Dynamics in Abstract Argumentation Frameworks with Recursive Attack and Support Relations

Gianvincento Alfano 1 and Andrea Cohen 2 and Sebastian Gottifredi 2 and Sergio Greco 1 and Francesco Parisi 1 and Guillermo Simari 2

Abstract. Argumentation is an important topic in the field of AI. There is a substantial amount of work about different aspects of Dung’s abstract Argumentation Framework (AF). Two relevant aspects considered separately so far are extending the framework to account for recursive attacks and supports, and considering dynamics, i.e., AFs evolving over time. In this paper, we jointly deal with these two aspects. We focus on Attack-Support Argumentation Frameworks (ASAFs) which allow for attack and support relations not only between arguments but also targeting attacks and supports at any level, and propose an approach for the incremental computation of extensions (sets of accepted arguments, attacks and supports) of updated ASAFs. Our approach assumes that an initial ASAF extension is given and uses it for first checking whether updates are irrelevant; for relevant updates, an extension of an updated ASAF is computed by translating the problem to the AF domain and leveraging on AF solvers. We experimentally show our incremental approach outperforms the direct computation of extensions for updated ASAFs.

1 Introduction

Argumentation has become an attractive and efficient paradigm for knowledge representation and reasoning within the field of Artificial Intelligence [17, 39, 43, 47]. In particular, Dung’s abstract Argumentation Framework (AF) [32] provides a simple yet powerful formalism for modelling and reasoning with information expressed in terms of arguments and their conflicts. Following the work by Dung, there have been many extensions of AFs allowing for bipolar interactions [26], second-order attacks [20] or, more generally, recursive attacks [10]. In addition to these approaches extending Dung’s AFs, the Attack-Support Argumentation Framework (ASAF) [37] allows for attacks and supports not only towards arguments but also targeting the attack and support relations at any level. This framework has the advantage of enabling a straightforward representation of reasoning situations (e.g. in the area of modelling decision processes), which are not easily accommodated within other frameworks such as Dung’s AF or flat bipolar AFs. In particular, the support in ASAF is interpreted as necessity [45]: if argument A supports argument B, then the acceptance of A is necessary to get the acceptance of B; equivalently, the acceptance of B implies the acceptance of A. However, a support can also target an attack or a support.

An ASAF can be represented by a graph as that in Figure 1(a), where arguments are denoted with calligraphic letters, attacks are denoted with single arrows (→), and supports are denoted with double arrows (⇒). In particular, the ASAF in Figure 1(a) models the following scenario. Suppose John is planning to spend his winter holidays in Bariloche and has to decide whether to rent a car to drive during his stay (D) or make use of public transportation (PT). Arguments D and PT are two alternatives John has, and the conflict between them is represented by the attacks α2 and α3. In general, John has a preference towards driving over using public transport (P). This is encoded by the attack α1 from P to α3. However, he has been told that in order to drive safely in Bariloche he needs to put a snow traction device on his car during winter, the current season in Bariloche (W). Hence, ST supports D (support β1) as the acceptance of ST is necessary for the acceptance of D, and W supports β1. In addition, rental car services ran out of such devices (RA). In this context, John will end up deciding to use public transportation in Bariloche.

However, in practice, argumentation frameworks can be dynamic systems [4, 7, 12, 13, 18, 29, 35, 42]. In fact, typically an ASAF represents a temporary situation, and new arguments, attacks and supports can be added/removed to take into account new available knowledge. For instance, in our running example, suppose now that John comes across with new information stating that it has not snowed in Bariloche for the last two months and will not snow during his stay, represented by an argument NS. Then, when considering the previous arguments and attacks together with the new argument, a new conflict arises, needing to update John’s knowledge: argument NS attacks β1, the support from ST to D, as it provides a context in which John will not need to put a snow traction device on his car in order to drive safely in Bariloche. The new scenario, with the addition of argument NS and then, of the attack α2, corresponds to the updated ASAF shown in Figure 1(b), according to which John will choose to rent and drive a car during his stay in Bariloche.

Recently, there has been a growing interest in studying dynamics of different argumentation systems, including Dung AFs [1, 3, 12, 19, 31, 38], Bipolar AFs and AFs with second order attacks [2]. Notwithstanding this, none of the developments regarding dynamics

1 DIMES, University of Calabria, Italy, email: {g.alfano, greco, fparisi}@dimes.unical.it
2 DCIC, Universidad Nacional del Sur, ICIC (CONICET-UNS), Argentina, email: {ac, sg, gvs}@cs.uns.edu.ar
in argumentation has so far considered bipolar recursive frameworks like the ASAF. However, incremental computation techniques could improve performance, as they only require to reconsider the acceptance status of those arguments and interactions that are affected by the new information. For instance, the acceptance status of $RN$ and $ST$, as well as that of $\alpha_1$, does not change after adding the attack $\alpha_2$ from $NS$ to the support from $ST'$ to $D$.

With the aim of exploiting features like those above, in this paper we propose an incremental approach for efficiently computing extensions of an ASAF after performing an update. Specifically, we propose a technique addressing the following problem: given an ASAF $\Delta$, an initial extension $E_0$ of $\Delta$, and an update $u$ consisting of the addition/removal of an attack/support, determine an extension $E$ of the updated ASAF $u(\Delta)$.

**Contributions.** Our main contributions are as follows:

1. We identify and formally characterize irrelevant updates for an ASAF, for which an extension $E$ of an updated ASAF $u(\Delta)$ can be directly obtained without requiring its overall computation. These results can be used to avoid wasted effort for any incremental algorithm, not just that proposed in this paper.
2. We characterize an ASAF in terms of an AF, and formally show that this AF yields equivalent extensions to those of the ASAF. Note that this AF can be used to compute ASAFs’s extensions even in the static case, where updates are not considered.
3. We define an incremental algorithm for computing an extension $E$ of an updated ASAF $u(\Delta)$, accounting for early termination conditions implied by irrelevant updates and leveraging on the incremental technique proposed in [1] for the computation on the (extensions-equivalent) AF for an updated ASAF $u(\Delta)$. The technique is able to incorporate any existing AF-solver to perform the incremental computation of ASAFs’s extensions.
4. We perform an experimental analysis showing that our incremental approach outperforms the computation from scratch, where the fastest solvers from the International Competition on Computational Models of Argumentation (ICCMAs) are used as baselines.

As a side contribution, we develop a benchmark for testing ASAF solvers, and we briefly introduce labellings for the ASAF.

**2 Essential Background**

We briefly review the essential background for abstract Argumentation Frameworks (AFs) [32] and Attack-Support Argumentation Frameworks (ASAFs) (for a full presentation see [37]).

An AF consists of a set of arguments whose origin is left unspecified, and a set of conflicts between them [32].

**Definition 1 (AF).** An abstract Argumentation Framework (AF) is a pair $\langle A, R \rangle$, where $A$ is a set of arguments and $R \subseteq A \times A$ is an attack relation.

In [32] some semantic notions are defined leading to the characterization of collectively acceptable sets of arguments. Given an AF $\langle A, R \rangle$ and a set $S \subseteq A$ of arguments, we say that:

- $S$ is **conflict-free** iff $\exists A, B \in S$ s.t. $(A, B) \in R$.
- $A \in A$ is **acceptable** w.r.t. $S$ iff $\forall B \in A$ s.t. $(B, A) \in R, \exists C \in S$ s.t. $(C, B) \in R$.
- $S$ is **admissible** iff it is conflict-free and $\forall A \in S, A$ is acceptable w.r.t. $S$.

Then, by adding restrictions to the notion of admissibility, the complete (co), preferred (pr), stable (st), and grounded (gr) extensions of an AF are defined as follows. Given an AF $\langle A, R \rangle$ and a set of arguments $S \subseteq A$, we say that:

- $S$ is a **complete extension** of $A$ if it is admissible and $\forall A \in A$, if $A$ is acceptable w.r.t. $S$, then $A \in S$.
- $S$ is a **preferred extension** of $A$ if it is a maximal (w.r.t. $\subseteq$) complete extension of $A$.
- $S$ is a **stable extension** of $A$ if it is a complete extension of $A$ and $\forall A \in A \setminus S, \exists B \in S$ s.t. $(B, A) \in R$.
- $S$ is the **grounded extension** of $A$ if it is the smallest (w.r.t. $\subseteq$) complete extension of $A$.

**Attack-Support Argumentation Framework.** The ASAF extends Dung’s AF by incorporating bipolar higher-order interactions. In that way, the ASAF allows for the representation and reasoning with attack and support relations not only between arguments, but also targeting the attack and support relations themselves. The support relation of the ASAF is interpreted as necessity [45]. That is, the necessary support relation in the ASAF imposes the following acceptability constraints on the elements it relates: the acceptance of $B$ implies the acceptance of $A$, or, equivalently, the non-acceptance of $A$ implies the non-acceptance of $B$.

**Definition 2 (ASAF).** An Attack-Support Argumentation Framework (ASAF) is a tuple $\langle A, R, S \rangle$ where $A$ is a set of arguments, $R \subseteq W$ is the attack relation, and $S \subseteq W$ is the support relation, where $W$ is the set iteratively defined as follows: $W = A \times A$ (basic step) and $W = A \times W$ (iterative step). It is assumed that $S$ is cyclic and $\forall R \cap S = \emptyset$.

As stated before, attacks and supports in an ASAF can also be attacked and supported. To simplify the notation, an attack $(A, B) \in R$ will be denoted as $\alpha_1 = (A, B)$; similarly, a support $(B, C) \in S$ will be denoted as $\beta_1 = (B, C)$. Then, for instance, an attack from $D$ to $\alpha_1$ will be denoted as $\alpha_2 = (D, \alpha_1)$. Moreover, given an attack $\alpha = (A, X) \in R$, $A$ is called the source of $\alpha$, denoted $\text{src}(\alpha) = A$, and $X$ is called the target of $\alpha$, denoted $\text{trg}(\alpha) = X$. Analogously, given a support $\beta = (B, Y) \in S$, $B$ is called the source of $\beta$, denoted $\text{src}(\beta) = B$, and $Y$ is called the target of $\beta$, denoted $\text{trg}(\beta) = Y$.

An ASAF $\Delta$ can be represented following a graph-like notation by $G_\Delta$: an argument $A \in \Delta$ will be a node in $G_\Delta$, an attack $\alpha = (A, X) \in R$ will be an edge $\alpha \rightarrow X \in G_\Delta$, and a support $\beta = (B, Y) \in S$ will be an edge $B \rightarrow Y \in G_\Delta$. Attacks and supports whose target is an argument are said to be first-level interactions, while attacks and supports whose target is an interaction of level $i$ are said to be interactions of level $i + 1$.

**Example 1.** The initial example from the introduction can be represented by the ASAF $\Delta_1 = \langle A_1, R_1, S_1 \rangle$ in Figure 1(a), with $A_1 = \{D, PT, P, ST, W, RN\}$, $R_1 = \{\alpha_1,\alpha_2,\alpha_3,\alpha_4\}$, and $S_1 = \{\beta_1,\beta_2\}$; where $\alpha_1 = (RN, ST)$, $\alpha_2 = (D, PT)$ and $\alpha_3 = (PT, P)$ are first-level attacks, $\beta_1 = (ST, D)$ is a first-level support, and the second-level interactions are the attack $\alpha_4 = (P, \alpha_3)$ and the support $\beta_2 = (W, \beta_1)$.

In standard graphs, paths are defined among nodes. Here, for graphs denoting ASAFs, we consider paths whose target can also be an edge. In particular, we define a support path from $A$ to $X$ as a path $A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n = X$, where each $A_i (1 \leq i < n)$ is an argument and $A_n = X$ is an argument, attack or support, whose set of edges $S$ contains the support links in the path;
a support path is empty iff $A = X$. Then, the conflicts between the elements of an ASAF, referred to as defeats, are defined as follows.

**Definition 3** (Defeats). Let $\Delta = (\mathcal{A}, \mathcal{R}, S)$ be an ASAF, $\alpha \in \mathcal{A}$, $X \in (\mathcal{A} \cup \mathcal{R} \cup S)$ and $S \subseteq S$. We say that $\alpha$ defeats $X$ (given $S$), denoted $\alpha \text{ def } X$ given $S$ (or simply $\alpha \text{ def } X$, whenever $S = \emptyset$) iff:

- there exists a (possibly empty) support path from $\text{trg}(\alpha)$ to $X$, whose set of edges is $S$; or
- $X \in \mathcal{R}$ and there exists a (possibly empty) support path from $\text{trg}(\alpha)$ to $\text{src}(X)$, whose set of edges is $S$.

The preceding definition not only accounts for defeats originated by the attack relation, but also accounts for defeats arising from the coexistence of the attack and support relations in the ASAFs.

**Example 2.** Given the ASAF $\Delta_1$ from Example 1, these defeats occur: $\alpha_1 \text{ def } ST$, $\alpha_2 \text{ def } PT$, $\alpha_2 \text{ def } \alpha_3$, $\alpha_3 \text{ def } D$, $\alpha_3 \text{ def } \alpha_2$, $\alpha_4 \text{ def } \alpha_3$, $\alpha_1 \text{ def } D$ given $\{\beta_1\}$ and $\alpha_1 \text{ def } \alpha_2$ given $\{\beta_1\}$.

We now define the sets of elements (arguments, attacks and supports) of an ASAF that can be collectively accepted under some criteria, called extensions. When not explicitly stated otherwise, attacks in an ASAF will be denoted with $\alpha$, supports will be denoted with $\beta$ (both, possibly with subscripts), and elements that can either be arguments, attacks or supports will be denoted with capital letters such as $X$ or $Y$. Similarly, sets of elements from an ASAF will be denoted with capital boldface letters such as $S$ or $S'$.

The notion of conflict-freeness establishes the minimum requirements a set of arguments, attacks, and supports should satisfy in order to be collectively accepted.

**Definition 4** (Conflict-freeness). Let $\Delta = (\mathcal{A}, \mathcal{R}, S)$ be an ASAF and $S \subseteq (\mathcal{A} \cup \mathcal{R} \cup S)$. We say that $S$ is conflict-free iff $\not\exists \alpha, X \in S, \not\exists S' \subseteq S$ s.t. $\alpha \text{ def } X$ given $S'$.

The notion of acceptability characterizes the defense by a set of arguments, attacks and supports against the defeat on its elements.

**Definition 5** (ASAF Acceptability). Let $\Delta = (\mathcal{A}, \mathcal{R}, S)$ be an ASAF, $X \in (\mathcal{A} \cup \mathcal{R} \cup S)$ and $S \subseteq (\mathcal{A} \cup \mathcal{R} \cup S)$. We say that $X$ is acceptable w.r.t. $S$ iff $\forall \alpha \in \mathcal{A}, \forall \beta \subseteq S$ s.t. $\alpha \text{ def } X$ given $T$: $\exists Y \in \{\alpha \cup \beta\}, \exists \beta' \subseteq S$ s.t. $\alpha' \text{ def } Y$ given $S'$.

Then, admissible sets are defined by combining the notions of conflict-freeness and acceptability.

**Definition 6** (ASAF Admissibility). Let $\Delta = (\mathcal{A}, \mathcal{R}, S)$ be an ASAF and $S \subseteq (\mathcal{A} \cup \mathcal{R} \cup S)$. $S$ is admissible iff it is conflict-free and $\forall X \in S$: $X$ is acceptable w.r.t. $S$.

**Example 3.** Given the ASAF $\Delta_1$ from Example 1, for instance, $D$ is not acceptable w.r.t. $\{\alpha_4\}$ because, even though $\alpha_4$ defends it against the defeat by $\alpha_3$ given $\{\beta_1\}$, it does not defend it against the defeat by $\alpha_2$ given $\{\beta_1\}$. In contrast, $PT$ is acceptable w.r.t. $\{\alpha_1, \beta_1\}$ because this set defends it against the defeat by $\alpha_2$. Consequently, for instance, the set $\{\alpha_1, \beta_1, PT\}$ is admissible whereas the set $\{\alpha_1, D\}$ is not.

Finally, the complete (co), preferred (pr), stable (st), and grounded (gr) extensions are defined as follows.

**Definition 7** (ASAF Extensions). Let $\Delta = (\mathcal{A}, \mathcal{R}, S)$ be an ASAF and $S \subseteq (\mathcal{A} \cup \mathcal{R} \cup S)$:

- $S$ is a complete extension of $\Delta$ iff it is admissible and $\forall X \in (\mathcal{A} \cup \mathcal{R} \cup S)$, if $X$ is acceptable w.r.t. $S$, then $X \in S$.
- $S$ is a preferred extension of $\Delta$ iff it is a maximal (w.r.t. $\subseteq$) complete extension of $\Delta$.
- $S$ is a stable extension of $\Delta$ iff it is a complete extension of $\Delta$ and $\forall X \in (\mathcal{A} \cup \mathcal{R} \cup S) \setminus S$, $\exists \alpha \in S, \exists S' \subseteq S$ s.t. $\alpha \text{ def } X$ given $S'$.
- $S$ is the grounded extension of $\Delta$ iff it is the smallest (w.r.t. $\subseteq$) complete extension of $\Delta$.

**Example 4.** The grounded extension of the ASAF $\Delta_1$ from Example 1 is $\{R \cup N, \alpha_1, \beta_1, W, \beta_2, PT, P, \alpha_3\}$ (it is also the only preferred and stable extension, thus it is the unique complete extension).

Similarly, to the case of AFs (see [8] for an overview), extensions of an ASAF can also be expressed through labellings. A labelling for an ASAF $\Delta = (\mathcal{A}, \mathcal{R}, S)$ is a total function $\text{Lab} : (\mathcal{A} \cup \mathcal{R} \cup S) \mapsto \{\text{IN}, \text{OUT}, \text{UNDEC}\}$. Given a labelling $\text{Lab}$, we define $\text{IN}(\text{Lab}) = \{X \mid \text{Lab}(X) = \text{IN}\}, \text{OUT}(\text{Lab}) = \{X \mid \text{Lab}(X) = \text{OUT}\}$, and $\text{UNDEC}(\text{Lab}) = \{X \mid \text{Lab}(X) = \text{UNDEC}\}$.

Then, the complete labellings can be defined as follows. $\text{Lab}$ is a complete labelling of an ASAF $\Delta = (\mathcal{A}, \mathcal{R}, S)$ iff for every $X \in (\mathcal{A} \cup \mathcal{R} \cup S)$ it holds that: (1) $\text{Lab}(X) = \text{IN}$ iff $\forall \alpha \in \mathcal{A}, \forall S' \subseteq S$ s.t. $\alpha \text{ def } X$ given $S'$, $\exists Y \in \{\alpha \cup S\}$ s.t. $\text{Lab}(Y) = \text{OUT}$; and (2) $\text{Lab}(X) = \text{OUT}$ iff $\exists \alpha \in \mathcal{A}, \exists S' \subseteq S$ s.t. $\alpha \text{ def } X$ given $S'$ and $\forall Y \in \{\alpha \cup S\}, \text{Lab}(Y) = \text{IN}$.

Hence, for $X$ to be labelled as IN by a complete labelling of the ASAF we require that, for every set of elements originating a defeat on $X$, one of the elements in the set is labelled as OUT (i.e., either the attack or one of the supports, if they exist). Analogously, for $X$ to be labelled as OUT, we require that there exists a set of elements originating a defeat on $X$ where every element in the set (i.e., the attack and every support) is IN. Finally, if $X$ is neither labelled as IN nor OUT, it is labelled as UNDEC.

As it holds for AFs (cf. [23]), a one-to-one correspondence between complete extensions and complete labellings of an ASAF can be established. Specifically, each complete extension $E$ is in one-to-one correspondence with a complete labelling $L = (E, E^+, (\mathcal{A} \cup \mathcal{R} \cup S)(E \cup E^+))$, where $E^+ = \{X \in (\mathcal{A} \cup \mathcal{R} \cup S) \mid \exists \alpha \in E, \exists S \subseteq E$ s.t. $\alpha \text{ def } X$ given $S\}$. In other words, the complete labelling $L$ corresponding to a complete extension $E$ of an ASAF is given by the triple $(\text{IN}(L), \text{OUT}(L), \text{UNDEC}(L))$, where $\text{IN}(L) = E$, $\text{OUT}(L) = E^+$, and $\text{UNDEC}(L) = (\mathcal{A} \cup \mathcal{R} \cup S)(E \cup E^+)$. Then, the preferred, stable and grounded labellings of an ASAF can be defined in terms of the complete labellings of the framework: $\text{Lab}$ is a preferred (resp. stable, grounded) labelling of $\Delta$ iff it is a complete labelling s.t. $\text{IN}(\text{Lab})$ is a preferred (resp. stable, grounded) extension of $\Delta$.

**Example 5.** Given the ASAF $\Delta_1$ and its extensions listed in Example 4, the grounded labelling is $((R \cup N, \alpha_1, \beta_1, W, \beta_2, PT, P, \alpha_3), (ST, D, \alpha_2, \alpha_3), \emptyset)$ (it is also the only complete, preferred and stable labelling of $\Delta_1$).

3 Dynamics: Updates and Translation into AF

We start by defining the notion of update for an ASAF and identify updates that can be considered as irrelevant, since they do not require to compute the corresponding extension of the updated ASAF. Then, we formally characterize an ASAF in terms of an AF, whose extensions are shown to be equivalent to the extensions of the ASAF.

We start by defining the universal sets of attacks and supports, which account for every conceivable relationship that may appear.
in an ASAF. For a set of arguments \( \mathcal{A} \), the set of "universal attacks \( \mathcal{A} \), the set of universal supports \( \mathcal{S} \) and the set of universal updates \( \mathcal{U} \) are such that \( \mathcal{A} \subseteq \mathcal{W} \), \( \mathcal{S} \subseteq \mathcal{W} \), and \( \mathcal{U} \cap \mathcal{W} = \emptyset \) (where \( \mathcal{W} \) is as defined in Definition 2).

An update consists of the addition (resp., removal) of an attack or a support not present (resp., present) in a given ASAF, as formalized in what follows.

**Definition 8** (Update for ASAF). Let \( \mathcal{A} \) be a set of arguments, \( \mathcal{U} \) and \( \mathcal{S} \) the universal sets of attacks and supports, and \( \Delta = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle \) an ASAF, where \( \mathcal{A} \subseteq \mathcal{U} \) and \( \mathcal{S} \subseteq \mathcal{S} \). An update \( u \) over \( \Delta \) belongs to any of the sets characterized below, yielding the updated ASAF \( \Delta' = \langle \mathcal{A}, \mathcal{R}', \mathcal{S}' \rangle \):

- \( u \in \{ +X | X \in \{ \mathcal{R} \cup \mathcal{S} \} \} \). If \( X \in \mathcal{R} \), then \( u = +X \); if \( X \in \mathcal{S} \), then \( u = -X \).

The ASAF \( \Delta' \) obtained by applying an update \( u \) over an ASAF \( \Delta \) will also be denoted as \( u(\Delta) \). Furthermore, for simplicity, we write \( \pm X \) for the addition or removal of an attack or a support \( X \) for the effect of adding/removing arguments to/from an ASAF.

### Tables 1–4: Illustration of All Cases of Irrelevant Updates for an ASAF

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Cases in which the update ( u = +\alpha ), with ( \alpha \in (\mathcal{R} \cup \mathcal{S}) ) is irrelevant. If ( L_0(\text{src}(\alpha)) = \in ), then ( E_0 \cup { \alpha } \in E_S(u(\Delta)) ); otherwise, ( E_0 \notin E_S(u(\Delta)) ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update ( u = +\alpha ). with ( \alpha \in (\mathcal{R} \cup \mathcal{S}) )</td>
<td>( L_0(\text{src}(\alpha)) )</td>
</tr>
<tr>
<td>IN</td>
<td>UNDEC</td>
</tr>
<tr>
<td>co.pr.st.gr</td>
<td>co.pr.st.gr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Cases in which the update ( u = -\alpha ), with ( \alpha \in \mathcal{R} ), is irrelevant. In all cases, ( E_0 \cup { \alpha } \in E_S(u(\Delta)) ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update ( u = -\alpha ), with ( \alpha \in \mathcal{R} )</td>
<td>( L_0(\text{src}(\alpha)) )</td>
</tr>
<tr>
<td>IN</td>
<td>UNDEC</td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Theorem 1.** Let \( \Delta = \langle \mathcal{A}, \mathcal{R}, \mathcal{S} \rangle \) be an ASAF, \( u = \pm X \) an update for \( \Delta \), and \( u(\Delta) \) the updated ASAF. Also, let \( E_0 \) be a \( S \)-extension of \( \Delta \) and \( \mathcal{L} \) the labelling corresponding to \( E_0 \), where \( \mathcal{S} \in \{ \text{co, pr, st, gr} \} \). Then, for each type of update considered in Tables 1–4, a semantics \( \mathcal{S} \) occurs in the cell \( \langle L_0(\text{src}(\alpha)), L_0(\text{trg}(\beta)) \rangle \) of Table i (i \( \in \{1, 4\} \)) if \( u \) is irrelevant for \( \Delta \) w.r.t. \( \mathcal{S} \) and an \( S \)-extension \( E \) of \( u(\Delta) \) can be obtained directly from \( E_0 \) as described in the caption of Table i.

**Proof.** (Sketch) The proof considers each kind of update separately and then moves to considering, for each of the four semantics appearing in a given cell, the initial labelling \( L_0(\Delta) \) of the source and target of the updated interaction. For each case of update that is not irrelevant, a counter-example showing that the initial labelling changes after performing an update is given. As an example, the proof corresponding to the top-middle cell in Table 3 follows. Consider the update \( u = +\beta \), with \( \beta \in (\mathcal{S}_0 \setminus \mathcal{S}) \). Also, let \( L_0 \) be an \( S \)-labelling of \( \Delta \), with \( \mathcal{S} \in \{ \text{co, pr, gr} \} \), s.t. \( L_0(\text{src}(\beta)) = \in \) and \( L_0(\text{trg}(\beta)) = \text{UNDEC} \). We have to show that there exists an \( S \)-labelling \( L \) of \( u(\Delta) \) s.t. \( \mathcal{S}_0(\Delta) = \mathcal{IN}(\mathcal{L}) \cup \{ \beta \} \). First, since no interaction in \( u(\Delta) \) can target \( \beta \), no attacks on \( \beta \) will exist in \( u(\Delta) \) and \( \beta \) will be in every co, pr and gr-labelling of \( u(\Delta) \). Then, we have to show that \( \mathcal{IN}(\Delta) \subseteq \mathcal{IN}(\mathcal{L}) \). The only way in which the addition of \( \beta \) can change the labelling of other elements in \( u(\Delta) \) is by generating new changes towards them, in which \( \beta \) will be involved. Since \( L_0(\text{src}(\beta)) = \in \), it holds that \( \forall \alpha_1 \in \mathcal{R}, \mathcal{S}_0 \subseteq \mathcal{S} \) s.t. \( \alpha_1 \text{def src}(\beta) \) given \( \mathcal{S}_0 \), \( \exists Y \in \{ \{ \alpha_1 \} \cup \mathcal{S}_0 \} \) s.t. \( L_0(Y) = \mathcal{OUT} \). Then, since every new defeat existing in \( u(\Delta) \) but not in \( \Delta \) also has \( \alpha_1 \) and the supports in \( \mathcal{S}_0 \) among its originating elements, the new defeats do not affect the labelling of the arguments, attacks or supports they defeat. Therefore, \( \forall Z \in \{ \mathcal{A} \cup \mathcal{R} \cup \mathcal{S} \} \), \( Z \) maintains in \( u(\Delta) \) the labelling assigned by \( L_0 \) in \( \Delta \). Consequently, there exists an \( S \)-labelling \( L \) of \( u(\Delta) \) corresponding to an \( S \)-extension \( E \) of \( u(\Delta) \) s.t. \( E = E_0 \cup \{ \beta \} \).
Table 3: Cases in which the update \( u = +\beta \), with \( \beta \in (S_1 \setminus S) \) is irrelevant. In all cases, \( E_0 \cup \{\beta\} \in \mathcal{E}(u(\Delta)) \).

<table>
<thead>
<tr>
<th>Update ( u = +\beta ), with ( \beta \in (S_1 \setminus S) )</th>
<th>( L_0(\text{src}(\beta)) )</th>
<th>( \text{IN} )</th>
<th>( \text{UNDEC} )</th>
<th>( \text{OUT} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_0(\text{src}(\beta)) )</td>
<td>( \text{co.pr.st.gr} )</td>
<td>( \text{co.pr.gr} )</td>
<td>( \text{co.pr.st.gr} )</td>
<td>( \text{co.gr} )</td>
</tr>
</tbody>
</table>

Function \( L_0(\text{src}(\beta)) \) returns the chain of attacks from \( \text{src} \) to \( \text{dest} \). The support \( \Delta \) corresponds to the chain of 3 attacks from \( \text{src} \) to \( \text{dest} \) in the AF for the ASAF shown in Figure 2. For instance, the attack \( \beta \) corresponds to the chain of attacks \( \{\alpha, \beta\} \) and \( \alpha \) is a support argument for the attack \( \beta \).

Table 4: Cases in which the update \( u = -\beta \), with \( \beta \in S \) is irrelevant. In all cases, \( E_0 \setminus \{\beta\} \in \mathcal{E}(u(\Delta)) \).

<table>
<thead>
<tr>
<th>Update ( u = -\beta ), with ( \beta \in S )</th>
<th>( L_0(\text{src}(\beta)) )</th>
<th>( \text{IN} )</th>
<th>( \text{UNDEC} )</th>
<th>( \text{OUT} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_0(\text{src}(\beta)) )</td>
<td>( \text{co.pr.st.gr} )</td>
<td>( \text{co.pr.gr} )</td>
<td>( \text{co.pr.st.gr} )</td>
<td>( \text{co.gr} )</td>
</tr>
</tbody>
</table>

Before showing the equivalence between \( S \)-extensions of an ASAF and \( S \)-extensions of its AF, we introduce functions mapping extensions of \( \Delta \) into extensions of \( \Delta_{AF} \) and vice-versa.

Definition 10 (ASAFtoAF and AFtoASAF Functions). Let \( \Delta = \langle A, R, S \rangle \) be an ASAF and \( \Delta_{AF} = \langle A, R \rangle \) its AF. We define the functions \( \text{ASAFtoAF} \) and \( \text{AFtoASAF} \), where \( E \) is a complete extension of \( \Delta \) and \( E' \) is a complete extension of \( \Delta_{AF} \):

- \( \text{ASAFtoAF}(E) = \{ \{ \alpha \} | \alpha \in \mathcal{A} \setminus \mathcal{E} \} \cup \{ \{ S \} | S \in \mathcal{E} \} \cup \{ \{ \beta \} | \beta \in \mathcal{B} \} \)
- \( \text{AFtoASAF}(E') = \{ \{ \alpha \} | \alpha \in \mathcal{A} \} \cup \{ \{ S \} | S \in \mathcal{E} \} \cup \{ \{ \beta \} | \beta \in \mathcal{B} \} \}

The following theorem states that the AF for a given ASAF yields extensions which are equivalent to those of the ASAF.

Theorem 2. Let \( \Delta = \langle A, R, S \rangle \) be an ASAF, \( \Delta_{AF} = \langle A, R \rangle \) its AF, \( E \subseteq \{ A \cup R \setminus S \} \), \( E' \subseteq A \) and \( S \subseteq \{ \text{co.pr.st.gr} \} \). It holds that \( E \) is an \( S \)-extension of \( \Delta \) iff \( \text{ASAFtoAF}(E) \) is an \( S \)-extension of \( \Delta_{AF} \). Equivalently, it holds that \( E' \) is an \( S \)-extension of \( \Delta_{AF} \) iff \( \text{AFtoASAF}(E') \) is an \( S \)-extension of \( \Delta \).

Then, given an ASAF and an update, we can obtain its AF and perform the update on this AF. For this purpose, we redefine an update \( u \) over an ASAF in terms of a set of updates \( u' \) over its AF, and say that \( u' \) is a set of updates corresponding to \( u \). Briefly, each update identified in Definition 8 will be expressed as a set of updates in terms of the elements of the AF, following the translation given in Definition 9. In case of an addition update \(+X\), the corresponding arguments \( X \) and \( X^* \) will be added first. Then, the set of updates on the AF can be directly obtained by applying Definition 9: it is the set of attacks belonging to the AF for the updated ASAF but not belonging to the AF for the initial ASAF. On the other hand, the set of updates on the AF corresponding to the removal of an attack or a support in the ASAF are those identified in Definition 9 by the attacks corresponding to the mapped attack or support.

4 Computing Extensions of Updated ASAFs

Given an ASAF \( \Delta_0 \), a semantics \( S \subseteq \{ \text{co.pr.st.gr} \} \), an extension \( E_0 \in \mathcal{E}(\Delta_0) \), and an update \( u = \pm X \), we define an incremental algorithm (Algorithm 1) for computing an \( S \)-extension \( E \) of the updated ASAF \( u(\Delta_0) \), if it exists.

Algorithm 1 first checks if the update is irrelevant at Line 1, where \( \text{checkIrrelevantUpdate}(\Delta_0, u, E_0, S) \) is a function returning true if some condition of Theorem 1 holds. If this is the case, Algorithm 1 returns the updated extension directly obtained from \( E_0 \) as given in Theorem 1. Otherwise, it computes the AF \( \Delta_{AF} \) for \( \Delta_0 \) (Line 4) according to Definition 9, and, if the update is an addition, augments \( \Delta_{AF} \) with the auxiliary arguments \( X \) and \( X^* \), obtaining the \( \Delta_{AF}' \). Then, the set \( u' \) of updates over \( \Delta_{AF}' \) corresponding to \( u \) is built and an \( S \)-extension \( E_0' \) of \( \Delta_{AF}' \) corresponding to \( E_0 \) is computed (Lines 6–7). Next, function \( \text{incrAlg} \) encoding the incremental algorithm proposed in [1] for Dung’s AFs is invoked. \( \text{incrAlg} \) takes as input the parameters \( \Delta_{AF}' \), \( u' \), \( S \), \( E_0' \), and an external solver \( \text{Solver} \) that computes an \( S \)-extension for the input AF. \( \text{incrAlg} \) returns an \( S \)-extension for \( u' \) (Line 8) by first identifying a suitable sub-graph of the updated AF, then computing an \( S \)-extension.
Algorithm 1 DynamicASAF($\Delta_0$, $\alpha$, $S$, $E_0$, Solver$_G$)

Input: $\Delta_0 = \langle A, R, S \rangle$, update $\alpha = \pm X$ ($X$ is either an attack or a support), semantics $S \in \{co, pr, at, gr\}$, initial $S$-extension $E_0$ of $\Delta_0$, function Solver$_G((A, R))$ returning an $S$-extension for an AF $(A, R)$ if it exists, $\perp$ otherwise.

Output: $S$-extension $E$ for $u(\Delta_0)$ if it exists, $\perp$ otherwise.

1: if checkIrrelevantUpdate($\Delta_0$, $u$, $E_0$, $S$) then
2:     Obtain $E$ from $E_0$ as per Theorem 1;
3:     return $E$;
4: else
5:     Let $\Delta_{AF} = (A, R)$ be the AF for $\Delta_0$;
6:     Let $\Delta'_{AF} = (A \cup \{X, X^{*}\}, R)$ if $u = +X$, otherwise $\Delta'_{AF} = \Delta_{AF}$;
7:     Let $u'$ be the set of updates over $\Delta'_{AF}$ corresponding to $\alpha$;
8:     Let $E' = $ incrAlg$(\Delta'_{AF}, u', S, E_0, $Solver$_G)$;
9:     if $(E' \neq \perp)$ then
10:         $E = AFtoASAF(E')$;
11:     else
12:         return $\perp$;
end if
end if

on this part only by using $Solver_G$, and finally merging it with $E_0$ to get an extension $E'$ of the updated AF. Finally, the extension of the updated ASAF (if any) is obtained by mapping the extension $E'$ obtained by incrAlg to the corresponding ASAF extension using the result of Theorem 2 (Line 10).

As stated next, Algorithm 1 is sound and complete.

Theorem 3. For any ASAF $\Delta_0$, extension $E_0 \in E_S(\Delta_0)$, update $u$, and semantics $S \in \{co, pr, at, gr\}$, if Solver$_G$ is sound and complete then Algorithm 1 computes $E \in E_S(u(\Delta_0))$ if $E_S(\Delta_0)$ is not empty, otherwise it returns $\perp$.

5 Empirical Evaluation

We implemented a C++ prototype and, for each semantics $S \in \{co, pr, at, gr\}$, compared the performance of Algorithm 1—where Solver$_G$ is a solver that won the last or second-to-last edition of the ICCMA competition for the task of determining an $S$-extension against the computation from scratch, that is, the computation of an extension of the updated ASAF by running Solver$_G$ directly on the corresponding updated AF. The latter is the best competitor we can compare with, as, to our knowledge, there is no other available ASAF solver. As for Solver$_G$, it is either $\mu$-toksia [44] that won the ICCMA’19 competition for all the considered semantics, or ArgSemSAT [28] for $S = pr$, goDIAMOND [48] for $S = at$, heureka [36] for $S = gr$, and cegariz [34] for $S = co$, the winners of ICCMA’17.

Dataset. Although there are several benchmark generators and solvers for Dung’s AFs [49], no benchmark is available for ASAFs. Thus, we generated a set of benchmark ASAFs by starting from AFs used as benchmarks at ICCMA for the tracks SE-pr and SE-at of determining an $S$-extension. Specifically, we use the AF datasets named B1 and B2 both consisting of 50 AFs, and B3 consisting of 100 AFs, and given a benchmark AF $(A, R)$, we generate an ASAF as follows: 30% of attacks in $R$ are transformed into first-level supports; 12% (resp. 3%) of attacks in $R$ are transformed into second-level supports towards a support (resp. an attack); 3% (resp. 2%) of attacks in $R$ are transformed into third-level supports supports towards a support (resp. an attack); 12% (resp. 3%) of attacks in $R$ are transformed into second-level attacks towards an attack (resp. a support); 2% (resp. 3%) of attacks in $R$ are transformed into third-level attacks towards an attack (resp. a support); the remaining 30% of attacks in $R$ are kept as first-level attacks of the resulting ASAF. This benchmark generation process aimed at preserving AFs’ topology as much as possible. However, the process of generating ASAF benchmarks starting from AF benchmarks is challenging because we require specific amounts of different kinds of attacks and supports, and we also need to check that the sub-graph induced by first-level supports is acyclic. Hence, to make it feasible, for each dataset, we generated an ASAF $\Delta$ if the number of arguments $A$ of the associated AF $\Delta_{AF}$ does not exceed the number of arguments of the biggest AF in the original dataset. Therefore, starting from the AF datasets B1, B2, and B3, we obtained an ASAF dataset B3 consisting of 139 ASAFs $(A, R, S)$ with a number of arguments $|A| \in [35, 2.2K]$ and a number of interactions $|R \cup S| \in [48, 271K]$. The associated set of AFs consists of 139 AFs $(A, R)$ with $|A| \in [138, 251K]$ and $|R| \in [128, 338K]$.

Methodology. For each semantics $S \in \{co, pr, at, gr\}$ and ASAF $\Delta_0$ in the datasets, we consider i) a randomly selected $S$-extension $E_0$ of $\Delta_0$ (initial extension), and ii) an update $u$ selected among one of the possible 12 types (addition/deletion of an attack/support towards an argument/attack/support). Next, we compute an $S$-extension $E$ of the updated ASAF $u(\Delta_0)$ by calling our incremental algorithm DynamicASAF. Finally, the average run time of DynamicASAF to compute an $S$-extension is compared with the average run time required by the competitor, which computes an $S$-extension of the updated ASAF by directly computing an $S$-extension of the updated AF for $u(\Delta_0)$. The experiments have been carried out on an Intel Core i7-3770K CPU 3.5GHz, 12GB RAM, running Ubuntu.

Results. Figure 3 reports the average run times (log scale) of the incremental computation (DynamicASAF) and the computation from scratch over $B_\Delta$ for the grounded (top), preferred (center) and stable (bottom) semantics. Each data point reported in the figure is the average over 12 runs, each of them corresponding to a different update. For the sake of readability, Figure 3 also shows the lines obtained by linear regression, and the results obtained for the complete semantics are not shown as they were analogous to those obtained for the grounded semantics. Moreover, the diagrams for the stable and grounded semantics contain less data points than those for the preferred one. As for the stable semantics, the missing points correspond to generated benchmark ASAFs having no stable extension (it is not possible to run Algorithm 1 without the input parameter $E_0$). As for the grounded semantics, the missing points correspond to ASAFs whose grounded extension is empty, for which we found that running the incremental algorithm gives worthless improvement—in fact, for the grounded semantics, one could run the incremental approach only if $E_0 \neq \emptyset$. This is due to the fact that incrAlg uses the initial extension $E_0$ to compute a sub-AF (called reduced AF) that is used for the computation of an extension of the updated AF. But if $E_0$ is empty and the considered semantics is polynomial, as for the grounded one [33], the overhead of computing the reduced AF may not pay off. Finally, the diagrams on the right-hand side do not contain data points for a large number of ASAF interactions because the solver ran into memory capacity saturation.

The results in Figure 3 show that, on average, the incremental algorithm outperforms the competitors that compute the extensions from scratch. However, the improvement obtained for the preferred and stable semantics—for which DynamicASAF is between one and two orders of magnitude faster than the competitors—is better than that obtained for the grounded semantics. This happens because the computation from scratch is for a given semantics, the larger the improvements are for incrAlg [1]. In fact, the improve-
the principles according to which the grounded extension of an AF does not change when the set of arguments/attacks are changed. [24, 25] addressed the problem of revising the set of extensions of an AF, and studied how the extensions can evolve when a new argument is considered. [18] have studied the evolution of the set of extensions after performing a change operation (addition/removal of arguments/interactions). Dynamic argumentation has been applied to decision-making of an autonomous agent by [6], where it is studied how the acceptability of arguments evolves when a new argument is added to the decision system. The division-based method, proposed by [42] and then refined by [12], divides the updated framework into two parts: affected and unaffected, where only the status of affected arguments is recomputed after updates. [41] investigated the efficient evaluation of the justification status of a subset of arguments in an AF (instead of the whole set of arguments), and proposed an approach based on answer-set programming for local computation. In [40], an AF is decomposed into a set of strongly connected components, yielding sub-AFs located in layers, which are then used for incrementally computing the semantics of the given AF by proceeding layer by layer. [50] introduced a matrix representation of argumentation frameworks and proposed a matrix reduction that, when applied to dynamic AFs, resembles the division-based method in [42].

Relevant work on dynamic aspects of Dung’s AFs also includes the following. [13] have proposed an approach exploiting the concept of splitting of logic programs to deal with dynamic argumentation. The technique considers weak expansions of the initial AF, where added arguments never attack previous ones. [16] have investigated whether and how it is possible to modify a given AF so that a desired set of arguments becomes an extension, whereas [46] have studied equivalence between two AFs when further information (another AF) is added to both AFs. [14] have focused on expansions where new arguments and attacks may be added but the attacks among the old arguments remain unchanged, while [15] characterized update and deletion equivalence, where adding/deleting arguments/attacks is allowed. (deletions were not considered by [46, 14]).

Bipolarity in argumentation is discussed in [5], where a survey of the use of bipolarity and a formal definition of bipolar argumentation framework (BAF) that extends Dung’s AF by including supports is provided. A survey of different approaches to support in argumentation can be found in [30]. Changes in bipolar argumentation frameworks (BAFs) have been studied in [27], where it is shown how the addition of one argument together with one support involving it (and without any attack) impacts the extensions of the updated BAF. The problem of incrementally computing extensions of dynamic BAFs, with deductive interpretation of support [20], has been first addressed in [1], and then extended in [2] to deal with second-order attacks [9]. However, none of the existing approaches deals with the incremental computation and the empirical evaluation of a general framework as the ASAF framework.

Finally, it is worth noting that our translation from an ASAF to an AF improves that of [37] from two standpoints: i) it is direct, not requiring the two-step process of [37] which first obtains an Argumentation Framework with Necessities (AFN) [45] and then an AF; ii) the size of the resulting AF is smaller than that of [37].

7 Conclusions and Future Work

To the best of our knowledge, this is the first paper addressing the problem of efficiently and incrementally computing extensions for dynamic ASAFs. Given the generality of the ASAF, our technique can be also applied to restricted frameworks such as Argumentation Framework with Necessities (AFN) [45] and then an AF; ii) the size of the resulting AF is smaller than that of [37].