You Can't Always Forget What You Want: on the limits of forgetting in answer set programming

Ricardo Gonçalves and **Matthias Knorr** and **João Leite**[1]

Abstract. Selectively forgetting information while preserving what matters the most is becoming an increasingly important issue in many areas, including in knowledge representation and reasoning. Depending on the application at hand, forgetting operators are defined to obey different sets of desirable properties. Of the myriad of desirable properties discussed in the context of forgetting in Answer Set Programming, *strong persistence*, which imposes certain conditions on the correspondence between the answer sets of the program pre- and post-forgetting, and a certain independence from non-forgotten atoms, seems to best capture its essence, and be desirable in general. However, it has remained an open problem whether it is always possible to forget a set of atoms from a program while obeying strong persistence. In this paper, after showing that it is not always possible to forget a set of atoms from a program while obeying this property, we move forward and precisely characterise what can and cannot be forgotten from a program, by presenting a necessary and sufficient criterion. This characterisation allows us to draw some important conclusions regarding the existence of forgetting operators for specific classes of logic programs, to characterise the class of forgetting operators that achieve the correct result whenever forgetting is possible, and investigate the related question of determining what we can forget from some specific logic program.

1 Introduction

In this paper, we show that it is not always possible to forget some set of atoms from an answer set program while preserving all existing relations between the atoms not to be forgotten, and investigate the *when*, *what*, and *how* related to adequately forgetting a set of atoms from an answer set program.

Whereas keeping memory of information and knowledge has always been at the heart of research in Knowledge Representation and Reasoning, with tight connections to broader areas such as Databases and Artificial Intelligence, we have recently observed a growing attention being devoted to the complementary problem of *forgetting*.

Forgetting – or variable elimination – is an operation that allows the removal of *middle* variables no longer deemed relevant. It is most useful when we wish to eliminate (temporary) variables introduced to represent auxiliary concepts, with the goal of restoring the declarative nature of some knowledge base, or just to simplify it. Furthermore, it is becoming increasingly necessary to properly deal with legal and privacy issues, including, for example, the implementation of court orders to eliminate certain pieces of illegal information. Recent applications of forgetting to cognitive robotics [28, 29, 33], resolving conflicts [20, 45, 12, 21], and ontology abstraction and comparison [42, 19, 17, 18], further witness its importance.

With its early roots in Boolean Algebra [24], forgetting has been extensively studied in the context of classical logic [3, 20, 22, 23, 30, 31, 43] and, more recently, in the context of logic programming, notably of Answer Set Programming (ASP). The non-monotonic rule-based nature of ASP called for the development of specific methods and techniques – just as it happened with other belief change operations such as revision and update, cf. [2, 8, 34, 35, 36, 6, 37] – resulting in a significant number of different forgetting operators [45, 12, 44, 40, 39, 16, 41, 7], obeying different sets of properties deemed desirable, and often defined for different classes of answer set programs. Such properties include the so-called *consequence persistence*, which requires that the answer sets of the result of forgetting correspond exactly to those of the original program, ignoring the atoms to be forgotten, or *existence* which requires that the result of forgetting belongs to the same class of programs admitted by the forgetting operator, so that the operator can be iterated, among many others. A complete picture of the existing forgetting operators and properties they obey can be found in a recent survey [15].

From observing the landscape of existing operators and properties, one can conclude that there cannot be a one-size-fits-all forgetting operator for ASP, but rather a family of operators, each obeying a specific set of properties. Furthermore, it is clear that not all properties bear the same relevance. Whereas some properties can be very important, such as *existence*, since it guarantees that we can use the same automated reasoners after forgetting, despite not being a property specific of forgetting operators, other properties are less important, sometimes perhaps even questionable, as discussed in [15].

There is nevertheless one property – *strong persistence* [16] – which seems to best capture the essence of forgetting in the context of ASP. The property of *strong persistence* essentially requires that all existing relations between the atoms not to be forgotten be preserved, captured by requiring that there be a correspondence between the answer sets of a program before and after forgetting a set of atoms, and that such correspondence be preserved in the presence of additional rules not containing the atoms to be forgotten. With a slight abuse of using notation that has not been introduced yet, an operator f is said to obey strong persistence if, for any program P and any set of atoms to be forgotten V, it holds that $\mathcal{AS}(\mathsf{f}(P,V) \cup R) = \mathcal{AS}(P \cup R)_{\|V}$, for all programs R not containing atoms in V, where $\mathsf{f}(P,V)$ denotes the result of forgetting V from P, $\mathcal{AS}(P)$ the answer sets of P, and $\mathcal{AS}(P)_{\|V}$ their restriction to atoms not in V.

Whereas it seems rather undisputed that *strong persistence* is a desirable property, it is not clear to what extent we can define operators that satisfy it. In [16], the authors propose an operator that obeys such

[1] NOVA LINCS, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, email: {rjrg|mkn|jleite}@fct.unl.pt.

property, but which is only defined for a restricted class of programs and can only be applied to forget a single atom from a program in a very limited range of situations.

In this paper, we investigate the limits of *forgetting* under *strong persistence*, by answering the following fundamental questions.

Can we always forget some set of atoms from an ASP while obeying strong persistence? This is perhaps the most fundamental open question that remained in [15], not being clear whether such operator does not exist, or simply no one had found it. As the reader can guess from the title of this paper, the answer to this question is negative: sometimes it is simply not possible to forget some set of atoms from a program, while maintaining the relevant relations between other atoms, since the atoms to be forgotten play a pivotal role. From this negative result, the following questions become central, which we will also address in this paper.

When can't we forget some set of atoms from an ASP while obeying strong persistence? We answer this by characterizing when a specific set of atoms cannot be forgotten from a specific program. We define a criterion (Ω) on a program and set of atoms which, when satisfied, implies that such atoms cannot be forgotten from the program.

When (and how) can we forget some set of atoms from an ASP while obeying strong persistence? We answer this by presenting a class of operators that satisfy strong persistence, among many other properties, and show that Ω is both sufficient and necessary to determine when some set of atoms can be forgotten from a program.

What can we forget from a specific ASP while obeying strong persistence? We answer this by providing a constructive definition of the sets of atoms that can be forgotten from a given program. While investigating the answer to this question, we uncover certain classes of programs from which we can always forget any single atom.

Throughout the paper, other relevant intermediate results are shown, and the main concepts illustrated with examples. After the next section with the background on ASP and on forgetting, the remainder of the paper is structured according to the questions above.

2 Forgetting in ASP

In this section, we recall the necessary notions on answer set programming and forgetting, following the presentation in [15].

We assume a propositional language $\mathcal{L}_\mathcal{A}$ over a *signature* $\mathcal{A}$, a finite set of propositional atoms[2]. The *formulas* of $\mathcal{L}_\mathcal{A}$ are inductively defined using connectives $\bot$, $\wedge$, $\vee$, and $\supset$:

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

where $p \in \mathcal{A}$. In addition, $\neg\varphi$ and $\top$ are resp. shortcuts for $\varphi \supset \bot$ and $\bot \supset \bot$. Given a finite set S of formulas, $\bigvee S$ and $\bigwedge S$ denote resp. the disjunction and conjunction of all formulas in S. In particular, $\bigvee \emptyset$ and $\bigwedge \emptyset$ stand for resp. $\bot$ and $\top$, and $\neg S$ and $\neg\neg S$ represent resp. $\{\neg\varphi \mid \varphi \in S\}$ and $\{\neg\neg\varphi \mid \varphi \in S\}$. Unless otherwise stated, we assume that the underlying signature for a particular formula φ is $\mathcal{A}(\varphi)$, the set of atoms appearing in φ.

HT-models Regarding the semantics of propositional formulas, we consider the monotonic logic here-and-there (HT) and equilibrium models [26]. An *HT-interpretation* is a pair $\langle H, T\rangle$ s.t. $H \subseteq T \subseteq \mathcal{A}$. The satisfiability relation in HT, denoted $\models_{\mathsf{HT}}$, is recursively defined as follows for $p \in \mathcal{A}$ and formulas φ and ψ:

- $\langle H, T\rangle \models_{\mathsf{HT}} p$ if $p \in H$;
- $\langle H, T\rangle \not\models_{\mathsf{HT}} \bot$;
- $\langle H, T\rangle \models_{\mathsf{HT}} \varphi \wedge \psi$ if $\langle H, T\rangle \models_{\mathsf{HT}} \varphi$ and $\langle H, T\rangle \models_{\mathsf{HT}} \psi$;
- $\langle H, T\rangle \models_{\mathsf{HT}} \varphi \vee \psi$ if $\langle H, T\rangle \models_{\mathsf{HT}} \varphi$ or $\langle H, T\rangle \models_{\mathsf{HT}} \psi$;
- $\langle H, T\rangle \models_{\mathsf{HT}} \varphi \supset \psi$ if both (i) $T \models \varphi \supset \psi$,[3] and (ii) $\langle H, T\rangle \models_{\mathsf{HT}} \varphi$ implies $\langle H, T\rangle \models_{\mathsf{HT}} \psi$.

An HT-interpretation $\langle H, T\rangle$ is an *HT-model* of a formula φ if $\langle H, T\rangle \models_{\mathsf{HT}} \varphi$. We denote by $\mathcal{HT}(\varphi)$ the set of *all HT-models of* φ. In particular, $\langle T, T\rangle \in \mathcal{HT}(\varphi)$ is an *equilibrium model* of φ if there is no $T' \subset T$ s.t. $\langle T', T\rangle \in \mathcal{HT}(\varphi)$.

Given two formulas φ and ψ, if $\mathcal{HT}(\varphi) \subseteq \mathcal{HT}(\psi)$, then φ *entails* ψ in HT, written $\varphi \models_{\mathsf{HT}} \psi$. Also, φ and ψ are *HT-equivalent*, written $\varphi \equiv_{\mathsf{HT}} \psi$, if $\mathcal{HT}(\varphi) = \mathcal{HT}(\psi)$.

The *V-exclusion* of a set of HT-interpretations $\mathcal{M}$, denoted $\mathcal{M}_{\|V}$, is $\{\langle X \setminus V, Y \setminus V\rangle \mid \langle X, Y\rangle \in \mathcal{M}\}$. Finally, determining if a formula has an HT-model is NP-complete [32].

Logic Programs An *(extended) logic program* P is a finite set of *(extended) rules*, i.e., formulas of the form

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

where all elements in $A = \{a_1, \ldots, a_k\}$, $B = \{b_1, \ldots, b_l\}$, $C = \{c_1, \ldots, c_m\}$, $D = \{d_1, \ldots, d_n\}$ are atoms.[4] Such rules r are also commonly written as

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, ..., b_l, not\ c_1, ..., not\ c_m, \\ not\ not\ d_1, ..., not\ not\ d_n\ , \tag{3}$$

and we will use both forms interchangeably. Given r, we distinguish its *head*, $head(r) = A$, and its *body*, $body(r) = B \cup \neg C \cup \neg\neg D$, representing a disjunction and a conjunction.

Any set of (propositional) formulas is HT-equivalent to an (extended) logic program [4], which is why we can focus solely on these.

This class of logic programs, $\mathcal{C}_e$, includes a number of special kinds of rules r: if $n = 0$, then we call r *disjunctive*; if, in addition, $k \leq 1$, then r is *normal*; if on top of that $m = 0$, then we call r *Horn*, and *fact* if also $l = 0$. The classes of *disjunctive*, *normal* and *Horn programs*, $\mathcal{C}_d$, $\mathcal{C}_n$, and $\mathcal{C}_H$, are defined resp. as a finite set of disjunctive, normal, and Horn rules. We have $\mathcal{C}_H \subset \mathcal{C}_n \subset \mathcal{C}_d \subset \mathcal{C}_e$.

We now recall the *answer set semantics* [14] for logic programs. Given a program P and a set I of atoms, the *reduct* P^I is defined as $P^I = \{A \leftarrow B : r \text{ of the form (3) in } P, C \cap I = \emptyset, D \subseteq I\}$. A set I' of atoms is a model of P^I if, for each $r \in P^I$, $I' \models B$ implies $I' \models A$. I is minimal in a set S, denoted by $I \in \mathcal{MIN}(S)$, if there is no $I' \in S$ s.t. $I' \subset I$. Then, I is an *answer set* of P iff I is a minimal model of P^I. Note that, for $\mathcal{C}_n$ and its subclasses, this minimal model is in fact unique. The set of all answer sets of P is denoted by $\mathcal{AS}(P)$. Note that, for $\mathcal{C}_d$ and its subclasses, all $I \in \mathcal{AS}(P)$ are pairwise incomparable. If P has an answer set, then P is *consistent*. Also, the *V-exclusion* of a set of answer sets $\mathcal{M}$, denoted $\mathcal{M}_{\|V}$, is $\{X \setminus V \mid X \in \mathcal{M}\}$. Two programs P_1, P_2 are *equivalent* if $\mathcal{AS}(P_1) = \mathcal{AS}(P_2)$ and *strongly equivalent* if $P_1 \equiv_{\mathsf{HT}} P_2$, i.e., if $\mathcal{AS}(P_1 \cup R) = \mathcal{AS}(P_2 \cup R)$ for any $R \in \mathcal{C}_e$. It is well-known that answer sets and equilibrium models coincide [26], but since the former notion is frequently used in the literature and arguably easier to use, we will mainly rely on it. Also, determining if program P has an answer set is Σ^p_2-complete, and NP-complete if P is normal [5].

Forgetting The principal idea of forgetting in ASP is to remove or hide certain atoms from a given program, while preserving its se-

[2] Often, the term propositional variable is used synonymously.

[3] $\models$ is the standard consequence relation from classical logic.

[4] Extended logic programs [27] are actually more expressive, but this form is sufficient here.

mantics for the remaining atoms. As the result, rather often, a representative up to some notion of equivalence between programs is considered. In this sense, many notions of forgetting for logic programs are defined semantically, i.e., they introduce a class of operators that satisfy a certain semantic characterization. Each single operator in such a class is then a concrete function that, given a program P and a non-empty set of atoms V to be forgotten, returns a unique program, the result of forgetting about V from P. Formally, given a class of logic programs $\mathcal{C}$ over $\mathcal{A}$, a *forgetting operator (over $\mathcal{C}$)* is a partial function $\mathsf{f} : \mathcal{C} \times 2^{\mathcal{A}} \to \mathcal{C}$ s.t. $\mathsf{f}(P, V)$ is a program over $\mathcal{A}(P) \setminus V$, for each $P \in \mathcal{C}$ and $V \in 2^{\mathcal{A}} \setminus \emptyset$. We call $\mathsf{f}(P, V)$ the *result of forgetting about V from P*. Whenever $\mathcal{C} = \mathcal{C}_e$, we leave $\mathcal{C}$ implicit. Furthermore, f is called *closed* for $\mathcal{C}' \subseteq \mathcal{C}$ if, for every $P \in \mathcal{C}'$ and $V \in 2^{\mathcal{A}}$, we have $\mathsf{f}(P, V) \in \mathcal{C}'$. A *class* F *of forgetting operators* is a set of forgetting operators. Often, F is defined for a (maximal) class of programs $\mathcal{C}$, denoted as *a class* F *of forgetting operators over $\mathcal{C}$*. The requirement for f being a partial function is a natural one given the existing literature, where some operators as well as classes of these are not closed for certain classes of programs.

Previous work on forgetting in ASP has introduced a variety of desirable properties. Unless stated otherwise, F is a class of forgetting operators, and $\mathcal{C}$ the class of programs over $\mathcal{A}$ of a given $\mathsf{f} \in \mathsf{F}$.

(sC) F satisfies *strengthened Consequence* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $\mathcal{AS}(\mathsf{f}(P, V)) \subseteq \mathcal{AS}(P)_{\|V}$.

(wE) F satisfies *weak Equivalence* if, for each $\mathsf{f} \in \mathsf{F}$, $P, P' \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $\mathcal{AS}(\mathsf{f}(P, V)) = \mathcal{AS}(\mathsf{f}(P', V))$ whenever $\mathcal{AS}(P) = \mathcal{AS}(P')$.

(SE) F satisfies *Strong Equivalence* if, for each $\mathsf{f} \in \mathsf{F}$, $P, P' \in \mathcal{C}$ and $V \subseteq \mathcal{A}$: if $P \equiv_{\mathsf{HT}} P'$, then $\mathsf{f}(P, V) \equiv_{\mathsf{HT}} \mathsf{f}(P', V)$.

(W) F satisfies *Weakening* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $P \models_{\mathsf{HT}} \mathsf{f}(P, V)$.

(PP) F satisfies *Positive Persistence* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$: if $P \models_{\mathsf{HT}} P'$, with $P' \in \mathcal{C}$ and $\mathcal{A}(P') \subseteq \mathcal{A} \setminus V$, then $\mathsf{f}(P, V) \models_{\mathsf{HT}} P'$.

(NP) F satisfies *Negative Persistence* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$: if $P \not\models_{\mathsf{HT}} P'$, with $P' \in \mathcal{C}$ and $\mathcal{A}(P') \subseteq \mathcal{A} \setminus V$, then $\mathsf{f}(P, V) \not\models_{\mathsf{HT}} P'$.

(SI) F satisfies *Strong (addition) Invariance* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $\mathsf{f}(P, V) \cup R \equiv_{\mathsf{HT}} \mathsf{f}(P \cup R, V)$ for all programs $R \in \mathcal{C}$ with $\mathcal{A}(R) \subseteq \mathcal{A} \setminus V$.

($\mathbf{E}_{\mathcal{C}}$) F satisfies *Existence for $\mathcal{C}$*, i.e., F is *closed for a class of programs $\mathcal{C}$* if there exists $\mathsf{f} \in \mathsf{F}$ s.t. f is closed for $\mathcal{C}$.

(CP) F satisfies *Consequence Persistence* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $\mathcal{AS}(\mathsf{f}(P, V)) = \mathcal{AS}(P)_{\|V}$.

(SP) F satisfies *Strong Persistence* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $\mathcal{AS}(\mathsf{f}(P, V) \cup R) = \mathcal{AS}(P \cup R)_{\|V}$, for all programs $R \in \mathcal{C}$ with $\mathcal{A}(R) \subseteq \mathcal{A} \setminus V$.

(wC) F satisfies *weakened Consequence* if, for each $\mathsf{f} \in \mathsf{F}$, $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $\mathcal{AS}(P)_{\|V} \subseteq \mathcal{AS}(\mathsf{f}(P, V))$.

Throughout the paper, whenever we write that a single operator f obeys some property, we mean that the singleton class composed of that operator, $\{\mathsf{f}\}$, obeys such property.

3 Can We Always Forget?

Among the desirable properties of classes of forgetting operators recalled in the previous section, *strong persistence* **(SP)** [16] is of particular interest, as it ensures that forgetting preserves all existing relations between all atoms occurring in the program, but the forgotten. In this sense, a class of operators satisfying **(SP)** removes the desired atoms, but has no negative semantical effects on the remainder. The importance of **(SP)** is also witnessed by the fact that a class of operators that satisfies **(SP)** also satisfies all the other previously mentioned properties with the exception of **(W)** and **(NP)**, which happen to be equivalent and can hardly be considered desirable [15].

However, determining a forgetting operator that satisfies **(SP)** is a difficult problem, since, for the verification whether a certain program P' should be the result of forgetting about V from P, none of the well-established equivalence relations can be used, i.e., neither equivalence nor strong equivalence hold in general between P and P', not even relativized equivalence [10], even though it is close in spirit to the ideas of **(SP)**. Hence, maybe not surprisingly, there is no known general class of operators that satisfies **(SP)** and which is closed (for the considered class of logic programs).

The two known positive results concerning the satisfiability of **(SP)** are the existence of several known classes of operators that satisfy **(SP)** *when restricted to Horn programs* [15], and the existence of one specific operator that, in a very restricted range of situations based on a non-trivial syntactical criterion, permits forgetting about V from P while satisfying **(SP)** [16]. However, the former result is probably of little relevance given the crucial role played by (default) negation in ASP, while the criterion required in the latter result is certainly too strong, excluding large classes of cases where forgetting about V from P is possible.

All this begs the question of whether there exists a forgetting operator, or a class of these, defined over a class of programs $\mathcal{C}$ beyond the class of Horn programs, that satisfies **(SP)**. The following theorem provides a negative answer to this question.

Theorem 1 *There is no forgetting operator over $\mathcal{C} \supseteq \mathcal{C}_n$ that satisfies* **(SP)**.

Proof: Let $\mathcal{C}$ be a class of programs with $\mathcal{C} \supseteq \mathcal{C}_n$ and suppose there exists f over $\mathcal{C}$ that satisfies **(SP)**. Then, for each $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$, we have $\mathcal{AS}(\mathsf{f}(P, V) \cup R) = \mathcal{AS}(P \cup R)_{\|V}$, for all programs $R \in \mathcal{C}$ with $\mathcal{A}(R) \subseteq \mathcal{A} \setminus V$. Consider $P \in \mathcal{C}_n$:

$$a \leftarrow p \qquad b \leftarrow q \qquad p \leftarrow not\, q \qquad q \leftarrow not\, p$$

We construct $\mathcal{HT}(\mathsf{f}(P, \{p, q\}))$, the set of HT-models of the result of forgetting about $V = \{p, q\}$ from P.

We know that $\langle ab, ab\rangle$[5] must be part of $\mathcal{HT}(\mathsf{f}(P, \{p, q\}))$, otherwise it would not be possible to obtain the answer set $\{a, b\}$ for the result when adding $R = \{a \leftarrow; b \leftarrow\}$ to $\mathsf{f}(P, \{p, q\})$.

At the same time, since $\{a, b\}$ (modulo V) is not an answer set of the original program, $\langle X, ab\rangle \in \mathcal{HT}(\mathsf{f}(P, \{p, q\}))$ for at least one $X \subset \{a, b\}$ to prevent $\{a, b\}$ from being an answer set of $\mathsf{f}(P, \{p, q\})$. Consider the three alternatives:

- $\langle \emptyset, ab\rangle \notin \mathcal{HT}(\mathsf{f}(P, \{p, q\}))$, as adding $R = \{a \leftarrow b; b \leftarrow a\}$, whose HT-models are $\{\langle \emptyset, \emptyset\rangle, \langle \emptyset, ab\rangle, \langle ab, ab\rangle\}$, yields one answer set $\{a, b\}$ for $P \cup R$ (modulo V) which the forgetting result has to preserve;
- $\langle a, ab\rangle \notin \mathcal{HT}(\mathsf{f}(P, \{p, q\}))$, since adding $R = \{a \leftarrow\}$, whose HT-models include $\langle a, ab\rangle$, yields one answer set $\{a, b\}$ for $P \cup R$ (modulo V) which the forgetting result has to preserve;
- $\langle b, ab\rangle \notin \mathcal{HT}(\mathsf{f}(P, \{p, q\}))$ symmetrically for $R = \{b \leftarrow\}$.

We derive a contradiction. ■

[5] We follow a common convention and abbreviate sets in HT-interpretations such as $\{a, b\}$ with the sequence of its elements, ab.

Consequently, it is not always possible to forget a set of atoms from a given logic program satisfying the property **(SP)**. In the next section, we address the question of when it is not possible to forget.

4 When Can't We Forget?

Whereas Thm. 1 shows that in general it is not always possible to forget while satisfying **(SP)**, its proof provides some hints on why this is the case. Some atoms play an important role in the program, being pivotal in establishing the relations between the remaining atoms. Therefore, it is simply not possible to forget them and expect that the relations between other atoms be preserved. That is precisely what happens with the pair of atoms p and q in the program

$$a \leftarrow p \qquad b \leftarrow q \qquad p \leftarrow not\, q \qquad q \leftarrow not\, p$$

presented in the proof of Thm. 1. It is simply not possible to forget them both and expect all the semantic relations between a and b to be kept. No program over atoms $\{a, b\}$ would have the same answer sets as those of the original program (modulo p and q), when both are extended with an arbitrary set of rules over $\{a, b\}$.

This observation immediately leads to one of the central questions here: under what circumstances is it not possible to forget about a given set of atoms V from P while satisfying **(SP)**? In particular, given a concrete program, which sets of atoms play such a pivotal role that they cannot be jointly forgotten without affecting the semantic relations between the remaining atoms in the original program?

To deal with these questions that no longer require the satisfaction of certain properties in general for all programs P and all sets of forgotten atoms V, but rather that **(SP)** holds for a concrete P and set V, we introduce the notion of a *forgetting instance*.

Definition 1 (Forgetting Instance) *Let $\mathcal{C}$ be a class of programs over $\mathcal{A}$. A* (forgetting) instance (over $\mathcal{C}$) *is a pair $\langle P, V\rangle$ s.t. $P \in \mathcal{C}$ and $V \subseteq \mathcal{A}$.*

We also introduce a restriction of property **(SP)** to operators of forgetting and such forgetting instances.

Definition 2 (Strong Persistence for Forgetting Instance) *A forgetting operator f over $\mathcal{C}$ satisfies* **(SP)**$_{\langle P,V\rangle}$ *if $\mathcal{AS}(\mathsf{f}(P, V) \cup R) = \mathcal{AS}(P \cup R)_{\|V}$, for all programs $R \in \mathcal{C}$ with $\mathcal{A}(R) \subseteq \mathcal{A} \setminus V$.*

Also, f satisfies **(SP)**$_V$ *if* f *satisfies* **(SP)**$_{\langle P,V\rangle}$ *for all $P \in \mathcal{C}$.*

First, we focus on specific classes of programs.

Proposition 1 *There is no forgetting operator over $\mathcal{C}_n$ that satisfies* **(SP)**$_V$ *for any V.*

Example 1 *Consider forgetting about q from P:*

$$p \leftarrow not\, q \qquad\qquad q \leftarrow not\, p$$

The only correct result that satisfies the condition of **(SP)** *is strongly equivalent to $\{p \leftarrow not\, not\, p\}$, which is not strongly equivalent to any normal program.*

This example shows that even if we can forget about some V from a normal program, the result will in general be an extended program, and we would have to revert to using operators over this general class for subsequent forgetting operations.

We could also wonder whether considering $\mathcal{C}_d$ instead of $\mathcal{C}_n$, in the former proposition, would yield better results, but to no avail.

Proposition 2 *There is no forgetting operator over $\mathcal{C}_d$ that satisfies* **(SP)**$_V$ *for any V.*

This can be verified by considering P in Ex. 1, since there is no disjunctive program which is strongly equivalent to the presented result.

The reason why Props. 1 and 2 hold is tied to the fact that the result of forgetting is not within the class of programs considered. Since we are interested in more essential reasons why a set of atoms cannot be forgotten from a program, in what follows, unless otherwise stated, we will focus on forgetting operators over the entire class $\mathcal{C}_e$.

We now proceed with the introduction of a criterion (Ω) which will play a fundamental role in characterizing the instances for which we cannot expect forgetting operators to satisfy **(SP)**$_{\langle P,V\rangle}$.

Definition 3 (Criterion Ω) *Let P be a program over $\mathcal{A}$ and $V \subseteq \mathcal{A}$. An instance $\langle P, V\rangle$* satisfies criterion Ω *if there exists $Y \subseteq \mathcal{A} \setminus V$ such that the set of sets*

$$\mathcal{R}^Y_{\langle P,V\rangle} = \{R^{Y,A}_{\langle P,V\rangle} \mid A \in Rel^Y_{\langle P,V\rangle}\}$$

is non-empty and has no least element, where

$$R^{Y,A}_{\langle P,V\rangle} = \{X \setminus V \mid \langle X, Y \cup A\rangle \in \mathcal{HT}(P)\}$$
$$Rel^Y_{\langle P,V\rangle} = \{A \subseteq V \mid \langle Y \cup A, Y \cup A\rangle \in \mathcal{HT}(P) \text{ and } \nexists A' \subset A \text{ such that } \langle Y \cup A', Y \cup A\rangle \in \mathcal{HT}(P)\}.$$

The following example illustrates how criterion Ω can be checked.

Example 2 *Recall $P \in \mathcal{C}_n$ used in the proof of Thm. 1:*

$$a \leftarrow p \qquad b \leftarrow q \qquad p \leftarrow not\, q \qquad q \leftarrow not\, p$$

To check if the instance $\langle P, \{p, q\}\rangle$ satisfies criterion Ω, we need to inspect the HT-models of P, $\mathcal{HT}(P)$, which contains 15 elements:

$$\begin{array}{cccc} \langle ap, ap\rangle & \langle bq, abq\rangle & \langle b, abpq\rangle & \langle abp, abpq\rangle \\ \langle bq, bq\rangle & \langle abq, abq\rangle & \langle ab, abpq\rangle & \langle abq, abpq\rangle \\ \langle ap, abp\rangle & \langle \emptyset, abpq\rangle & \langle ap, abpq\rangle & \langle abpq, abpq\rangle \\ \langle abp, abp\rangle & \langle a, abpq\rangle & \langle bq, abpq\rangle & \end{array}$$

To prove that instance $\langle P, \{p, q\}\rangle$ satisfies Ω, we need to find $Y \subseteq \mathcal{A} \setminus V = \{a, b, p, q\} \setminus \{p, q\} = \{a, b\}$ such that $\mathcal{R}^Y_{\langle P,V\rangle}$ is non-empty and has no least element. For that, we only need to focus on those sets $Y' \subseteq \{a, b\}$ for which there exists $\langle H, T\rangle \in \mathcal{HT}(P)$ with $Y' = T \setminus \{p, q\}$, since for all other Y' the set $\mathcal{R}^{Y'}_{\langle P,V\rangle}$ is necessarily empty. In this case, such sets are $Y' = \{b\}$, since there is an HT-model of P of the form $\langle X, bq\rangle$, $Y' = \{a\}$, since there is an HT-model of P of the form $\langle X, ap\rangle$, and $Y' = \{a, b\}$, since there are HT-models of P of the form $\langle X, abp\rangle$, $\langle X, abq\rangle$ and $\langle X, abpq\rangle$. For $Y' = \{b\}$, $\mathcal{R}^{Y'}_{\langle P,V\rangle}$ has only one element and therefore necessarily a least one. The same holds for $Y' = \{a\}$.

We are left to inspect $Y' = \{a, b\}$. For such Y' we need to focus on HT-models of the form $\langle X, abp\rangle$, $\langle X, abq\rangle$ and $\langle X, abpq\rangle$.

Those of the form $\langle X, abpq\rangle$ are however not relevant (for property **(SP)***), since $\{a, b, p, q\}$ can never be an answer set of $P \cup R$ with $\mathcal{A}(R) \subseteq \{a, b\}$. This happens since, besides $\langle abpq, abpq\rangle$, $\langle ab, abpq\rangle$, $\langle abp, abpq\rangle$ and $\langle abq, abpq\rangle$ are also HT-models of P, and since any program R over $\{a, b\}$ cannot distinguish these HT-models, i.e., either all are HT-models of R or none is. Therefore, $\{p, q\} \notin Rel^{\{a,b\}}_{\langle P,V\rangle}$. Then, $\mathcal{R}^{\{a,b\}}_{\langle P,V\rangle}$ has only two elements, $R^{\{a,b\},\{p\}}_{\langle P,V\rangle} = \{\{a, b\}, \{a\}\}$ and $R^{\{a,b\},\{q\}}_{\langle P,V\rangle} = \{\{a, b\}, \{b\}\}$. Since these two sets are incomparable, $\mathcal{R}^{\{a,b\}}_{\langle P,V\rangle}$ has no least element. Therefore, taking $Y = \{a, b\}$, we conclude that $\langle P, \{p, q\}\rangle$ satisfies Ω.*

Since criterion Ω heavily relies on HT-models, it helps to observe that, for all instances $\langle P, V\rangle$, any forgetting operator that satisfies **(SP)**$_{\langle P,V\rangle}$ produces a result whose HT-models are a subset of the HT-models of the original program (modulo the forgotten atoms).

Proposition 3 *If a forgetting operator* f *over* $\mathcal{C}$ *satisfies* **(SP)**$_{\langle P,V\rangle}$ *for an instance* $\langle P, V\rangle$ *over* $\mathcal{C}$ *then*

$$\mathcal{HT}(\mathsf{f}(P,V)) \subseteq \mathcal{HT}(P)_{\|V}.$$

Proof: We first prove that $\langle Y,Y\rangle \in \mathcal{HT}(P)_{\|V}$ if $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V))$. Let R be a program over $\mathcal{A}(P) \setminus V$ whose only HT-model of the form $\langle X,Y\rangle$ is $\langle Y,Y\rangle$. Then $Y \in \mathcal{AS}(\mathsf{f}(P,V) \cup R)$. Since f satisfies **(SP)**$_{\langle P,V\rangle}$, we have that $Y \in \mathcal{AS}(P \cup R)_{\|V}$. Then, there exists $A \subseteq V$ such that $\langle Y \cup A, Y \cup A\rangle \in \mathcal{HT}(P \cup R)$. In particular $\langle Y \cup A, Y \cup A\rangle \in \mathcal{HT}(P)$, and hence $\langle Y,Y\rangle \in \mathcal{HT}(P)_{\|V}$.

Now let $\langle X,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V))$ with $X \subset Y$. We then have that $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V))$, and, as we proved above, $\langle Y,Y\rangle \in \mathcal{HT}(P)_{\|V}$. Let R be a program over $\mathcal{A}(P) \setminus V$ whose only HT-models of the form $\langle X',Y\rangle$ are $\langle X,Y\rangle$ and $\langle Y,Y\rangle$. Then, $Y \notin \mathcal{AS}(\mathsf{f}(P,V) \cup R)$. Since f satisfies **(SP)**$_{\langle P,V\rangle}$, we have that $Y \notin \mathcal{AS}(P \cup R)_{\|V}$. Since R is a program over $\mathcal{A}(P) \setminus V$, the only HT-models of R of the form $\langle X', Y \cup A\rangle$ with $A \subseteq V$ are of the form $\langle X \cup A', Y \cup A\rangle$ or $\langle Y \cup A', Y \cup A\rangle$ with $A' \subseteq A$. It is easy to prove that there is $A' \subseteq A \subseteq V$ such that $\langle X \cup A', Y \cup A\rangle \in \mathcal{HT}(P)$. Otherwise, taking R' over $\mathcal{A}(P) \setminus V$ whose only HT-model of the form $\langle X,Y\rangle$ is $\langle Y,Y\rangle$, we would have that $Y \notin \mathcal{AS}(P \cup R')$, but $Y \in \mathcal{AS}(P \cup R')$. Therefore, $\langle X,Y\rangle \in \mathcal{HT}(P)_{\|V}$. ■

We are now ready to state that Ω is a sufficient condition to determine that some set of atoms V cannot be forgotten from a program P while satisfying *strong persistence*.

Theorem 2 *If* $\langle P,V\rangle$ *satisfies* Ω*, then no forgetting operator* f *satisfies* **(SP)**$_{\langle P,V\rangle}$.

Proof: Let $\langle P,V\rangle$ be an instance satisfying Ω, and assume that there is a forgetting operator f that satisfies **(SP)**$_{\langle P,V\rangle}$.

Since $\langle P,V\rangle$ satisfies Ω, we have that there is $Y \subseteq \mathcal{A} \setminus V$ such that the set $\mathcal{R}^Y_{\langle P,V\rangle} = \{R^{Y,A}_{\langle P,V\rangle} \mid A \in Rel^Y_{\langle P,V\rangle}\}$ – as defined in Def. 3 – has no least element.

First, we prove that $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V))$. Assume it is not the case. Consider the program $R = \{a \leftarrow \mid a \in Y\}$ over $\mathcal{A} \setminus V$. The only HT-model of R of the form $\langle X,Y\rangle$ is $\langle Y,Y\rangle$. Since $\mathcal{R}^Y_{\langle P,V\rangle}$ is non-empty, P must contain an HT-model of the form $\langle Y', Y'\rangle$ such that $Y = Y' \setminus V$ and there is no $\langle X', Y'\rangle$ with $X' \subset Y'$ and $Y = X' \setminus V$. Therefore, $Y \in \mathcal{AS}(P \cup R)_{\|V}$, but $Y \notin \mathcal{AS}(\mathsf{f}(P,V) \cup R)$, contradicting the assumption that f satisfies **(SP)**$_{\langle P,V\rangle}$.

We now show that, for each $\langle X, Y \cup A\rangle \in \mathcal{HT}(P)$ such that $A \in Rel^Y_{\langle P,V\rangle}$ and $X \setminus V \notin \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, we have that $\langle X \setminus V, Y\rangle \notin \mathcal{HT}(\mathsf{f}(P,V))$. Assume that $\langle X \setminus V, Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V))$. Consider a program R over $\mathcal{A} \setminus V$ whose HT-models over $\mathcal{A} \setminus V$ of the form $\langle X', Y\rangle$ are only $\langle Y,Y\rangle$ and $\langle X \setminus V, Y\rangle$. Since $X \setminus V \notin \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, there exists $A' \in Rel^Y_{\langle P,V\rangle}$ such that $X \setminus V \notin R^{Y,A'}_{\langle P,V\rangle}$. Therefore, there is no $A'' \subseteq A'$ such that $\langle (X \setminus V) \cup A'', Y \cup A'\rangle \in \mathcal{HT}(P)$. Since R is a program over $\mathcal{A} \setminus V$ then $\langle Y,Y\rangle \in \mathcal{HT}(R)$ implies that, for any $A'' \subseteq V$, we have that $\langle Y \cup A'', Y \cup A''\rangle \in \mathcal{HT}(R)$ over $\mathcal{A}$. Then, $Y \cup A' \in \mathcal{AS}(P \cup R)$, which implies that $Y \in \mathcal{AS}(P \cup R)_{\|V}$. But $Y \notin \mathcal{AS}(\mathsf{f}(P,V) \cup R)$, since $\langle X \setminus V, Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V) \cup R)$. This contradicts the assumption that f satisfies **(SP)**$_{\langle P,V\rangle}$.

Recall from Prop. 3 that $\mathcal{HT}(\mathsf{f}(P,V)) \subseteq \mathcal{HT}(P)_{\|V}$. This, together with $\langle X,Y\rangle \notin \mathcal{HT}(\mathsf{f}(P,V))$ for $X \notin \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, implies that $\{X \mid \langle X,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V))\} \subseteq \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$.

Since we are assuming that $\mathcal{R}^Y_{\langle P,V\rangle}$ has no least element, then, for each $A \in Rel^Y_{\langle P,V\rangle}$, there exists $X_A \in R^{Y,A}_{\langle P,V\rangle}$ s.t. $X_A \notin \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$. Now take a program R over $\mathcal{A} \setminus V$ whose HT-models of the form $\langle X', Y\rangle$ are exactly $\langle Y,Y\rangle$ and $\langle X_A, Y\rangle$ for each $A \in Rel^Y_{\langle P,V\rangle}$. Then we clearly have that $Y \notin \mathcal{AS}(P \cup R)_{\|V}$. Since each $\langle X_A, Y\rangle$ cannot belong to $\mathcal{HT}(\mathsf{f}(P,V))$ because $X_A \notin \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, we have that $Y \in \mathcal{AS}(\mathsf{f}(P,V) \cup R)$. This is a contradiction to the assumption that f satisfies **(SP)**$_{\langle P,V\rangle}$. Therefore, f cannot satisfy **(SP)**$_{\langle P,V\rangle}$. ■

Example 3 *Recall* $P \in \mathcal{C}_n$ *used in the proof of Thm. 1 and Ex. 2.*

$$a \leftarrow p \qquad b \leftarrow q \qquad p \leftarrow not\ q \qquad q \leftarrow not\ p$$

Since $\langle P, \{p,q\}\rangle$ *satisfies criterion* Ω *(cf. Ex. 2), no forgetting operator* f *satisfies* **(SP)**$_{\langle P,\{p,q\}\rangle}$.

We can determine the complexity of testing Ω for a given instance.

Theorem 3 *The complexity of checking if an instance* $\langle P,V\rangle$ *satisfies criterion* Ω *is in* Π^p_3.

Proof: We need to find $R^{Y,A_1}_{\langle P,V\rangle}, R^{Y,A_2}_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle}$ s.t. both are minimal in $\mathcal{R}^Y_{\langle P,V\rangle}$ and do not coincide to ensure that Ω is satisfied.

We can guess Y, A_1 and A_2 in polynomial time.

The test of whether some $R^{Y,A}_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle}$ requires verifying whether $A \in Rel^Y_{\langle P,V\rangle}$. This can be solved in each case by verifying whether (a) $\langle Y \cup A, Y \cup A\rangle \in \mathcal{HT}(P)$ and whether (b) there is no proper subset A' of A such that $\langle Y \cup A', Y \cup A\rangle \in \mathcal{HT}(P)$. Testing whether (a) a given HT-interpretation is a HT-model can be done in PTIME, and (b) corresponds to a coNP problem with an NP oracle (for finding HT-models). Hence, this test can be done in Π^p_2.

Checking whether each $R^{Y,A_i}_{\langle P,V\rangle}$ is minimal corresponds to a coNP problem using a Σ^p_2 oracle for guessing and checking suitable $R^{Y,A}_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle}$. Hence, this test can be done in Π^p_3.

Finally, the test of whether the two $R^{Y,A_i}_{\langle P,V\rangle}$ do not coincide can be done by finding a single HT-interpretation $\langle X, Y \cup A_1\rangle$ such that $X \setminus V \in R^{Y,A_1}_{\langle P,V\rangle}$ but no corresponding $\langle X', Y \cup A_2\rangle \in HT(P)$ such that $X' \setminus V \in R^{Y,A_2}_{\langle P,V\rangle}$. This test can is also a coNP problem with an NP oracle, hence in Π^p_2. Thus, the overall test is in Π^p_3. ■

Even though showing hardness for this result remains open, it seems clear that testing for Ω is non-trivial, the intuitive reason being that besides the complexity of determining HT-models, there are two minimizations to consider, one for $Rel^Y_{\langle P,V\rangle}$ and one for $\mathcal{R}^Y_{\langle P,V\rangle}$. Still, the complexity is lower than that of computing results of forgetting, which is commonly exponential [15] – *it's not easy to forget*. Nevertheless, if the HT-models have already been computed, checking Ω can be done in linear time on the number of HT-models.

Turning back to the reasons why we cannot forget some sets of atoms from a given program, we can observe – both in the proof of Thm. 1 as well as when the criterion Ω is satisfied – that they seem to be strongly connected to the presence of atoms to be forgotten that depend on themselves via an even (and non-zero) number of negations. This could raise the question of whether it is possible to forget about sets of atoms under certain restrictions. For example, we can verify that criterion Ω is not satisfied for forgetting only about p from the normal program used in the proof of Thm. 1, i.e., we can forget about p in this case. So it would seem that forgetting about certain sets of atoms from normal programs should be a more admissible problem. Unfortunately, this is not the case, since even if we were to delimit V, e.g., by restricting its size, in general, no forgetting operator exists that satisfies **(SP)**$_V$.

Proposition 4 *There is no forgetting operator over* $\mathcal{C}_e$ *that satisfies* **(SP)**$_V$ *for any* V *of fixed size.*

Example 4 *Consider forgetting about p from P:*

$$a \leftarrow p \qquad b \leftarrow not\, p \qquad p \leftarrow not\, not\, p\,,$$

Even though we forget only about one atom p, the argument in the proof of Thm. 1 applies just as well, and no program $\mathsf{f}(P,\{p\})$ exists whose HT-models match the requirements to satisfy **(SP)**$_V$.

5 When (and How) Can We Forget?

So far we have focused on what cannot be forgotten. We now turn our attention to what can be forgotten.

While Ω allows us to test when it is not possible to forget without sacrificing **(SP)**, we do not yet know whether this is a necessary criterion. That being the case would ensure that whenever we forget about V from P, and Ω is not satisfied, **(SP)**$_{\langle P,V\rangle}$ could be obeyed. In this section, we investigate this matter with the aim of being able to determine when we can forget, and, if possible, to characterise those operators that can obtain the desirable result. To this end, we start by introducing a new class of forgetting operators that computes the HT-models that represent a result of forgetting about V from P.

Definition 4 (SP-Forgetting) *Let $\mathsf{F_{SP}}$ be the class of forgetting operators defined by the following set:*

$$\{\mathsf{f} \mid \mathcal{HT}(\mathsf{f}(P,V)) = \{\langle X,Y\rangle \mid Y \subseteq \mathcal{A}(P)\setminus V \wedge X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}\}\}$$

As expected, the definition of this class of operators strongly relies on the sets of sets $\mathcal{R}^Y_{\langle P,V\rangle}$, where computing the intersections amounts to determining the least elements in each $\mathcal{R}^Y_{\langle P,V\rangle}$ if they exist, i.e., whenever criterion Ω is not satisfied.

Example 5 *Consider the following normal program $P \in \mathcal{C}_n$.*

$$a \leftarrow not\, p \qquad p \leftarrow not\, b$$

Forgetting $\{p\}$ from P while satisfying **(SP)** *should preserve all dependencies between a and b. Such dependencies are internalized in the set of HT-models of P. In this case, all models in $\mathcal{HT}(P)$ are of the form $\langle X,Y\rangle$ with $Y=\{p\}$, $Y=\{a,p\}$, $Y=\{b,p\}$, $Y=\{a,b\}$ or $Y=\{a,b,p\}$. Some of these models are however not relevant from the point of view of* **(SP)**. *For example, since $\langle b, bp\rangle \in \mathcal{HT}(P)$, we have that $\{b,p\}$ can never be an answer set of $(P\cup R)$, for a program R over $\{a,b\}$. This is due to the fact that any R over $\{a,b\}$ cannot distinguish the models $\langle b,bp\rangle$ and $\langle bp,bp\rangle$, i.e., one is a model of R iff the other is. The same type of argument applies to $Y=\{a,b,p\}$ since $\langle ab,abp\rangle \in \mathcal{HT}(P)$. Therefore, $\{p\} \notin Rel^{\{b\}}_{\langle P,V\rangle}$ and $\{p\} \notin Rel^{\{a,b\}}_{\langle P,V\rangle}$, which intuitively means that models of P of the form $\langle X, bp\rangle$ and of the form $\langle X, abp\rangle$ are not relevant for the existence of models of the form $\langle X,b\rangle$ and $\langle X,ab\rangle$ in $\mathsf{f}(P,\{p\})$, respectively. In the case of $Y=\{b\}$, we have that $\mathcal{R}^{\{b\}}_{\langle P,V\rangle}=\emptyset$, which means that $\mathsf{f}(P,V)$ has no HT-model of the form $\langle X,\{b\}\rangle$. For the other possible models we have $\{p\} \in Rel^{\emptyset}_{\langle P,\{p\}\rangle}$, $\{a\} \in Rel^{\{a\}}_{\langle P,\{p\}\rangle}$ and $\{a,b\} \in Rel^{\{a,b\}}_{\langle P,\{p\}\rangle}$. For each such Y, the set $\mathcal{R}^Y_{\langle P,V\rangle}$ has only one element, thus the intersection is precisely that element. This immediately implies that the forgetting instance $\langle P,\{p\}\rangle$ does not satisfy criterion Ω. The resulting set of HT-models of $\mathsf{f}(P,V)$ for $f \in \mathsf{F_{SP}}$, is $\{\langle\emptyset,\emptyset\rangle, \langle\emptyset,a\rangle, \langle a,a\rangle, \langle a,ab\rangle, \langle ab,ab\rangle\}$. This means that for any $f \in \mathsf{F_{SP}}$, the program $\mathsf{f}(P,V)$ is strongly equivalent to $\{a \leftarrow not\, not\, b\}$.*

The close connection between the definition of $\mathsf{F_{SP}}$ and criterion Ω is not mere coincidence. It turns out that every operator in $\mathsf{F_{SP}}$ in fact satisfies **(SP)** for those instances $\langle P,V\rangle$ that do not satisfy Ω.

Theorem 4 *Every $\mathsf{f} \in \mathsf{F_{SP}}$ satisfies* **(SP)**$_{\langle P,V\rangle}$ *for every $\langle P,V\rangle$ that does not satisfy Ω.*

Proof: Let $\mathsf{f} \in \mathsf{F_{SP}}$ and let $\langle P,V\rangle$ be an instance such that $\langle P,V\rangle$ does not satisfy **(SP)**$_{\langle P,V\rangle}$. Recall that by construction, $\mathcal{HT}(\mathsf{f}(P,V)) = \{\langle X,Y\rangle \mid Y \subseteq \mathcal{A}(P)\setminus V \wedge X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}\}\}$. Let R be a program over $\mathcal{A}(P)\setminus V$.

We start by proving that $\mathcal{AS}(\mathsf{f}(P,V)\cup R) \subseteq \mathcal{AS}(P\cup R)_{\|V}$. Suppose that $Y \in \mathcal{AS}(\mathsf{f}(P,V)\cup R)$. Then, $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V)\cup R)$ and there is no $X \subset Y$ such that $\langle X,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V)\cup R)$. We can then conclude that there is no $X \subset Y$ with $X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$ and $\langle X,Y\rangle \in \mathcal{HT}(R)$. Since $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V)\cup R)$, we can conclude that $\mathcal{R}^Y_{\langle P,V\rangle}$ is non-empty, and given that we are assuming that $\langle P,V\rangle$ does not satisfy Ω, the $\mathcal{R}^Y_{\langle P,V\rangle}$ has a least element, which is precisely $\bigcap \mathcal{R}^Y_{\langle P,V\rangle}$. Let $A \in Rel^Y_{\langle P,V\rangle}$ be such that $R^{Y,A}_{\langle P,V\rangle} = \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$. Then, since $\langle Y,Y\rangle \in \mathcal{HT}(R)$ and $\mathcal{A}(R) \subseteq \mathcal{A}(P)\setminus V$, we have that $\langle Y\cup A, Y\cup A\rangle \in \mathcal{HT}(R)$. Therefore, $\langle Y\cup A, Y\cup A\rangle \in \mathcal{HT}(P\cup R)$. Since $A \in Rel^Y_{\langle P,V\rangle}$ and there is no $X \subset Y$ with $X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$ such that $\langle X,Y\rangle \in \mathcal{HT}(R)$, we can conclude that there is no $X \subset Y\cup A$ such that $\langle X, Y\cup A\rangle \in \mathcal{HT}(P\cup R)$. Therefore, $Y\cup A \in \mathcal{AS}(P\cup R)$, and so $Y \in \mathcal{AS}(P\cup R)_{\|V}$.

We now prove that $\mathcal{AS}(P\cup R)_{\|V} \subseteq \mathcal{AS}(\mathsf{f}(P,V)\cup R)$. Suppose $Y \in \mathcal{AS}(P\cup R)_{\|V}$. Then there exists $A \subseteq V$ such that $\langle Y\cup A, Y\cup A\rangle \in \mathcal{HT}(P\cup R)$ and there is no $X \subset Y\cup A$ such that $\langle X, Y\cup A\rangle \in \mathcal{HT}(P\cup R)$. Therefore, $\mathcal{R}^Y_{\langle P,V\rangle}$ is non-empty and $A \in Rel^Y_{\langle P,V\rangle}$. Also, there is no $X' \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$ such that $\langle X',Y\rangle \in \mathcal{HT}(R)$. Therefore, there is no $\langle X',Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V)\cup R)$ such that $X' \subset Y$. In order to conclude that $Y \in \mathcal{AS}(\mathsf{f}(P,V)\cup R)$ we just need to prove that $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V)\cup R)$. Since $\langle Y\cup A, Y\cup A\rangle \in \mathcal{HT}(R)$ and $\mathcal{A}(R) \subseteq \mathcal{A}(P)\setminus V$, we have that $\langle Y,Y\rangle \in \mathcal{HT}(R)$. Since $Y \in R^{Y,A}_{\langle P,V\rangle}$ for all $A \in Rel^Y_{\langle P,V\rangle}$, we clearly have that $Y \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$ and therefore, by construction, $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V))$. We can then conclude that $\langle Y,Y\rangle \in \mathcal{HT}(\mathsf{f}(P,V)\cup R)$. Therefore, $Y \in \mathcal{AS}(\mathsf{f}(P,V)\cup R)$. ■

It immediately follows from Thms. 2 and 4 that Ω is a necessary and sufficient criterion for determining whether it is possible to forget about V from P and preserve **(SP)**.

Corollary 1 *There is a forgetting operator f that satisfies* **(SP)**$_{\langle P,V\rangle}$ *iff $\langle P,V\rangle$ does not satisfy Ω.*

Thus, given an instance $\langle P,V\rangle$, we can test whether Ω is not satisfied, i.e., whether we are allowed to forget V from P while preserving **(SP)**, in which case we can compute the HT-models that characterise a result using the definition of $\mathsf{F_{SP}}$.

Also, whenever it is possible to forget, the class $\mathsf{F_{SP}}$ precisely characterises the result of forgetting.

Proposition 5 *Let $\langle P,V\rangle$ be an instance that does not satisfy Ω. Then, for every forgetting operator f satisfying* **(SP)**$_{\langle P,V\rangle}$ *and every $\mathsf{f}' \in \mathsf{F_{SP}}$ we have that $\mathsf{f}(P,V) \equiv_{\mathsf{HT}} \mathsf{f}'(P,V)$.*

For the particular case of Horn programs, it is possible to simplify the construction of the result since the set of HT-models of the result of forgetting coincides with the set of HT-models of the original program, modulo the forgotten atoms. A side-effect of this is that the result is guaranteed to be itself a Horn program.

Proposition 6 *Let f be in $\mathsf{F_{SP}}$. Then, for every $V \subseteq \mathcal{A}$:*

$$\mathcal{HT}(\mathsf{f}(P,V)) = \mathcal{HT}(P)_{\|V} \text{ for } P \in \mathcal{C}_H \text{ and } \mathsf{f}(P,V) \in \mathcal{C}_H\,.$$

Proof: We know from [9] that the HT-models M of Horn programs are characterized by the following conditions:

(A) $\langle X, Y\rangle \in M, Y \subseteq Y'$ and $\langle Y', Y'\rangle \in M \Rightarrow \langle X, Y'\rangle \in M$
(B) $\langle X, Y\rangle \in M$ iff $X \subseteq Y, \langle X, X\rangle \in M$ and $\langle Y, Y\rangle \in M$
(C) $\langle X, Y\rangle \in M$ and $\langle H, T\rangle \in M \Rightarrow \langle X \cap H, Y \cap T\rangle \in M$

We first prove that $\mathcal{HT}(\mathsf{f}(P, V)) = \mathcal{HT}(P)_{\|V}$ for $P \in \mathcal{C}_H$. Let us prove the two inclusions. Consider $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. Then $Y \subseteq \mathcal{A}(P) \setminus V$ and $X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, i.e., there is at least one $A \in Rel^Y_{\langle P,V\rangle}$ such that $X \in R^{Y,A}_{\langle P,V\rangle}$. Hence, there is $\langle X', Y \cup A\rangle \in \mathcal{HT}(P)$ with $X = X' \setminus V$.

Now consider $\langle X, Y\rangle \in \mathcal{HT}(P)_{\|V}$. Then, there exists $A' \subseteq A \subseteq V$ such that $\langle X \cup A', Y \cup A\rangle \in \mathcal{HT}(P)$. Let $A'' \in Rel^Y_{\langle P,V\rangle}$. Then $\langle Y \cup A'', Y \cup A''\rangle \in \mathcal{HT}(P)$. Using condition (C) we can conclude that $\langle (X \cup A') \cap (Y \cup A''), (Y \cup A) \cap (Y \cup A'')\rangle \in \mathcal{HT}(P)$, i.e., $\langle X \cup (A' \cap A''), Y \cup (A \cap A'')\rangle \in \mathcal{HT}(P)$. Since $Y \cup (A \cap A'') \subseteq Y \cup A''$, and using condition (A) we can conclude that $\langle X \cup (A' \cap A''), Y \cup A''\rangle \in \mathcal{HT}(P)$, and consequently $X \in R^{Y,A''}_{\langle P,V\rangle}$. Hence $X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, and therefore $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$.

We now prove that for $P \in \mathcal{C}_H$ we have $\mathsf{f}(P, V) \in \mathcal{C}_H$. So, we need to prove that conditions (A), (B) and (C) hold for $\mathcal{HT}(\mathsf{f}(P, V))$.

For condition (A) take $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ and $\langle Y', Y'\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ such that $Y \subseteq Y'$. We aim to prove that $\langle X, Y'\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. By definition $X \in \mathcal{R}^Y_{\langle P,V\rangle}$ and $Y' \in \mathcal{R}^{Y'}_{\langle P,V\rangle}$. Therefore, $\langle X \cup A, Y \cup A'\rangle \in \mathcal{HT}(P)$ for some $A \subseteq A' \subseteq V$. Also, $\langle Y' \cup A'', Y' \cup A''\rangle \in \mathcal{HT}(P)$ for every $A'' \in Rel^{Y'}_{\langle P,V\rangle}$. Then, since $\mathcal{HT}(P)$ satisfies condition (B), we can easily conclude that $\langle X \cup (A \cap A''), Y' \cup A''\rangle \in \mathcal{HT}(P)$ for every $A'' \in Rel^{Y'}_{\langle P,V\rangle}$. Therefore, $X \in \bigcap \mathcal{R}^{Y'}_{\langle P,V\rangle}$, hence $\langle X, Y'\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$.

For condition (B), first take $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. We aim to prove that $\langle X, X\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ and $\langle Y, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. Since $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ we have that $X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$. Then, $\langle X \cup A', Y \cup A''\rangle \in \mathcal{HT}(P)$ for some $A' \subseteq A'' \subseteq V$. Using condition (B) for $\mathcal{HT}(P)$ we have that $\langle X \cup A', X \cup A'\rangle \in \mathcal{HT}(P)$ and $\langle Y \cup A'', Y \cup A''\rangle \in \mathcal{HT}(P)$. Using again condition (B), we can conclude that for every $A''' \in Rel^X_{\langle P,V\rangle}$ we have $\langle X \cup (A' \cap A'''), X \cup A'''\rangle \in \mathcal{HT}(P)$. Hence, $X \in \bigcap \mathcal{R}^X_{\langle P,V\rangle}$, and therefore $\langle X, X\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. In the same way, we can conclude that for every $A''' \in Rel^Y_{\langle P,V\rangle}$ we have $\langle Y \cup (A'' \cap A'''), Y \cup A'''\rangle \in \mathcal{HT}(P)$. Hence, $Y \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, and therefore $\langle Y, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$.

For the converse, assume that $\langle X, X\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ and $\langle Y, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. We aim to prove that $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. Since $\langle X, X\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ and $\langle Y, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ we have that $X \in \bigcap \mathcal{R}^X_{\langle P,V\rangle}$ and $Y \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$.

Then, there is $A' \subseteq V$ such that $\langle X \cup A', X \cup A'\rangle \in \mathcal{HT}(P)$. Also, for every $A'' \in Rel^Y_{\langle P,V\rangle}$ we have $\langle Y \cup A'', Y \cup A''\rangle \in \mathcal{HT}(P)$. Since $X \subseteq Y$ and using condition (C) we have that $\langle X \cup (A' \cap A''), X \cup (A' \cap A'')\rangle \in \mathcal{HT}(P)$. Since $X \cup (A' \cap A'') \subseteq Y \cup A''$ we can use condition (B) to conclude that $\langle X \cup (A' \cap A''), Y \cup A''\rangle \in \mathcal{HT}(P)$, therefore $X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$. Then $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$.

For condition (C) take $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ and $\langle H, T\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. We aim to prove that $\langle X \cap H, Y \cap T\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. Since $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ and $\langle H, T\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$ we have that $X \in \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$ and $H \in \bigcap \mathcal{R}^T_{\langle P,V\rangle}$. Then, there are $A, A', A'', A''' \subseteq V$ such that $\langle X \cup A, Y \cup A'\rangle \in \mathcal{HT}(P)$ and $\langle H \cup A'', T \cup A'''\rangle \in \mathcal{HT}(P)$. Using condition (C) for $\mathcal{HT}(P)$ we have that $\langle (X \cap H) \cup (A \cap A''), (Y \cap T) \cup (A' \cap A''')\rangle \in \mathcal{HT}(P)$. Therefore, it follows easily from condition (B) that $\langle (X \cap H) \cup (A \cap A'' \cap A^*), (Y \cap T) \cup A^*\rangle \in \mathcal{HT}(P)$ for any $A^* \in Rel^{Y\cap T}_{\langle P,V\rangle}$. Therefore $(X \cap H) \in \bigcap \mathcal{R}^{Y\cap T}_{\langle P,V\rangle}$, and hence $\langle X \cap H, Y \cap T\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. ■

We can now show that F_{SP} is closed in the general case and for Horn programs, but not for disjunctive or normal programs. This turns out to be similar as for previous classes of operators defined for the class of extended programs that are based on manipulating HT-models, namely HT-forgetting [41] and SM-forgetting [39].

Theorem 5 F_{SP} *is closed for extended programs and Horn programs, but neither for disjunctive programs nor normal programs.*

Proof: F_{SP} is naturally closed for $\mathcal{C}_e$, since $\langle X, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$, by construction, implies $\langle Y, Y\rangle \in \mathcal{HT}(\mathsf{f}(P, V))$. The negative results for $(\mathbf{E}_{\mathcal{C}_d})$ and $(\mathbf{E}_{\mathcal{C}_n})$ follow from Props. 1 and 2.

Regarding $(\mathbf{E}_{\mathcal{C}_H})$, by Prop. 6, $\mathsf{f}(P, V)) \in \mathcal{C}_H$ for $P \in \mathcal{C}_H$. ■

We state the result in the general case here, i.e., independently of whether we are allowed to forget or not, but these results obviously apply in exactly the same way if we restrict our attention to the cases where we can forget about some V from a given program P.

If we restrict our attention to the cases where we can forget, i.e., where the considered instance does not satisfy Ω, then most of the properties mentioned in Sec. 2 are satisfied.

Theorem 6 *Restricted to instances $\langle P, V\rangle$ that do not satisfy Ω,* F_{SP} *satisfies* **(sC)**, **(wE)**, **(SE)**, **(PP)**, **(SI)**, **(CP)**, **(SP)** *and* **(wC)**.

Proof: By Thm 4, every $\mathsf{f} \in \mathsf{F}_{\mathsf{SP}}$ satisfies $\mathbf{(SP)}_{\langle P,V\rangle}$ for every $\langle P, V\rangle$ that does not satisfy Ω. Hence, F_{SP} satisfies **(SP)** under this restriction. Then, by Prop. 1 of [15], F_{SP} also satisfies **(sC)**, **(wE)**, **(SE)**, **(PP)**, **(SI)**, **(CP)**, **(wC)**, but neither **(W)** nor **(NP)**. ■

The properties which are not satisfied – **(W)** and **(NP)** – have been proved orthogonal to **(SP)** [15], hence of little relevance in our view.

Theorem 7 *Let* f *be in* F_{SP}, *$P, P' \in \mathcal{C}_e$ and $V \subseteq \mathcal{A}$. Deciding if $P' \equiv_{\mathsf{HT}} \mathsf{f}(P, V)$ is in Π^p_3.*

Proof: If $P' \not\equiv_{\mathsf{HT}} \mathsf{f}(P, V)$, then there exists an HT-interpretation $\langle X, Y\rangle$ such that either (a) $\langle X, Y\rangle \models_{\mathsf{HT}} P'$ and $\langle X, Y\rangle \not\models_{\mathsf{HT}} \mathsf{f}(P, V)$, or (b) $\langle X, Y\rangle \not\models_{\mathsf{HT}} P'$ and $\langle X, Y\rangle \models_{\mathsf{HT}} \mathsf{f}(P, V)$.

Consider (a). Checking whether $\langle X, Y\rangle \models_{\mathsf{HT}} P'$ can be done in PTIME. Checking whether $\langle X, Y\rangle \not\models_{\mathsf{HT}} \mathsf{f}(P, V)$ amounts to verifying that $X \notin \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$. This holds if either $\bigcap \mathcal{R}^Y_{\langle P,V\rangle}$ is empty or there is at least one $R^{Y,A}_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle}$ such that $X \notin R^{Y,A}_{\langle P,V\rangle}$. The first case corresponds to the co-problem of finding at least one $R^{Y,A}_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle}$, which is in Π^p_2 (see proof of Thm. 3). The second also requires finding such $R^{Y,A}_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle}$, but also to ensure that $\langle X \cup V', Y \cup A\rangle$ is not an HT-model of P for any $V' \subseteq A$ using an NP-oracle. Hence, (a) is in Π^p_3. A similar argument holds for (b). ■

Still, computing the actual result (and not just its representation in terms of HT-models) is exponential [15]. This high computational complexity of computing a result, together with the fact that the test used in the proof of Thm. 7 does not check if Ω holds, justifies the use of our test for criterion Ω before proceeding with the operation.

6 What Can We Forget?

We now approach the problem from a different angle, and determine which sets of atoms can be forgotten from a specific program.

We begin with the case where the set of atoms to be forgotten is a singleton and the program normal. As it turns out, we can always forget single atoms from normal programs without having to test Ω.

Proposition 7 *There is a forgetting operator satisfying* **(SP)**$_{\langle P,V\rangle}$*, for every $P \in \mathcal{C}_n$ and every V such that $|V|= 1$.*

Proof: Let $P \in \mathcal{C}_n$ and assume that $|V|= 1$. We aim to prove that $\langle P, V\rangle$ does not satisfy Ω. Since $|V|= 1$, for every $Y \subseteq \mathcal{A}(P) \setminus V$, the set $\mathcal{R}^Y_{\langle P,V\rangle}$ has at most two elements: $R^{Y,\emptyset}_{\langle P,V\rangle}$ and $R^{Y,V}_{\langle P,V\rangle}$. If $\mathcal{R}^Y_{\langle P,V\rangle}$ has at most one element then either the set is empty or it has a least element. Suppose both $R^{Y,\emptyset}_{\langle P,V\rangle}$ and $R^{Y,V}_{\langle P,V\rangle}$ belong to $\mathcal{R}^Y_{\langle P,V\rangle}$. Then both $\langle Y, Y\rangle$ and $\langle Y \cup V, Y \cup V\rangle$ are HT-models of P.

We now prove that $R^{Y,\emptyset}_{\langle P,V\rangle} \subseteq R^{Y,V}_{\langle P,V\rangle}$. Let $X \in R^{Y,\emptyset}_{\langle P,V\rangle}$. Then $\langle X, Y\rangle \in \mathcal{HT}(P)$. We can apply one of the conditions that characterize the possible classes of models of programs in $\mathcal{C}_n$: if $\langle H, T\rangle \in \mathcal{HT}(P)$ and $\langle T', T'\rangle \in \mathcal{HT}(P)$ such that $T \subseteq T'$, then $\langle H, T'\rangle \in \mathcal{HT}(P)$. Since $\langle X, Y\rangle \in \mathcal{HT}(P)$ and $\langle Y \cup V, Y \cup V\rangle \in \mathcal{HT}(P)$ we have that $\langle X, Y \cup V\rangle \in \mathcal{HT}(P)$, and therefore $X \setminus V = X \in R^{Y,V}_{\langle P,V\rangle}$. We can conclude that $R^{Y,\emptyset}_{\langle P,V\rangle} \subseteq R^{Y,V}_{\langle P,V\rangle}$, and so $R^{Y,\emptyset}_{\langle P,V\rangle}$ is the least element of $\mathcal{R}^Y_{\langle P,V\rangle}$. Therefore, $\langle P, V\rangle$ does not satisfy Ω for every $P \in \mathcal{C}_n$ and V such that $|V|= 1$. ■

This result also holds for the class of disjunctive programs, whose proof, which we omit for lack of space, follows a similar strategy.

Proposition 8 *There is a forgetting operator satisfying* **(SP)**$_{\langle P,V\rangle}$*, for every $P \in \mathcal{C}_d$ and every V such that $|V|= 1$.*

As indicated in Thm. 5, no operator in F_{SP} is closed for normal or disjunctive programs, hence quite likely the result will not be applicable throughout an iterative process of forgetting one atom at a time. But for a one-time forgetting operation, they might be useful.

We now provide a general way to determine which sets of atoms can be forgotten from a given program.

Theorem 8 *Let P be a program. Then the set of sets of atoms*

$$\mathcal{V}_P = \{V \subseteq \mathcal{A}(P) \mid \bigcap \mathcal{R}^Y_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle} \text{ for every } \mathcal{R}^Y_{\langle P,V\rangle} \neq \emptyset\}$$

is the set of all sets V of atoms for which it is possible to forget V from P while satisfying **(SP)**$_{\langle P,V\rangle}$*.*

Proof: Let P be a program. Using Cor. 1 we need to check that $\mathcal{V}_P$ is exactly the set of all $V \subseteq \mathcal{A}(P)$ such that the instance $\langle P, V\rangle$ does not satisfies criterion Ω.

Let $V \in \mathcal{V}_P$. By definition of $\mathcal{V}_P$, we have that for every $Y \subseteq \mathcal{A}(P) \setminus V$ either $\mathcal{R}^Y_{\langle P,V\rangle} = \emptyset$ or $\bigcap \mathcal{R}^Y_{\langle P,V\rangle} \in \mathcal{R}^Y_{\langle P,V\rangle}$. Since clearly $\bigcap \mathcal{R}^Y_{\langle P,V\rangle}$ is the least element of $\mathcal{R}^Y_{\langle P,V\rangle}$, $\langle P, V\rangle$ cannot satisfy Ω.

Now let $V \notin \mathcal{V}_P$. Then, by definition of $\mathcal{V}_P$ there exists $Y \subseteq \mathcal{A}(P) \setminus V$ such that $\mathcal{R}^Y_{\langle P,V\rangle} \neq \emptyset$ and $\bigcap \mathcal{R}^Y_{\langle P,V\rangle} \notin \mathcal{R}^Y_{\langle P,V\rangle}$. Suppose that $\mathcal{R}^Y_{\langle P,V\rangle}$ has a least element, call it L. Then $L \subseteq R$, for every $R \in \mathcal{R}^Y_{\langle P,V\rangle}$. Therefore $L \subseteq \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$. Since $L \in \mathcal{R}^Y_{\langle P,V\rangle}$ we have that $\bigcap \mathcal{R}^Y_{\langle P,V\rangle} \subseteq L$. Thus $L = \bigcap \mathcal{R}^Y_{\langle P,V\rangle}$, which contradicts the fact that $\bigcap \mathcal{R}^Y_{\langle P,V\rangle} \notin \mathcal{R}^Y_{\langle P,V\rangle}$. Therefore, $\mathcal{R}^Y_{\langle P,V\rangle}$ has no least element, and we can conclude that $\langle P, V\rangle$ satisfies criterion Ω. ■

Thus, this result provides a general way to obtain all sets of atoms V that can be forgotten from a given program P while preserving **(SP)**. Notably, all possible sets contained in $\mathcal{V}_P$ have to be determined individually as neither sub- nor supersets are necessarily contained in the result. For example, for program P given in Ex. 2, $\mathcal{V}_P$ contains $\{p\}$, $\{q\}$, and, e.g., $\{p, q, a\}$, but not $\{p, q\}$.

7 Conclusions

We have studied forgetting in ASP, focusing on what is perhaps its most crucial property – *strong persistence* **(SP)** – which captures the essence of forgetting in ASP by ensuring that all semantic relations between the atoms not forgotten are preserved.

We began by answering an important open question, showing that it is not always possible to forget a set of atoms while obeying **(SP)**.

Departing from this impossibility result, we conducted a thorough study of the limits of forgetting in ASP, including a necessary and sufficient criterion (Ω) to determine whether a particular set of atoms can be forgotten from a program while obeying **(SP)**. Whereas at a technical level, criterion Ω is closely tied to certain conditions on the HT-models of the program at hand, it seems that what cannot be forgotten from a program are atoms used in rules that are somehow equivalent to *choice rules* [25], and those atoms are pivotal in the sense that they play an active role in determining the truth of other atoms in some answer sets i.e., there are rules whose bodies mention these atoms and they are true at least in some answer sets. Further investigating this conjecture is an interesting line of future work.

We have also introduced a new class of operators that allows us to show how to forget a set of atoms from a given program while preserving **(SP)**. It is worth noting that of the many classes of operators investigated in [15], one, dubbed F_{Sas}, also satisfies **(SP)**. However, F_{Sas} is only partially defined, witnessed by the fact that the only known operator in F_{Sas}, dubbed f_{Sas}, is only defined for a non-standard class of programs (permitting double negation but no disjunctions), and can only be applied to forget about single atoms p if a sufficient (but not necessary) criterion, called p-forgettable (see [16]), holds, which is considerably stronger than Ω, hence unecessarily excluding many possible cases. Nevertheless, for the sake of completeness, we formally relate such operator and the class F_{SP}.

Corollary 2 *Let f_{Sas} be the operator defined for F_{Sas} and P a program for which f_{Sas} is defined. For any $\mathsf{f} \in \mathsf{F}_{\mathsf{SP}}$, if a single atom p is p-forgettable from P, then $\mathcal{HT}(\mathsf{f}(P, \{p\})) = \mathcal{HT}(\mathsf{f}_{Sas}(P, \{p\}))$.*

We also provided a general condition to determine all sets of atoms that can be forgotten from a given program, as well as special cases in which a set of atoms can always be forgotten, namely singleton sets in the case of normal and disjunctive programs.

Left open, for future work, is the definition of a syntactic operator similar in style to that defined for strong as-forgetting [16], as well as its implementation. Also, we may investigate the relation of Ω to projections of answer sets [11] and investigate the limits of forgetting for semantics other than ASP, such as [41] based on the FLP-semantics [38], or [1, 16] based on the well-founded semantics [13].

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their comments, which helped improve this paper. R. Gonçalves, M. Knorr and J. Leite were partially supported by FCT under strategic project NOVA LINCS (PEst/UID/CEC/04516/2013). R. Gonçalves was partially supported by FCT grant SFRH/BPD/100906/2014 and M. Knorr by FCT grant SFRH/BPD/86970/2012.

REFERENCES

[1] José Júlio Alferes, Matthias Knorr, and Kewen Wang, 'Forgetting under the well-founded semantics', in *Procs. of LPNMR*, eds., Pedro Cabalar and Tran Cao Son, volume 8148 of *LNCS*, pp. 36–41. Springer, (2013).

[2] José Júlio Alferes, João Alexandre Leite, Luís Moniz Pereira, Halina Przymusinska, and Teodor C. Przymusinski, 'Dynamic updates of non-monotonic knowledge bases', *The Journal of Logic Programming*, **45**(1-3), 43–70, (September/October 2000).

[3] W. W. Bledsoe and Larry M. Hines, 'Variable elimination and chaining in a resolution-based prover for inequalities', in *Procs. of CADE*, eds., Wolfgang Bibel and Robert A. Kowalski, volume 87 of *LNCS*, pp. 70–87. Springer, (1980).

[4] Pedro Cabalar and Paolo Ferraris, 'Propositional theories are strongly equivalent to logic programs', *TPLP*, **7**(6), 745–759, (2007).

[5] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov, 'Complexity and expressive power of logic programming', *ACM Comput. Surv.*, **33**(3), 374–425, (2001).

[6] James P. Delgrande, Torsten Schaub, Hans Tompits, and Stefan Woltran, 'A model-theoretic approach to belief change in answer set programming', *ACM Trans. Comput. Log.*, **14**(2), 14, (2013).

[7] James P. Delgrande and Kewen Wang, 'A syntax-independent approach to forgetting in disjunctive logic programs', in *Procs. of AAAI*, eds., Blai Bonet and Sven Koenig, pp. 1482–1488. AAAI Press, (2015).

[8] Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits, 'On properties of update sequences based on causal rejection', *Theory and Practice of Logic Programming (TPLP)*, **2**(6), 721–777, (2002).

[9] Thomas Eiter, Michael Fink, Hans Tompits, and Stefan Woltran, 'Simplifying logic programs under uniform and strong equivalence', in *Procs. of LPNMR*, eds., Vladimir Lifschitz and Ilkka Niemelä, volume 2923 of *LNCS*, pp. 87–99. Springer, (2004).

[10] Thomas Eiter, Michael Fink, and Stefan Woltran, 'Semantical characterizations and complexity of equivalences in answer set programming', *ACM Trans. Comput. Log.*, **8**(3), (2007).

[11] Thomas Eiter, Hans Tompits, and Stefan Woltran, 'On solution correspondences in answer-set programming', in *Procs. of IJCAI*, eds., Leslie Pack Kaelbling and Alessandro Saffiotti, pp. 97–102. Professional Book Center, (2005).

[12] Thomas Eiter and Kewen Wang, 'Semantic forgetting in answer set programming', *Artif. Intell.*, **172**(14), 1644–1672, (2008).

[13] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf, 'The well-founded semantics for general logic programs', *J. ACM*, **38**(3), 620–650, (1991).

[14] Michael Gelfond and Vladimir Lifschitz, 'Classical negation in logic programs and disjunctive databases', *New Generation Comput.*, **9**(3-4), 365–385, (1991).

[15] Ricardo Goncalves, Matthias Knorr, and João Leite, 'The ultimate guide to forgetting in ASP', in *Procs. of KR*, eds., Chitta Baral, James Delgrande, and Frank Wolter. AAAI, (2016).

[16] Matthias Knorr and José Júlio Alferes, 'Preserving strong equivalence while forgetting', in *Procs. of JELIA*, eds., Eduardo Fermé and João Leite, volume 8761 of *LNCS*, pp. 412–425. Springer, (2014).

[17] Boris Konev, Michel Ludwig, Dirk Walther, and Frank Wolter, 'The logical difference for the lightweight description logic EL', *J. Artif. Intell. Res. (JAIR)*, **44**, 633–708, (2012).

[18] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter, 'Model-theoretic inseparability and modularity of description logic ontologies', *Artif. Intell.*, **203**, 66–103, (2013).

[19] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev, 'Logic-based ontology comparison and module extraction, with an application to dl-lite', *Artif. Intell.*, **174**(15), 1093–1141, (2010).

[20] Jérôme Lang, Paolo Liberatore, and Pierre Marquis, 'Propositional independence: Formula-variable independence and forgetting', *J. Artif. Intell. Res. (JAIR)*, **18**, 391–443, (2003).

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

[22] Javier Larrosa, 'Boosting search with variable elimination', in *Procs. of CP*, ed., Rina Dechter, volume 1894 of *LNCS*, pp. 291–305. Springer, (2000).

[23] Javier Larrosa, Enric Morancho, and David Niso, 'On the practical use of variable elimination in constraint optimization problems: 'still-life' as a case study', *J. Artif. Intell. Res. (JAIR)*, **23**, 421–440, (2005).

[24] C. I. Lewis, *A survey of symbolic logic*, University of California Press, 1918. Republished by Dover, 1960.

[25] Vladimir Lifschitz, 'What is answer set programming?', in *Procs. of AAAI*, eds., Dieter Fox and Carla P. Gomes, pp. 1594–1597. AAAI Press, (2008).

[26] Vladimir Lifschitz, David Pearce, and Agustín Valverde, 'Strongly equivalent logic programs', *ACM Trans. Comput. Log.*, **2**(4), 526–541, (2001).

[27] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner, 'Nested expressions in logic programs', *Ann. Math. Artif. Intell.*, **25**(3-4), 369–389, (1999).

[28] Fangzhen Lin and Raymond Reiter, 'How to progress a database', *Artif. Intell.*, **92**(1-2), 131–167, (1997).

[29] Yongmei Liu and Ximing Wen, 'On the progression of knowledge in the situation calculus', in *Procs. of IJCAI*, ed., Toby Walsh, pp. 976–982. IJCAI/AAAI, (2011).

[30] Aart Middeldorp, Satoshi Okui, and Tetsuo Ida, 'Lazy narrowing: Strong completeness and eager variable elimination', *Theor. Comput. Sci.*, **167**(1&2), 95–130, (1996).

[31] Yves Moinard, 'Forgetting literals with varying propositional symbols', *J. Log. Comput.*, **17**(5), 955–982, (2007).

[32] David Pearce, Hans Tompits, and Stefan Woltran, 'Characterising equilibrium logic and nested logic programs: Reductions and complexity' ', *TPLP*, **9**(5), 565–616, (2009).

[33] David Rajaratnam, Hector J. Levesque, Maurice Pagnucco, and Michael Thielscher, 'Forgetting in action', in *Procs. of KR*, eds., Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter. AAAI Press, (2014).

[34] Chiaki Sakama and Katsumi Inoue, 'An abductive framework for computing knowledge base updates', *Theory and Practice of Logic Programming (TPLP)*, **3**(6), 671–713, (2003).

[35] Martin Slota and João Leite, 'Robust equivalence models for semantic updates of answer-set programs', in *Procs. of KR*, eds., Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, pp. 158–168. AAAI Press, (2012).

[36] Martin Slota and João Leite, 'A unifying perspective on knowledge updates', in *Procs. of JELIA*, eds., Luis Fariñas del Cerro, Andreas Herzig, and Jérôme Mengin, volume 7519 of *LNAI*, pp. 372–384. Springer, (2012).

[37] Martin Slota and João Leite, 'The rise and fall of semantic rule updates based on se-models', *TPLP*, **14**(6), 869–907, (2014).

[38] Miroslaw Truszczynski, 'Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs', *Artif. Intell.*, **174**(16-17), 1285–1306, (2010).

[39] Yisong Wang, Kewen Wang, and Mingyi Zhang, 'Forgetting for answer set programs revisited', in *Procs. of IJCAI*, ed., Francesca Rossi. IJCAI/AAAI, (2013).

[40] Yisong Wang, Yan Zhang, Yi Zhou, and Mingyi Zhang, 'Forgetting in logic programs under strong equivalence', in *Procs. of KR*, eds., Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, pp. 643–647. AAAI Press, (2012).

[41] Yisong Wang, Yan Zhang, Yi Zhou, and Mingyi Zhang, 'Knowledge forgetting in answer set programming', *J. Artif. Intell. Res. (JAIR)*, **50**, 31–70, (2014).

[42] Zhe Wang, Kewen Wang, Rodney W. Topor, and Jeff Z. Pan, 'Forgetting for knowledge bases in DL-Lite', *Ann. Math. Artif. Intell.*, **58**(1-2), 117–151, (2010).

[43] Andreas Weber, 'Updating propositional formulas', in *Expert Database Conf.*, pp. 487–500, (1986).

[44] Ka-Shu Wong, *Forgetting in Logic Programs*, Ph.D. dissertation, The University of New South Wales, 2009.

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