Incremental Computation of Grounded Semantics for Dynamic Abstract Argumentation Frameworks

Sergio Greco and Francesco Parisi(✉)

Department of Informatics Modeling, Electronics and System Engineering, University of Calabria, Rende, Italy
{greco,fparisi}@dimes.unical.it

Abstract. Abstract argumentation has emerged as a central field in Artificial Intelligence. Although the underlying idea is very simple and intuitive, most of the semantics proposed so far suffer from a high computational complexity. For this reason, in recent years, an increasing amount of work has been done to define efficient algorithms. However, so far, the research has concentrated on the definition of algorithms for static frameworks, whereas argumentation frameworks (AFs) are highly dynamic in practice. Surprisingly, the definition of evaluation algorithms taking into account such dynamic aspects has been mostly neglected. In this paper, we address the problem of efficiently recomputing the extensions of AFs which are updated by adding/deleting arguments or attacks. In particular, after identifying some properties that hold for updates of AFs under several well-known semantics, we focus on the most popular unique-status semantics (namely, the *grounded* semantics) and present an algorithm for its incremental computation, well-suited to dynamic applications where updates to an initial AF are frequently performed to take into account new available knowledge.

1 Introduction

Abstract argumentation has emerged as a central field in Artificial Intelligence [3,10,26,42,44,45]. Although the underlying idea is very simple and intuitive, most of the semantics proposed so far suffer from a high computational complexity [22–25,28–32]. Complexity bounds and evaluation algorithms for argumentation frameworks (AFs) have been deeply studied in the literature, but this research focused on 'static' frameworks, whereas, in practice, AFs are not static systems [4,5,19,27,39]. Typically an AF represents a temporary situation as new arguments and attacks continuously can be added/removed to take into account new available knowledge. This may change significantly the conclusions that can be derived. For instance, when a new attack is added to an AF, existing attacks may cease to apply and new attacks become applicable.

Surprisingly, the definition of evaluation algorithms and the analysis of the computational complexity taking into account such dynamic aspects have been mostly neglected, whereas in these situations incremental computation techniques can greatly improve performance. Sometimes changes to the AF can

© Springer International Publishing AG 2017
R. Aydoğan et al. (Eds.): COREDEMA 2016, LNAI 10238, pp. 66–81, 2017.
DOI: 10.1007/978-3-319-57285-7_5

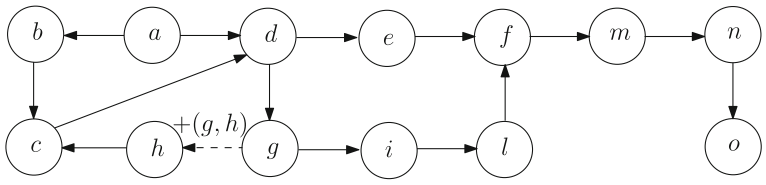

Fig. 1. AFs $\mathcal{A}_0$ and $\mathcal{A} = +(g,h)(\mathcal{A}_0)$

make small changes to the set of conclusions, and recomputing the whole semantics from scratch can be avoided. For instance, consider the situation shown in Fig. 1: the initial AF $\mathcal{A}_0$, where h is not attacked by any other argument, is updated to AF $\mathcal{A}$ by adding attack (g, h). According to the most popular argumentation semantics, i.e. *grounded*, *complete*, *ideal*, *preferred*, *stable*, and *semi-stable* [15,20,21], the initial AF $\mathcal{A}_0$ admits the extension $E_0 = \{a, h, g, e, l, m, o\}$, whereas the extension for the updated framework $\mathcal{A}$ becomes $E = \{a, c, g, e, l, m, o\}$. As it will be shown later, for the grounded semantics the extension E can be efficiently computed incrementally by looking only at a small part of the AF, which is "influenced by" the update operation. This part is just $\{h, c\}$ in our example, and we will show that the membership of the other arguments to E does not depend on the update operation, and thus we do not need to compute them again after performing update $+(g, h)$.

Contributions. The main contributions are as follows:

– We introduce the concept of *influenced set* consisting of the arguments whose status could change after an update. The influenced set refines the previously proposed set of *affected arguments* [4,39] and makes the computation more efficient.
– We present an incremental algorithm for recomputing the grounded extension. It is very efficient as it (iteratively) computes the status of influenced arguments only and when it finds that the status of arguments derived at some step cannot be changed by subsequent steps then it stops.
– We present experimental results showing the effectiveness of our approach.

Plan of the paper. We start by discussing related works in Sect. 2, and reviewing Dung's abstract argumentation framework in Sect. 3, where updates are introduced. Next, we identify some sufficient conditions on the updates that guarantee that the semantics of an AF does not change, and introduce the concept of influenced set in Sect. 4. Then we provide our algorithm for incrementally computing the grounded semantics in Sect. 5. In Sect. 6 we present experimental results on two datasets showing that our incremental approach outperforms the computation from scratch of the grounded semantics. Finally, we draw conclusions and discuss future work in Sect. 7, where we also discuss how the results presented in the paper carry over to the case of sets of updates to be performed simultaneously.

2 Related Work

There have been several efforts coping with dynamics aspects of abstract argumentation. In [12,13] the principles according to which the extension does not change when the set of arguments/attacks are changed have been studied. However, this work does not consider how the extensions of an AF evolve when new arguments are added or some of the old ones are removed. [17,18] 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. However, they focus on adding only one argument interacting with one initial argument (i.e. an argument which is not attacked by any other argument). The work in [17,18] has been extended in [11], where the evolution of the set of extensions after performing a change operation (addition/removal of arguments/interaction) is studied. Dynamic argumentation has been applied to decision-making of an autonomous agent in [1], where it is studied how the acceptability of arguments evolves when a new argument is added to the decision system. However, they do not compute the whole extensions and also focused on the case where only one argument is added to the system.

The division-based method, proposed in [39] and refined in [4], divides the updated framework into two parts: *affected* and *unaffected*, where only the status of affected arguments is recomputed after updates. However, the set of affected arguments consists of those that are reachable from the updated arguments, which is often larger than the set that actually needs to be considered when recomputing the extension. For the AF of Fig. 1, all the arguments in the chains originated by h turn out to be 'affected'. But we only need to recompute the status of h and c after the update. Recently, [48] introduced a matrix representation of AFs and proposed a matrix reduction that, when applied to dynamic AFs, resembles the division-based method in [39]. In [5,9] an approach exploiting the concept of splitting of logic programs [40] was adopted to deal with dynamic argumentation. However, the technique considers weak expansions of the initial AF, where added arguments never attack previous ones. Recently, [16] studied the relationship between argumentation and logic programming [14,35,36].

[8] investigated whether and how it is possible to modify a given AF in such a way that a desired set of arguments becomes an extension, whereas [43] studied equivalence between two AFs when further information (another AF) is added to both simultaneously. [6] focused on specific expansions where new arguments and attacks may be added but the attacks among the old arguments remain unchanged, while [7] characterized update and deletion equivalence, where adding as well as deleting arguments and attacks is allowed (deletions were not considered in [6,43]).

To the best of our knowledge, this is the first paper that exploits the initial extension E_0 of an AF $\mathcal{A}_0$ not only for computing the set $\mathcal{I}(u, \mathcal{A}_0, E_0)$ of arguments influenced by an update u but also for recomputing the status of the arguments in $\mathcal{I}(u, \mathcal{A}_0, E_0)$ by applying early termination conditions. A short version of this paper appeared in [37].

3 Preliminaries

We assume the existence of a set Arg whose elements are called *arguments*. An *(abstract) argumentation framework* [20] (AF) is a pair $\langle A, \Sigma \rangle$, where $A \subseteq Arg$ is a finite set whose elements are referred to as *arguments*, and $\Sigma \subseteq A \times A$ is a binary relation over A whose elements are referred to as *attacks*. Essentially, an AF is a directed graph in which the arguments are represented by the nodes and the attack relation is represented by the set of directed edges. An argument is an abstract entity whose role is entirely determined by its relationships with other arguments.

Given arguments $a, b \in A$, we say that a *attacks* b iff $(a, b) \in \Sigma$. An argument a *attacks* a set $S \subseteq A$ iff $\exists\, b \in S$ such that a *attacks* b.

We use $S^+ = \{b \mid \exists a \in S : (a, b) \in \Sigma\}$ and $S^- = \{b \mid \exists a \in S : (b, a) \in \Sigma\}$ to denote the sets of all arguments that are attacked by S and attack S, respectively.

A set $S \subseteq A$ *defends* a iff $\forall b \in A$ such that b *attacks* a, there is $c \in S$ such that c *attacks* b.

A set $S \subseteq A$ of arguments, is said to be

(i) *conflict-free*, if there are no $a, b \in S$ such that a *attacks* b;
(ii) *admissible*, if it is conflict-free and it defends all its arguments.

An argumentation semantics specifies the criteria for identifying a set of arguments considered to be "reasonable" together, called *extension*. A *complete extension* ($\texttt{co}$) is an admissible set that contains all the arguments that it defends. A complete extension S is said to be:

- *preferred (*$\texttt{pr}$*)* iff it is maximal (w.r.t. $\subseteq$);
- *semi-stable (*$\texttt{ss}$*)* iff $S \cup S^+$ is maximal (w.r.t. $\subseteq$);
- *stable (*$\texttt{st}$*)* iff it attacks each argument in $A \setminus S$;
- *grounded (*$\texttt{gr}$*)* iff it is minimal (w.r.t. $\subseteq$);
- *ideal (*$\texttt{id}$*)* iff it is contained in every preferred extension and it is maximal (w.r.t. $\subseteq$).

Given an AF $\mathcal{A}$ and a semantics $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{st}, \texttt{gr}, \texttt{id}\}$, we use $\mathcal{E}_{\mathcal{S}}(\mathcal{A})$ to denote the set of $\mathcal{S}$-extensions of $\mathcal{A}$.

All the above-mentioned semantics except the stable admit at least one extension, and the grounded and ideal admit exactly one extension [15,20,21]. That is, for $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{gr}, \texttt{id}\}$ it is the case that $\mathcal{E}_{\mathcal{S}}(\mathcal{A}) \neq \emptyset$, while $\mathcal{E}_{\texttt{st}}(\mathcal{A})$ may be empty. Semantics $\texttt{gr}$ and $\texttt{id}$ are called *unique status* semantics as $|\mathcal{E}_{\texttt{gr}}(\mathcal{A})| = |\mathcal{E}_{\texttt{id}}(\mathcal{A})| = 1$, whereas the others are called *multiple status* semantics. It is well-known that, for any AF $\mathcal{A}$, $\mathcal{E}_{\texttt{gr}}(\mathcal{A}) \subseteq \mathcal{E}_{\texttt{co}}(\mathcal{A})$ and $\mathcal{E}_{\texttt{id}}(\mathcal{A}) \subseteq \mathcal{E}_{\texttt{co}}(\mathcal{A})$, and $\mathcal{E}_{\texttt{st}}(\mathcal{A}) \subseteq \mathcal{E}_{\texttt{ss}}(\mathcal{A}) \subseteq \mathcal{E}_{\texttt{pr}}(\mathcal{A}) \subseteq \mathcal{E}_{\texttt{co}}(\mathcal{A})$.

Example 1. Consider the AF $\mathcal{A}_0$ shown in Fig. 2. Then, the set of admissible sets is $\{\ \emptyset,\ \{a\},\ \{d\},\ \{a, d\},\ \{b, d\}\ \}$, and $\mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$ with $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{st}, \texttt{gr}, \texttt{id}\}$ is as reported in the second column of Fig. 3. □

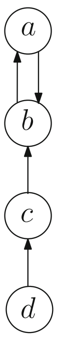

Fig. 2. AF $\mathcal{A}_0$ Example 1.

$\mathcal{S}$	$\mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$	$\mathcal{E}_{\mathcal{S}}(\mathcal{A}_1)$	$\mathcal{E}_{\mathcal{S}}(\mathcal{A}_2)$
co	$\{\{d\},\{a,d\},\{b,d\}\}$	$\{\emptyset,\{a,d\}\}$	$\{\emptyset,\{a,d\},\{b,c\}\}$
pr	$\{\ \{a,d\},\{b,d\}\ \}$	$\{\ \{a,d\}\ \}$	$\{\ \{a,d\},\{b,c\}\ \}$
ss	$\{\ \{a,d\},\{b,d\}\ \}$	$\{\ \{a,d\}\ \}$	$\{\ \{a,d\},\{b,c\}\ \}$
st	$\{\ \{a,d\},\{b,d\}\ \}$	$\{\ \{a,d\}\ \}$	$\{\ \{a,d\},\{b,c\}\ \}$
gr	$\{\ \{d\}\ \}$	$\{\ \emptyset\ \}$	$\{\ \emptyset\ \}$
id	$\{\ \{d\}\ \}$	$\{\ \{a,d\}\ \}$	$\{\ \emptyset\ \}$

Fig. 3. Sets of extensions for the AF of Example 1, and changes in the sets after performing updates $+(b,d)$ and $-(c,b)$.

The argumentation semantics can be also defined in terms of *labelling*. A labelling for an AF $\mathcal{A} = \langle A, \Sigma \rangle$ is a total function $L : A \rightarrow \{\text{IN}, \text{OUT}, \text{UN}\}$ assigning to each argument a label. $L(a) = \text{IN}$ means that argument a is accepted, $L(a) = \text{OUT}$ means that a is rejected, while $L(a) = \text{UN}$ means that a is undecided.

Let $in(L) = \{a \mid a \in A \wedge L(a) = \text{IN}\}$, $out(L) = \{a \mid a \in A \wedge L(a) = \text{OUT}\}$, and $un(L) = \{a \mid a \in A \wedge L(a) = \text{UN}\}$. In the following, we also use the triple $\langle in(L), out(L), un(L)\rangle$ to represent L.

A labelling L is said to be *admissible (or legal)* if $\forall a \in in(L) \cup out(L)$ (i) if $L(a) = \text{OUT}$ then $\exists\, b \in A$ such that $(b, a) \in \Sigma$ and $L(b) = \text{IN}$; and (ii) if $L(a) = \text{IN}$ then $L(b) = \text{OUT}$ for all $b \in A$ such that $(b, a) \in \Sigma$. L is a complete labelling iff conditions (i) and (ii) hold for all $a \in A$.

Between complete extensions and complete labellings there is a bijective mapping defined as follows: for each extension E there is a unique labelling $L = \langle E, E^+, A \setminus (E \cup E^+)\rangle$ and for each labelling L there is a unique extension $in(L)$. We say that L is the labelling *corresponding* to E.

In the following, we say that the *status of an argument* a w.r.t. a labelling L (or its corresponding extension $in(L)$) is IN (resp., OUT, UN) iff $L(a) = \text{IN}$ (resp., $L(a) = \text{OUT}$, $L(a) = \text{UN}$). We will avoid to mention explicitly the labelling (or the extension) whenever it is understood.

Updates. An *update* u for an AF $\mathcal{A}_0$ consists in modifying $\mathcal{A}_0$ into an AF $\mathcal{A}$ by adding or removing arguments or attacks.

In the following, we focus on updates consisting of adding/deleting one attack between arguments belonging to $\mathcal{A}_0$. As we discuss in Sect. 7, focusing on single attack updates is not a limitation as *multiple (attack) updates* to be performed simultaneously can be simulated by means of a single attack update.

Concerning the addition (resp. deletion) of a set of isolated arguments, it is easy to see that if $\mathcal{A}$ is obtained from $\mathcal{A}_0$ through the addition (resp. deletion) of a set S of isolated arguments, then, let E_0 be an extension for $\mathcal{A}_0$, $E = E_0 \cup S$ (resp. $E = E_0 \setminus S$) is an extension for $\mathcal{A}$ that can be trivially computed. Of course, if arguments in S are not isolated, we can first delete all attacks involving arguments in S; adding an attack between an argument in $\mathcal{A}_0$ and a new argument can be simulated as well.

We use $+(a,b)$ (resp. $-(a,b)$) to denote the addition (resp. deletion) of an attack (a,b), and $u(\mathcal{A}_0)$ to denote the application of update $u = \pm(a,b)$ to $\mathcal{A}_0$.

Updating an AF implies that its semantics (sets of extensions or labellings) changes, as shown in the following example.

Example 2. Consider the AF $\mathcal{A}_0$ of Example 1. For each semantics $\mathcal{S}$, the set $\mathcal{E}_\mathcal{S}(\mathcal{A}_1)$ of extensions for $\mathcal{A}_1 = +(b,d)(\mathcal{A}_0)$ is reported in the third column of Fig. 3. If update $-(c,b)$ is performed on $\mathcal{A}_1$, then $\mathcal{E}_\mathcal{S}(\mathcal{A}_2)$ with $\mathcal{A}_2 = -(c,b)(\mathcal{A}_1)$ is as shown on the last column of Fig. 3. $\square$

4 Influenced Arguments

In this section, we first identify conditions ensuring that a given $\mathcal{S}$-extension continues to be an $\mathcal{S}$-extension after an update, and then introduce the *influenced set* that will be used to limit the set of arguments that needs to be recomputed after an update. In addition, arguments not in the influenced set can be used to derive extensions for the updated AF.

The following two propositions introduce sufficient conditions guaranteeing that a given $\mathcal{S}$-extension is still an $\mathcal{S}$-extension after performing an update.

Proposition 1. *Let $\mathcal{A}_0$ be an AF, $u = +(a,b)$ an update, $\mathcal{S}$ a semantics, $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ an extension of $\mathcal{A}_0$ under semantics $\mathcal{S}$, and L_0 the labelling corresponding to E_0. Then $E_0 \in \mathcal{E}_\mathcal{S}(u(\mathcal{A}_0))$ if*

- *$\mathcal{S} \in \{\texttt{co}, \texttt{st}, \texttt{gr}\}$ and one of the following conditions holds:*
  - *$L_0(a) \neq \text{IN}$ and $L_0(b) \neq \text{IN}$,*
  - *$L_0(a) = \text{IN}$ and $L_0(b) = \text{OUT}$;*
- *$\mathcal{S} \in \{\texttt{pr}, \texttt{ss}, \texttt{id}\}$ and $L_0(b) = \text{OUT}$.*

Proposition 2. *Let $\mathcal{A}_0$ be an AF, $u = -(a,b)$, $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{st}, \texttt{gr}\}$, and $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ an extension of $\mathcal{A}_0$ under $\mathcal{S}$. Then $E_0 \in \mathcal{E}_\mathcal{S}(u(\mathcal{A}_0))$ if one of the following conditions holds:*

(1) $L_0(a) = \text{OUT}$;
(2) $L_0(a) = \text{UN}$ and $L_0(b) = \text{OUT}$.

Example 3. Consider the AFs $\mathcal{A}_1 = +(b,d)(\mathcal{A}_0)$ and $\mathcal{A}_2 = -(c,b)(\mathcal{A}_1)$, where $\mathcal{A}_0$ is the AF of Example 2. For $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{st}\}$, extension $\{a,d\}$ of $\mathcal{A}_1$ is still an extension of $\mathcal{A}_2$ as $L_0(c) = \text{OUT}$ (see Fig. 3). The grounded extension $\emptyset$ of $\mathcal{A}_1$ is still a grounded extension of $\mathcal{A}_2$, whereas the ideal extension $\{a,d\}$ of $\mathcal{A}_1$ is not the ideal extension of $\mathcal{A}_2$. $\square$

Given an AF $\mathcal{A} = \langle A, \Sigma \rangle$ and an argument $b \in A$, we denote as $Reach_\mathcal{A}(b)$ the set of arguments that are reachable from b in $\mathcal{A}$. We now introduce the *influenced set*.

Definition 1 (Influenced set). Let $\mathcal{A} = \langle A, \Sigma \rangle$ be an AF, $u = \pm(a, b)$ an update, E an extension of $\mathcal{A}$ under a given semantics $\mathcal{S}$, and let

–
$$\mathcal{I}_0(u, \mathcal{A}, E) = \begin{cases} \emptyset \;\; if \;\; E \in \mathcal{E}_{\mathcal{S}}(u(\mathcal{A})) \;\; \text{[i.e., the conditions of Prop. 1/2 hold]} \; or \\ \quad \exists (z, b) \in \Sigma \;\, s.t. \;\, z \in E \wedge z \notin Reach_{\mathcal{A}}(b); \\ \\ \{b\} \;\; otherwise; \end{cases}$$

– $\mathcal{I}_{i+1}(u, \mathcal{A}, E) = \mathcal{I}_i(u, \mathcal{A}, E) \cup \{y \mid \exists (x, y) \in \Sigma \;\, s.t. \;\, x \in \mathcal{I}_i(u, \mathcal{A}, E) \wedge \nexists (z, y) \in \Sigma \;\, s.t. \;\, z \in E \wedge z \notin Reach_{\mathcal{A}}(b)\}$.

The *influenced set* of update u w.r.t. AF $\mathcal{A}$ and the extension E is $\mathcal{I}(u, \mathcal{A}, E) = \mathcal{I}_n(u, \mathcal{A}, E)$ such that $\mathcal{I}_n(u, \mathcal{A}, E) = \mathcal{I}_{n+1}(u, \mathcal{A}, E)$. □

Thus, the set of arguments that are influenced by an update of the status of b are those that can be reached from b without using any intermediate argument y whose status is known to be OUT because it is determined by an argument $z \in E$ which is not reachable from (and thus not influenced by) b.

Example 4. For the AF $\mathcal{A}_0 = \langle A_0, \Sigma_0 \rangle$ of Fig. 1, whose grounded extension is $E_0 = \{a, h, g, e, l, m, o\}$, we have that $Reach_{\mathcal{A}_0}(h) = A_0 \setminus \{a, b\}$, and the influenced set of $u = +(g, h)$ is $\mathcal{I}(u, \mathcal{A}_0, E_0) = \{h, c\}$. Note that $d \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ since it is attacked by $a \in E_0$. Thus the arguments that can be reached only using d cannot belong to $\mathcal{I}(u, \mathcal{A}_0, E_0)$ either.

For $\mathcal{A} = u(\mathcal{A}_0)$, whose the grounded extension is $E = \{a, c, g, e, l, m, o\}$, we have that $S = \mathcal{I}(u, \mathcal{A}, E)$ is still $\{h, c\}$. Therefore, only the status of arguments in S could change and their status can be determined by considering a restricted AF containing only arguments in $S \cup S^-$. □

Proposition 3. *Given an AF $\mathcal{A} = \langle A, \Sigma \rangle$, an update $u = \pm(a, b)$, and an extension E, the complexity of computing the influenced set of u w.r.t. $\mathcal{A}$ and E is $O(|\Sigma|)$.*

All the arguments not belonging to the influenced set of an update will still belong to an extension of the updated AF.

Theorem 1. *Let $\mathcal{A}_0$ be an AF, and $\mathcal{A} = u(\mathcal{A}_0)$ be the AF resulting from performing update $u = \pm(a, b)$ on $\mathcal{A}_0$. Let $E_0 \in \mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$ be an extension for $\mathcal{A}_0$ under any semantics $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{st}, \texttt{gr}, \texttt{id}\}$. Let $\overline{\mathcal{I}} = Arg \setminus \mathcal{I}(u, \mathcal{A}_0, E_0)$ be the set of the arguments that are not influenced by u in $\mathcal{A}_0$ w.r.t. E_0. Then, either $\mathcal{E}_{\mathcal{S}}(\mathcal{A}) = \emptyset$ or there is an extension $E \in \mathcal{E}_{\mathcal{S}}(\mathcal{A})$ for $\mathcal{A}$ such that $(E \cap \overline{\mathcal{I}}) = (E_0 \cap \overline{\mathcal{I}})$.*

Observe that the set of extensions may be empty only for the stable semantics. For all of the other semantics, the theorem suggests the following strategy for computing an extension of the updated AF: derive it by first projecting out the set of arguments not influenced by the update and then extend the so-obtained set by using the information provided by it.

We conclude this section by introducing a refinement of Proposition 1 that makes use of the influenced set. This result will be used in the next section to restrict the input AF.

Proposition 4. *Let $\mathcal{A}_0$ be an AF, $u = +(a, b)$, $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{st}, \texttt{gr}\}$, and $E_0 \in \mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$ an extension of $\mathcal{A}_0$ under $\mathcal{S}$. Then $E_0 \in \mathcal{E}_{\mathcal{S}}(u(\mathcal{A}_0))$ if*

- *one of the conditions of Proposition 1 holds or*
- *the next three conditions hold:*
  (1) $L_0(a) =$ OUT,
  (2) $L_0(b) =$ IN, and
  (3) either (i) $\mathcal{S} \in \{\texttt{co}, \texttt{st}, \texttt{ss}, \texttt{pr}\}$ or (ii) $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ and $\mathcal{S} = \texttt{gr}$.

Example 5. Consider AFs $\mathcal{A}_0$ and $\mathcal{A}_1 = +(b, d)(\mathcal{A}_0)$ of Examples 1 and 3. For $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{ss}, \texttt{st}\}$, extension $E_0 = \{a, d\}$ for $\mathcal{A}_0$ is still an extension of the AF $\mathcal{A}_1$ as $L_0(b) =$ OUT and $L_0(d) =$ IN (see Fig. 3). However, the grounded extension $E'_0 = \{d\}$ for $\mathcal{A}_0$ is not guarantee to be a grounded extension for $\mathcal{A}_1$ as neither Proposition 1 nor conditions 1) and 3.*ii*) of Proposition 4 hold (b is UN and $b \in \mathcal{I}(+(b, d), \mathcal{A}_0, E'_0)$). □

5 Recomputing the Grounded Semantics

Given an AF $\mathcal{A}_0$, the grounded extension E_0 for $\mathcal{A}_0$, an update u for $\mathcal{A}_0$ yielding $\mathcal{A} = u(\mathcal{A}_0)$, we address the problem of efficiently computing the grounded extension E of the updated AF $\mathcal{A}$ starting from E_0.

For any AF $\mathcal{A} = \langle A, \Sigma \rangle$ and set $S \subseteq A$ of arguments, we denote with $\Pi(S, \mathcal{A}) = \langle S, \Sigma \cap S \times S \rangle$ the subgraph of $\mathcal{A}$ induced by the nodes in S. Moreover, given two AFs $\mathcal{A}_1 = \langle A_1, \Sigma_1 \rangle$ and $\mathcal{A}_2 = \langle A_2, \Sigma_2 \rangle$, we denote as $\mathcal{A}_1 \sqcup \mathcal{A}_2 = \langle A_1 \cup A_1, \Sigma_1 \cup \Sigma_2 \rangle$ the union of the two AFs.

Our algorithm first identifies the restricted subgraph of the given AF containing the arguments influenced by the update.

Definition 2. (Restricted AF for grounded semantics). Given an AF $\mathcal{A} = \langle A, \Sigma \rangle$, a grounded extension E for $\mathcal{A}$, and an update $u = \pm(a, b)$, the restricted AF of $\mathcal{A}$ w.r.t. E and u (denoted as $\mathcal{R}_{\texttt{gr}}(u, \mathcal{A}, E)$) is as follows.

- $\mathcal{R}_{\texttt{gr}}(u, \mathcal{A}, E)$ is empty if $\mathcal{I}(u, \mathcal{A}, E)$ is empty or one of the conditions of Proposition 4 holds.
- $\mathcal{R}_{\texttt{gr}}(u, \mathcal{A}, E) = \Pi(\mathcal{I}(u, \mathcal{A}, E), u(\mathcal{A})) \sqcup T_1 \sqcup T_2$ where:
  - T_1 is the union of the AFs $\langle \{a, b\}, \{(a, b)\} \rangle$ s.t. (a, b) is an attack of $u(\mathcal{A})$ and $a \notin \mathcal{I}(u, \mathcal{A}, E)$, $a \in E$, and $b \in \mathcal{I}(u, \mathcal{A}, E)$;
  - $T_2 = \langle \{c \,|\, Check(c)\}, \{(c, c) \,|\, Check(c)\} \rangle$, where $Check(c)$ is true if $\exists (e, c) \in \Sigma$ such that $c \in \mathcal{I}(u, \mathcal{A}, E)$ and $e \notin \mathcal{I}(u, \mathcal{A}, E)$ and $e \notin E \cup E^+$. □

Hence, AF $\mathcal{R}_{\texttt{gr}}(u, \mathcal{A}, E)$ contains, in addition to the subgraph of $u(\mathcal{A})$ induced by $\mathcal{I}(u, \mathcal{A}, E)$, additional nodes and edges containing needed information on the "external context", i.e. information about the status of arguments which are attacking some argument in $\mathcal{I}(u, \mathcal{A}, E)$. Specifically, if there is in $u(\mathcal{A})$ an edge from node $a \notin \mathcal{I}(u, \mathcal{A}, E)$ whose status is IN to node $b \in \mathcal{I}(u, \mathcal{A}, E)$, then we add the edge (a, b) so that, as a does not have incoming edges in $\mathcal{R}_{\texttt{gr}}(u, \mathcal{A}, E)$, its status is confirmed to be IN. Moreover, if there is in $u(\mathcal{A})$ an edge from a node

$e \notin \mathcal{I}(u, \mathcal{A}, E)$ to a node $c \in \mathcal{I}(u, \mathcal{A}, E)$ such that e is UN, we add edge (c, c) to $\mathcal{R}_{\mathbf{gr}}(u, \mathcal{A}, E)$ so that the status of c cannot be IN. Using fake arguments/attacks to represent external contexts has been exploited in a similar way in [2], where decomposability properties of argumentation semantics are studied.

Example 6. Continuing Example 4, $\mathcal{R}_{\mathbf{gr}}(+(g, h), \mathcal{A}_0, E_0)$ consists of the subgraph induced by $\mathcal{I}(u, \mathcal{A}_0, E_0) = \{h, c\}$ as well as the edge (g, h) which is an attack towards argument $h \in \mathcal{I}(u, \mathcal{A}_0, E_0)$ coming from argument g outside $\mathcal{I}(u, \mathcal{A}_0, E_0)$ labelled as IN. Hence, $\mathcal{R}_{\mathbf{gr}}(+(g, h), \mathcal{A}_0, E_0) = \langle A_d, \Sigma_d \rangle$ with $A_d = \{g, h, c\}$ and $\Sigma_d = \{(g, h), (h, c)\}$. □

Example 7. Consider the AF $\mathcal{A}_0 = \langle \{a, b, c, d, e, f, g\}, \{(a, b), (b, a), (c, d), (d, c), (a, c), (b, c), (f, c), (g, f)\} \rangle$ and the update $u = +(e, d)$. We have that

(i) the grounded extension of $\mathcal{A}_0$ is $E_0 = \{g, e\}$ (i.e. arguments a, b, c, d are all labeled as UN);
(ii) the influenced set is $\mathcal{I}(u, \mathcal{A}_0, E_0) = \{c, d\}$; and
(iii) the restricted AF is $\mathcal{R}_{\mathbf{gr}}(u, \mathcal{A}_0, E_0) = \langle \{c, d\}, \{(c, d), (d, c)\} \rangle \sqcup T_1 \sqcup T_2$ where $T_1 = \langle \{e, d\}, \{(e, d)\} \rangle$ and $T_2 = \langle \{c\}, \{(c, c)\} \rangle$.

That is, $\mathcal{R}_{\mathbf{gr}}(u, \mathcal{A}_0, E_0) = \langle \{c, d, e\}, \{(c, d), (d, c), (e, d), (c, c)\} \rangle$. □

Algorithm 1 first checks if the restricted AF (computed w.r.t. update $u = \pm(a, b)$) is empty (Line 3). If this is the case, then $E = E_0$. Otherwise, the status of arguments in $S = \mathcal{I}(u, \mathcal{A}_0, E_0)$ needs to be recomputed and the extension E of $u(\mathcal{A}_0)$ is constructed at Line 6 by combining the arguments in E_0 not belonging to the influenced part and the arguments returned by Function *IFP* (incremental fixpoint), which is invoked with AF $\mathcal{A}_d = \langle A_d, \Sigma_d \rangle$ (the restricted graph of $\mathcal{A}$) and starting extension $E_0 \cap A_d$ (the restriction of E_0 to A_d).

Function *IFP* first computes the set of nodes which are labelled IN and an initial set of nodes which are labelled OUT. If no argument can be labelled IN, it returns the empty set. Otherwise, it iteratively applies function G that takes as input the set of arguments S_{OUT} which have been labeled OUT so far and the subset $\Delta_{\text{OUT}} \subseteq S_{\text{OUT}}$ of arguments which have been labelled OUT in the last step, and returns the arguments $b \in \Delta^{+}_{\text{OUT}}$ such that for every attack $(a, b) \in$

Algorithm 1. Incr-Grounded-Sem($\mathcal{A}_0, u, E_0$)

```
Input: AF $\mathcal{A}_0 = \langle A_0, \Sigma_0 \rangle$, $u = \pm(a, b)$, grounded extension $E_0$;
Output: Revised grounded extension $E$
1: Let $S = \mathcal{I}(u, \mathcal{A}_0, E_0)$;
2: Let $\mathcal{A}_d = \langle A_d, \Sigma_d \rangle = \mathcal{R}_{\mathbf{gr}}(u, \mathcal{A}_0, E_0)$;
3: if ($A_d = \emptyset$) then
4:     $E = E_0$;
5: else
6:     $E = (E_0 \setminus S) \cup IFP(\mathcal{A}_d, E_0 \cap A_d)$;
```

Function 1. *IFP*$(\mathcal{A}, E_0)$

```
Input: AF $\mathcal{A} = \langle A, \Sigma \rangle$, Extension $E_0$;
Output: Extension $E$
 1: $S_{\text{IN}} = \Delta_{\text{IN}} = \{ a \mid \nexists (c, a) \in \Sigma \}$;
 2: if ($S_{\text{IN}} = \emptyset$) then
 3:     return $S_{\text{IN}}$
 4: $S_{\text{OUT}} = \Delta_{\text{OUT}} = \Delta_{\text{IN}}^{+}$;
 5: repeat
 6:     $\Delta_{\text{IN}} = G(S_{\text{OUT}}, \Delta_{\text{OUT}}) \setminus S_{\text{IN}}$;
 7:     $\Delta_{\text{OUT}} = \Delta_{\text{IN}}^{+} \setminus S_{\text{OUT}}$;
 8:     $S_{\text{IN}} = S_{\text{IN}} \cup \Delta_{\text{IN}}$;
 9:     $S_{\text{OUT}} = S_{\text{OUT}} \cup \Delta_{\text{OUT}}$;
10: until $\Delta_{\text{IN}} \subseteq E_0$
11: if ($\Delta_{\text{IN}} = \emptyset$) then
12:     return $S_{\text{IN}}$;
13: else
14:     return $S_{\text{IN}} \cup (E_0 \setminus (S_{\text{IN}} \cup S_{\text{OUT}}))$;
```

Σ, argument $a \in S_{\text{OUT}}$ (i.e. a is labelled OUT).[1] Function G returns the set Δ_{IN} of arguments which are labeled IN at Line 6. Arguments labeled OUT are immediately derived by taking Δ_{IN}^{+}, that is the arguments which are attacked by some argument which has been labelled as IN (Line 7). Function G is iteratively applied until, in the last step of the **repeat** loop, all arguments derived are confirmed to be in the extension E_0 of the AF $\mathcal{A}_0$ being updated (i.e., $\Delta_{\text{IN}} \subseteq E_0$). Finally, if Δ_{IN} is empty, then it just returns the set of arguments labeled as IN (in this case, the status of all the arguments in the restricted AF has been recomputed by function G), otherwise it returns S_{IN} union the arguments in E_0 whose status has not been recomputed by function G.

Example 8. Consider the AF $\mathcal{A}_0$ of Fig. 1 where $E_0 = \{a, h, g, e, l, m, o\}$ and $\mathcal{I}(u, \mathcal{A}_0, E_0) = \{h, c\}$. Algorithm 1 computes the grounded extension E of the AF $\mathcal{A} = +(g, h)(\mathcal{A}_0)$ as follows. The restricted AF $\mathcal{A}_d = \langle A_d, \Sigma_d \rangle = \mathcal{R}_{\mathbf{gr}}(u, \mathcal{A}_0, E_0)$ is computed (at Line 2) obtaining $A_d = \{g, h, c\}$ and $\Sigma_d = \{(g, h), (h, c)\}$. As A_d is not empty, Function IFP with actual parameters $\mathcal{A}_d$ and $E_0 \cap A_d = \{g, h\}$ is called at Line 6. Function IFP first computes $S_{\text{IN}} = \Delta_{\text{IN}} = \{g\}$ and $S_{\text{OUT}} = \Delta_{\text{OUT}} = \{h\}$. Next, at the first iteration of the **repeat** loop, it is computed $\Delta_{\text{IN}} = G(\{h\}, \{h\}) = \{c\}$ (Line 6) and $\Delta_{\text{OUT}} = \emptyset$ (Line 7) as there is no argument attacked by c in $\mathcal{A}_d$. Then the function terminates returning the set $\{g, c\}$ and E turns out to be the set $\{a, g, e, l, m, o\} \cup \{g, c\}$. □

Theorem 2. *For any AF $\mathcal{A} = \langle A, \Sigma \rangle$, the complexity of computing IFP$(\mathcal{A}, E_0)$, with $E_0 \subset A$, is $O(|A| \times \bar{d}^2)$, where $\bar{d}$ is the maximum input degree of a node (i.e., the maximum number of attacks towards an argument in A).*

[1] Similarly to the characteristic function F of an AF [20], function G infers new arguments that can be labelled IN. But it is more efficient as it only uses arguments labelled in the last step.

Theorem 3. *For any AF $\mathcal{A}_0 = \langle A_0, \Sigma_0\rangle$ with grounded extension E_0, and $u = \pm(a, b)$, the complexity of Algorithm* Incr-Grounded-Sem$(\mathcal{A}_0, u, E_0)$ *is $O(|\Sigma_0| + |\mathcal{I}(u, \mathcal{A}_0, E_0)| \times \bar{d}^{\,2})$, where $\bar{d}$ is the maximum input degree of a node.*

Theorem 4. *Given an AF $\mathcal{A}_0$, an update $u{=}\pm(a, b)$ for $\mathcal{A}_0$ yielding $\mathcal{A} = u(\mathcal{A}_0)$, and the grounded extension E_0 of $\mathcal{A}_0$, Algorithm 1 computes the grounded extension E of $\mathcal{A}$.*

6 Experimental Results

We implemented a prototype for incremental computation of argumentation semantics using the Java argumentation libraries provided by the *Tweety* project [47].

Datasets. We used two datasets taken from the International Competition on Computational Models of Argumentation (ICCMA)[2]:

(i) `REAL` consists of 19 AFs $\langle A_0, \Sigma_0\rangle$ with $|A_0| \in [5K, 100K]$ and $|\Sigma_0| \in [7K, 143K]$;
(ii) `SYN` consists of 24 AFs $\langle A_0, \Sigma_0\rangle$ with $|A_0| \in [1K, 4K]$ and $|\Sigma_0| \in [14K, 172K]$.

The AFs in the two datasets have a different structure: on average, $|Reach_{\mathcal{A}_0}(a)|$ is around 2200 for arguments a in `SYN`, while it is about 10 for `REAL`; moreover, the average number of attacks per argument for `REAL` is 1.5 while it is 26 for `SYN`.

Algorithms. For each AF $\mathcal{A}_0 = \langle A_0, \Sigma_0\rangle$ in each dataset, we first computed the grounded extension E_0. Then, we randomly selected an update u of the form $+(a, b)$ (with $a, b \in A_0$ and $(a, b) \notin \Sigma_0$) or $-(a, b)$ (with $(a, b) \in \Sigma_0$). Next, we executed the following algorithms:

(i) *BaseG* which computes the grounded semantics E of the updated AF $u(\mathcal{A}_0)$ from scratch. It finds the fixpoint of the characteristic function of an AF as implemented in the libraries of *Tweety* [47]. This algorithm was also used to compute the initial extension E_0 which is taken as input by the incremental algorithms;
(ii) *Incr-Grounded-Sem* (*IncrG* for short) which incrementally computes the grounded extension E by implementing Algorithm 1 (note that it also includes the computation of the influenced set and the restricted AF).

All experiments have been carried out on an Intel Core i7-4790 CPU 3.60 GHz with 16 GB RAM running Ubuntu 14.04 64bit. All data points reported on the figures are averages over 20 trials (except those for *BaseG* which are averages of 5 trials, as it can take hours in some cases due to the huge size of the datasets considered).

[2] http://argumentationcompetition.org.

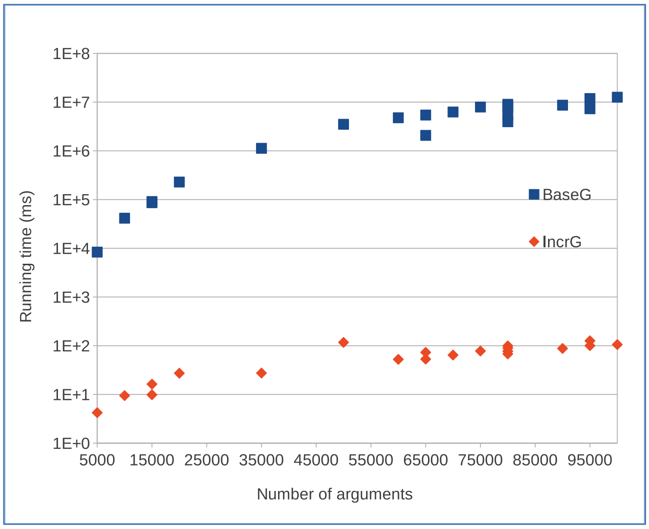

Fig. 4. Run times (ms) of *BaseG* and *IncrG* over `REAL`.

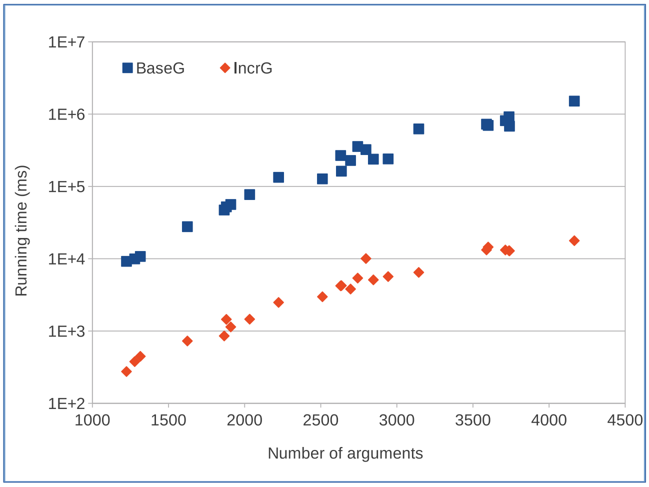

Fig. 5. Run times (ms) of *BaseG* and *IncrG* over `SYN`.

Results. Figures 4 and 5 report the run times (log scale) of *BaseG* and *IncrG* for computing the grounded extensions of the updated AFs versus the number of arguments over `REAL` and `SYN`, respectively.[3] The experiments also showed that, on average, the size of the influenced set w.r.t. that of the input AF for `REAL` (resp. `SYN`) is about 0.01% (resp. 1%).

[3] Data points with the same x-axis value are due to AFs in the datasets having the same number of arguments.

From these results, we can draw the following conclusions:

- The overall time needed by our algorithm for incrementally computing the grounded extension is orders of magnitude better than the time needed to recompute the whole extension from scratch.
- The definition of influenced set substantially restricts the portion of the AF to be analysed for recomputing the semantics of an AF after performing an update. It is worth noting that this means that even using *any non-incremental* algorithm taking as input the restricted AF would result in a performance improvement, since the size of the input data to be processed would be significantly smaller.

7 Conclutions and Future Work

We presented an incremental approach for computing the grounded extension of updated AFs. Our algorithm exploits the initial extension of an AF for computing the set of arguments influenced by an update, and for detecting early termination conditions during the recomputation of the status of the arguments. The experiments showed that the incremental computation outperforms the base (non-incremental) computation. In fact, the time needed by our algorithm for incrementally computing the grounded extension is orders of magnitude better than the time needed to recompute the whole extension from scratch.

Although in this paper we focused on updates consisting of adding/removing only one attack, our technique can be extended to deal with the case of multiple updates. Indeed, in [38] a construction is provided for reducing the application of a set of updates $\{+(a_1, b_1), \ldots, +(a_n, b_n),\ -(a'_1, b'_1),\ \ldots, -(a'_m, b'_m)\}$ on AF $\mathcal{A}_0$ to the application of a single attack update $+(v, w)$ on an AF obtained from $\mathcal{A}_0$ by adding some new arguments/attacks and replacing some existing ones. Thus, this construction can be used to simulate the simultaneous application of a set of updates by a single attack update of the type considered in this paper.

Moreover, our approach can be extended to work in the case of the incremental computation of the ideal extension of an AF. In fact, the definition of influenced set can be used as it is to compute the part of the AF consisting of the arguments whose status can change after performing an update, and an appropriate definition of restricted AF $\mathcal{R}_{id}(u, \mathcal{A}, E)$ for the ideal semantics can be provided [38]. Once the restricted AF is identified, even a non-incremental algorithm taking as input the restricted AF could be used to recompute the status of influenced arguments.

We plan to continue our work along two directions. First, we will investigate the application of the techniques developed in this paper to other (multiple status) semantics. Indeed, the influenced set is defined already for non-deterministic semantics, and the identification of restricted AFs for these semantics would enable the use of existing (non-incremental) algorithms taking as input a smaller AF for computing extensions. We envisage the definition of incremental algorithms that make use of initial extensions for computing extensions after updates for multiple status semantics where we need to deal with the additional issue that extensions can be split/merged after an update.

Our second direction of research for future work is related to the recent investigations of the integration of argumentation and database repairing techniques [46]. In fact, database reparation is often modelled as interactive reasoning process [41], and the user can profitably exploit argumentation techniques to identify and resolve the conflicts between tuples, possibly specifying preferences among repairs suggested by the system [33,34]. Given the interactive nature of this process, we believe it would benefit from the use of incremental algorithms for the computation of the arguments justifying repairs, and thus plan to explore this issue in the future.

References

1. Amgoud, L., Vesic, S.: Revising option status in argument-based decision systems. J. Log. Comput. **22**(5), 1019–1058 (2012)
2. Baroni, P., Boella, G., Cerutti, F., Giacomin, M., van der Torre, L.W.N., Villata, S.: On the input/output behavior of argumentation frameworks. Artif. Intell. **217**, 144–197 (2014)
3. Baroni, P., Caminada, M., Giacomin, M.: An introduction to argumentation semantics. Knowl. Eng. Rev. **26**(4), 365–410 (2011)
4. Baroni, P., Giacomin, M., Liao, B.: On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: a division-based method. Artif. Intell. **212**, 104–115 (2014)
5. Baumann, R.: Splitting an argumentation framework. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS (LNAI), vol. 6645, pp. 40–53. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20895-9_6
6. Baumann, R.: Normal and strong expansion equivalence for argumentation frameworks. Artif. Intell. **193**, 18–44 (2012)
7. Baumann, R.: Context-free and context-sensitive kernels: update and deletion equivalence in abstract argumentation. In: Proceedings of European Conference on Artificial Intelligence (ECAI), pp. 63–68 (2014)
8. Baumann, R., Brewka, G.: Expanding argumentation frameworks: enforcing and monotonicity results. In: Proceedings of International Conference Computational Models of Argument (COMMA), pp. 75–86 (2010)
9. Baumann, R., Brewka, G., Dvořák, W., Woltran, S.: Parameterized splitting: a simple modification-based approach. In: Erdem, E., Lee, J., Lierler, Y., Pearce, D. (eds.) Correct Reasoning. LNCS, vol. 7265, pp. 57–71. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30743-0_5
10. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artif. Intell. **171**(1015), 619–641 (2007). argumentation in Artificial Intelligence
11. Bisquert, P., Cayrol, C., de Saint-Cyr, F.D., Lagasquie-Schiex, M.: Characterizing change in abstract argumentation systems. Trends Belief Revision Argumentation Dyn. **48**, 75–102 (2013)
12. Boella, G., Kaci, S., Torre, L.: Dynamics in argumentation with single extensions: abstraction principles and the grounded extension. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS (LNAI), vol. 5590, pp. 107–118. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02906-6_11

13. Boella, G., Kaci, S., Torre, L.: Dynamics in argumentation with single extensions: attack refinement and the grounded extension (extended version). In: McBurney, P., Rahwan, I., Parsons, S., Maudet, N. (eds.) ArgMAS 2009. LNCS (LNAI), vol. 6057, pp. 150–159. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12805-9_9
14. Calautti, M., Greco, S., Trubitsyna, I.: Detecting decidable classes of finitely ground logic programs with function symbols. In: Proceedings of International Symposium on Principles and Practice of Declarative Programming (PPDP), pp. 239–250 (2013)
15. Caminada, M.: Semi-stable semantics. In: Proceedings of International Conference on Computational Models of Argument (COMMA), pp. 121–130 (2006)
16. Caminada, M., Sá, S., Alcântara, J., Dvorák, W.: On the equivalence between logic programming semantics and argumentation semantics. Int. J. Approx. Reasoning **58**, 87–111 (2015)
17. Cayrol, C., de Saint-Cyr, F.D., Lagasquie-Schiex, M.: Revision of an argumentation system. In: Proceedings of International Conference on Principles of Knowledge Represent and Reasoning (KR), pp. 124–134 (2008)
18. Cayrol, C., de Saint-Cyr, F.D., Lagasquie-Schiex, M.: Change in abstract argumentation frameworks: adding an argument. J. Artif. Intell. Res. (JAIR) **38**, 49–84 (2010)
19. Charwat, G., Dvorák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Methods for solving reasoning problems in abstract argumentation - a survey. Artif. Intell. **220**, 28–63 (2015)
20. Dung, P.M.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–358 (1995)
21. Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artif. Intell. **171**(10–15), 642–674 (2007)
22. Dunne, P.E.: The computational complexity of ideal semantics. Artif. Intell. **173**(18), 1559–1591 (2009)
23. Dunne, P.E., Wooldridge, M.: Complexity of abstract argumentation. In: Simari, G., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 85–104. Springer, USA (2009)
24. Dvorák, W., Pichler, R., Woltran, S.: Towards fixed-parameter tractable algorithms for argumentation. In: Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR) (2010)
25. Dvorák, W., Woltran, S.: Complexity of semi-stable and stage semantics in argumentation frameworks. Inform. Process. Lett. **110**(11), 425–430 (2010)
26. Eiter, T., Strass, H., Truszczyński, M., Woltran, S. (eds.): Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation. LNCS (LNAI), vol. 9060. Springer, Cham (2015)
27. Falappa, M.A., Garcia, A.J., Kern-Isberner, G., Simari, G.R.: On the evolving relation between belief revision and argumentation. Knowl. Eng. Rev. **26**(1), 35–43 (2011)
28. Fazzinga, B., Flesca, S., Parisi, F.: Efficiently estimating the probability of extensions in abstract argumentation. In: Proceedings of International Conference on Scalable Uncertainty Management (SUM), pp. 106–119 (2013)
29. Fazzinga, B., Flesca, S., Parisi, F.: On the complexity of probabilistic abstract argumentation. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pp. 898–904 (2013)
30. Fazzinga, B., Flesca, S., Parisi, F.: On the complexity of probabilistic abstract argumentation frameworks. ACM Trans. Comput. Log. **16**(3), 22 (2015)

31. Fazzinga, B., Flesca, S., Parisi, F.: On efficiently estimating the probability of extensions in abstract argumentation frameworks. Int. J. Approx. Reasoning **69**, 106–132 (2016)
32. Fazzinga, B., Flesca, S., Parisi, F., Pietramala, A.: PARTY: a mobile system for efficiently assessing the probability of extensions in a debate. In: Chen, Q., Hameurlain, A., Toumani, F., Wagner, R., Decker, H. (eds.) DEXA 2015. LNCS, vol. 9261, pp. 220–235. Springer, Cham (2015). doi:10.1007/978-3-319-22849-5_16
33. Flesca, S., Furfaro, F., Parisi, F.: Preferred database repairs under aggregate constraints. In: Proceedings of First International Conference Scalable Uncertainty Management (SUM), pp. 215–229 (2007)
34. Flesca, S., Furfaro, F., Parisi, F.: Range-consistent answers of aggregate queries under aggregate constraints. In: Proceedings of International Conference Scalable Uncertainty Management (SUM), pp. 163–176 (2010)
35. Greco, S., Molinaro, C., Trubitsyna, I.: Logic programming with function symbols: checking termination of bottom-up evaluation through program adornments. TPLP **13**(4–5), 737–752 (2013)
36. Greco, S., Molinaro, C., Trubitsyna, I., Zumpano, E.: NP datalog: a logic language for expressing search and optimization problems. TPLP **10**(2), 125–166 (2010)
37. Greco, S., Parisi, F.: Efficient computation of deterministic extensions for dynamic abstract argumentation frameworks. In: Proceedings of European Conference on Artificial Intelligence (ECAI), pp. 1668–1669 (2016)
38. Greco, S., Parisi, F.: Incremental computation of deterministic extensions for dynamic argumentation frameworks. In: Michael, L., Kakas, A. (eds.) JELIA 2016. LNCS (LNAI), vol. 10021, pp. 288–304. Springer, Cham (2016). doi:10.1007/978-3-319-48758-8_19
39. Liao, B.S., Jin, L., Koons, R.C.: Dynamics of argumentation systems: a division-based method. Artif. Intell. **175**(11), 1790–1814 (2011)
40. Lifschitz, V., Turner, H.: Splitting a logic program. In: Proceedings of International Conference on Logic Programming (ICLP), pp. 23–37 (1994)
41. Martinez, M.V., Parisi, F., Pugliese, A., Simari, G.I., Subrahmanian, V.S.: Policy-based inconsistency management in relational databases. Int. J. Approx. Reasoning **55**(2), 501–528 (2014)
42. Modgil, S., Prakken, H.: Revisiting preferences and argumentation. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pp. 1021–1026 (2011)
43. Oikarinen, E., Woltran, S.: Characterizing strong equivalence for argumentation frameworks. Artif. Intell. **175**(14–15), 1985–2009 (2011)
44. Pollock, J.L.: Perceiving and reasoning about a changing world. Comput. Intell. **14**(4), 498–562 (1998)
45. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence, 1st edn. Springer Publishing Company, Incorporated (2009)
46. Santos, E., Martins, J.P., Galhardas, H.: An argumentation-based approach to database repair. In: Proceedings of European Conference on Artificial Intelligence (ECAI), pp. 125–130 (2010)
47. Thimm, M.: Tweety: A comprehensive collection of java libraries for logical aspects of artificial intelligence and knowledge representation. In: Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR) (2014)
48. Xu, Y., Cayrol, C.: The matrix approach for abstract argumentation frameworks. In: Black, E., Modgil, S., Oren, N. (eds.) TAFA 2015. LNCS (LNAI), vol. 9524, pp. 243–259. Springer, Cham (2015). doi:10.1007/978-3-319-28460-6_15