Limits and Possibilities of Forgetting in Abstract Argumentation

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

Abstract

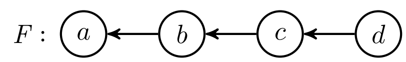
The topic of *forgetting* has been extensively studied in the field of knowledge representation and reasoning for many major formalisms. Quite recently it has been introduced to abstract argumentation. However, many already known as well as essential aspects about forgetting like *strong persistence* or *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 *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 $stb(F) = \{\{b, d\}\}$. Assume now that we want to forget the argument b. Hence, one reasonable forgetting result is thus an AF F', s.t. $stb(F') = \{\{d\}\}$. Note that simply deleting b would yield an AF F_b, s.t. $stb(F_b) = \{\{a, d\}\}$. This means, such a syntactical removal would render the previously unaccepted argument a acceptable.*



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

$$P: \quad a \leftarrow not\, b \qquad b \leftarrow not\, c$$
$$c \leftarrow not\, d \qquad d$$

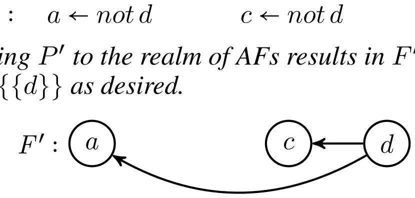
Now we may apply the already defined forgetting operator f_{SP} [Berthold et al.*, 2019b]. More precisely, forgetting b from P results in $\mathsf{f}_{SP}(P, b)$ as given below.*

$$\mathsf{f}_{SP}(P, b): \quad a \leftarrow not\, not\, c \qquad c \leftarrow not\, d \qquad d$$

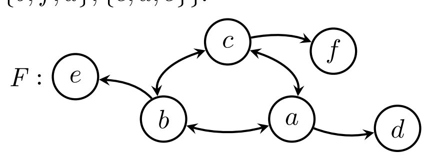
Unfortunately, $\mathsf{f}_{SP}(P, b)$ is a non-AF-like program. Therefore, it is generally not possible to simply reverse the standard translation. However, in case of $\mathsf{f}_{SP}(P, b)$ we may find an equivalent LP P' which is indeed AF-like.

$$P': \quad a \leftarrow not\, d \qquad c \leftarrow not\, d \qquad d$$

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



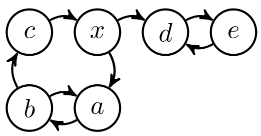
Example 2 (Representational Limits)**.** *Consider now the slightly more involved AF F. We observe $stb(F) = \{\{a, e, f\}, \{b, f, d\}, \{c, d, e\}\}$.*



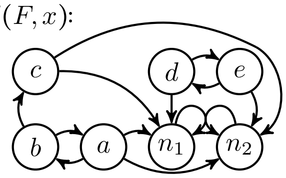
Let us assume again that we want to forget the argument b. The favored forgetting result is thus the extension set $D = \{\{a, e, f\}, \{f, d\}, \{c, d, e\}\}$. Since D forms a $\subseteq$-antichain there is an LP P realizing it [Eiter et al.*, 2013]. However, we will never find an equivalent AF-like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

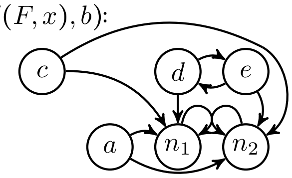
The final example deals with forgetting multiple arguments. It reveals that this task cannot be simply reduced to forgetting single arguments.

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

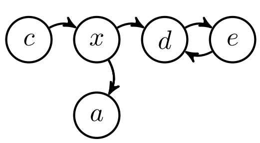
F:


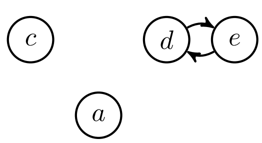
One natural approach for forgetting multiple arguments is to iteratively apply an existing forgetting operator for single arguments. The following frameworks illustrate this procedure for the operator f firstly presented in [Baumann et al.*, 2020, Algorithm 1, Example 4].*

$\mathsf{f}(F, x)$:


$\mathsf{f}(\mathsf{f}(F, x), b)$:


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.

$\mathsf{f}(F, b)$:


$\mathsf{f}(\mathsf{f}(F, b), x)$:


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 $a_1 \vee \ldots \vee a_k \leftarrow b_1, ..., b_l,\ \mathit{not}\, c_1, ..., \mathit{not}\, c_m,\ \mathit{not}\,\mathit{not}\, d_1, ..., \mathit{not}\,\mathit{not}\, d_n$. 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 $P^I = \{H(r) \leftarrow B^{+}(r) \mid r \in P, B^{-}(r) \cap I = \varnothing, B^{--}(r) \subseteq I\}$. An interpretation I is an *answer set* of P if $I \vDash P$, where $\vDash$ is the classical satisfaction relation, and for each interpretation I' we have: If $I' \vDash P^I$, then $I' \not\subset I$. The set of all answer sets of P is denoted by $\mathcal{AS}(P)$. Two programs P_1, P_2 are *equivalent* if $\mathcal{AS}(P_1) = \mathcal{AS}(P_2)$ ($P_1 \equiv P_2$) and *strongly equivalent*, whenever $\mathcal{AS}(P_1 \cup R) = \mathcal{AS}(P_2 \cup R)$ for any program R ($P_1 \equiv_s P_2$) [Lifschitz *et al.*, 2001]. Given a set $V \subseteq \mathcal{U}$, the *V-exclusion* of a set of answer sets $\mathcal{M}$, denoted $\mathcal{M}_{\|V}$, is $\{X \backslash V \mid X \in \mathcal{M}\}$.

Forgetting: Desiderata and Operators Let $\mathcal{P}$ be the set of all logic programs over $\mathcal{U}$. A *forgetting operator* is a (partial) function $\mathsf{f} : \mathcal{P} \times 2^{\mathcal{U}} \to \mathcal{P}$ with $(P, V) \mapsto \mathsf{f}(P, V)$. The program $\mathsf{f}(P, V)$ is interpreted as the *result of forgetting about V from P*. Moreover, $\mathcal{U}(\mathsf{f}(P, V)) \subseteq \mathcal{U}(P) \setminus V$ 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 $\mathsf{f}(P, V)$ 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: $\mathcal{AS}(\mathsf{f}(P, V) \cup R) = \mathcal{AS}(P \cup R)_{\|V}$ for all programs R with $\mathcal{U}(R) \subseteq \mathcal{U} \backslash V$.

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: $\mathsf{f}(P, V) \cup R \equiv_s \mathsf{f}(P \cup R, V)$ for all programs R with $\mathcal{U}(R) \subseteq \mathcal{U} \backslash V$.

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 P and each set of atoms V: $\mathcal{AS}(\mathsf{f}(P, V)) = \mathcal{AS}(P)_{\|V}$.

(sC) f satisfies *strengthened consequence* if, for each P and each set of atoms V: $\mathcal{AS}(\mathsf{f}(P, V)) \subseteq \mathcal{AS}(P)_{\|V}$.

(wC) f satisfies *weakened consequence* if, for each P and each set of atoms V: $\mathcal{AS}(\mathsf{f}(P, V)) \supseteq \mathcal{AS}(P)_{\|V}$.

2.2 Argumentation Theory

Syntax and Semantics Let $\mathcal{U}$ be an infinite background set. An *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 $E \subseteq A$ we use E^+ for $\{b \mid (a,b) \in R, a \in E\}$ and define $E^\oplus = E \cup E^+$. 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 $F \sqcup G$, as expected: $F \sqcup G = (A(F) \cup A(G), R(F) \cup R(G))$. 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* $\sigma : \mathcal{F} \to 2^{2^{\mathcal{U}}}$ is a function which assigns to any AF F 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 σ*-extension*. The most basic requirements of an extension are called *conflict-freeness* (cf) and *admissibility* (ad). Other well-studied semantics include stage (stg), stable (stb), semi-stable (ss), complete (co), preferred (pr), grounded (gr), ideal (il) and eager (eg). The requirements of 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.* σ ($F \equiv^\sigma G$) if $\sigma(F) = \sigma(G)$.

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* $\mathcal{I} \in cf(F)$, $E^\oplus \subset \mathcal{I}^\oplus$,
5. $E \in stb(F)$ *iff* $E \in cf(F)$ *and* $E^\oplus = A$,
6. $E \in ss(F)$ *iff* $E \in ad(F)$ *and for no* $\mathcal{I} \in ad(F)$, $E^\oplus \subset \mathcal{I}^\oplus$,
7. $E \in pr(F)$ *iff* $E \in co(F)$ *and for no* $\mathcal{I} \in co(F)$, $E \subset \mathcal{I}$,
8. $E \in gr(F)$ *iff* $E \in co(F)$ *and for any* $\mathcal{I} \in co(F)$, $E \subseteq \mathcal{I}$,
9. $E \in il(F)$ *iff* $E \in co(F)$, $E \subseteq \bigcap pr(F)$ *and there is no* $\mathcal{I} \in co(F)$ *satisfying* $\mathcal{I} \subseteq \bigcap pr(F)$ *s.t.* $E \subset \mathcal{I}$,
10. $E \in eg(F)$ *iff* $E \in co(F)$, $E \subseteq \bigcap ss(F)$ *and there is no* $\mathcal{I} \in co(F)$ *satisfying* $\mathcal{I} \subseteq \bigcap ss(F)$ *s.t.* $E \subset \mathcal{I}$.

Existence, Reasoning and Expressibility A semantics σ is *universally defined*, if $\sigma(F) \neq \emptyset$ for any $F \in \mathcal{F}$. If even $|\sigma(F)| = 1$ we say that σ 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 σ, an AF F, and an argument $a \in A(F)$, we say that a is *credulously accepted w.r.t.* σ if $a \in \bigcup \sigma(F)$ and that a is *skeptically accepted w.r.t.* σ if $\sigma(F) \neq \emptyset$ and $a \in \bigcap \sigma(F)$.

We say that a set of sets $\mathcal{E} \subseteq 2^{\mathcal{U}}$ is *realizable w.r.t.* σ if there is an AF F s.t. $\sigma(F) = \mathcal{E}$. Realizability under stable semantics is given if and only if i) $\mathcal{E}$ forms a $\subseteq$-antichain[1] and ii) $\mathcal{E}$ 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).

[1]Within the argumentation community this property is usually referred to as *I-maximality* [Baroni and Giacomin, 2007].

3 Desiderata for Forgetting

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

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 $e_{\mathbf{CP}}$ and $e_{\mathbf{SP}}$ (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. $\sigma(\mathsf{f}(F,X)) = \{E \setminus X \mid E \in \sigma(F)\}$ *(X-adjusted extension)*

$e_{\mathbf{wC}}$. $\sigma(\mathsf{f}(F,X)) \supseteq \{E \setminus X \mid E \in \sigma(F)\}$ *(no such extension is lost)*

$e_{\mathbf{sC}}$. $\sigma(\mathsf{f}(F,X)) \subseteq \{E \setminus X \mid E \in \sigma(F)\}$ *(no further extensions are added)*

e_2. $\sigma(\mathsf{f}(F,X) \sqcup H) = \{E \setminus X \mid E \in \sigma(F \sqcup H)\}$ *for any* H *with* $A(H) \subseteq \mathcal{U} \setminus X$ *(delete X even from any future extension)*

$e_{3_\subseteq}$. $\sigma(\mathsf{f}(F,X)) = \{T(E) \mid E \in \sigma(F)\}$ *with* $T : \sigma(F) \to 2^{\mathcal{U}}$ *and* $E \mapsto T(E) \subseteq E \setminus X$ *(subsets of X-adjusted extension)*

$e_{3_\supseteq}$. $\sigma(\mathsf{f}(F,X)) = \{T(E) \mid E \in \sigma(F)\}$ *with* $T : \sigma(F) \to 2^{\mathcal{U}}$ *and* $E \mapsto T(E) \supseteq E \setminus X$ *(supersets of X-adjusted extension)*

e_4. $\sigma(\mathsf{f}(F,X)) = \sigma(F) \setminus \{E \mid E \in \sigma(F), E \cap X \neq \emptyset\}$ *(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. $\bigcap \sigma(\mathsf{f}(F,X)) \cap X = \emptyset$ *(X is not skept. accepted)*

r_2. $\bigcup \sigma(\mathsf{f}(F,X)) \cap X = \emptyset$ *(X is not cred. accepted)*

r_3. $\bigcap \sigma(\mathsf{f}(F,X)) = (\bigcap \sigma(F)) \setminus X$ *(rigid skept. accept.)*

r_4. $\bigcup \sigma(\mathsf{f}(F,X)) = (\bigcup \sigma(F)) \setminus X$ *(rigid cred. accept.)*

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 syntatic requirements.

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

m_1. $\bigcap \sigma(\mathsf{f}(F, X)) \subseteq A(F) \setminus X$
(skept. acceptance is among unforgotten old arguments)

m_2. $\bigcup \sigma(\mathsf{f}(F, X)) \subseteq A(F) \setminus X$
(cred. acceptance is among unforgotten old arguments)

m_3. $\bigcap \sigma(\mathsf{f}(F, X) \sqcup H) \subseteq (A(H) \cup A(F)) \setminus X$ *for all AFs H with $A(H) \subseteq \mathcal{U} \setminus X$*
(forgotten arguments are never skept. accepted)

m_4. $\bigcup \sigma(\mathsf{f}(F, X) \sqcup H) \subseteq (A(H) \cup A(F)) \setminus X$ *for all AFs H with $A(H) \subseteq \mathcal{U} \setminus X$*
(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 f we define:*

s_1. $A(\mathsf{f}(F, X)) \cap X = \varnothing$ *(no arguments from X)*

s_2. $A(\mathsf{f}(F, X)) = A(F) \setminus X$ *(precise set of arguments)*

s_3. $\mathsf{f}(F, X) = F|_{A(F) \setminus X}$ *(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 f we define:*

v_1. *If $\bigcap \sigma(F) \cap X = \varnothing$, then $F = \mathsf{f}(F, X)$. (skept. vacuity)*

v_2. *If $\bigcup \sigma(F) \cap X = \varnothing$, then $F = \mathsf{f}(F, X)$. (cred. vacuity)*

v_3. *If $A(F) \cap X = \varnothing$, then $F = \mathsf{f}(F, X)$. (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 $(\mathbf{SI})$. 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 f we define:*

l_0. $\mathsf{f}(F, X) \sqcup H \equiv \mathsf{f}(F \sqcup H, X)$ *for all AFs H with $A(H) \subseteq \mathcal{U} \setminus X$* *($\mathsf{f}$ and $\sqcup$ are compatible)*

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

We proceed with an analysis of their dependencies.

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

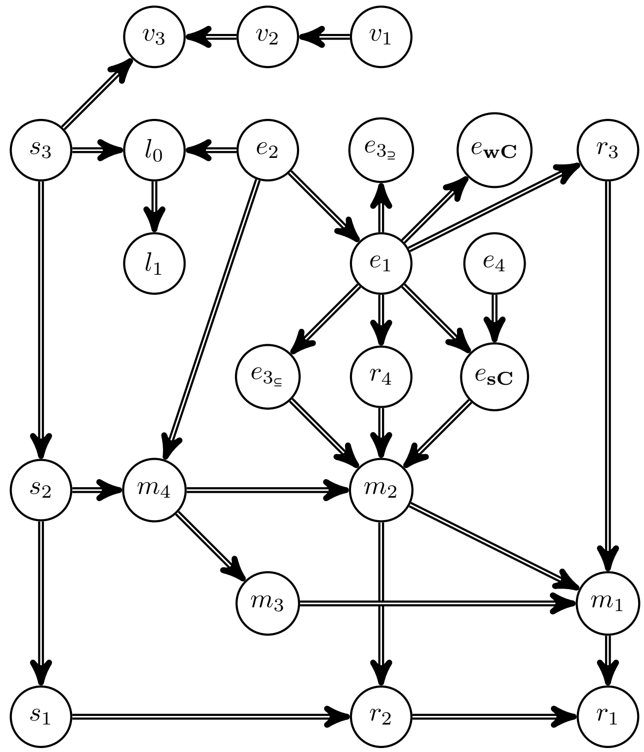


Figure 1: Dependencies

Apart from the relationships concerning single conditions there are more complex implications. In the realm logic programming it was already shown that $(\mathbf{SP})$ is necessary and sufficient for $(\mathbf{SI})$ and $(\mathbf{CP})$ [Gonçalves *et al.*, 2016a]. Beside other interesting relations we state the analogous result for abstract argumentation in Item 5 of the following proposition. Please note that the proof is astonishingly simple.

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

1. *s_2, l_0 and $e_{3_\subseteq}$ imply e_2,*
2. *s_2, l_0 and $e_{3_\supseteq}$ imply e_2.*

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

3. *$e_{3_\subseteq}$ and $e_{3_\supseteq}$ if and only if e_1,*
4. *$e_{\mathbf{sC}}$ and $e_{\mathbf{wC}}$ 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 satisfiablity. In fact, 19 conditions are satisfiable under any considered semantics if considered in isolation.

Proposition 3. *Desideratum $d \in \{e_{3_\supseteq}, e_{3_\subseteq}, e_{\mathbf{sC}}, r_1, r_2, r_3, r_4, s_1, s_2, s_3, m_1, m_2, m_3, m_4, v_1, v_2, v_3, l_0, l_1\}$ 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 $e_{\mathbf{wC}}$. We mention that the corresponding forgetting property $(\mathbf{CP})$ in the realm of LPs is satisfiable.

Proposition 4. *Desiderata e_1 and $e_{\mathbf{wC}}$ 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_4 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. *$\{s_2, l_0, e_{3_\subseteq}\}$ as well as $\{s_2, l_0, e_{3_\supseteq}\}$ are unsatisfiable for any semantics $\mu \in \{stg, stb\}$.*
2. *Moreover, $\{e_{3_\subseteq}, e_{3_\supseteq}\}$ and $\{e_{\mathbf{sC}}, e_{\mathbf{wC}}\}$ 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. *$\{l_0, e_{\mathbf{sC}}\}$ as well as $\{l_1, e_{\mathbf{sC}}\}$ 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 "$\checkmark$"/"$\times$" in cell (l,c) indicates whether or not the conditions in line l and column c are simultaneously satisfiable under σ. The symbol "τ" restricts the satisfiability to the semantics gr, il and eg, the symbol "stb" to stable semantics and the symbol "$\neg stb$" to all semantics but stable respectively. The combinations in a dark background are trivial.*

	s_1	s_2	s_3	m_1	m_2	m_3	m_4
r_1	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
r_2	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
r_3	$\checkmark$	$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
r_4	$\checkmark$	$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
e_1	τ	τ	$\times$	τ	τ	τ	τ
e_2	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
$e_{3_\subseteq}$	$\checkmark$	$\neg stb$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
$e_{3_\supseteq}$	$\checkmark$	$\neg stb$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
e_4	stb	$\times$	$\times$	stb	stb	stb	stb
$e_{\mathbf{sC}}$	$\checkmark$	$\neg stb$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
$e_{\mathbf{wC}}$	τ	τ	$\times$	τ	τ	τ	τ

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 $(\sigma(\mathsf{f}(F, X)) = \{E \setminus X \mid E \in \sigma(F)\})$. In contrast, Condition e_4 requires to delete any extension carrying arguments which has to be forgotten $(\sigma(\mathsf{f}(F, X)) = \sigma(F) \setminus \{E \mid E \in \sigma(F), E \cap X \neq \varnothing\})$. 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 $\mathsf{f}^*(F, X) = F|_{A(F) \setminus X}$ as $stb(F|_{A(F) \setminus X}) = \{\{a, c, d\}, \{a, c, e\}\}$.*

Algorithm 1: Construct $G = \mathsf{f}_1^*(F, X)$

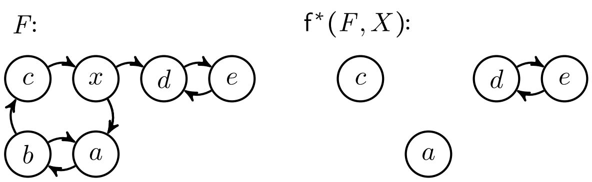
Input : AF F; arguments $X \subseteq \mathcal{U}$
Output : AF G satisfying $\{s_1, e_4, v_3, m_4\}$

Function `compute_G`(F, X)
  if $X \cap A(F) = \emptyset$ **then** $G \leftarrow F$;
  else
    $G_0 \leftarrow F|_{A(F) \setminus X}$;
    $A = A(G_0)$; $R \leftarrow R(G_0)$;
    foreach $E_i \in stb(G_0) \setminus stb(F)$ **do**
      Let a_i be a fresh argument s.t. $a_i \notin A(F) \cup A$;
      $A \leftarrow A \cup \{a_i\}$; $R \leftarrow R \cup \{(a_i, a_i)\}$
      foreach $y \in \bigcup A(G_0) \setminus E_i$ **do**
        $R \leftarrow R \cup \{(y, a_i)\}$;
    $G \leftarrow (A, R)$;
  return G;

Algorithm 2: Construct $G = \mathsf{f}_2^*(F, X)$

Input : AF F; arguments $X \subseteq \mathcal{U}$
Output : AF G satisfying $\{e_4, v_3, m_4\}$

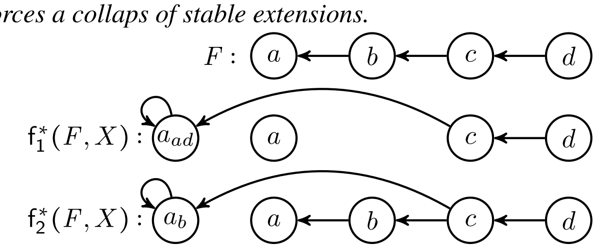
Function `compute_G`(F, X)
  if $X \cap A(F) = \emptyset$ **then** $G \leftarrow F$;
  else
    $G_0 \leftarrow F$;
    $A = A(G_0)$; $R \leftarrow R(G_0)$;
    foreach $x \in X$ **do**
      Let a_x be a fresh argument s.t. $a_x \notin A$;
      $A \leftarrow A \cup \{a_x\}$; $R \leftarrow R \cup \{(a_x, a_x)\}$
      foreach $b \in A : (b, x) \in R$ **do**
        $R \leftarrow R \cup \{(b, a_x)\}$;
      $G \leftarrow (A, R)$;
  return G;



The attentive reader may have already noticed that $\mathsf{f}^*(F, X) = \mathsf{f}(\mathsf{f}(F, b), x)$. This means, forgetting x and b simultaneously yields the more compact outcome if applying the former operator f iteratively ($\mathsf{f}(\mathsf{f}(F, b), x)$ vs. $\mathsf{f}(\mathsf{f}(F, x), b)$). A detailed comparison between iteratively applying f and the one-shot f^* will be part of future work.

Now lets turn to the question of whether e_4 can be satisfied when removing arguments is impossible, unintended or just undesired for some reason. This means, only expanding the initial framework F is allowed. The following Algorithm 2 gives an affirmative answer. The simple idea is to manipulate as follows: For any argument $x \in X$ we add a self-attacking argument a_x which is attacked by any attacker of x. In doing so, any former extension E not containing x remains stable, but former ones E' with $x \in E'$ does not survive as they do not attack a_x (confer Appendix). It is important to note that the presented algorithm relies on syntactic information only as it does note require the computation of any extension.

Example 5 (Example 1 cont.)**.** *Consider again AF F. We have $stb(F) = \{\{b, d\}\}$. Let $X = \{b\}$. Note that Desideratum e_4 forces a collaps of stable extensions.*



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 $\subseteq$-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 $e_{3_\subseteq}$, $e_{3_\supseteq}$ or $e_{\mathbf{sC}}$. 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 expansions 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 labellings persist/new labellings 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 desidered 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

[Baroni and Giacomin, 2007] Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, 171:675–700, 2007.

[Baroni *et al.*, 2018a] P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. *Handbook of Formal Argumentation*. College Publications, 2018.

[Baroni *et al.*, 2018b] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. Abstract argumentation frameworks and their semantics. In *Handbook of Formal Argumentation*, chapter 4. College Publications, 2018.

[Baumann and Spanring, 2015] Ringo Baumann and Christof Spanring. Infinite argumentation frameworks - On the existence and uniqueness of extensions. In *Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday*, volume 9060, pages 281–295. Springer, 2015.

[Baumann *et al.*, 2020] Ringo Baumann, Dov M. Gabbay, and Odinaldo Rodrigues. Forgetting an argument. In *Proceedings of (IJCAI-20)*, pages 2750–2757. AAAI Press, 2020.

[Baumann, 2018] Ringo Baumann. On the nature of argumentation semantics: Existence and uniqueness, expressibility, and replaceability. In *Handbook of Formal Argumentation*, chapter 18. College Publications, 2018. also appears in IfCoLog Journal of Logics and their Applications 4(8):2779-2886.

[Berthold *et al.*, 2019a] Matti Berthold, Ricardo Gonçalves, Matthias Knorr, and João Leite. Forgetting in answer set programming with anonymous cycles. In *Proceedings of (EPIA-19)*, pages 552–565, Cham, 2019. Springer.

[Berthold *et al.*, 2019b] Matti Berthold, Ricardo Gonçalves, Matthias Knorr, and João Leite. A syntactic operator for forgetting that satisfies strong persistence. *Theory and Practice in Logic Programming*, 19(5-6):1038–1055, 2019.

[Berthold, 2022] Matti Berthold. On syntactic forgetting with strong persistence. In *Proceedings of (KR-22)*, 2022. To appear.

[Bisquert *et al.*, 2011] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Change in argumentation systems: Exploring the interest of removing an argument. In *Proceedings of (SUM-11)*, pages 275–288, 2011.

[Dung, 1995] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[Dunne *et al.*, 2015] Paul E. Dunne, Wolfgang Dvořák, Thomas Linsbichler, and Stefan Woltran. Characteristics of multiple viewpoints in abstract argumentation. *Artificial Intelligence*, 228:153–178, 2015.

[Eiter and Kern-Isberner, 2018] Thomas Eiter and Gabriele Kern-Isberner. A brief survey on forgetting from a knowledge representation and reasoning perspective. *KI - Künstliche Intelligenz*, 2018.

[Eiter *et al.*, 2013] Thomas Eiter, Michael Fink, Jörg Pührer, Hans Tompits, and Stefan Woltran. Model-based recasting in answer-set programming. *Journal of Applied Non-Classical Logics*, 23(1-2):75–104, 2013.

[Gonçalves *et al.*, 2016a] Ricardo Gonçalves, Matthias Knorr, and João Leite. The ultimate guide to forgetting in answer set programming. In *Proceedings of (KR-16)*, pages 135–144, 2016.

[Gonçalves *et al.*, 2016b] Ricardo Gonçalves, Matthias Knorr, and João Leite. You can't always forget what you want: On the limits of forgetting in answer set programming. In *Proceedings of (ECAI-16)*, pages 957–965, 2016.

[Gonçalves *et al.*, 2017] Ricardo Gonçalves, Matthias Knorr, João Leite, and Stefan Woltran. When you must forget: Beyond strong persistence when forgetting in answer set programming. *Theory and Practice in Logic Programming*, 17(5-6):837–854, 2017.

[Knorr and Alferes, 2014] Matthias Knorr and José Júlio Alferes. Preserving strong equivalence while forgetting. In *Proceedings of (JELIA-14)*, pages 412–425, 2014.

[Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):369–389, 1999.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[Lin and Reiter, 1994] Fangzhen Lin and Ray Reiter. Forget it. In *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159, 1994.

[Rienstra *et al.*, 2020] Tjitze Rienstra, Chiaki Sakama, Leendert van der Torre, and Beishui Liao. A principle-based robustness analysis of admissibility-based argumentation semantics. *Argument & Computation*, 11(3):305–339, 2020.

[Simari and Rahwan, 2009] Guillermo Ricardo Simari and Iyad Rahwan, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.

[Strass, 2013] Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence*, 205:39–70, 2013.