv \varphi'$.

(ii) $\varphi' \models_\star \varphi$.

Proof: (i) $T \models \varphi$
iff $\langle T, T\rangle \models_\star \varphi$ by (i) of Proposition 3
iff $\langle T, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ since $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$
iff $\exists \langle Y, Y\rangle \models_\star \psi$ such that $\langle T, T\rangle \sim_V \langle Y, Y\rangle$ by Definition 8
iff $\exists Y \models \psi$ such that $T \sim_V Y$ by (i) of Proposition 3
iff $T \models \mathsf{Forget}(\psi, V)$ by Corollary 5
iff $T \models \varphi'$ since $\varphi' \equiv \mathsf{Forget}(\psi, V)$.

(ii) $\langle H, T\rangle \models_\star \varphi'$
$\Rightarrow T \models \varphi'$ by (i) of Proposition 3
$\Rightarrow T \models \mathsf{Forget}(\neg\psi, V)$ since $\varphi' \equiv \mathsf{Forget}(\neg\psi, V)$
$\Rightarrow \exists Y \models \neg\psi$ such that $Y \sim_V T$ by Corollary 5
$\Rightarrow \exists \langle H \setminus V, Y\rangle \models_\star \neg\psi$ such that $Y \sim_V T$ by (ii) of Proposition 3
$\Rightarrow \langle H, T\rangle \models_\star \mathsf{Forget}_\star(\neg\psi, V)$ due to $\langle H \setminus V, Y\rangle \sim_V \langle H, T\rangle$ and Definition 8
$\Rightarrow \langle H, T\rangle \models_\star \varphi$ due to $\mathsf{Forget}_\star(\neg\psi, V) \equiv_\star \varphi$. $\Box$

Proposition 14 *Let Π and Π' be two Horn logic programs, and V a set of atoms such that $\Pi' \equiv \mathsf{Forget}(\Pi, V)$. Then $\Pi' \equiv_\star \mathsf{Forget}_\star(\Pi, V)$.*

Proof: By Proposition 2, it suffices to show $\star = \text{HT}$.
($\Rightarrow$) $\langle H', T' \rangle \models_{\text{HT}} \Pi'$
$\Rightarrow H' \models \Pi'$ and $T' \models \Pi'$ by Lemma 2
$\Rightarrow \exists H, T$ such that $H \models \Pi$, $T \models \Pi$, $H \sim_V H'$ and $T \sim_V T'$ by $\Pi' \equiv \mathsf{Forget}(\Pi, V)$
$\Rightarrow \exists H, T$ such that $H \cap T \models \Pi$, $T \models \Pi$, $H \cap T \sim_V H'$ and $T \sim_V T'$
$\Rightarrow \exists H, T$ such that $\langle H \cap T, T \rangle \models_{\text{HT}} \Pi$ and $\langle H \cap T, T \rangle \sim_V \langle H', T' \rangle$
$\Rightarrow \langle H', T' \rangle \models_{\text{HT}} \mathsf{Forget}_{\text{HT}}(\Pi, V)$.
($\Leftarrow$) $\langle H', T' \rangle \models_{\text{HT}} \mathsf{Forget}_{\text{HT}}(\Pi, V)$
$\Rightarrow \exists \langle H, T \rangle \models_{\text{HT}} \Pi$ such that $\langle H', T' \rangle \sim_V \langle H, T \rangle$
$\Rightarrow \exists H \subseteq T$ such that $H \models \Pi$, $T \models \Pi$ and $\langle H', T' \rangle \sim_V \langle H, T \rangle$ by Lemma 2
$\Rightarrow H' \models \mathsf{Forget}(\Pi, V)$ and $T' \models \mathsf{Forget}(\Pi, V)$
$\Rightarrow H' \models \Pi'$ and $T' \models \Pi'$ due to $\Pi' \equiv \mathsf{Forget}(\Pi, V)$
$\Rightarrow \langle H', T' \rangle \models_{\text{HT}} \Pi'$. $\square$

Proposition 15 *Let ψ and φ be two formulas and V a set of atoms.*

(i) $\varphi \equiv \mathsf{Forget}(\psi, V)$ *iff* $\neg\neg\varphi \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$.

(ii) $\mathsf{Forget}(\varphi, V) \equiv \mathsf{Forget}(\psi, V)$ *iff* $\mathsf{Forget}_\star(\neg\neg\varphi, V) \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$.

Proof: (i) ($\Rightarrow$) $\langle H, T \rangle \models_\star \neg\neg\varphi$
iff $T \models \neg\neg\varphi$, i.e. $T \models \varphi$ by (ii) of Proposition 3
iff $T \models \mathsf{Forget}(\psi, V)$ since $\varphi \equiv \mathsf{Forget}(\psi, V)$
iff $\exists Y \models \psi$ i.e. $Y \models \neg\neg\psi$ such that $Y \sim_V T$ by Corollary 5
iff $\langle H \setminus V, Y \rangle \models_\star \neg\neg\psi$ ($H \setminus V \subseteq T \setminus V = Y \setminus V$) by (ii) of Proposition 3
iff $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$ by Definition 8.
($\Leftarrow$) $T \models \varphi$ i.e. $T \models \neg\neg\varphi$
iff $\langle H, T \rangle \models_\star \neg\neg\varphi$ by (ii) of Proposition 3
iff $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$ for $H \subseteq T$ since $\neg\neg\varphi \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$
iff $\exists \langle X, Y \rangle \models_\star \neg\neg\psi$ such that $\langle H, T \rangle \sim_V \langle X, Y \rangle$ by Definition 8
iff $\exists Y \models \neg\neg\psi$ such that $Y \sim_V T$ by (ii) or Proposition 3
iff $T \models \mathsf{Forget}(\neg\neg\psi, V)$ by Corollary 5.
(ii) ($\Rightarrow$) $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\neg\neg\varphi, V)$
iff $\exists \langle X, Y \rangle \models_\star \neg\neg\varphi$ such that $\langle X, Y \rangle \sim_V \langle H, T \rangle$ by Definition 8
iff $\exists Y \models \neg\neg\varphi$ i.e. $Y \models \varphi$ such that $Y \sim_V T$ by (ii) of Proposition 3
iff $T \models \mathsf{Forget}(\varphi, V)$ by Corollary 5
iff $T \models \mathsf{Forget}(\psi, V)$ since $\mathsf{Forget}(\varphi, V) \equiv \mathsf{Forget}(\psi, V)$
iff $\exists Y' \models \psi$ i.e. $Y' \models \neg\neg\psi$ such that $Y' \sim_V T$ by Definition 8
iff $\exists \langle X \setminus V, Y' \rangle \models_\star \neg\neg\psi$ by (ii) of Proposition 3 ($X \setminus V \subseteq Y \setminus V = Y' \setminus V$)
iff $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$ by $\langle H, T \rangle \sim_V \langle X \setminus V, Y' \rangle$ and Definition 8.
($\Leftarrow$) $T \models \mathsf{Forget}(\varphi, V)$
iff $\exists Y \models \varphi$ i.e. $Y \models \neg\neg\varphi$ such that $Y \sim_V T$ by Corollary 5
iff $\exists \langle X, Y \rangle \models_\star \neg\neg\varphi$ such that $Y \sim_V T$ by (ii) of Proposition 3
iff $\langle X \setminus V, T \rangle \models_\star \mathsf{Forget}_\star(\neg\neg\varphi, V)$ $\langle X \setminus V, T \rangle \sim_V \langle X, Y \rangle$ and by Definition 8
iff $\langle X \setminus V, T \rangle \models_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$ since $\mathsf{Forget}_\star(\neg\neg\varphi, V) \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$
iff $\exists \langle X', Y' \rangle \models_\star \neg\neg\psi$ such that $\langle X \setminus V, T \rangle \sim_V \langle X', Y' \rangle$ by Definition 8

iff $\exists Y' \models \neg\neg\psi$ i.e. $Y' \models \psi$ such that $T \sim_V Y'$ by (ii) of Proposition 3
iff $T \models \mathsf{Forget}(\psi, V)$ by Corollary 5. $\square$

Theorem 12 ($\star$-forgetting vs propositional forgetting) *Let ψ and φ be two formulas of $\mathcal{L}_\mathcal{A}$ and $V \subseteq \mathcal{A}$. Then*

$$\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V) \textit{ iff } \Delta(\mathcal{A}) \models \tau_\star[\varphi] \leftrightarrow \mathsf{Forget}(\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}, V \cup V').$$

Proof: ($\Rightarrow$) Let $M = M_\mathcal{A} \cup M^*_{\mathcal{A}'}$ be a model of $\Delta(\mathcal{A})$.
$M \models \Delta(\mathcal{A}) \cup \{\tau_\star[\varphi]\}$
iff $\langle M_\mathcal{A}, M^*_{\mathcal{A}'}\rangle \models_\star \varphi$ by Proposition 8
iff $\langle M_\mathcal{A}, M^*_{\mathcal{A}'}\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ since $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$
iff $\exists\langle H, T\rangle \models_\star \psi$ such that $\langle H, T\rangle \sim_V \langle M_\mathcal{A}, M^*_{\mathcal{A}'}\rangle$ by Definition 8
iff $\exists\langle H, T\rangle \models_\star \psi$ such that $H \sim_V M_\mathcal{A}$ and $T \sim_V M^*_{\mathcal{A}'}$
iff $\exists H \cup T' \models \Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}$ and $H \sim_V M_\mathcal{A}$ and $T' \sim_{V'} M_{\mathcal{A}'}$ by Proposition 8
iff $\exists H \cup T' \models \Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}$ and $H \cup T' \sim_{V \cup V'} M_\mathcal{A} \cup M_{\mathcal{A}'}$ by Lemma 4
iff $M_\mathcal{A} \cup M_{\mathcal{A}'} \models \mathsf{Forget}(\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}, V \cup V')$ by Definition 8
iff $M \models \mathsf{Forget}(\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}, V \cup V')$.
($\Leftarrow$) $\langle X, Y\rangle \models_\star \varphi$
iff $X \cup Y' \models \Delta(\mathcal{A}) \cup \{\tau_\star[\varphi]\}$ by Proposition 8
iff $X \cup Y' \models \Delta(\mathcal{A}) \cup \mathsf{Forget}(\Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}, V \cup V')$
iff $\exists M \models \Delta(\mathcal{A}) \cup \{\tau_\star[\psi]\}$ such that $M \sim_{V \cup V'} X \cup Y'$
iff $\exists\langle M_\mathcal{A}, M^*_{\mathcal{A}'}\rangle \models_\star \psi$ such that $M_\mathcal{A} \cup M^*_{\mathcal{A}'} \sim_V X \cup Y$ by Proposition 8
iff $\langle X, Y\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ due to $\langle X, Y\rangle \sim_V \langle M_\mathcal{A}, M^*_{\mathcal{A}'}\rangle$ by Definition 8. $\square$

Proposition 16 *Let ψ and φ be two formulas of $\mathcal{L}_\mathcal{A}$ and V a set of atoms. Then* $\mathsf{Forget}_\star(\psi, V) \equiv_\star \mathsf{Forget}_\star(\varphi, V)$ *iff the following condition holds:*

$$\mathsf{Forget}(\{\tau_\star[\psi]\} \cup \Delta(\mathcal{A}), V \cup V') \equiv \mathsf{Forget}(\{\tau_\star[\varphi]\} \cup \Delta(\mathcal{A}), V \cup V').$$

Proof: ($\Rightarrow$) We show $\mathsf{Forget}(\{\tau_\star[\psi] \cup \Delta(\mathcal{A}), V \cup V') \models \mathsf{Forget}(\{\tau_\star[\varphi] \cup \Delta(\mathcal{A}), V \cup V')$. The other side can be similarly proved.
$M \models \mathsf{Forget}(\{\tau_\star[\psi]\} \cup \Delta(\mathcal{A}), V \cup V')$
$\Rightarrow \exists N \subseteq \mathcal{A} \cup \mathcal{A}'$ such that $N \sim_{V \cup V'} M$ and $N \models \{\tau_\star[\psi]\} \cup \Delta(\mathcal{A})$
$\Rightarrow \exists\langle X, Y\rangle \models_\star \psi$ with $N = X \cup Y'$ by Proposition 8
$\Rightarrow \langle X, Y\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ by (iii) of Proposition 10
$\Rightarrow \langle X, Y\rangle \models_\star \mathsf{Forget}_\star(\varphi, V)$ as $\mathsf{Forget}_\star(\psi, V) \equiv_\star \mathsf{Forget}_\star(\varphi, V)$
$\Rightarrow \exists\langle H, T\rangle \models_\star \varphi$ such that $\langle H, T\rangle \sim_V \langle X, Y\rangle$ by Definition 8
$\Rightarrow H \cup T' \models \tau_\star[\varphi] \cup \Delta(\mathcal{A})$ by Proposition 8
$\Rightarrow X \cup Y' \models \mathsf{Forget}(\{\tau_\star[\varphi]\} \cup \Delta(\mathcal{A}), V \cup V')$ as $H \cup T' \sim_{V \cup V'} X \cup Y'$
$\Rightarrow M \models \mathsf{Forget}(\{\tau_\star[\varphi]\} \cup \Delta(\mathcal{A}), V \cup V')$ by $M \sim_{V \cup V'} X \cup Y' (= N)$.
($\Leftarrow$) We show $\mathsf{Forget}_\star(\psi, V) \models_\star \mathsf{Forget}_\star(\varphi, V)$. The other side is similar.
$\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$
$\Rightarrow \exists\langle X, Y\rangle \models_\star \psi$ such that $\langle H, T\rangle \sim_V \langle X, Y\rangle)$ by Definition 8
$\Rightarrow X \cup Y' \models \{\tau_\star[\psi]\} \cup \Delta(\mathcal{A})$ by Proposition 8
$\Rightarrow X \cup Y' \models \mathsf{Forget}(\{\tau_\star[\psi]\} \cup \Delta(\mathcal{A}), V \cup V')$

$\Rightarrow X \cup Y' \models \mathsf{Forget}(\{\tau_\star[\varphi]\} \cup \Delta(\mathcal{A}), V \cup V')$
$\Rightarrow \exists H_1 \cup T_1' \models \{\tau_\star[\varphi]\} \cup \Delta(\mathcal{A})$ such that $H_1 \cup T_1' \sim_{V \cup V'} X \cup Y'$
$\Rightarrow \langle H_1, T_1 \rangle \models_\star \varphi$ by Proposition 8
$\Rightarrow \langle X, Y \rangle \models_\star \mathsf{Forget}_\star(\varphi, V)$ as $\langle X, Y \rangle \sim_V \langle H_1, T_1 \rangle$ by Definition 8
$\Rightarrow \langle H, T \rangle \models_\star \mathsf{Forget}_\star(\varphi, V)$ as $\langle X, Y \rangle \sim_V \langle H, T \rangle$. □

Theorem 14 *Let ψ and φ be two formulas and V a set of atoms.*

(i) The problem of deciding if $\psi \equiv_\star \mathsf{Forget}_\star(\psi, V)$ is co-NP*-complete.*

(ii) The problem of deciding if $\mathsf{Forget}_\star(\varphi, V) \equiv_\star \mathsf{Forget}_\star(\psi, V)$ is Π_2^P-complete.

(iii) The problem of deciding if $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$ is Π_2^P-complete.

Proof: (i) Membership. Recall that $\psi \models_\star \mathsf{Forget}_\star(\psi, V)$ by (iii) of Proposition 10. We have
$\psi \not\equiv_\star \mathsf{Forget}_\star(\psi, V)$
iff $\mathsf{Forget}_\star(\psi, V) \not\models_\star \psi$
iff $\exists \langle X, Y \rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ and $\langle X, Y \rangle \not\models_\star \psi$
iff $\exists \langle H, T \rangle \models_\star \psi$ such that $\langle H, T \rangle \sim_V \langle X, Y \rangle$ and $\langle X, Y \rangle \not\models_\star \psi$.

Since both guessing $\langle H, T \rangle$, $\langle X, Y \rangle$ and checking the $\star$-satisfiability can be done in polynomial time in the size of ψ and V. Thus the complement of $\psi \not\equiv_\star \mathsf{Forget}_\star(\psi, V)$, i.e. $\psi \equiv_\star \mathsf{Forget}_\star(\psi, V)$, is in co-NP.

The hardness follows from the fact that, by (i) of Proposition 15, $\neg\neg\psi \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$ iff $\psi \equiv \mathsf{Forget}(\psi, V)$, which is co-NP-complete (cf., see Lang et al., 2003, Prop. 10).

(ii) Membership. If $\mathsf{Forget}_\star(\varphi, V) \not\equiv_\star \mathsf{Forget}_\star(\psi, V)$ then there exists a $\star$-interpretation $\langle H, T \rangle$ such that either

(a) $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\varphi, V)$ and $\langle H, T \rangle \not\models_\star \mathsf{Forget}_\star(\psi, V)$, or

(b) $\langle H, T \rangle \not\models_\star \mathsf{Forget}_\star(\varphi, V)$ and $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\psi, V)$.

On the one hand, to guess a $\star$-interpretation $\langle H, T \rangle$ is feasible by a nondeterministic Turing machine. On the other hand, checking if $\langle H, T \rangle \models_\star \varphi$ is feasible by a deterministic Turing machine; and $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\varphi, V)$ iff there exists $\langle X, Y \rangle \models_\star \varphi$ such that $\langle X, Y \rangle \sim_V \langle H, T \rangle$. Thus checking the conditions (a) and (b) can be done in polynomial time in the size of ψ and φ by calling a nondeterministic Turing machine. Thus the problem is in Π_2^P.

Note that, by (ii) of Proposition 15, $\mathsf{Forget}_\star(\neg\neg\varphi, V) \equiv_\star \mathsf{Forget}_\star(\neg\neg\psi, V)$ iff $\mathsf{Forget}(\varphi, V) \equiv \mathsf{Forget}(\psi, V)$, which is Π_2^P-complete (cf., see Lang et al., 2003, Prop. 24). Thus the hardness follows.

(iii) Membership. Note that $\varphi \not\equiv_\star \mathsf{Forget}_\star(\psi, V)$ iff there is a $\star$-interpretation $\langle H, T \rangle$ such that

- $\langle H, T \rangle \models_\star \varphi$ and $\langle H, T \rangle \not\models_\star \mathsf{Forget}_\star(\psi, V)$, or
- $\langle H, T \rangle \not\models_\star \varphi$ and $\langle H, T \rangle \models_\star \mathsf{Forget}_\star(\psi, V)$.

Similar to the case of (ii), the guessing and checking are in polynomial time in the size of φ, ψ and V by calling a nondeterministic Turing machine. Thus the problem is in Π_2^P.

Note that $\varphi \equiv_\star \mathsf{Forget}_\star(\psi, V)$ iff $\varphi \equiv_\star \mathsf{Forget}_\star(\varphi, V)$ and $\mathsf{Forget}_\star(\varphi, V) \equiv_\star \mathsf{Forget}_\star(\psi, V)$, the latter is Π_2^P-hard by (ii). Then the hardness follows. □

Proposition 17 *Let ψ and φ be two formulas and V a set of atoms.*

(i) The problem of deciding whether $\psi \models_\star \mathsf{Forget}_\star(\varphi, V)$ is Π_2^P-complete.

(ii) The problem of deciding whether $\mathsf{Forget}_\star(\psi, V) \models_\star \varphi$ is co-NP-*complete.*

Proof: (i) Membership. Recall that $\psi \not\models_\star \mathsf{Forget}_\star(\varphi, V)$ iff there exists a $\star$-model $\langle H, T\rangle$ of ψ such that $\langle H, T\rangle \not\models \mathsf{Forget}_\star(\varphi, V)$. As $\langle H, T\rangle \not\models \mathsf{Forget}_\star(\varphi, V)$ iff $\langle X, Y\rangle \not\models \varphi$ for every $\star$-interpretation $\langle X, Y\rangle$ such that $\langle X, Y\rangle \sim_V \langle H, T\rangle$. Such $\langle H, T\rangle$ can be guessed in polynomial time in the size of φ, ψ and V. Checking $\langle H, T\rangle \not\models \mathsf{Forget}_\star(\varphi, V)$ is possible in polynomial time in the size of φ, ψ and V by calling a nondeterministic Turing machine. Thus the original problem is in Π_2^p.

Hardness. It follows from the following fact:
$\top \models_\star \mathsf{Forget}_\star(\neg\neg\varphi, V)$
iff $\top \equiv_\star \mathsf{Forget}_\star(\neg\neg\varphi, V)$
iff $\top \equiv \mathsf{Forget}(\varphi, V)$ by (i) of Proposition 15 ($\neg\neg\top \equiv_\star \top$)
iff the QBF $\forall\overline{V}\exists V\varphi$ is valid, which is Π_2^P-complete (Papadimitriou, 1994).

(ii) Membership. Note that
$\mathsf{Forget}_\star(\psi, V) \not\models_\star \varphi$
iff $\exists\langle H, T\rangle \models_\star \mathsf{Forget}_\star(\psi, V)$ such that $\langle H, T\rangle \not\models \varphi$
iff $\exists\langle X, Y\rangle \models_\star \psi$ such that $\langle X, Y\rangle \sim_V \langle H, T\rangle$ and $\langle H, T\rangle \not\models \varphi$.
Since the guessing and checking are both polynomial in the size of ψ, φ and V, the original problem is in co-NP.

Hardness follows from the fact that
$\mathsf{Forget}_\star(\psi, V) \models_\star \bot$
iff $\psi \models_\star \bot$ by (ii) of Proposition 10
iff ψ has no $\star$-model, which is co-NP-complete by Proposition 9. □

Appendix D. Forgetting Operators $\mathsf{F_W}$ and $\mathsf{F_S}$

Wong proposed six postulates and argued that the postulates should to respected by all forgetting operators in disjunctive logic programs under strong equivalence:

(F-1) If $\Pi \models_{\text{HT}} \Sigma$ then $F(\Pi, a) \models_{\text{HT}} F(\Sigma, a)$;

(F-2) If a does not appear in Δ, then $F(\{r\} \cup \Delta, a) \equiv_{\text{HT}} F(\{r\}, a) \cup \Delta$;

(F-3) $F(\Pi, a)$ does not contain any atoms not in Π;

(F-4) If $F(\Pi, a) \models_{\text{HT}} r$ then $F(\{s\}, a) \models_{\text{HT}} r$ for some $s \in \text{Cn}(\Pi)$;

(F-5) If $F(\Pi, a) \models_{\text{HT}} (A \leftarrow B, not\, C)$, then $\Pi \models_{\text{HT}} (A \leftarrow B, not\, C, not\, a)$;

(F-6) $F(F(\Pi, a), b) \equiv_{\text{HT}} F(F(\Pi, b), a)$

where F is a forgetting operator, Π, Σ and Δ are disjunctive logic programs, a and b are atoms, r is a disjunctive rule, and

$$\text{Cn}(\Pi) = \{r \mid r \text{ is a disjunctive rule such that } \Pi \models_{\text{HT}} r \text{ and } var(r) \subseteq var(\Pi)\}.$$

where $var(\alpha)$ is the set of atoms occurring in α.

Accordingly, he proposed two forgetting operators $\mathsf{F_S}$ and $\mathsf{F_W}$: the result of forgetting an atom a from a disjunctive logic program Π is defined by the below procedure:

(1) Let $\Pi_1 = \mathrm{Cn}(\Pi)$.

(2) Form Π_1, remove rules of the form $(A \leftarrow B, a, not\, C)$, replace each rule of the form $(A \cup \{a\} \leftarrow B, not\, C, not\, a)$ with $(A \leftarrow B, not\, C, not\, a)$. Let the resulting logic program be Π_2.

(3) Replace or remove each rule in Π_2, of the form $(A \leftarrow B, not\, C, not\, a)$ or $(A \cup \{a\} \leftarrow B, not\, C)$ according to the following table:

	$A \leftarrow B, not\, C, not\, a$	$A \cup \{a\} \leftarrow B, not\, C$
S	(remove)	(remove)
W	$A \leftarrow B, not\, C$	$A \leftarrow B, not\, C$

Let Π_3 be the resulting logic program.

The logic program Π_3 is the result of forgetting p from Π.

References

Alfred, H. (1951). On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, *16*(1), 14–21.

Bobrow, D. G., Subramanian, D., Greiner, R., & Pearl, J. (Eds.). (1997). *Special issue on relevance 97 (1-2)*. Artificial Intelligence Journal.

Cabalar, P., & Ferraris, P. (2007). Propositional theories are strongly equivalent to logic programs. *Theory and Practice of Logic Programming*, *7*(6), 745–759.

Delgrande, J. P., Schaub, T., Tompits, H., & Woltran, S. (2013). A model-theoretic approach to belief change in answer set programming. *ACM Transactions on Computational Logic*, *14*(2), A:1–A:42.

Eiter, T., Fink, M., Tompits, H., & Woltran, S. (2004). On eliminating disjunctions in stable logic programming. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pp. 447–458, Whistler, Canada. AAAI Press.

Eiter, T., & Wang, K. (2008). Semantic forgetting in answer set programming. *Artificial Intelligence*, *172*(14), 1644–1672.

Faber, W., Pfeifer, G., & Leone, N. (2011). Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, *175*(1), 278–298.

Ferraris, P. (2005). Answer sets for propositional theories. In *Logic Programming and Nonmonotonic Reasoning, 8th International Conference*, Vol. 3662 of *Lecture Notes in Computer Science*, pp. 119–131, Diamante, Italy. Springer.

Ferraris, P., Lee, J., & Lifschitz, V. (2011). Stable models and circumscription. *Artificial Intelligence*, *175*(1), 236–263.

Ferraris, P., & Lifschitz, V. (2005). Mathematical foundations of answer set programming. In Artëmov, S. N., Barringer, H., d'Avila Garcez, A. S., Lamb, L. C., & Woods, J. (Eds.), *We Will Show Them! Essays in Honour of Dov Gabbay*, Vol. 1, pp. 615–664. College Publications.

Gabbay, D. M., Pearce, D., & Valverde, A. (2011). Interpolable formulas in equilibrium logic and answer set programming. *Journal of Artificial Intelligence Research*, *42*, 917–943.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pp. 1070–1080, Seattle, Washington. MIT Press.

Goranko, V., & Otto, M. (2007). *Handbook of Modal Logic*, Vol. 3, chap. 5 Model Theory Of Modal Logic, pp. 249–329. Elsevier.

Jongh, D. D., & Hendriks, L. (2003). Characterization of strongly equivalent logic programs in intermediate logics. *Theory and Practice of Logic Programming*, *3*(3), 259–270.

Kontchakov, R., Wolter, F., & Zakharyaschev, M. (2008). Can you tell the difference between dl-lite ontologies?. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008*, pp. 285–295, Sydney, Australia. AAAI Press.

Lang, J., Liberatore, P., & Marquis, P. (2003). Propositional independence: Formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, *18*, 391–443.

Lang, J., & Marquis, P. (2010). Reasoning under inconsistency: A forgetting-based approach. *Artificial Intelligence*, *174*(12-13), 799–823.

Lifschitz, V., Pearce, D., & Valverde, A. (2001). Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, *2*(4), 526–541.

Lifschitz, V., Tang, L. R., & Turner, H. (1999). Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, *25*(3-4), 369–389.

Lin, F. (2001). On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, *128*(1-2), 143–159.

Lin, F. (2002). Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02)*, pp. 170–176, Toulouse, France. Morgan Kaufmann.

Lin, F., & Chen, Y. (2007). Discovering classes of strongly equivalent logic programs. *Journal of Artificial Intelligence Research*, *28*, 431–451.

Lin, F., & Reiter, R. (1994). Forget it!. In *In Proceedings of the AAAI Fall Symposium on Relevance*, pp. 154–159.

Lin, F., & Zhou, Y. (2011). From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence*, *175*(1), 264–277.

Liu, Y., & Wen, X. (2011). On the progression of knowledge in the situation calculus. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 976–982, Barcelona, Catalonia, Spain. IJCAI/AAAI.

Lutz, C., & Wolter, F. (2011). Foundations for uniform interpolation and forgetting in expressive description logics. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 989–995, Barcelona, Catalonia, Spain. IJCAI/AAAI.

Osorio, M., & Cuevas, V. (2007). Updates in answer set programming: An approach based on basic structural properties. *TPLP*, *7*(4), 451–479.

Osorio, M., & Zacarias, F. (2004). On updates of logic programs: A properties-based approach. In Seipel, D., & Torres, J. M. T. (Eds.), *FoIKS*, Vol. 2942 of *Lecture Notes in Computer Science*, pp. 231–241. Springer.

Packer, H. S., Gibbins, N., & Jennings, N. R. (2011). An on-line algorithm for semantic forgetting. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 2704–2709, Barcelona, Catalonia, Spain. IJCAI/AAAI.

Papadimitriou, C. H. (1994). *Computational complexity*. Addison Wesley.

Pearce, D., Tompits, H., & Woltran, S. (2001). Encodings for equilibrium logic and logic programs with nested expressions. In *Proceedings of the10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving*, pp. 306–320, London, UK. Springer-Verlag.

Pearce, D., Tompits, H., & Woltran, S. (2009). Characterising equilibrium logic and nested logic programs: Reductions and complexity. *Theory and Practice of Logic Programming*, *9*(5), 565–616.

Su, K., Sattar, A., Lv, G., & Zhang, Y. (2009). Variable forgetting in reasoning about knowledge. *Journal of Artificial Intelligence Research*, *35*, 677–716.

Truszczynski, M. (2010). Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence*, *174*(16-17), 1285–1306.

van Ditmarsch, H. P., Herzig, A., Lang, J., & Marquis, P. (2009). Introspective forgetting. *Synthese*, *169*(2), 405–423.

Visser, A. (1996). Uniform interpolation and layered bisimulation. In *Gödel'96*, pp. 139–164.

Wang, Y., Wang, K., & Zhang, M. (2013). Forgetting for answer set programs revisited. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 1162–1168, Beijing, China. IJCAI/AAAI.

Wang, Y., Zhang, Y., Zhou, Y., & Zhang, M. (2012). Forgetting in logic programs under strong equivalence. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference*, pp. 643–647, Rome, Italy. AAAI Press.

Wang, Z., Wang, K., Topor, R. W., & Pan, J. Z. (2010). Forgetting for knowledge bases in dl-lite. *Annuals of Mathematics and Artificial Intelligence*, *58*(1-2), 117–151.

Wong, K.-S. (2009). *Forgetting in Logic Programs*. Ph.D. thesis, The University of New South Wales.

Zhang, Y., & Foo, N. Y. (2006). Solving logic program conflict through strong and weak forgettings. *Artificial Intelligence*, *170*(8-9), 739–778.

Zhang, Y., & Zhou, Y. (2009). Knowledge forgetting: Properties and applications. *Artificial Intelligence*, *173*(16-17), 1525–1537.

Zhou, Y., & Zhang, Y. (2011). Bounded forgetting. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*, pp. 280–285, San Francisco, California, USA. AAAI Press.