Limits and Possibilities of Forgetting in Abstract Argumentation

Ringo Baumann and Matti Berthold
Universität Leipzig
{baumann, berthold}@informatik.uni-leipzig.de

Abstract
The topic of \textit{forgetting} has been extensively studied in the field of knowledge representation and reasoning for many major formalisms. Quite recently it has been introduced to abstract argumentation. However, many already known as well as essential aspects about forgetting like \textit{strong persistence} or \textit{strong invariance} have been left unconsidered. We show that forgetting in abstract argumentation cannot be reduced to forgetting in logic programming. In addition, we deal with the more general problem of forgetting whole sets of arguments and show that iterative application of existing operators for single arguments does not necessarily yield a desirable result as it may not produce an informationally economic argumentation framework. As a consequence we provide a systematic and exhaustive study of forgetting desiderata and associated operations adapted to the intrinsics of abstract argumentation. We show the limits and shed light on the possibilities.

1 Introduction
The notion of \textit{forgetting} has been extensively studied in the field of knowledge representation and reasoning for many major formalisms like classical logic [Lin and Reiter, 1994], logic programming [Gonçalves et al., 2016a; Eiter and Kern-Isberner, 2018] and more recently for abstract argumentation [Baumann et al., 2020]. Roughly speaking, forgetting is about getting rid of some variables, atoms or arguments while keeping as much as possible of the reasoning not concerned with the forgotten. The ability of forgetting is often exploited to make reasoning more efficient. In this paper we want to further elaborate the limits and possibilities of forgetting in abstract argumentation. The latter is a vibrant research area in AI [Simari and Rahwan, 2009; Baroni et al., 2018a] with Dung-style argumentation frameworks (AFs) and their associated semantics at the heart of this field [Dung, 1995].

In order to obtain reasonable forgetting operators for abstract reasoning we may try to convey ideas from other formalisms. The area of logic programming with its plenty of approaches to forgetting is a good candidate (cf. [Gonçalves et al., 2016b] for an excellent overview). However, the following two examples show that forgetting in abstract argumentation cannot be reduced to forgetting in logic programming in a straightforward manner.

Example 1 (Limits of the Standard Translation). Consider the following AF $F$. We have $\text{stb}(F) = \{\{b, d\}\}$. Assume now that we want to forget the argument $b$. Hence, one reasonable forgetting result is thus an AF $F'$, s.t. $\text{stb}(F') = \{\{d\}\}$. Note that simply deleting $b$ would yield an AF $F_b$, s.t. $\text{stb}(F_b) = \{\{a, d\}\}$. This means, such a syntactical removal would render the previously unaccepted argument $a$ acceptable.

$$F : a \leftrightarrow \neg b \quad b \leftrightarrow \neg c \quad c \leftrightarrow \neg d \quad d$$

Let us consider instead the standard translation from AFs to LPs [Strass, 2013]. This yields the following equivalent logic program $P$.

$$P : a \leftarrow \neg b \quad b \leftarrow \neg c \quad c \leftarrow \neg d \quad d$$

Now we may apply the already defined forgetting operator $f_{SP}$ [Berthold et al., 2019b]. More precisely, forgetting $b$ from $P$ results in $f_{SP}(P, b)$ as given below.

$$f_{SP}(P, b) : \quad a \leftarrow \neg \neg c \quad c \leftarrow \neg d \quad d$$

Unfortunately, $f_{SP}(P, b)$ is a non-AF-like program. Therefore, it is generally not possible to simply reverse the standard translation. However, in case of $f_{SP}(P, b)$ we may find an equivalent LP $P'$ which is indeed AF-like.

$$P' : \quad a \leftarrow \neg d \quad c \leftarrow \neg d \quad d$$

Retranslating $P'$ to the realm of AFs results in $F'$. Note that $\text{stb}(F') = \{\{d\}\}$ as desired.

$$F' : \quad a \quad c \leftrightarrow d$$

Example 2 (Representational Limits). Consider now the slightly more involved AF $F$. We observe $\text{stb}(F) = \{\{a, e, f\}, \{b, f, d\}, \{c, d, e\}\}$.

$$F : \quad c \leftrightarrow f$$

Let us consider instead the standard translation. However, in case of $f_{SP}(P, b)$ we may find an equivalent LP $P'$ which is indeed AF-like.
Let us assume again that we want to forget the argument \( b \). The favored forgetting result is thus the extension set \( D = \{ \{ a, e, f \}, \{ f, d \}, \{ c, d, e \} \} \). Since \( D \) forms a \( \subseteq \)-antichain there is an LP \( P \) realizing it [Eiter et al., 2013]. However, we will never find an equivalent AF-like LP \( P' \) since \( D \) does not satisfy so-called tightness [Dunne et al., 2015]. In particular, \( \{ f, d \} \cup \{ e \} \notin D \) but \( \{ e, f \} \in \{ a, e, f \} \) and \( \{ d, e \} \in \{ c, d, e \} \).

The final example deals with forgetting multiple arguments. It reveals that this task cannot be simply reduced to forgetting single arguments.

**Example 3 (Forgetting Sets vs. Arguments). Consider the following AF \( F \). We have \( \text{stb}(F) = \{ \{ x, b, e \}, \{ a, c, d \}, \{ a, c, e \} \} \). Assume that we want to forget a set of arguments, say \( \{ x, b \} \). One reasonable forgetting result is thus an AF \( F' \), s.t. \( \text{stb}(F') = \{ \{ a, c, d \}, \{ a, c, e \} \} = D \).

\[
F: \quad \begin{array}{ccc}
  & c & d \\
  b & x & e \\
a & & \\
\end{array}
\]

One natural approach for forgetting multiple arguments is to iteratively apply an existing forgetting operator for single arguments. The following frameworks illustrate this procedure for the operator \( f \) firstly presented in [Baumann et al., 2020, Algorithm 1, Example 4].

\[
f(F, x): \quad \begin{array}{ccc}
  & c & d \\
  e & x & e \\
a & & \end{array} \quad \text{and} \quad f(f(F, x), b): \quad \begin{array}{ccc}
  & c & d \\
  & & e \\
b & a & n_1 \quad n_2 \\
\end{array}
\]

If forgetting \( b \) first and subsequently \( x \) reveals that this approach is sensitive to the order of forgetting and might not yield an informationally economic result.

\[
f(F, b): \quad \begin{array}{ccc}
  & c & d \\
  e & x & e \\
a & & \end{array} \quad \text{and} \quad f(f(F, b), x): \quad \begin{array}{ccc}
  & c & d \\
  & & e \\
a & & \end{array}
\]

The three examples above show that we need further investigation on how sets of arguments can be forgotten in case of AFs. As a consequence we provide a systematic and comprehensive analysis of forgetting desiderata and associated operations adapted to the intrinsics of abstract argumentation. We hereby draw a lot of inspiration from logic programming. We show the limits and shed light on the possibilities. In particular, we study the relations between desiderata, their individual as well as combined satisfiability and look for promising combinations. Moreover, we consider forgetting under stable semantics as it shows a quite different behavior regarding the fulfillment of combined desiderata. Finally, we conclude and discuss related work.

## 2 Background

### 2.1 Logic Programming

**Syntax and Semantics** A logic program \( P \) over a propositional signature \( \mathcal{U} \) [Lifschitz et al., 1999] is a finite set of rules of the form \( a_1 \lor \ldots \lor a_k \leftarrow b_1, \ldots, b_l, \neg c_1, \ldots, \neg c_m, \neg d_1, \ldots, \neg d_n \).

For such a rule \( r \) let \( H(r) = \{ a_1, \ldots, a_k \}, B^+(r) = \{ b_1, \ldots, b_l \}, B^-(r) = \{ c_1, \ldots, c_m \} \) and \( B^-(r) = \{ d_1, \ldots, d_n \} \). We define \( \mathcal{U}(P) = \bigcup_{r \in P} H(r) \cup B^+(r) \cup B^-(r) \cup B^-(r) \).

A set of atoms \( I \subseteq \mathcal{U} \) is called an interpretation. The reduct of \( P \) w.r.t. \( I \) is given as \( P^I = \{ H(r) \leftarrow B^-(r) \mid r \in P, B^+(r) \cap I = \emptyset, B^-(r) \subseteq I \} \). An interpretation \( I \) is an answer set of \( P \) if \( I = P^I \), where \( \emptyset \) is the classical satisfaction relation, and for each interpretation \( I' \) we have: If \( I' = P^I \), then \( I' \notin I \). The set of all answer sets of \( P \) is denoted by \( \mathcal{AS}(P) \). Two programs \( P_1, P_2 \) are equivalent if \( \mathcal{AS}(P_1) = \mathcal{AS}(P_2) \) and strongly equivalent, whenever \( \mathcal{AS}(P_1 \cup R) = \mathcal{AS}(P_2 \cup R) \) for any program \( R \) [Lifschitz et al., 2001]. Given a set \( V \subseteq \mathcal{U} \), the \( V \)-exclusion of a set of answer sets \( \mathcal{M} \), denoted \( \mathcal{M}_{\neg V} \), is \( \{ X \setminus V \mid X \in \mathcal{M} \} \).

**Forgetting: Desiderata and Operators** Let \( P \) be the set of all logic programs over \( \mathcal{U} \). A forgetting operator is a (partial) function \( f : \mathcal{P} \times 2^d \rightarrow \mathcal{P} \) with \( (P, V) \mapsto f(P, V) \). The program \( f(P, V) \) is interpreted as the result of forgetting about \( V \) from \( P \). Moreover, \( \mathcal{U}(f(P, V)) \subseteq \mathcal{U}(P) \setminus V \) is usually required. In the following we introduce some well-known properties for forgetting operators [Gonçalves et al., 2016a].

**Strong persistence** is presumably the best known one [Knorr and Alferes, 2014]. It requires that the result of forgetting \( f(P, V) \) is strongly equivalent to the original program \( P \), modulo the forgotten atoms.

**(SP)** \( f \) satisfies strong persistence if, for each program \( P \) and each set of atoms \( V \), we have: \( \mathcal{AS}(f(P, V) \cup R) = \mathcal{AS}(P \cup R) \) for all programs \( R \) with \( \mathcal{U}(R) \not\subseteq \mathcal{U}(V) \).

**Strong invariance** requires that rules not mentioning atoms to be forgotten can be added before or after forgetting.

**(SI)** \( f \) satisfies strong invariance if, for each program \( P \) and each set of atoms \( V \), we have: \( f(P, V) \cup R \equiv_s f(P \cup R, V) \) for all programs \( R \) with \( \mathcal{U}(R) \not\subseteq \mathcal{U}(V) \).

**Consequence persistence** and its two variations are weaker forms of strong persistence dealing with ordinary equivalence only.

**(CP)** \( f \) satisfies consequence persistence if, for each program \( P \) and each set of atoms \( V \), we have: \( \mathcal{AS}(f(P, V)) = \mathcal{AS}(P) \) for all programs \( R \) with \( \mathcal{U}(R) \not\subseteq \mathcal{U}(V) \).

**(SC)** \( f \) satisfies strengthened consequence if, for each program \( P \) and each set of atoms \( V \), we have: \( \mathcal{AS}(f(P, V)) \subseteq \mathcal{AS}(P) \) for all programs \( R \) with \( \mathcal{U}(R) \not\subseteq \mathcal{U}(V) \).

**(wC)** \( f \) satisfies weakened consequence if, for each program \( P \) and each set of atoms \( V \), we have: \( \mathcal{AS}(f(P, V)) \not\supseteq \mathcal{AS}(P) \).

### 2.2 Argumentation Theory

**Syntax and Semantics** Let \( \mathcal{U} \) be an infinite background set. An abstract argumentation framework (AF) [Dung, 1995] is a directed graph \( F = (A, R) \) with \( A \subseteq \mathcal{U} \) representing arguments and \( R \subseteq A \times A \) interpreted as attacks. If \( (a, b) \in R \) we say that \( a \)
attacks $b$ or $a$ is an attacker of $b$. Moreover, a set $E$ defends an argument $a$ if any attacker of $a$ is attacked by some argument of $E$. In this paper we consider finite AFs only and use the symbol $\mathcal{F}$ to denote the set of all finite AFs. Moreover, for a set $E \in \mathcal{A}$ we use $E^+ = \{ b \mid (a, b) \in E, a \in E \}$ and define $E^0 = E \cup E^+$. Given an AF $F = (B, S)$, we use $A(F)$ to refer to the set $B$ and $R(F)$ to refer to the relation $S$. For two AFs $F$ and $G$, we define the expansion of $F$ by $G$, in symbols $F \cup G$, which is an AF $F \cup G = (A(F) \cup A(G), R(F) \cup R(G))$. Finally, the restriction of an AF $F$ to a set of arguments $C \subseteq \mathcal{U}$ is defined as $F|_C = (A(F) \cap C, R(F) \cap (C \times C))$.

An extension-based semantics $\sigma : \mathcal{F} \rightarrow 2^{2^\mathcal{A}}$ is a function which assigns to any AF a set of sets of arguments $\sigma(F) \subseteq 2^{A(F)}$. Each set of arguments $E \in \sigma(F)$ is considered to be acceptable with respect to $F$ and is called a $\sigma$-extension. The most basic requirements of an extension are called conflict-freeness (cf) and admisibility (ad). Other well-studied semantics are grounded (gr), semi-stable (ss), complete (co), preferred (pr), semi-stable (ss), and eager (eq). The requirements for each semantics are summarized below. A recent overview of argumentation semantics can be found in [Baroni et al., 2018b]. Two AFs $F$ and $G$ are equivalent w.r.t. $\sigma (F) = \sigma(G)$.

**Definition 1.** Let $F = (A, R)$ be an AF and $E \subseteq A$.
1. $E \in cf(F)$ iff for no $a, b \in E$, $(a, b) \in R$.
2. $E \in ad(F)$ iff $E \in cf(F)$ and $E$ defends all its elements.
3. $E \in co(F)$ iff $E \in ad(F)$ and for any $a \in A$ defended by $E$, $a \in E$.
4. $E \in stg(F)$ iff $E \in cf(F)$ and for no $I \subseteq cf(F), E^0 \subseteq I^0$.
5. $E \in sbl(F)$ iff $E \in cf(F)$ and $E^0 = A$.
6. $E \in ss(F)$ iff $E \in ad(F)$ and for no $I \subseteq cf(F), E^0 \subseteq I^0$.
7. $E \in pr(F)$ iff $E \in co(F)$ and for no $I \subseteq co(F), E \subseteq I$.
8. $E \in gr(F)$ iff $E \in co(F)$ and for any $I \subseteq co(F), E \subseteq I$.
9. $E \in il(F)$ iff $E \in co(F)$, $E \subseteq \cap pr(F)$ and there is no $I \subseteq co(F)$ satisfying $I \subseteq \cap pr(F)$ s.t. $E \subseteq I$.
10. $E \in eg(F)$ iff $E \in co(F)$, $E \subseteq \cap ss(F)$ and there is no $I \subseteq co(F)$ satisfying $I \subseteq \cap ss(F)$ s.t. $E \subseteq I$.

**Existence, Reasoning and Expressibility** A semantics $\sigma$ is universally defined, if $\sigma(F) \neq \emptyset$ for any $F \in \mathcal{F}$. If even $|\sigma(F)| = 1$ we say that $\sigma$ is uniquely defined. Apart from stable semantics all considered semantics are universally defined. The grounded, ideal and eager semantics are uniquely defined (cf. [Baumann and Spanring, 2015] for an overview).

With respect to the acceptability of arguments, we consider the two main reasoning modes. Given a semantics $\sigma$, an AF $F$, and an argument $a \in A(F)$, we say that $a$ is credulously accepted w.r.t. $\sigma$ if $a \in \bigcap \sigma(F)$ and that $a$ is skeptically accepted w.r.t. $\sigma$ if $\sigma(F) \neq \emptyset$ and $a \in \bigcap \sigma(F)$.

We say that a set of sets $E \subseteq 2^\mathcal{A}$ is realizable w.r.t. $\sigma$ if there is an AF $F$ s.t. $\sigma(F) = E$. Realizability under stable semantics is given if and only if i) $E$ forms a $\subseteq$-antichain\(^1\) and ii) $E$ is tight [Dunne et al., 2015]. Tightness is fulfilled if for all $E \in \mathcal{E}$ and $a \in \bigcup \mathcal{E}$ we have: if $E \cup \{ a \} \notin \mathcal{E}$ then there exists an $e \in E$, s.t. $(a, e) \notin \{ (b, c) \mid \exists E' \in \mathcal{E} : \{ b, c \} \subseteq E' \}$. See Example 2 for an illustration. Moreover, we will frequently use that stage, semi-stable as well as preferred semantics satisfy I-maximality too (cf. [Baumann, 2018] for an overview).

### 3 Desiderata for Forgetting

Given an AF $F$ and a set of arguments $X \subseteq \mathcal{U}$, we use $f_\sigma(F, X)$ to denote the result of forgetting the arguments $X$ in $F$ under semantics $\sigma$. This means, we consider a function $f_\sigma : \mathcal{F} \times 2^\mathcal{U} \rightarrow \mathcal{F}$ mapping a pair $(F, X)$ to an AF $f_\sigma(F, X)$. If clear from context or irrelevant we will omit $\sigma$.

In the following we collect and define a large number of desiderata for forgetting in abstract argumentation. Some of them have been already considered in [Baumann et al., 2020] for the case of single arguments. We generalize them to sets of arguments as done in the LP case. Moreover we introduce further important conditions firstly considered in the realm of LPs [Gonçalves et al., 2016a]. We will see that there are many dependencies that are not clear at first glance. Note that desiderata $e_1$ as well as $e_2$ could be alternatively renamed as $e_{cp}$ and $e_{sp}$ (see Section 2 for more details.) However, we decided to keep in line with the notation chosen in [Baumann et al., 2020].

**Desiderata 1.** Given an AF $F$ and a set of arguments $X \subseteq \mathcal{U}$, For a forgetting operator $f$ we define:

\[ e_1. \quad \sigma(f(F, X)) = \{ E \setminus X \mid E \in \sigma(F) \} \]  
\[ \text{ (X-adjusted extension) } \]

\[ e_{wC}. \quad \sigma(f(F, X)) \supseteq \{ E \setminus X \mid E \in \sigma(F) \} \]  
\[ \text{ (no such extension is lost) } \]

\[ e_{sc}. \quad \sigma(f(F, X)) \subseteq \{ E \setminus X \mid E \in \sigma(F) \} \]  
\[ \text{ (no further extensions are added) } \]

\[ e_2. \quad \sigma(f(F, X) \cup H) = \{ E \setminus X \mid E \in \sigma(F \cup H) \} \]  
\[ \text{ for any } H \text{ with } A(H) \subseteq \mathcal{U} \setminus X \]  
\[ \text{ (delete X even from any future extension) } \]

\[ e_{3a}. \quad \sigma(f(F, X)) = \{ T(E) \mid E \in \sigma(F) \} \]  
\[ \text{ with } T : \sigma(F) \rightarrow 2^\mathcal{U} \]  
\[ \text{ and } E \mapsto T(E) \subseteq E \setminus X \]  
\[ \text{ (subsets of X-adjusted extension) } \]

\[ e_{3b}. \quad \sigma(f(F, X)) = \{ T(E) \mid E \in \sigma(F) \} \]  
\[ \text{ with } T : \sigma(F) \rightarrow 2^\mathcal{U} \]  
\[ \text{ and } E \mapsto T(E) \supseteq E \setminus X \]  
\[ \text{ (supersets of X-adjusted extension) } \]

\[ e_4. \quad \sigma(f(F, X)) = \sigma(F) \setminus \{ E \mid E \in \sigma(F), E \cap X = \emptyset \} \]  
\[ \text{ (remove X-overlapping extensions) } \]

The next four desiderata are concerned with skeptical and credulous reasoning.

**Desiderata 2.** Given an AF $F$ and a set of arguments $X \subseteq \mathcal{U}$, For a forgetting operator $f$ we define:

\[ r_1. \quad \bigcap \sigma(f(F, X)) \cap X = \emptyset \]  
\[ \text{ (X is not skept. accepted) } \]

\[ r_2. \quad \bigcup \sigma(f(F, X)) \cap X = \emptyset \]  
\[ \text{ (X is not cred. accepted) } \]

\[ r_3. \quad \sigma(f(F, X)) = \{ \sigma(F) \} \setminus X \]  
\[ \text{ (rigid skept. accept.) } \]

\[ r_4. \quad \sigma(f(F, X)) = \{ \cup \sigma(F) \} \setminus X \]  
\[ \text{ (rigid cred. accept.) } \]  

\(^1\)Within the argumentation community this property is usually referred to as I-maximality [Baroni and Giacomin, 2007].
Arguably the presented reasoning desiderata either describe too strictly or too loosely what is skeptically or credulously accepted. For $r_1$ and $r_2$ to be satisfied, it suffices to syntactically remove $X$. In contrast, to satisfy $r_3$ or $r_4$, the resulting AF must entail a precise set of arguments. As a compromise between them, we suggest the following desiderata, that bridge semantic and syntactic requirements.

**Desiderata 3.** Given two AFs $F$ and $H$ as well as a set of arguments $X \subseteq \mathcal{U}$. For a forgetting operator $\mathbf{f}$ we define:

$m_1. \quad \cap \sigma(f(F,X)) \subseteq A(F) \setminus X$ (skept. acceptance is among unforgotten old arguments)

$m_2. \quad \cup \sigma(f(F,X)) \subseteq A(F) \setminus X$ (cred. acceptance is among unforgotten old arguments)

$m_3. \quad (\forall f : f(F,X) \cup H \subseteq A(H) \cup A(F)) \setminus X \text{ for all } AFs H$ (forgotten arguments are never skept. accepted)

$m_4. \quad (\forall f : f(F,X) \cup H \subseteq A(H) \cup A(F)) \setminus X$ (forgotten arguments are never cred. accepted)

Condition $m_1$ (resp. $m_2$) requires that, if there are new arguments added while forgetting, they be irrelevant to skeptical (resp. credulous) reasoning. In other words, that these arguments are purely administrative. Then $m_3$ (resp. $m_4$) require new arguments to be irrelevant, even under the addition of new information.

The following three conditions are purely syntactical ones. Desideratum $s_1$ makes explicit what is often implicitly assumed for forgetting operators in other formalisms. Condition $s_3$ presents the most straightforward way of forgetting a set of arguments. Such an syntactical approach was firstly considered in [Bisquert et al., 2011].

**Desiderata 4.** Given an AF $F$ and a set of arguments $X \subseteq \mathcal{U}$. For a forgetting operator $\mathbf{f}$ we define:

$s_1. \quad A(f(F,X)) \cap X = \emptyset$ (no arguments from $X$)

$s_2. \quad A(f(F,X)) = A(F) \setminus X$ (precise set of arguments)

$s_3. \quad f(F,X) = f|_{A(F)\setminus X}$ (rigid AF)

The following vacuity desiderata provide conditions under which a given framework does not require any changes.

**Desiderata 5.** Given an AF $F$ and a set of arguments $X \subseteq \mathcal{U}$. For a forgetting operator $\mathbf{f}$ we define:

$v_1. \quad \cap \sigma(F \cap X) = \emptyset$, then $F = f(F,X)$. (skept. vacuity)

$v_2. \quad \cup \sigma(F \cap X) = \emptyset$, then $F = f(F,X)$. (cred. vacuity)

$v_3. \quad A(F \cap X) = \emptyset$, then $F = f(F,X)$. (argument vacuity)

When deriving a forgetting result it would be advantageous to be able to confine the construction in some way. For comparison, some forgetting operators in LP have been shown to be able to disregard rules that do not mention the atoms to be forgotten, i.e. they satisfy the discussed property (SI). Similarly, when forgetting arguments from an AF we could require that arguments that do not stand in (close) contact to the arguments to be forgotten can be left unchanged.

**Desiderata 6.** Given an AF $F$ and a set of arguments $X \subseteq \mathcal{U}$. For a forgetting operator $\mathbf{f}$ we define:

$l_0. \quad f(F,X) \cup H = f(F \cup H,X)$ for all AFs $H$ with $A(H) \subseteq \mathcal{U} \setminus X$ (f and $\cup$ are compatible)

$l_1. \quad f(F,X) \cup H = f(F \cup H,X)$ for all AFs $H$ with $A(H) \subseteq \mathcal{U} \setminus (X \cup \{a \mid \exists x \in X, s.t. (a,x) \in R \text{ or } (x,a) \in R\})$ (less tolerant refinement of compatibility)

We proceed with an analysis of their dependencies.

**Proposition 1.** For $\sigma \in \{\text{stg, stb, ss, pr, gr, il, eg}\}$ and conditions $c \text{ and } c'$ in the diagram below, a path from $c$ to $c'$ indicates that any function $f_\sigma$ satisfying $c$ under $\sigma$ also satisfies $c'$ under $\sigma$. Moreover, only these relations hold.

![Figure 1: Dependencies](image)

Apart from the relationships concerning single conditions there are more complex implications. In the realm logic programming it was already shown that (SP) is necessary and sufficient for (SI) and (CP) [Goncalves et al., 2016a]. Besides other interesting relations we state the analogous result for abstract argumentation in Item 5 of the following proposition.

**Proposition 2.** For any semantics $\sigma \in \{\text{stg, stb}\}$:

1. $s_2, l_0$ and $c_{3d}$ imply $e_2$.
2. $s_2, l_0$ and $c_{3d}$ imply $e_2$.

Moreover, for any $\tau \in \{\text{stg, stb, ss, pr, gr, il, eg}\}$ we have:

3. $c_{3d}$ and $c_{3d'}$ if and only if $e_1$.
4. $e_{aC}$ and $e_{wC}$ if and only if $e_1$.
5. $e_1$ and $l_0$ if and only if $e_2$. 
4 Satisfiability and Unsatisfiability

In this section we consider the satisfiability of single conditions as well as whole sets of desiderata. Most of the results underline the intrinsic limits of forgetting in abstract argumentation as they prove unsatisfiability.

4.1 Individual Desiderata

We start with a positive result regarding individual satisfiability. In fact, 19 conditions are satisfiable under any considered semantics if considered in isolation.

Proposition 3. Desideratum \( d \in \{e_3, e_5, e_{wC}, r_1, r_2, r_3, \)
\( r_4, s_1, s_2, s_3, m_1, m_2, m_3, m_4, v_1, v_2, v_3, l_1, \) \( \overline{l_1} \) \} is satisfiable under any semantics \( \sigma \in \{stg, stb, ss, pr, gr, il, eg\} \).

The following proposition shows a dividing line between uniquely and universally defined semantics. The I-maximality of the latter family prevent the satisfiability of \( e_1 \) and \( e_{wC} \).

Proposition 4. Desiderata \( e_1 \) and \( e_{wC} \) are satisfiable under any \( \tau \in \{gr, il, eg\} \), but not under \( \sigma \in \{stb, stg, ss, pr\} \).

The following two propositions are mainly due to already shown results in [Baumann et al., 2020].

Proposition 5. Desiderata \( e_3 \) is satisfiable under stable semantics, but not under any \( \tau \in \{stg, ss, pr, gr, il, eg\} \).

Proposition 6. Desiderata \( e_2 \) is unsatisfiable under any semantics \( \sigma \in \{stg, stb, ss, pr, gr, il, eg\} \).

4.2 Combined Desiderata

In the following we consider whole sets of conditions.

Proposition 7. We have the following satisfiability results:
1. \( \{s_2, l_0, e_3, e_5\} \) as well as \( \{s_2, l_0, e_3\} \) are unsatisfiable for any semantics \( \mu \in \{stg, stb\} \).
2. Moreover, \( \{e_3, e_5\} \) and \( \{e_{wC}, e_{wC}\} \) are unsatisfiable for any semantics \( \sigma \in \{stg, stb, ss, pr\} \), but satisfiable for each \( \tau \in \{gr, il, eg\} \).

The next result underline the exceptional potential of stable semantics regarding forgetting.

Proposition 8. \( \{l_0, e_{wC}\} \) as well as \( \{l_1, e_{wC}\} \) are satisfiable under stable semantics but not under \( \sigma \in \{gr, stg, ss, pr\} \).

4.3 Testing the Limits: Promising Combinations

The strongest syntactical desideratum \( s_3 \) is incompatible with all semantical ones and can only be trivially combined with \( r_1 \) and \( r_2 \). In this context, trivial means, that one condition already implies the other (cf. Proposition 1). Stable semantics is able to collapse for certain AFs. This unique property is nicely reflected in Table 1 via exceptional behaviour regarding fulfilling combined desiderata.

Proposition 9. Table 1 summarizes the compatibility under semantics \( \sigma \in \{stg, stb, ss, pr, gr, il, eg\} \). A “✓”/“✓” in cell \( (l,c) \) indicates whether or not the conditions in line \( l \) and column \( c \) are simultaneously satisfiable under \( \sigma \). The symbol “✓” restricts the satisfiability to the semantics \( gr, il \) and \( eg \), the symbol “\( \neg \)stb” to stable semantics and the symbol “\( \neg \)stb” to all semantics but stable respectively. The combinations in a dark background are trivial.

<table>
<thead>
<tr>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
<th>( m_1 )</th>
<th>( m_2 )</th>
<th>( m_3 )</th>
<th>( m_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: Compatibility of syntactical/semantical conditions

5 Forgetting under Stable Semantics

Let us reflect on the proposed extension-based conditions as listed in Desiderata 1. At first we observe that \( e_1 \) and \( e_4 \) represent two opposing philosophies about the concept of forgetting. Desideratum \( e_1 \) requires that any former extension has to survive in an adjusted fashion, namely new extensions are obtained from initial ones via deleting the arguments which has to be forgotten \( (\sigma(f(F,X)) = \{E \setminus X \mid E \in \sigma(F)\}) \). In contrast, Condition \( e_4 \) requires to delete any extension carrying arguments which has to be forgotten \( (\sigma(f(F,X)) = \sigma(F) \cap \{E \mid E \notin \sigma(F)\}) \). Both interpretations of forgetting are independent as shown in Proposition 1. Any other considered extension-based condition is either a relaxation of \( e_1 \), or a lifting of this interpretation to the level of strong equivalence. In the following we will consider these two main desiderata in more detail. We restrict ourselves to stable semantics and leave the consideration of other semantics for future work.

Forgetting via \( e_4 \) Quite recently, an \( e_4 \)-operator \( f \) for forgetting single arguments was presented [Baumann et al., 2020, Algorithm 1]. In Example 3 of the introductory part we have seen that applying this operator \( f \) iteratively does not necessarily produce a desirable outcome. Moreover, this procedure is sensitive to the order of forgetting. How to adapt the existing procedures for multiple arguments?

It turned out that the main idea can be directly conveyed to the task of forgetting a whole set of arguments \( X \). More precisely, in a first step we simply restrict the initial framework \( F \) to \( A(F) \setminus X \), i.e. we consider \( F|_{A(F) \setminus X} \). In doing so we get that any former extension containing arguments from \( X \) is not stable anymore but any other survives. Now, in a second step, we eliminate any unwanted extensions via the addition of self-attacking arguments. This yields the following Algorithm 1.

Example 4 (Example 3 cont.). Consider again AF \( F \). Let \( X = \{x, b\} \). Applying Algorithm 1 immediately yields \( f^*(F, X) = F|_{A(F) \setminus X} \) as \( stb(F|_{A(F) \setminus X}) = \{\{a, c, d\}, \{a, e, e\}\} \).
Forgetting via $e_1$ The main reason for the impossibility to find an operator satisfying $e_1$ under stable semantics is an intrinsic one, namely realizability. More precisely, there are certain instances $(F, X)$ which would enforce a forgetting result $F'$ violating the $\leq$-antichain property or tightness (see Example 2). Consequently, one reasonable strategy is to look for forgetting operators satisfying $e_1$ whenever possible, and if not, try to satisfy a certain relaxation of it. Natural candidates would be $e_{2b}, e_{3b},$ or $e_{4b}$. A similar procedure was suggested and also implemented for strong persistence in the realm of logic programming [Gonçalves et al., 2017].

The choice of how to relax $e_1$ depends on the application in mind. Each relaxation has its particular advantages and drawbacks. Moreover, as we have seen in Figure 1 there are no dependencies between them. Further research on this subject is left for future work.

6 Discussion and Conclusion

The paper sheds more light on forgetting in abstract argumentation. One main motivation was to convey desiderata from recent studies of forgetting in LP [Knorr and Alferes, 2014; Gonçalves et al., 2016a; Berthold et al., 2019a; Berthold, 2022]. We redefined several principles and provided a comprehensive study regarding satisfiability and relations. We further demonstrated that already existing operators from LP cannot be unconditionally applied to abstract argumentation. The two main reasons are non-AF-like forgetting results and the essential differences regarding expressibility. Finally, we presented two specific forgetting operators, one excluding the arguments to be forgotten, and the other one using extensions only.

A relevant work in this context is [Rienstra et al., 2020] dealing with so-called robustness principles. The paper studies the question to which extent old labelings persist/new labelings arise if a certain structural change of a given AF is performed. Such results are highly relevant for the theory of forgetting as they can be used to show the satisfiability/unsatisfiability of desired properties. The consideration of such results will be fruitful for the development of concrete forgetting operators.
Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF, 1IS18026A) by funding the competence center for Big Data and AI “ScaDS.AI” Dresden/Leipzig.

References


