A Unified View on Forgetting and Strong Equivalence Notions in Answer Set Programming

Zeynep G. Saribatur, Stefan Woltran

Institute of Logic and Computation, TU Wien, Austria
{zeynep.saribatur,stefan.woltran}@tuwien.ac.at

Abstract

Answer Set Programming (ASP) is a prominent rule-based language for knowledge representation and reasoning with roots in logic programming and non-monotonic reasoning. The aim to capture the essence of removing (ir)relevant details in ASP programs led to the investigation of different notions, from strong persistence **(SP)** forgetting, to faithful abstractions, and, recently, strong simplifications, where the latter two can be seen as relaxed and strengthened notions of forgetting, respectively. Although it was observed that these notions are related, especially given that they have characterizations through the semantics for strong equivalence, it remained unclear whether they can be brought together. In this work, we bridge this gap by introducing a novel relativized equivalence notion, which is a relaxation of the recent simplification notion, that is able to capture all related notions from the literature. We provide necessary and sufficient conditions for relativized simplifiability, which shows that the challenging part is for when the context programs do not contain all the atoms to remove. We then introduce an operator that combines projection and a relaxation of **(SP)**-forgetting to obtain the relativized simplifications. We furthermore present complexity results that complete the overall picture.

Introduction

Forgetting or discarding information that are not deemed necessary is crucial in human reasoning, as it allows to focus on the important details and to abstract over the rest. Such active or *intentional* forgetting is argued to enhance decision-making through flexibility under changing conditions and the ability to generalize (Richards and Frankland 2017). Over the years, the desire to abstract over details led to different theories (e.g., (Giunchiglia and Walsh 1992)) and applications of abstraction in various areas of AI, among many are planning (Knoblock 1994), constraint satisfaction (Bistarelli, Codognet, and Rossi 2002), and model checking (Clarke, Grumberg, and Long 1994). Getting rid of (ir)relevant details through forgetting continues to motivate works in different subfields of AI (Beierle and Timm 2019), such as knowledge representation and reasoning (KR) (Eiter and Kern-Isberner 2018) and symbolic machine learning (Siebers and Schmid 2019). Recent examples of forgetting within KR appear in action theories (Luo et al. 2020), explanations for planning (Vasileiou and Yeoh 2022) and argumentation (Berthold, Rapberger, and Ulbricht 2023; Baumann and Berthold 2022).

The theoretical underpinnings of forgetting has been investigated for classical logic and logic programming for over decades. Answer Set Programming (ASP), is a well established logic programming language, characterized by non-monotonic declarative semantics. Its non-monotonic nature resulted in various forgetting operators satisfying different desirable properties (see recent survey (Gonçalves, Knorr, and Leite 2023)). The property *strong persistence* **(SP)** (Knorr and Alferes 2014) is considered to best capture the essence of forgetting in the context of ASP. The aim is to preserve all existing relations between the remaining atoms, by requiring that there be a correspondence between the answer sets of a program before and after forgetting a set of atoms, which is preserved in the presence of additional rules. This correspondence is formally defined as

$$AS(P \cup R)_{|\overline{A}} = AS(f(P, A) \cup R) \tag{1}$$

for all programs R over the universe $\mathcal{U}$ without containing atoms from A, where $f(P, A)$ is the resulting program of applying an operator f on P to forget about the set A of atoms, $AS(\cdot)$ denotes the collection of answer sets of a program, and $AS(\cdot)_{|\overline{A}}$ is their projection onto the remaining atoms.

When nothing is forgotten, **(SP)** matches the notion of *strong equivalence (SE)* (Lifschitz, Pearce, and Valverde 2001) among programs, denoted as $AS(P \cup R) = AS(Q \cup R)$ for all programs R. Gonçalves et al. (2020) showed that **(SP)**-forgetting can only be done when the SE-models of the program adheres to certain conditions, which is motivated by *relativized* strong equivalence (Woltran 2004; Eiter, Tompits, and Woltran 2005), a relaxation of strong equivalence where the context programs can exclude some atoms.

The motivation to obtain ASP programs with a reduced signature also led to notion of abstraction by omission (Saribatur and Eiter 2018) by means of *over-approximation*, i.e., any answer set in program P can be mapped to some answer set in the abstracted program Q, which is denoted by $AS(P)_{|\overline{A}} \subseteq AS(Q)$, and also has been referred as *weakened Consequence* **(wC)** within forgetting (Gonçalves, Knorr, and Leite 2016a). Saribatur and Eiter (2018) introduce a syntactic operator that obtains abstracted programs, and an auto-

A	B	Strong A-simplification relative to B
$\emptyset$	$\emptyset$	equivalence
$\emptyset$	$\mathcal{U}$	Strong Equivalence (Turner 2001)
$\emptyset$	B	relativized Strong Equivalence (Woltran 2004)
A	$\mathcal{U}$	Strong Simplification (Saribatur and Woltran 2023)
A	$\overline{A}$	Strong Persistence (Knorr and Alferes 2014)
A	C	$C \subseteq \overline{A}$, relativized Strong Persistence (this paper)
A	$\emptyset$	Faithful Abstraction (Saribatur and Eiter 2018)

Table 1: Overview of the full spectrum of the relativized strong simplification notion introduced in this paper.

mated abstraction and refinement methodology, that starts with a coarse abstraction and refines it upon encountering *spurious* answer sets (which do not have correspondence in P) until a fine-grained abstraction is achieved.

A desired abstraction property was considered to be *faithfulness* where Q does not contain a spurious answer set, i.e.,

$$AS(P)_{|\overline{A}} = AS(Q), \tag{2}$$

matching an instance of *Consequence Persistence* **(CP)**-forgetting (Wang, Wang, and Zhang 2013). The notion however does not truly preserve the semantics w.r.t. projection. The recent equivalence notion, called *strong simplification* (Saribatur and Woltran 2023), defined as[1]

$$AS(P \cup R)|_{\overline{A}} = AS(Q \cup R_{|\overline{A}}) \tag{3}$$

for all programs R, allows to capture the atoms that can be disregarded from the original program and also the context program, so that the simplified program can reason over the reduced vocabulary while ensuring that the semantics of the original program is preserved w.r.t. projection.

It is known that strong simplifications imply **(SP)**-forgetting (Saribatur and Woltran 2023) and the relation between omission abstraction and forgetting has also been studied (Saribatur and Eiter 2020). The characterizations for all of the mentioned notions have been established through the SE-models of programs, which characterizes strong equivalence. However until now it remained unclear how these notions come together.

In this paper we bridge this gap through a relaxation of the recent simplification notion, where on the context programs we allow for excluding some dedicated atoms: for sets A, B of atoms, we define the notion of *strong A-simplification relative to B* where (3) holds for all programs R over B. All of the above mentioned notions such as (relativized) strong equivalence, strong persistence, faithful abstractions and strong simplifications, then become special cases of this novel relativized equivalence notion, of which a summary can be seen in Table 1. Furthermore we show the conditions for relativized simplifiability and observe that the challenging part is for when the context programs do not contain all the atoms to remove/forget. We then show how the desired simplifications can be obtained by an operator that combines projection and a relaxation of **(SP)**-forgetting.

[1] $R_{|\overline{A}}$ projects the positive body of the rules in R onto $\overline{A}$ and removes the rules with a negative body or head containing an atom from A.

Our main contributions are thus as follows (i) We propose the novel concept of relativized strong simplification between programs, provide the necessary and sufficient conditions for testing relativized strong simplifiability, give semantical characterizations of relativized strong simplifications and discuss the full spectrum of this notion; (ii) we introduce a novel forgetting operator which is a combination of projection and a relaxation of SP-forgetting, which we introduce as relativized SP-forgetting; (iii) we conclude with complexity results.

Background

Answer Set Programming An *(extended) logic program (ELP)* is a finite set of *(extended) rules* of form

$$A_1 \vee \cdots \vee A_l \leftarrow A_{l+1}, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n, \\ not\ not\ A_{n+1}, \ldots, not\ not A_k$$

where A_i ($1 \leq i \leq k$, $0 \leq l \leq m \leq n \leq k$) are atoms from a first-order language, and not is default negation. We also write a rule r as $H(r) \leftarrow B(r)$ or $H(r) \leftarrow B^+(r),\ not\ B^-(r),\ not\ not\ B^{--}(r)$. We call $H(r) = \{A_1, \ldots, A_l\}$ the *head* of r, $B^+(r) = \{A_{l+1}, \ldots, A_m\}$ the *positive body*, $B^-(r) = \{A_{m+1}, \ldots, A_n\}$ the *negative body* and $B^{--}(r) = \{A_{n+1}, \ldots, A_k\}$ the *double-negated body* of r. If $H(r) = \emptyset$, then r is a *constraint*. A rule r is *disjunctive* if $k = n$; if, in addition, $l \leq 1$ then r is *normal*; r is *positive* if $k = m$ and it is a (non-disjunctive) *fact* if $B(r) = \emptyset$ and $l \leq 1$; for $H(r) = \emptyset$, we occasionally write $\perp$.

In the what follows, we focus on propositional programs over a set of atoms from universe $\mathcal{U}$. Programs with variables reduce to their ground versions as usual. Unless stated otherwise the term *program* refers to a (propositional) ELP.

Let $I \subseteq \mathcal{U}$ be an interpretation. The *GL-reduct* of a program P w.r.t. I is given by $P^I = \{H(r) \leftarrow B^+(r) \mid r \in P, B^-(r) \cap I = \emptyset, B^{--}(r) \subseteq I\}$. An interpretation I is a *model* of a program P (in symbols $I \models P$) if, for each $r \in P$, $(H(r) \cup B^-(r)) \cap I \neq \emptyset$ or $(B^+(r) \cup B^{--}(r)) \not\subseteq I$; I is an *answer set*, if it is a minimal model of P^I. We denote the set of all answer sets by $AS(P)$. Two programs P_1, P_2 are *equivalent* if $AS(P_1) = AS(P_2)$ and *strongly equivalent (SE)*, denoted by $P_1 \equiv P_2$, if $AS(P_1 \cup R) = AS(P_2 \cup R)$ for every R over $\mathcal{U}$.

An *SE-interpretation* is a pair $\langle X, Y \rangle$ such that $X \subseteq Y \subseteq \mathcal{U}$; it is *total* if $X = Y$ and *non-total* otherwise. An SE-interpretation $\langle X, Y \rangle$ is an *SE-model* of a program P if $Y \models P$ and $X \models P^Y$. The set of all SE-models of P is denoted by $SE(P)$. Note that a set Y of atoms is an answer set of P if $\langle Y, Y \rangle \in SE(P)$ and no non-total $\langle X, Y \rangle \in SE(P)$ exists. Two programs P_1 and P_2 are strongly equivalent iff $SE(P_1) = SE(P_2)$ (Turner 2001).

Lastly, for a set $S \subseteq \mathcal{U}$ of atoms, $S_{|A}$ denotes the projection to the atoms in A and $\overline{S}$ is a shorthand for $\mathcal{U} \setminus S$. We also use the notion on pairs, i.e. $\langle X, Y \rangle_{|A} = \langle X_{|A}, Y_{|A} \rangle$ and on sets of objects, i.e. $\mathcal{S}_{|A} = \{S_{|A} \mid S \in \mathcal{S}\}$.

We next summarize the notions needed for our purposes.

Relativized Equivalence Woltran (2004) relaxed the notion of strong equivalence to have the added programs, R, in a specific language $B \subseteq \mathcal{U}$. Its semantical characterization requires a generalization of SE-models as follows.

Definition 1. *A pair of interpretations $\langle X, Y\rangle$ is a (relativized) B-SE-interpretation iff either $X = Y$ or $X \subset (Y \cap B)$. The former are called total and the latter non-total B-SE-interpretations. Moreover, a B-SE-interpretation $\langle X, Y\rangle$ is a (relativized) B-SE-model of a program P iff:*

(i) $Y \models P$;
(ii) for all $Y' \subset Y$ with $(Y' \cap B) = (Y \cap B)$, $Y' \not\models P^Y$; and
(iii) $X \subset Y$ implies existence of a $X' \subseteq Y$ with $X' \cap B = X$, such that $X' \models P^Y$ holds.

The set of B-SE-models of P is given by $SE^B(P)$.

Two programs P_1 and P_2 are strongly equivalent relative to B iff $SE^B(P_1) = SE^B(P_2)$.

Forgetting We refer to (Eiter and Kern-Isberner 2018; Gonçalves, Knorr, and Leite 2023) for recent surveys on forgetting, and briefly define **(SP)**-forgetting. For a class F of forgetting operators and a class $\mathcal{C}$ of programs

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

Here $f(P, A)$ denotes the result of forgetting about A from P. Strong persistence is also considered for a particular forgetting instance $\langle P, A\rangle$, for $P \in \mathcal{C}$ and $A \subseteq \mathcal{U}$, denoted by **(SP)**$_{\langle P,A\rangle}$. Gonçalves, Knorr, and Leite (2016b) introduce a criterion Ω to characterize the instances for which an operator achieving **(SP)**$_{\langle P,A\rangle}$ is impossible, which has relations with $\overline{A}$-SE-models as shown below.

Definition 2. *Let P be a program over $\mathcal{U}$ and $A \subseteq \mathcal{U}$. An instance $\langle P, A\rangle$ satisfies criterion Ω if there exists $Y \subseteq \mathcal{U} \setminus A$ such that the set of sets*

$$\mathcal{R}^Y_{\langle P,A\rangle} = \{\{X \setminus A \mid \langle X, Y \cup A'\rangle \in SE^{\overline{A}}(P)\} \mid A' \subseteq A, \langle Y \cup A', Y \cup A'\rangle \in SE^{\overline{A}}(P)\}$$

is non-empty and has no least element.

It is not possible to forget about A from P while satisfying strong persistence exactly when $\langle P, A\rangle$ satisfies criterion Ω.

Gonçalves, Knorr, and Leite (2016b) also show that the resulting program obtained from forgetting A from program P by applying an operator f from the class F_{SP} of **(SP)**-forgetting operators has the SE-models over $\overline{A}$ as $SE(f(P, A)) = \{\langle X, Y\rangle \mid Y \subseteq \mathcal{U} \setminus A \wedge X \in \bigcap \mathcal{R}^Y_{\langle P,A\rangle}\}$.

Abstraction and Simplification The general notion of abstraction as an over-approximation is defined as follows.

Definition 3 ((Saribatur and Eiter 2018))**.** *For programs P (over $\mathcal{U}$) and Q (over $\mathcal{U}'$) with $|\mathcal{U}| \geq |\mathcal{U}'|$, and a mapping $m : \mathcal{U} \to \mathcal{U}' \cup \{\top\}$, Q is an* abstraction *of P w.r.t. m, if $m(AS(P)) \subseteq AS(Q)$.*

For an *omission abstraction mapping* that omits a set A of atoms from $\mathcal{U}$, it becomes $AS(P)_{|\overline{A}} \subseteq AS(Q)$. An abstraction Q is called *faithful* if $AS(P)_{|\overline{A}} = AS(Q)$.

Saribatur and Woltran (2023) generalized this notion for disjunctive logic programs (DLP) to consider newly added rules or facts that also get abstracted. For that they consider context programs R over $\mathcal{U}$ to be A-*separated*, which means they are of form $R = R_1 \cup R_2$ for programs R_1 and R_2 that are defined over $\mathcal{U} \setminus A$ and A, respectively.

Definition 4 ((Saribatur and Woltran 2023))**.** *Given $A \subseteq \mathcal{U}$ and a program P (over $\mathcal{U}$), a program Q (over $\mathcal{U} \setminus A$) is* a strong A-simplification of P *if for any program R over $\mathcal{U}$ that is A-separated, we have*

$$AS(P \cup R)_{|\overline{A}} = AS(Q \cup R_{|\overline{A}}) \tag{4}$$

We say that P is strong A-simplifiable if there is a program Q such that (4) holds.

It was shown that the SE-models of P need to satisfy the below conditions, where A is semantically behaving as facts, in order to ensure the existence of such a simplification.

Theorem 1 ((Saribatur and Woltran 2023))**.** *There exists a strong A-simplification of P iff P satisfies the following*

Δ_{s_1}*: $\langle Y, Y\rangle \in SE(P)$ implies $A \subseteq Y$.*
Δ_{s_2}*: For any $\langle X, Y\rangle \in SE(P)$, $X_{|\overline{A}} = Y_{|\overline{A}}$ implies $X = Y$.*
Δ_{s_3}*: $\langle X, Y\rangle \in SE(P)$ implies $\langle X \cup (Y \cap A), Y\rangle \in SE(P)$.*

The simplifications are shown to have SE-models equaling $SE(P)_{|\overline{A}}$. For strong A-simplifiable programs, projecting away the atoms in A achieves the desired simplification.

Theorem 2 ((Saribatur and Woltran 2023))**.** *Let P be a strong A-simplifiable program. Then $P_{|\overline{A}}$ is a strong A-simplification of P.*

Here $P_{|\overline{A}}$ refers to removing the atoms in A from the positive bodies of rules, and omitting the rule all together if an atom from A appears in the negative body or the head.

We will later show that such a projection can still be partially applicable in the relaxation of the simplification notion, while an additional operator more close to forgetting will be needed as well.

Relaxing Strong Simplifications

A natural relaxation for strong simplification is to allow excluding some atoms from the added programs. Thus we propose the following notion.

Definition 5. *Given $A, B \subseteq \mathcal{U}$ and a program P (over $\mathcal{U}$), a program Q (over $\mathcal{U} \setminus A$) is* a (strong) A-simplification of P relative to B *if for any program R over B that is A-separated, we have*

$$AS(P \cup R)_{|\overline{A}} = AS(Q \cup R_{|\overline{A}}) \tag{5}$$

We say that P is B-relativized (strong) A-simplifiable if there is a program Q such that (5) holds.

This relaxed notion of strong simplifiability allows to identify programs which are originally not strong simplifiable, but are relativized strong simplifiable when some atoms are not taken into account in the context programs. Below are examples of such programs.

Example 1. *Let program P_1 consist of rules*

$$a \leftarrow b, c. \quad c \leftarrow d. \quad b.$$

and program P_2 consist of rules

$$a \leftarrow not\ b. \quad b \leftarrow not\ a. \quad c.$$

P_1 and P_2 are not strong $\{b, c\}$-simplifiable (though the programs $P_1 \cup \{c.\}$ and $P_2 \cup \{b.\}$ are).

However P_1 is strong $\{b, c\}$-simplifiable relative to $\{a, b, d\}$, since the program $Q_1 = \{a \leftarrow d.\}$ is such a simplification, and P_2 is strong $\{b, c\}$-simplifiable relative to $\{a, c\}$, since the program $Q_2 = \{a \leftarrow not\ not\ a.\}$ is such a simplification.

Note that in Definition 5 there are no restrictions on how B and A might relate. Thus there can be cases where not all atoms in the set A appear in R. We will see that such cases are the cause for the notion of relativized (strong) simplifications being more challenging than strong simplifications.

The context programs that do not contain any atoms from A would be trivially A-separated, thus the relativized simplification notion gets reduced to **(SP)**-forgetting.

Proposition 3. *A forgetting operator f satisfies* **(SP)**$_{\langle P,A\rangle}$ *iff $f(P, A)$ is an A-simplification of P relative to $\overline{A}$.*

Similar to strong simplifications, not every program might have a relativized simplification. By investigating the undesired case that prevents a program from being relativized simplifiable, which is similar to Proposition 2 from (Saribatur and Woltran 2023), thus omitted for brevity, we obtain our first result which adjusts the conditions in Theorem 1 to the relativized case considering the B-SE models.[2]

Proposition 4 ($\star$). *Let P be a program and A, B be sets of atoms. If there exists an A-simplification of P relative to B then P satisfies following*

$\Delta^r_{s_1}$: *$\langle Y, Y\rangle \in SE^B(P)$ implies $A \cap B \subseteq Y$.*

$\Delta^r_{s_2}$: *For any $\langle X, Y\rangle \in SE(P)$ with $\langle Y, Y\rangle \in SE^B(P)$, $X_{|\overline{A}} = Y_{|\overline{A}}$ implies $X = Y$.*

$\Delta^r_{s_3}$: *$\langle X, Y\rangle \in SE^B(P)$ implies $\langle X \cup (Y \cap (A \cap B)), Y\rangle \in SE^B(P)$.*

One can see that the restrictive conditions that were required from the SE-models in Theorem 1 are relaxed to only hold for the B-SE-models, since those are the ones of importance for the answer sets of $P \cup R$ for R over B.

We shortly say that P *satisfies* Δ^r if it satisfies the conditions $\Delta^r_{s_i}$ for $1 \le i \le 3$. The following example illustrates checking the Δ^r conditions.

Example 2 (Ex. 1 ctd). *The SE-models of P_1 are*

$$\begin{array}{cccc} \langle bcad, bcad\rangle & \langle bca, bca\rangle & \langle ba, ba\rangle & \langle b, b\rangle \\ \langle bca, bcad\rangle & \langle ba, bca\rangle & \langle b, ba\rangle & \\ \langle ba, bcad\rangle & \langle b, bcad\rangle & \langle b, bca\rangle & \end{array}$$

For $B = \{a, b, d\}$, $SE^B(P_1) = \{\langle bcad, bcad\rangle, \langle ba, ba\rangle, \langle b, b\rangle, \langle ba, bcad\rangle, \langle b, bcad\rangle, \langle b, ba\rangle\}$. Now for $A = \{b, c\}$, we can easily see that $\Delta^r_{s_1}$ and $\Delta^r_{s_3}$ are satisfied since each B-SE-model contains $A \cap B = \{b\}$, and $\Delta^r_{s_2}$ is trivially satisfied since there is no relevant model.

Observe that, for $B = \mathcal{U}$, the conditions $\Delta^r_{s_i}$ become the same with the conditions Δ_{s_i} of strong simplification, for $1 \le i \le 3$. On the other hand, if B is such that $A \cap B = \emptyset$, i.e., $B \subseteq \overline{A}$, the conditions become immaterial.

Proposition 5. *Any program P satisfies Δ^r, for any A, B with $B \subseteq \overline{A}$.*

Proof (Sketch). $\Delta^r_{s_1}$ and $\Delta^r_{s_3}$ trivially holds as $A \cap B = \emptyset$. For some $\langle X, Y\rangle \in SE(P)$ to violate $\Delta^r_{s_2}$, X and Y need to differ on the atoms from A, while $X \cap \overline{A} = Y \cap \overline{A}$ holds which contradicts $\langle Y, Y\rangle \in SE^B(P)$. ☐

Unsurprisingly, the Δ^r conditions are not sufficient for B-relativized A-simplifiability in general. This can easily be seen for the case when the context programs do not contain atoms to remove, making use of Proposition 3 and the knowledge that not every program has a set of atoms which can be forgotten by satisfying **(SP)**.

Example 3. *Let program P_3 consist of rules*

$$a \leftarrow p. \quad b \leftarrow q. \quad p \leftarrow not\ q. \quad q \leftarrow not\ p.$$

For $A = \{p, q\}$ and $B = \{a, b\}$, P_3 satisfies Δ^r (Proposition 5), but is not B-relativized A-simplifiable, since no forgetting operator satisfies **(SP)**$_{\langle P_3,A\rangle}$ *(Gonçalves, Knorr, and Leite 2016b).*

Note that, when $A \subseteq B$, due to the definition of B-SE-models, in order to satisfy $\Delta^r_{s_3}$ there cannot be non-total $\langle X, Y\rangle \in SE^B(P)$ with $X \subset Y \cap B$. In addition to $\Delta^r_{s_1}$ and $\Delta^r_{s_2}$, these become quite restrictive conditions on the SE-models of P. In fact, as we shall see later, the Δ^r condition turns out to be sufficient for relativized simplifiability when $A \subseteq B$. Though first we need to understand the semantical characterization of such simplifications.

From B-SE-Models to A-B-SE-Models

We investigate the semantical characterization of relativized simplifications of a program. For that we first introduce the following notion of A-B-SE-models, which project those B-SE-models of importance w.r.t. A.

Definition 6. *Given program P over $\mathcal{U}$ and $A, B \subseteq \mathcal{U}$, the A-B-SE-models of P are given by the set*

$$\begin{aligned} SE^B_A(P) = &\{\langle Y_{|\overline{A}}, Y_{|\overline{A}}\rangle \mid \langle Y, Y\rangle \in SE^B(P)\} \cup \\ &\{\langle X_{|\overline{A}}, Y_{|\overline{A}}\rangle \mid \langle X, Y\rangle \in SE^B(P), X \subset Y, \\ &\text{and for all } \langle Y', Y'\rangle \in SE^B(P) \text{ with } Y'_{|\overline{A}} = Y_{|\overline{A}}, \\ &\langle X', Y'\rangle \in SE^B(P) \text{ with } X'_{|\overline{A}} = X_{|\overline{A}}\} \end{aligned}$$

The set of A-B-SE-models collects the projection of all total B-SE-models $\langle Y, Y\rangle$ and all non-total B-SE-models $\langle X, Y\rangle$ for which a respective non-total B-SE-model $\langle X', Y'\rangle$ can be found that agree on the projection, among all total B-SE-models $\langle Y', Y'\rangle$ that agree on the projection with $\langle Y, Y\rangle$.

[2]For proofs of theorems marked by $\star$ see the extended version at http://arxiv.org/abs/2312.07993.

Example 4 (Ex. 1 ctd). *For* $A{=}\{b,c\}$ *and* $B{=}\{a,b,d\}$*, none of the total* B*-SE-models agree on the projection onto* $\overline{A} = \{a,d\}$*. So* SE^B_A *simply collects the projection of those models and their non-total models. Thus* $SE^B_A(P_1) = \{\langle ad,ad\rangle, \langle a,a\rangle, \langle\emptyset,\emptyset\rangle\} \cup \{\langle a,ad\rangle, \langle\emptyset,ad\rangle, \langle\emptyset,a\rangle\}$*.*

Now assume that another program P_1' *has the SE-models* $SE(P_1') = SE(P_1) \setminus \langle ba,bca\rangle$*. Then* $\langle bca,bca\rangle$ *is added to* $SE^B(P_1')$ *in addition to* $SE^B(P_1)$*. Since* $\langle bca,bca\rangle_{|\{a,d\}} = \langle ba,ba\rangle_{|\{a,d\}} = \langle a,a\rangle$*, in order for* $\langle\emptyset,a\rangle$ *to be in* $SE^B_A(P_1')$ *there needs to be some non-total* B*-SE-model of form* $\langle .,bca\rangle$ *that can be projected onto* $\langle\emptyset,a\rangle$ *which is not the case. Thus* $SE^B_A(P_1') = SE^B_A(P_1) \setminus \langle\emptyset,a\rangle$*.*

For referring to the relativized SE-models of the simplifications in our next result, let us introduce a notation for the set of SE-models of a program over A_1 relativized to A_2.

Definition 7. *Let* P *be a program. The relativization of SE-models of* P *over* A_1 *to the set* A_2 *of atoms is denoted by*

$$SE^{A_1,A_2}(P) = \{\langle X,Y\rangle \mid \langle X,Y\rangle \in SE^{A_2}(P), Y \subseteq A_1\}.$$

Interestingly, the relativized SE-models of any A-simplification for a program P relative to B, if exists, need to adhere with the A-B-SE-models of P.

Proposition 6 ($\star$). *If* Q *is an* A*-simplification for* P *relative to* B*, then it satisfies*

$$SE^B_A(P) = SE^{\overline{A},B\setminus A}(Q). \tag{6}$$

Example 5 (Ex. 1 ctd). *Equation* (6) *holds for* P_1 *and* Q_1*.*

Now we can show the sufficiency of Δ^r for when $A \subseteq B$.

Theorem 7. *Given program* P *over* $\mathcal{U}$ *and* $A,B \subseteq \mathcal{U}$ *such that* $A \subseteq B$*.* P *satisfies* Δ^r *iff there exists an* A*-simplification of* P *relative to* B*.*

Proof (Sketch). When $A{\subseteq}B$, due to $\Delta^r_{s_1}$, there cannot be different total B-SE-models agreeing on the projection, thus $SE^B_A(P)$ amounts to $SE^B(P)_{|\overline{A}}$. Due to Proposition 4, what remains is to show that a program with $(B\setminus A)$-SE-models matching $SE^B(P)_{|\overline{A}}$ is a relativized simplification of P, which follows a very similar proof to that of Theorem 1. □

This result shows that the challenge of relativized simplifiability is in fact due to having atoms to be removed that do not appear in the context programs. Then the B-SE-models might differ in terms of the atoms from $A \setminus B$ which cannot be distinguished in the projection, making the Δ^r condition no longer sufficient. Thus in order to characterize relativized simplifiability in general, an additionaly property is needed.

Characterizing Relativized Simplifiability

We introduce the following criterion that will help us in obtaining the sufficient conditions.

Definition 8. *Let* P *be a program over* $\mathcal{U}$ *and* $A,B \subseteq \mathcal{U}$*.* P *satisfies criterion* $\Omega_{A,B}$ *if there exists* $Y \subseteq \mathcal{U} \setminus A$ *such that the set of sets*

$$\mathcal{R}^Y_{\langle P,A,B\rangle} = \{\{X\setminus A \mid \langle X, Y\cup A'\rangle \in SE^B(P)\} \mid A' \subseteq A, \langle Y\cup A', Y\cup A'\rangle \in SE^B(P)\}$$

is non-empty and has no least element.

The difference of $\Omega_{A,B}$ from Ω is that instead of $\overline{A}$-SE-models, now B-SE-models are taken into account. In fact, $\mathcal{R}^Y_{\langle P,A,\overline{A}\rangle} = \mathcal{R}^Y_{\langle P,A\rangle}$. Thus we have the following.

Proposition 8. P *does not satisfy* $\Omega_{A,\overline{A}}$ *iff* $\langle P,A\rangle$ *does not satisfy* Ω*.*

We illustrate how the criterion $\Omega_{A,B}$ can be checked with the following example.

Example 6. *Consider a program* P *with SE-models*

$$\begin{array}{ccc}\langle abc,abc\rangle & \langle abd,abd\rangle & \langle abcd,abcd\rangle\\ \langle ac,abc\rangle & \langle bd,abd\rangle & \langle abc,abcd\rangle\\ \langle ac,abcd\rangle & \langle bd,abcd\rangle & \langle abd,abcd\rangle\end{array}$$

Let $A = \{c,d\}$ *and* $B = \{a,b,c\}$*. For* $Y = \{a,b\}$*, only* $\langle abc,abc\rangle$ *and* $\langle abd,abd\rangle$ *appear in* $SE^B(P)$*. There are* $\langle abc,abc\rangle, \langle ac,abc\rangle \in SE^B(P)$ *of form* $\langle X,abc\rangle$*, and* $\langle abd,abd\rangle, \langle bd,abd\rangle \in SE^B(P)$ *of form* $\langle X,abd\rangle$*. Thus* $\mathcal{R}^{\{a,b\}}_{\langle P,A,B\rangle} = \{\{\{a,b\},\{a\}\},\{\{a,b\},\{b\}\}\}$ *is non-empty and does not have a least element. So* $\Omega_{A,B}$ *is satisfied.*

As the set $\mathcal{R}^Y_{\langle P,A,B\rangle}$ collects all B-SE-models that can be projected onto $\langle\cdot,Y\rangle$, the criterion $\Omega_{A,B}$ can in fact be characterized through A-B-SE-models.

Proposition 9 ($\star$). P *does not satisfy criterion* $\Omega_{A,B}$ *iff* $\{\langle X,Y\rangle \mid Y \subseteq \mathcal{U}\setminus A \wedge X \in \bigcap \mathcal{R}^Y_{\langle P,A,B\rangle}\} = SE^B_A(P)$*.*

It turns out that criterion $\Omega_{A,B}$ is also necessary for relativized simplifiability.

Proposition 10 ($\star$). *If* P *satisfies criterion* $\Omega_{A,B}$*, then a* B*-relativized* A*-simplification cannot exist.*

We next show that the Δ^r conditions together with the $\Omega_{A,B}$ criterion are sufficient for relativized simplifiability.

Theorem 11 ($\star$). *If* P *satisfies* Δ^r *and does not satisfy criterion* $\Omega_{A,B}$ *then* P *is* B*-relativized* A*-simplifiable.*

The result is obtained by proving that a program Q satisfying (6) is a relativized simplification of P.

Thus we have the necessary and sufficient conditions on B-relativized A-simplifiability, while also providing a characterization for such simplifications, when exist.

Corollary 12. P *satisfies* Δ^r *and does not satisfy criterion* $\Omega_{A,B}$ *iff* P *is* B*-relativized* A*-simplifiable.*

The Full Spectrum

We now discuss how the notion of relativized simplification captures all the related notions from the literature (shown in Table 1), by looking at the special cases of removing atoms and relativization.

When $A{=}\emptyset$, the Δ^r conditions are trivially satisfied, and as $SE^B_\emptyset(P){=}SE^B(P)$, the criterion $\Omega_{\emptyset,B}$ is not satisfied for any B. Also any program is $\emptyset$-separated. Then B-relativized $\emptyset$-simplification simply amounts to relativized strong equivalence (Woltran 2004). If, in addition, we set $B{=}\mathcal{U}$, we reach strong equivalence.

Proposition 13. P *and* Q *are strong equivalent relative to* B *iff* Q *is a* $\emptyset$*-simplification relative to* B*. Every program is* B*-relativized* $\emptyset$*-simplifiable, for any* B*.*

For $B = \emptyset$, the $\emptyset$-SE models only contain the total models of the answer sets, which captures the notion of faithful omission abstraction. Moreover Δ^r is trivially satisfied and $\Omega_{A,\emptyset}$ is not satisfied for any A.

Proposition 14. *Q is an A-simplification of P relative to $\emptyset$ iff Q is a faithful abstraction of P for omission of A. Every program is $\emptyset$-relativized A-simplifiable, for any A.*

When we additionally set A to $\emptyset$, then we reach equivalence of two programs.

Setting $A = \mathcal{U}$ results in omitting all the atoms from the program. Thus as potential relativized simplifications over $\emptyset$ we have either $Q = \emptyset$ or $Q' = \{\bot \leftarrow .\}$. From above we know that every program is $\emptyset$-relativized $\mathcal{U}$-simplifiable. So a satisfiable program has Q as its relativized simplification, while an unsatisfiable program has Q'. Though for $B \neq \emptyset$, relativized simplifiability might not always hold.

When the context programs are set to be over the remaining atoms, i.e., $B = \overline{A}$, we reach **(SP)**-forgetting. In the next section we introduce a relativization of **(SP)**-forgetting to consider the case of $B \subseteq \overline{A}$, which will be needed in defining operators that obtain the relativized simplifications.

Combination of Projection and Forgetting

In this section we introduce an operator that can achieve relativized simplifications. As we know, whenever a relativized simplification exists, it satisfies the equation (6). Following the notation from forgetting operators, we introduce a class of operators that achieves these simplifications. For this, instead of a forgetting instance $\langle P, A\rangle$ we consider *relativized forgetting instance* $\langle P, A, B\rangle$.

Definition 9. *Let F_{rSS} be the class of forgetting operators defined by the following set: $\{f \mid SE^{\overline{A},B\setminus A}(f(P,A,B)) = \{\langle X,Y\rangle \mid Y \subseteq \mathcal{U}\setminus A \wedge X \in \bigcap \mathcal{R}^Y_{\langle P,A,B\rangle}\}\}$.*

Proposition 9 and Corollary 12 leads to the following.

Corollary 15. *For $f \in F_{\text{rSS}}$, $f(P,A,B)$ is a B-relativized A-simplification of P iff P is B-relativized A-simplifiable.*

Note that above class of operators is similar to F_{SP}, where instead of $\mathcal{R}^Y_{\langle P,A\rangle}$ we focus on $\mathcal{R}^Y_{\langle P,A,B\rangle}$, and instead of giving the characterization over the SE-models of the resulting program, we consider its B-SE models.

Clearly when $B = \overline{A}$, since $\mathcal{R}^Y_{\langle P,A,\overline{A}\rangle} = \mathcal{R}^Y_{\langle P,A\rangle}$, the resulting program after applying an operator in F_{rSS} is strongly equivalent to the result of an **(SP)**-forgetting operator.

Proposition 16. *Let P be a program, $A \subseteq \mathcal{U}$, $f_{SP} \in F_{\text{SP}}$, $f_{rSS} \in F_{\text{rSS}}$. Then $f_{SP}(P,A) \equiv f_{rSS}(P,A,\overline{A})$.*

Thus any operator in F_{rSS} can be applied as an **(SP)**-forgetting operator. As we shall show next, the forgetting operators in F_{rSS} can also be used to achieve a relaxed notion of **(SP)**-forgetting.

Relativized Strong Persistence

We introduce a relaxed **(SP)**-forgetting notion, where non-forgotten atoms can be excluded from the context program.

Definition 10. *A forgetting operator f satisfies* relativized strong persistence *for a relativized forgetting instance $\langle P,A,S\rangle$, $S \subseteq \overline{A}$, denoted by* **(rSP)**$_{\langle P,A,S\rangle}$*, if for all programs R over S, $AS(f(P,A,S)\cup R) = AS(P \cup R)_{|\overline{A}}$.*

Above definition naturally leads to the following.

Proposition 17. *If a forgetting operator f satisfies* **(SP)**$_{\langle P,A\rangle}$ *then it satisfies* **(rSP)**$_{\langle P,A,S\rangle}$*, for any $S \subseteq \overline{A}$.*

In fact, every operator in F_{rSS} satisfies **(rSP)**$_{\langle P,A,S\rangle}$, when possible, which we get by Proposition 5 and Corollary 15.

Theorem 18. *Every $f \in F_{\text{rSS}}$ satisfies* **(rSP)**$_{\langle P,A,S\rangle}$*, $S \subseteq \overline{A}$, for every relativized forgetting instance $\langle P,A,S\rangle$, where P does not satisfy $\Omega_{A,S}$.*

An Operator That Projects and Forgets

We begin with showing that as long as the Δ^r conditions are satisfied for those of A which appear in B it is possible to project them away[3] from P while preserving the semantics.

We observe that the relativized SE-models of the resulting program after the projecting away atoms occurring in $A \cap B$ equals the B-SE-models of P projected onto the remaining atoms. This then also equals its $(A \cap B)$-B-SE-models.

Proposition 19 ($\star$)**.** *Let P satisfy Δ^r for $A, B \subseteq \mathcal{U}$, and $A' = A \cap B$. Then it holds that*

$$SE^{\overline{A'},B\setminus A}(P_{|\overline{A'}}) = SE^B(P)_{|\overline{A'}} = SE^B_{A'}(P).$$

This observation leads to the following result.

Corollary 20. *Let P satisfy Δ^r for $A, B \subseteq \mathcal{U}$, and $A' = A \cap B$. $P_{|\overline{A'}}$ is an A'-simplification of P relative to B.*

This result shows that whenever P satisfies Δ^r, even if $\Omega_{A,B}$ criterion is satisfied, preventing B-relativized A-simplifiability, it is still possible to project away atoms in $B \cap A$ to reach a program with a reduced signature.

Interestingly, if a program is B-relativized A-simplifiable, obtaining the desired simplification is possible by first syntactically projecting away those atoms in $B \cap A$ and then applying an operator from F_{rSS} for **(rSP)**-forgetting that forgets those atoms remaining outside of the context programs.

Theorem 21. *Let P be B-relativized A-simplifiable, and $f \in F_{\text{rSS}}$. Then $f(P_{|\overline{A\cap B}}, A\setminus B, B\setminus A)$ is a B-relativized A-simplification of P.*

Proof (Sketch). By Proposition 19, the $(B\setminus A)$-SE-models of $P' = P_{|\overline{A\cap B}}$ amount to $SE^B(P)_{|\overline{A\cap B}}$. Thus for any Y, $\bigcap \mathcal{R}^Y_{\langle P',A\setminus B,B\setminus A\rangle} = \bigcap \mathcal{R}^Y_{\langle P,A,B\rangle}$, which means that if f achieves a B-relativized A-simplification of P with $f(P,A,B)$, then it can achieve such a simplification with $f(P_{|\overline{A\cap B}}, A\setminus B, B\setminus A)$ as well. □

We see that the challenging part of obtaining a relativized simplification when there are atoms to remove that do not appear in the context programs brings us closer to **(SP)**-forgetting. In order to obtain a fully syntactic operator, an interesting follow-up work would be to see whether the existing syntactic **(SP)**-forgetting operators (Berthold et al. 2019; Berthold 2022) can be adjusted to consider **(rSP)**.

[3]When projecting from ELPs, the atoms in A are removed from the double negated body of the rules as well.

Computational Complexity

We provide the complexity of deciding simplifiability through checking the Δ^r and $\Omega_{A,B}$ conditions, and simplification testing. We assume familiarity with basic concepts of complexity theory. For comprehensive details we refer to (Papadimitriou 2003; Arora and Barak 2009).

We begin with checking the Δ^r conditions.

Proposition 22. *Let P be a program over $\mathcal{U}$ and $A, B \subseteq \mathcal{U}$. Deciding whether P satisfies Δ^r is in Π_2^P.*

Violation of any $\Delta^r_{s_i}$ can be checked in Σ_2^P since B-SE-model checking is in D^P (Eiter, Fink, and Woltran 2007).

Next we move on to checking the $\Omega_{A,B}$ criterion. For this we follow the results from (Gonçalves et al. 2020), with the condition that the given program satisfies Δ^r. Remember that for the case of **(SP)**-forgetting, Δ^r is trivially satisfied. For the below two results, we make use of $\Delta^r_{s_1}$ and $\Delta^r_{s_3}$, which gives us that whenever $\langle Y, Y\rangle \in SE^B(P)$ some $\langle X, Y\rangle \in SE^B(P)$ exists iff $\langle X \cup (A \cap B), Y\rangle \in SE^B(P)$.

Proposition 23. *Given program P satisfying Δ^r for $A, B \subseteq \mathcal{U}$, (i) given SE-interpretation $\langle X, Y\rangle$ with $Y \subseteq \mathcal{U} \setminus A$, deciding whether $X \in \bigcap \mathcal{R}^Y_{\langle P,A,B\rangle}$ is in Π_2^P; (ii) deciding whether P satisfies criterion $\Omega_{A,B}$ is Σ_3^P-complete.*

Proof (Sketch). For the complemetary problem, it suffices to guess an $A' \supseteq A \cap B$, and check that $\langle Y \cup A', Y \cup A'\rangle \in SE^B(P)$ and $\langle X \cup (A \cap B), Y \cup A'\rangle \notin SE^B(P)$ (the former ensures also that $\mathcal{R}^Y_{\langle P,A,B\rangle} \neq \emptyset$). (i) then follows by B-SE-model checking being in D^P. For (ii), we just need to additionally guess Y and check that $\mathcal{R}^Y_{\langle P,A,B\rangle}$ is non-empty (see above) and has no least element. For the latter, we additionally guess X and use (i) together with Proposition 9. Σ_3^P-hardness follows from the special case $B = \overline{A}$, cf. Thm 16, (Gonçalves et al. 2020), where Δ^r is trivially satisfied. ☐

Recall Corollary 12. The results in Proposition 22 and Proposition 23 are then used to determine the complexity of deciding relativized simplifiability.

Theorem 24. *Let P be a program over $\mathcal{U}$, and $A, B \subseteq \mathcal{U}$. Deciding whether P is A-simplifiable relative to B is in Π_3^P.*

By making use of the characterizing equality (6), Propositions 9 and 23, we finally provide the complexity result for relativized simplification testing.

Theorem 25. *Given program P which is B-relativized A-simplifiable, and program Q, checking whether Q is a B-relativized A-simplification of P is Π_3^P-complete.*

Proof (Sketch). Making use of the equality (6), we show the complementary problem to be in Σ_3^P, by guessing an SE-interpretation $\langle X, Y\rangle$ and checking the containment in $SE^B_A(P)$ or in $SE^{\overline{A},B}(Q)$ but not both. By Proposition 9, deciding $\langle X, Y\rangle \in SE^B_A(P)$ amounts to checking that $Y \subseteq \mathcal{U} \setminus A$ and $X \in \bigcap \mathcal{R}^Y_{\langle P,A,B\rangle}$. By Proposition 23 the latter is in Π_2^P, and B-SE-model checking is in D^P.

For hardness, we use the case $B = \emptyset$, where the problem reduces to decide $AS(P)|_{\overline{A}} = AS(Q)$ for a program P being $\emptyset$-relativized A-simplifiable. To this end, we extend the hardness-construction from (Eiter and Gottlob 1995) and reduce from $(3, \forall)$-QSAT. Let $\Phi = \forall U \exists V \forall W \phi$ with $\phi = \bigvee_{i=1}^n (l_{i,1} \wedge l_{i,2} \wedge l_{i_3})$. We use copies of atoms, e.g., $\widetilde{U} = \{\widetilde{u} \mid u \in U\}$. We construct P as follows:

$$\begin{aligned} P = &\{x \vee \widetilde{x} \leftarrow . \mid x \in U \cup V \cup W\} \cup \\ &\{w \leftarrow s.\ \widetilde{w} \leftarrow s.\ s \leftarrow w, \widetilde{w}. \mid w \in W\} \cup \\ &\{s \leftarrow \widetilde{l}_{i,1}, \widetilde{l}_{i,2}, \widetilde{l}_{i,3}. \mid 1 \leq i \leq n\} \cup \\ &\{\bot \leftarrow not\ s.\} \end{aligned}$$

where $\widetilde{l}_{i,j}$ is given by $\widetilde{a}$ if $l_{i,j}$ is $\neg a$ and by $\widetilde{l_{i,j}} = l_{i,j}$ if $l_{i,j}$ is a positive literal. Note that P is $\emptyset$-relativized A-simplifiable no matter how Φ looks like. Moreover, we set

$$Q = \{u \vee \widetilde{u} \leftarrow . \mid u \in U\}$$

and $A = V \cup \widetilde{V} \cup W \cup \widetilde{W} \cup \{s\}$. It can be checked that $AS(P)|_{\overline{A}} = AS(Q)$ holds iff Φ is true. ☐

	$\emptyset$	B	$\mathcal{U}$
$\emptyset$	Π_2^P-complete	Π_2^P-complete	coNP-complete
A	**Π_3^P-complete**	**Π_3^P-complete**	in Π_2^P

Table 2: Complexity landscape of testing B-relativized A-simplification. Results in bold-face are given in this paper.

Table 2 presents the full complexity landscape, using results from (Eiter and Gottlob 1995; Eiter, Fink, and Woltran 2007) for (relativized) (strong) equivalence and (Saribatur and Woltran 2023) for strong simplifications. Saribatur and Eiter (2018) provides the Π_2^P-completeness of $B = \emptyset$ for normal programs, which here we lifted to ELPs.

Conclusion

We introduced a novel relativized equivalence notion, which is a relaxation of the recent strong simplification notion (Saribatur and Woltran 2023), that provides a unified view over all related notions of forgetting and strong equivalence in the literature. We provided the necessary and sufficient conditions to ensure such relativized simplifiability. We observed that the challenge is when the context programs do not contain all the atoms to remove, that requires a criterion on the program that focuses on its relativized SE-models, which also captures the case of **(SP)**-forgetting.

We furthermore introduced an operator that can obtain relativized simplifications, when possible. We showed that at least for those atoms to be removed that appear in the context programs, it is possible to simply project them away, while for those that are outside of the context programs a relaxed version of an **(SP)**-forgetting operator will need to be applied. We provided complexity results that fill the gap in the landscape of the introduced notion.

Investigating the relativized simplification notion for the uniform case to bring together uniform persistence forgetting (Gonçalves et al. 2019), (relativized) uniform equivalence and uniform simplifications (Saribatur and Woltran 2023) would be an interesting extension of this work.

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF) projects T-1315, P32830, and by the Vienna Science and Technology Fund (WWTF) under grant ICT19-065. We thank the anonymous reviewers for their valuable feedback.

References

Arora, S.; and Barak, B. 2009. *Computational complexity: a modern approach*. Cambridge University Press.

Baumann, R.; and Berthold, M. 2022. Limits and Possibilities of Forgetting in Abstract Argumentation. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 2539–2545. ijcai.org.

Beierle, C.; and Timm, I. J. 2019. Intentional forgetting: An emerging field in AI and beyond. *KI-Künstliche Intelligenz*, 33: 5–8.

Berthold, M. 2022. On syntactic forgetting with strong persistence. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022*, 43–52. IJCAI Organization.

Berthold, M.; Gonçalves, R.; Knorr, M.; and Leite, J. 2019. A syntactic operator for forgetting that satisfies strong persistence. *Theory and Practice of Logic Programming*, 19(5-6): 1038–1055.

Berthold, M.; Rapberger, A.; and Ulbricht, M. 2023. Forgetting Aspects in Assumption-Based Argumentation. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, 86–96. IJCAI Organization.

Bistarelli, S.; Codognet, P.; and Rossi, F. 2002. Abstracting soft constraints: Framework, properties, examples. *Artificial Intelligence*, 139(2): 175–211.

Clarke, E. M.; Grumberg, O.; and Long, D. E. 1994. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1512–1542.

Eiter, T.; Fink, M.; and Woltran, S. 2007. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic (TOCL)*, 8(3): 17.

Eiter, T.; and Gottlob, G. 1995. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Ann. Math. Artif. Intell.*, 15(3-4): 289–323.

Eiter, T.; and Kern-Isberner, G. 2018. A Brief Survey on Forgetting from a Knowledge Representation and Reasoning Perspective. *KI – Künstliche Intelligenz*. Online http://link.springer.com/article/10.1007/s13218-018-0564-6.

Eiter, T.; Tompits, H.; and Woltran, S. 2005. On Solution Correspondences in Answer Set Programming. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 97–102.

Giunchiglia, F.; and Walsh, T. 1992. A theory of abstraction. *AIJ*, 57(2-3): 323–389.

Gonçalves, R.; Janhunen, T.; Knorr, M.; Leite, J.; and Woltran, S. 2019. Forgetting in modular answer set programming. In *Proc. of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2843–2850. AAAI Press.

Gonçalves, R.; Knorr, M.; and Leite, J. 2016a. The Ultimate Guide to Forgetting in Answer Set Programming. In *Proc. of the 15th International Conference on Principles of Knowledge Representation and Reasoning*, 135–144.

Gonçalves, R.; Knorr, M.; and Leite, J. 2016b. You can't always forget what you want: On the limits of forgetting in answer set programming. In *Proc. of the 22nd European Conference on Artificial Intelligence*, 957–965. IOS Press.

Gonçalves, R.; Knorr, M.; and Leite, J. 2023. Forgetting in Answer Set Programming A Survey. *Theory and Practice of Logic Programming*, 23(1): 111156.

Gonçalves, R.; Knorr, M.; Leite, J.; and Woltran, S. 2020. On the limits of forgetting in Answer Set Programming. *Artif. Intell.*, 286: 103307.

Knoblock, C. A. 1994. Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2): 243–302.

Knorr, M.; and Alferes, J. J. 2014. Preserving strong equivalence while forgetting. In *Logics in Artificial Intelligence: 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings 14*, 412–425. Springer.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4): 526–541.

Luo, K.; Liu, Y.; Lespérance, Y.; and Lin, Z. 2020. Agent Abstraction via Forgetting in the Situation Calculus. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and Lang, J., eds., *Proceedings of the 24th European Conference on Artificial Intelligence, ECAI 2020*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 809–816. IOS Press.

Papadimitriou, C. H. 2003. *Computational complexity*. John Wiley and Sons Ltd.

Richards, B. A.; and Frankland, P. W. 2017. The persistence and transience of memory. *Neuron*, 94(6): 1071–1084.

Saribatur, Z. G.; and Eiter, T. 2018. Omission-based Abstraction for Answer Set Programs. In *Proc. of the 16th International Conference on Principles of Knowledge Representation and Reasoning*, 42–51. AAAI Press.

Saribatur, Z. G.; and Eiter, T. 2020. A Semantic Perspective on Omission Abstraction in ASP. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, 733–737. IJCAI Organization.

Saribatur, Z. G.; and Woltran, S. 2023. Foundations for Projecting Away the Irrelevant in ASP Programs. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, 614–624. IJCAI Organization.

Siebers, M.; and Schmid, U. 2019. Please delete that! Why should I? Explaining learned irrelevance classifications of digital objects. *KI-Künstliche Intelligenz*, 33(1): 35–44.

Turner, H. 2001. Strong equivalence for logic programs and default theories (made easy). In *Logic Programming and Nonmotonic Reasoning: 6th International Conference, LPNMR 2001 Vienna, Austria, September 17–19, 2001 Proceedings 6*, 81–92. Springer.

Vasileiou, S. L.; and Yeoh, W. 2022. On Generating Abstract Explanations via Knowledge Forgetting. In *ICAPS 2022 Workshop on Explainable AI Planning*.

Wang, Y.; Wang, K.; and Zhang, M. 2013. Forgetting for Answer Set Programs Revisited. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 1162–1168. IJCAI/AAAI.

Woltran, S. 2004. Characterizations for Relativized Notions of Equivalence in Answer Set Programming. In Alferes, J. J.; and Leite, J. A., eds., *Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27-30, 2004, Proceedings*, volume 3229 of *Lecture Notes in Computer Science*, 161–173. Springer.