On the Incremental Computation of Semantics in Dynamic Argumentation

Gianvincenzo Alfano, Sergio Greco, Francesco Parisi
Department of Informatics, Modeling, Electronics and System Engineering, University of Calabria, Italy
{g.alfano,greco,fparisi}@dimes.unical.it

Gerardo I. Simari, Guillermo R. Simari
Department of Computer Science and Engineering, Universidad Nacional del Sur (UNS) & Institute for Computer Science and Engineering (ICIC UNS–CONICET), Bahia Blanca, Argentina
gis@cs.uns.edu.ar, grsimari@gmail.com

Abstract

Argumentation frameworks often model dynamic situations where arguments and their relationships (*e.g.,* attacks) frequently change over time. As a consequence, the sets of conclusions (*e.g.,* extensions of abstract argumentation frameworks, or warranted literals for structured argumentation frameworks) often need to be computed again after performing an update. However, as most of the argumentation semantics proposed so far suffer from high computational complexity, computing the set of conclusions from scratch is costly in general. In this work, we address the problems of efficiently recomputing extensions of dynamic abstract argumentation frameworks and warranted literals in dynamic defeasible knowledge bases. In particular, we first present an incremental algorithmic solution whose main idea is that of using an initial extension and the update to identify a (potentially small) portion of an abstract argumentation framework, which is sufficient to compute an extension of the updated framework.

1 Introduction

Computational Argumentation is an established research field in the area of Knowledge Representation and Reasoning (KR) [29; 91; 21; 84; 61], which is central in Artificial Intelligence (AI). An (abstract) argumentation framework [55] is a simple,

yet powerful formalism for modeling disputes between agents. The formal meaning of an argumentation framework is given in terms of argumentation semantics, which intuitively tell us the sets of arguments (referred to as *extensions*) that can be accepted together to support a point of view in a discussion. For an abstract argumentation framework, an argument is an abstract entity whose role is entirely determined by its relationships with other arguments. In contrast, DeLP [69] is a well-known argumentation formalism where arguments have an explicit structure as they derive from a knowledge base (DeLP program) consisting of facts and strict and defeasible rules. By considering the structure of arguments, *i.e.,* their inner construction, it becomes possible to analyze reasons for and against a conclusion closely, and the *warrant status* of a claim in the context of a knowledge base represents the main output of a dialectical process.

Although the ideas underlying abstract and structured argumentation frameworks are intuitive and straightforward, most of the argumentation semantics proposed so far suffer from high computational complexity [58; 57; 60; 77; 50]. Most research in the domain of formal argumentation (both in the abstract and structured settings) have focused on *static* frameworks (*i.e.,* frameworks whose structure does not change over time), whereas argumentation frameworks are frequently used for modeling dynamic systems [25; 62; 81; 24; 51; 36; 37]; since, as a matter of fact, the argumentation process is inherently dynamic, this is not surprising. For instance, consider how many times we change our minds after learning something new about a situation that is the focus of our reasoning. There is evidence of that in social network threads [76], where users frequently post new arguments against or supporting other posts, often made by the same users that change their minds. 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 could significantly improve performance. In many cases, especially when few updates at a time are performed, the changes made to a framework can result in small changes to the set of its conclusions—extensions of abstract argumentation frameworks; warranted literals for structured argumentation—and recomputing the whole semantics from scratch can be avoided.

The following is a summary of the contributions of this work:

- By focusing on the most popular argumentation semantics for abstract frameworks, *i.e., complete*, *preferred*, *stable*, *ideal*, and *grounded*, we present a general approach for incrementally solving the following computational task: given an argumentation framework AF, an extension for AF under semantics σ, and an update u, obtain an extension of the updated argumentation framework $u(AF)$

under σ. In other words, we explore the possibility of incrementally solving the task σ-SE of the International Competition on Computational Models of Argumentation (ICCMA) [93]: given an argumentation framework, obtain some σ-extension. The technique consists of the following main steps: (*i*) identification of the *influenced set*, which intuitively consists of the set of arguments whose acceptance status may change after performing an update; (*ii*) identification of a (possibly) smaller argumentation framework, called *reduced* argumentation framework, based on the influenced set and additional information provided by the initial extension; (*iii*) using *any* non-incremental algorithm to compute an extension of the reduced argumentation framework; and (*iv*) obtaining the final extension by merging a portion of the initial extension with the one computed for the reduced argumentation framework.

- We show that the main idea behind the above-described incremental approach can be adapted to *extended* abstract argumentation frameworks, *i.e.,* bipolar argumentation frameworks allowing the presence of attacks and supports, as well as argumentation frameworks with second-order interactions (*e.g.,* attacks towards attacks). This is achieved by leveraging meta-argumentation approaches, which provide ways to transform a more general abstract framework into a Dung framework.

- Intending to minimize wasted effort in the computation of the warrant status of literals of a DeLP program after performing an update, we summarize the necessary elements to develop the updating techniques in DeLP's structured argumentation. Particularly, we focus only on literals that are potentially affected by a given update (namely, *influenced* and *core* literals), and avoids the computation of the status of *inferable* and *preserved* literals.

Organization. As a prelude, we first briefly recall basic notions of abstract argumentation frameworks [55] and then introduce updates in Section 2. The incremental technique for recomputing an extension of an updated abstract argumentation framework under different semantics is presented in Section 3. The main idea behind the above-described incremental approach is then adapted to cope with extended argumentation frameworks in Section 4. Next, in Section 5, we discuss the critical aspects of the technique dealing with structured argumentation in an easy-to-read manner. Related work is discussed in Section 6, and conclusions and directions for future work are drawn in Section 7.

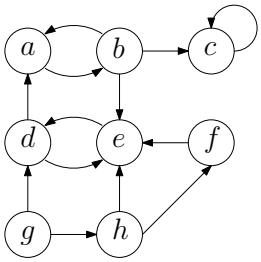

Figure 1: AF_0 of Example 2.1.

2 Abstract Argumentation Frameworks and Updates

We assume the existence of a set $\mathcal{A}rgs$ of *arguments*. An *(abstract) argumentation framework* [55] is a pair $\langle Ar, att\rangle$, where $Ar \subseteq \mathcal{A}rgs$ is a finite set of *arguments*, and $att \subseteq Ar \times Ar$ is a binary relation over Ar whose elements are called *attacks*. Thus, an argumentation framework can be viewed as a directed graph where nodes correspond to arguments and edges correspond to attacks.

Example 2.1 (Running example for abstract argumentation)**.** *Let* $AF_0 = \langle Ar_0, att_0\rangle$ *be an argumentation framework, where* $Ar_0 = \{a, b, c, d, e, f, g, h\}$ *and* $att_0 = \{(a,b), (b,a), (b,c), (c,c), (d,a), (d,e), (e,d), (b,e), (f,e), (g,d), (g,h), (h,e), (h,f)\}$*. The argumentation framework* AF_0 *is shown in Figure 1.*

Given an argumentation framework $\langle Ar, att\rangle$ and arguments $a, b \in Ar$, we say that a *attacks* b iff $(a,b) \in att$, and that a set $S \subseteq Ar$ attacks b iff there is $a \in S$ attacking b. We use $S^+ = \{b \mid \exists a \in S : (a,b) \in att\}$ to denote the set of all arguments that are attacked by S.

Moreover, we say that $S \subseteq Ar$ *defends* a iff $\forall b \in Ar$ such that b *attacks* a, there is $c \in S$ such that c *attacks* b. A set $S \subseteq Ar$ of arguments is said to be:

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

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

- *preferred* ($\mathcal{PR}$) iff it is maximal (w.r.t. $\subseteq$);
- *stable* ($\mathcal{ST}$) iff it attacks every argument in $A \setminus S$;

σ	$\mathcal{E}_\sigma(AF_0)$	$\mathcal{E}_\sigma(AF)$
$\mathcal{CO}$	$\{\{f,g\},\{a,f,g\},\{b,f,g\}\}$	$\{\{g\},\{a,g\},\{b,f,g\}\}$
$\mathcal{PR}$	$\{\{a,f,g\},\{b,f,g\}\}$	$\{\{a,g\},\{b,f,g\}\}$
$\mathcal{ST}$	$\{\{b,f,g\}\}$	$\{\{b,f,g\}\ \}$
$\mathcal{ID}$	$\{\{f,g\}\}$	$\{\{g\}\}$
$\mathcal{GR}$	$\{\{f,g\}\}$	$\{\{g\}\}$

Table 1: Sets of extensions for AF_0 and $AF = +(c,f)(AF_0)$.

- *grounded ($\mathcal{GR}$)* iff it is minimal (w.r.t. $\subseteq$).
- *ideal ($\mathcal{ID}$)* iff it is contained in every preferred extension and it is maximal (w.r.t. $\subseteq$).

Given an argumentation framework AF and a semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}, \mathcal{ID}\}$, we use $\mathcal{E}_\sigma(AF)$ to denote the set of σ-extensions for AF, *i.e.,* the set of extensions for AF according to the given semantics σ.

All the above-mentioned semantics except the stable admit at least one extension, and the grounded and ideal admits exactly one extension [55; 56; 41]. Grounded and ideal semantics are called *deterministic* or *unique status* as $|\mathcal{E}_{\mathcal{GR}}(AF)| = |\mathcal{E}_{\mathcal{ID}}(AF)| = 1$, whereas the other above recalled semantics are called *nondeterministic* or *multiple status*. For any AF AF, it holds that $\mathcal{E}_{\mathcal{ST}}(AF) \subseteq \mathcal{E}_{\mathcal{PR}}(AF) \subseteq \mathcal{E}_{\mathcal{CO}}(AF)$, $\mathcal{E}_{\mathcal{GR}}(AF) \subseteq \mathcal{E}_{\mathcal{CO}}(AF)$, and $\mathcal{E}_{\mathcal{ID}}(AF) \subseteq \mathcal{E}_{\mathcal{CO}}(AF)$.

Example 2.2. *The set of admissible sets for the argumentation framework AF_0 shown in Figure 1 is $\{\emptyset, \{b\}, \{g\}, \{a,g\}, \{b,g\}, \{f,g\}, \{a,g,f\}, \{b,g,f\}\}$, and the set $\mathcal{E}_\sigma(AF_0)$ of extensions, with $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{ID}, \mathcal{GR}\}$ is as reported in the second column of Table 1.*

2.1 Labelling and Status of Arguments

The argumentation semantics can be also defined in terms of *labelling* [21]. A labelling for an argumentation framework $\langle Ar, att\rangle$ is a total function $\mathcal{L}ab : Ar \to \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ assigning to each argument a label. $\mathcal{L}(a) = \texttt{in}$ means that argument a is accepted, $\mathcal{L}(a) = \texttt{out}$ means that a is rejected, while $\mathcal{L}(a) = \texttt{undec}$ means that a is undecided.

Let $in(\mathcal{L}) = \{a \mid a \in Ar \wedge \mathcal{L}(a) = \texttt{in}\}$, $out(\mathcal{L}) = \{a \mid a \in Ar \wedge \mathcal{L}(a) = \texttt{out}\}$, and $un(\mathcal{L}) = \{a \mid a \in Ar \wedge \mathcal{L}(a) = \texttt{undec}\}$. In the following, we also use the triple $\langle in(\mathcal{L}), out(\mathcal{L}), un(\mathcal{L})\rangle$ to represent the labelling $\mathcal{L}$.

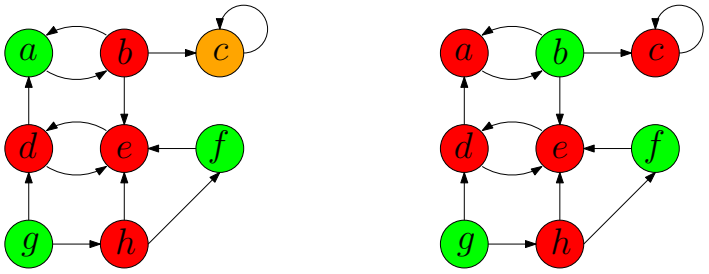

Figure 2: Labelling $\mathcal{L}$ corresponding to the preferred extensions $E_{\mathcal{PR}} \in \mathcal{E}_{AF_0}(\mathcal{PR}) = \langle\{a, f, g\}, \{b, d, e, h\}, \{c\}\rangle$ (left-hand side) and $E'_{\mathcal{PR}} \in \mathcal{E}_{AF_0}(\mathcal{PR}) = \langle\{b, f, g\}, \{a, d, e, h\}, \{c\}\rangle$ (right-hand side). A green (resp., red, orange) node x is such that $\mathcal{L}(x) =$ `in` (resp., `out`, `undec`).

Given an argumentation framework $AF = \langle Ar, att\rangle$, a labelling $\mathcal{L}$ for AF is said to be *admissible (or legal)* if $\forall a \in in(\mathcal{L}) \cup out(\mathcal{L})$ it holds that

(i) $\mathcal{L}(a) =$ `out` iff $\exists\, b \in Ar$ such that $(b, a) \in att$ and $\mathcal{L}(b) =$ `in`; and

(ii) $\mathcal{L}(a) =$ `in` iff $\mathcal{L}(b) =$ `out` for all $b \in Ar$ such that $(b, a) \in att$.

Moreover, $\mathcal{L}$ is a *complete* labelling iff conditions (i) and (ii) hold for all $a \in Ar$.

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

Example 2.3. *Continuing from Example 2.2,* $\langle\{a, f, g\}, \{b, d, e, h\}, \{c\}\rangle$ *is the labelling corresponding to the preferred extension* $E_{\mathcal{PR}} \in \mathcal{E}_{\mathcal{PR}}(AF_0) = \{a, f, g\}$*, as shown in Figure 2.*

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

2.2 Updating a Dung Argumentation Framework

An argumentation framework typically models a temporary situation, and new arguments and attacks can be added or retracted to take into account new available knowledge.

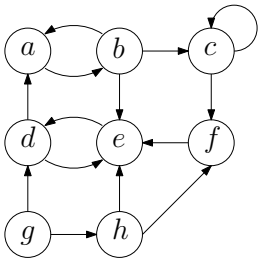

Figure 3: $AF = +(c, f)(AF_0)$

Performing an update on an argumentation framework AF_0 means modifying it into an argumentation framework AF by adding or removing arguments or attacks. We use $+(a, b)$, with $a, b \in Ar_0$ and $(a, b) \notin att_0$, (resp. $-(a, b)$, with $(a, b) \in att_0$) to denote the addition (resp. deletion) of an attack (a, b), and $u(AF_0)$ to denote the application of update $u = \pm(a, b)$ to AF_0 (where $\pm$ means either $+$ or $-$). Applying an update u to an argumentation framework implies that its semantics (set of extensions or labellings). Table 1 reports the sets of extensions for the argumentation framework AF_0 of Figure 1 and for $AF = +(c, f)(AF_0)$ of Figure 3 which is obtained from AF_0 by performing the update $+(c, f)$.

Concerning the addition (resp. deletion) of a set of isolated arguments, it is easy to see that if AF is obtained from AF_0 through the addition (resp. deletion) of a set S of isolated argument, then, let E_0 be an extension for AF_0, $E = E_0 \cup S$ (resp. $E = E_0 \setminus S$) is an extension for AF that can be trivially computed. Of course, if arguments in S are not isolated, for addition we can first add isolated arguments and then add attacks involving these arguments, while for deletion we can first delete all attacks involving arguments in S. Thus we do not consider these kinds of update in the following.

3 Incremental Computation of Extensions in Dynamic Argumentation Frameworks

We tackle the problem of incrementally computing extensions of dynamic argumentation frameworks: given an initial extension and an update (or a set of updates), we devise a technique for computing an extension of the updated argumentation framework under five well-known semantics (*i.e., complete*, *preferred*, *stable*, *grounded*, and *ideal*).

The idea, initially proposed in [74; 75] and then developed in [4], is that of identifying a reduced (updated) argumentation framework sufficient to compute an

update $+(a,b)$		$\mathcal{L}_0(b)$		
		in	undec	out
$\mathcal{L}_0(a)$	in			$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{ST}$, $\mathcal{GR}$
	undec		$\mathcal{CO}$, $\mathcal{GR}$	$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{GR}$
	out	$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{ST}$	$\mathcal{CO}$, $\mathcal{GR}$	$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{ST}$, $\mathcal{GR}$

Table 2: Cases for which $E_0 \in \mathcal{E}_\sigma(u(AF_0))$ for $u = +(a,b)$.

extension of the whole argumentation framework and use state-of-the-art algorithms to recompute an extension of the reduced argumentation framework only.

For the sake of presentation, we first present the technique for semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$, and then show how to deal with the ideal semantics in Section 3.3, since the definition of the reduced argumentation framework for the ideal semantics is different from that for the other semantics.

We first give some sufficient conditions ensuring that a given σ-extension for an argumentation framework AF continues to be a σ-extension for the updated argumentation framework $u(AF)$. Then, we introduce the *influenced set* that intuitively consists of the set of arguments whose status may change after performing an update.

Updates Preserving a Given Initial Extension

Given an update $\pm(a,b)$ and an initial extension E_0 corresponding to $\mathcal{L}_0$, for each pair of initial statuses $\mathcal{L}_0(a)$ and $\mathcal{L}_0(b)$ of the arguments involved in the update, Tables 2 and 3 tell us the semantics for which E_0 is still an extension after the update, as stated in the following proposition.

Proposition 3.1 (Irrelevant Updates [5])**.** *Let AF_0 be an argumentation framework, σ a semantics, $E_0 \in \mathcal{E}_\sigma(AF_0)$ an extension of AF_0 under semantics σ, $\mathcal{L}_0$ the labelling corresponding to E_0, and u an update. If σ is in the cell $\langle \mathcal{L}_0(a), \mathcal{L}_0(b)\rangle$ of Table 2 and $u = +(a,b)$ (resp., of Table 3 and $u = -(a,b)$), then $E_0 \in \mathcal{E}_\sigma(u(AF_0))$.*

The results in Tables 2 and 3 concerning the grounded semantics follow from those in [39; 40], where the principles according to which the grounded extension does not change when attacks are added or removed have been studied.

In the following, given an argumentation framework AF_0 and a σ-extension E_0 for it, we say that an update u is *irrelevant* w.r.t. E_0 and σ iff the conditions of Proposition 3.1 hold. Otherwise, u is said to be *relevant*.

update $-(a,b)$		$\mathcal{L}_0(b)$		
		in	undec	out
$\mathcal{L}_0(a)$	in	NA	NA	
	undec	NA		$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{GR}$
	out	$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{ST}$, $\mathcal{GR}$	$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{GR}$	$\mathcal{CO}$, $\mathcal{PR}$, $\mathcal{ST}$, $\mathcal{GR}$

Table 3: Cases for which $E_0 \in \mathcal{E}_\sigma(u(AF_0))$ for $u = -(a,b)$.

Example 3.2. *Consider AF_0 of Figure 1 and its sets of extensions listed in the second column of Table 1. $E_0 = \{b, f, g\}$ is an extension according to semantics $\sigma \in \{\mathcal{CO},\mathcal{PR},\mathcal{ST}\}$. Thus, $\mathcal{L}_0(c) =$ `out` and $\mathcal{L}_0(f) =$ `in`, and using Proposition 3.1 it follows that for update $u = +(c,f)$ E_0 is still an extension of $u(AF_0)$ (see the last column of Table 1). Thus $+(c,f)$ is irrelevant w.r.t. E_0 and σ.*

In contrast, $+(c,f)$ is relevant w.r.t. $E_0 = \{a, f, g\}$ and any semantics (in this case $\mathcal{L}_0(c) =$ `undec` and $\mathcal{L}_0(f) =$ `in`, and no semantics is listed in the cell $\langle$`undec`, `in`$\rangle$ of Table 2).

It is important to note that Tables 2 and 3 are not meant to be exhaustive, as more conditions can be found for which a σ-extension is preserved after an update. For instance, for the grounded semantics, the initial extension is preserved also if $\mathcal{L}_0(a) =$ `out` and $\mathcal{L}_0(b) =$ `in` and argument a of updated $+(a,b)$ is not reachable from b. Here we provided a simple set of conditions that can be easily checked by just looking at the initial labelling $\mathcal{L}_0$. The technique for the incremental computation can be trivially extended by considering a more general set of such conditions.

Influenced Set

Given an argumentation framework, an update, and an initial σ-extension of the considered framework, the influenced set consists of the arguments whose acceptance status (according to the semantics σ) may change after performing the update. For irrelevant updates, the influenced set will be empty, as in this case, the initial extension can be immediately returned as an extension of the updated argumentation framework. If none of the conditions of Proposition 3.1 hold (*i.e.,* the update is relevant), then the influenced set may turn out to be not empty. In such case, the influenced set will be used to delineate a portion of the argumentation framework, called *reduced* argumentation framework, that we will use to recompute (a portion of) an extension for the updated argumentation framework.

Given an argumentation framework $AF = \langle Ar, att\rangle$ and an argument $b \in Ar$, we use $Reach_{AF}(b)$ to denote the set of arguments that are reachable from b in the

graph AF.

Definition 3.3 (Influenced Set [5])**.** Let $AF = \langle Ar, att \rangle$ be an argumentation framework, $u = \pm(a, b)$, E an extension of AF under semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{ID}, \mathcal{GR}\}$, and let

- $INF_0(u, AF, E) = \begin{cases} \emptyset \;\; \textit{if } u \textit{ is irrelevant w.r.t. } E \textit{ and } \sigma \textit{ or } \exists(z, b) \in att \\ \qquad \textit{s.t. } z \neq a \wedge z \in E \;\wedge\; z \notin Reach_{AF}(b); \\ \{b\} \;\; otherwise; \end{cases}$
- $INF_{i+1}(u, AF, E) = INF_i(u, AF, E) \;\cup\; \{y \mid \exists(x, y) \in att \textit{ s.t. } x \in INF_i(u, AF, E) \wedge \nexists(z, y) \in att \textit{ s.t. } z \in E \wedge z \notin Reach_{AF}(b)\}$.

The *influenced set* of u w.r.t. AF and E is $INF(u, AF, E) = INF_n(u, AF, E)$ such that $INF_n(u, AF, E) = INF_{n+1}(u, AF, E)$.

Thus, the set of arguments that are influenced by an update of b's status 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 $z \in E$ that is not reachable from (and thus not influenced by) b.

Example 3.4. *Consider the argumentation framework $AF_0 = \langle Ar_0, att_0 \rangle$ of Figure 1 and the update $u = +(c, f)$. We have that $Reach_{AF_0}(f) = Ar_0 \setminus \{g, h\}$. The influenced set depends on the initial extension chosen. For the (preferred) extension $\{b, f, g\}$ of Example 3.2, we have that the influenced set is empty as u is irrelevant. In contrast, for the (preferred) extension $E_0 = \{a, f, g\}$, the influenced set is $INF(u, AF_0, E_0) = \{f, e\}$. Indeed, $d \notin INF(u, AF_0, E_0)$ since it is attacked by $g \in E_0$ which is not reachable from f. Thus the arguments that can be reached from d do not belong to $INF(u, AF_0, E_0)$. If we consider the initial grounded extension $\{f, g\}$, then $\{f, e\}$ turns out once again to be the influenced set.*

Reduced Argumentation Framework

Given the influenced set, we define a subgraph, called *reduced* argumentation framework, that will be used to compute the status of the influenced arguments, thus providing an extension that will be combined with that of initial argumentation framework to obtain an extension of the updated argumentation framework, for every semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$.

For any argumentation framework $AF = \langle Ar, att \rangle$ and set $S \subseteq Ar$ of arguments, we denote with $AF{\downarrow_S} = \langle S, att \cap (S \times S) \rangle$ the subgraph of AF induced by arguments in S. Moreover, given two argumentation frameworks $AF_1 = \langle Ar_1, att_1 \rangle$ and $AF_2 =$

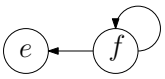

Figure 4: $RAF(+(c,f), AF_0, \{a,f,g\})$

$\langle Ar_2, att_2\rangle$, we denote as $AF_1 \sqcup AF_2 = \langle Ar_1 \cup Ar_2, att_1 \cup att_2\rangle$ the *union* of the two argumentation frameworks.

Definition 3.5 (Reduced Argumentation Framework [5])**.** *Let $AF_0 = \langle Ar_0, att_0\rangle$ be an argumentation framework, $E_0 \in \mathcal{E}_\sigma(AF_0)$ an extension for AF_0 under a semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$, and $u = \pm(a,b)$ an update. Let $AF = \langle Ar, att\rangle$ be the argumentation framework updated using u. The reduced argumentation framework for AF_0 w.r.t. E_0 and u (denoted as $RAF(u, AF_0, E_0)$) is as follows.*

- *$RAF(u, AF_0, E_0)$ is empty if $INF(u, AF_0, E_0)$ is empty.*
- *$RAF(u, AF_0, E_0) = AF{\downarrow_{INF(u,AF_0,E_0)}} \sqcup AF_1 \sqcup AF_2$ where:*
  - *(i) AF_1 is the union of the frameworks $\langle\{a,b\}, \{(a,b)\}\rangle$ s.t. $(a,b) \in att$, $a \notin INF(u, AF_0, E_0)$, $a \in E_0$, and $b \in INF(u, AF_0, E_0)$;*
  - *(ii) AF_2 is the union of the frameworks $\langle\{c\}, \{(c,c)\}\rangle$ s.t. there is $(e,c) \in att$, $e \notin INF(u, AF_0, E_0)$, $e \notin (E_0 \cup E_0^+)$, and $c \in INF(u, AF_0, E_0)$.*

Hence, the argumentation framework $RAF(u, AF_0, E_0)$ contains, in addition to the subgraph of $u(AF_0)$ induced by $INF(u, AF_0, E_0)$, 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 $INF(u, AF_0,\ E_0)$. Specifically, if there is in $u(AF_0)$ an edge from an uninfluenced node a whose status in `in` to an influenced node b, then we add the edge (a,b) so that, as a does not have incoming edges in $RAF(u, AF_0, E_0)$, its status is confirmed to be `in`. Moreover, if there is in $u(AF_0)$ an edge from an uninfluenced node e to an influenced node c such that e is `undec`, we add edge (c,c) to $RAF(u, AF_0, E_0)$ so that the status of c cannot be `in`. Using fake arguments/attacks to represent external contexts has been exploited in [20] where decomposability properties of argumentation semantics are investigated.

Example 3.6. *For our running example, if $E_0 = \{a,f,g\}$ and $u = +(c,f)$, the reduced argumentation framework $RAF(+(c,f), AF_0, E_0)$ consists of the subgraph induced by $INF(u, AF_0, E_0) = \{f,e\}$ plus the edge (f,f) as there is the attack (c,f) in the updated argumentation framework from a non influenced argument c labelled as* `undec` *toward the influenced argument f. Hence, $RAF(+(c,f), AF_0, E_0)$ $= \langle\{e,f\}, \{(f,f),(f,e)\}\rangle$ as shown in Figure 4.*

The following theorem states that, for every semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$, an extension for the updated argumentation framework can be obtained by the union of an extension of the reduced argumentation framework and the projection of the initial extension on the uninfluenced part.

Theorem 3.7 ([5]). *Let AF_0 be an argumentation framework, $AF = u(AF_0)$ be the argumentation framework resulting from performing update $u = \pm(a, b)$ on AF_0, and $E_0 \in \mathcal{E}_\sigma(AF_0)$ be an extension for AF_0 under a semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$. If $\mathcal{E}_\sigma(RAF(u, AF_0, E_0))$ is not empty, then there is an extension $E \in \mathcal{E}_\sigma(AF)$ for the updated argumentation framework AF such that $E = (E_0 \setminus INF(u, AF_0, E_0)) \cup E_d$, where E_d is a σ-extension for reduced argumentation framework $RAF(u, AF_0, E_0)$.*

Example 3.8. *Continuing with our example, for the preferred semantics, let $E_0 = \{a, f, g\}$ and $u = +(c, f)$, we have that $INF(u, AF_0, E_0) = \{f, e\}$, and $RAF(+(c, f), AF_0, E_0) = \langle\{e, f\}, \{(f, f), (f, e)\}\rangle$. Thus, using the theorem, there is an extension E of the updated argumentation framework such that $E = (\{a, f, g\} \setminus \{f, e\}) \cup E_d$ where $E_d = \emptyset$ is a preferred extension of the reduced argumentation framework. In fact, $E = \{a, g\} \in \mathcal{E}_{\mathcal{PR}}(u(AF_0))$.*

It is worth noting that the set of extensions of an argumentation framework can be empty only for the stable semantics. Thus, in the case that this happens for the reduced argumentation framework (*i.e.*, $\mathcal{E}_\sigma(RAF(u, AF_0, E_0)) = \emptyset$), the theorem does not give a method to determine an extension of the updated argumentation framework, as shown in the following example.

Example 3.9. *Consider the two stable extensions $\{a, c\}$ and $\{a, d, e\}$ for AF_0 and the update $u = +(d, d)$. Depending on the initial extension, the influenced set is either $INF(u, AF, \{a, c\}) = \emptyset$ (as u is irrelevant w.r.t. $\{a, c\}$ and $\mathcal{ST}$) or $INF(u, AF, \{a, d, e\})$ $= \{d\}$. Thus, starting from the extension $\{a, c\}$ we directly know $\{a, c\}$ is a stable extension of the updated argumentation framework. However, starting from $\{a, d, e\}$, the reduced argumentation framework will be $RAF(u, AF_0, \{a, d, e\}) = \langle\{d\}, \{(d, d)\}\rangle$, which has no stable extension. In this case, the theorem does not provide a stable extension of the updated argumentation framework, thought a stable extension exists: that obtained by starting from the initial extension $\{a, c\}$.*

Note that, if we consider the preferred semantics, for which the starting extensions are again $\{a, c\}$ and $\{a, d, e\}$, a preferred extension of the updated argumentation framework can be obtained no matter what starting extension is chosen. In particular, as the preferred extension for reduced argumentation framework $\langle\{d\}, \{(d, d)\}\rangle$ is the empty set, it follows that $(\{a, d, e\} \setminus \{d\}) \cup \emptyset = \{a, e\}$ is a preferred extension of the updated argumentation framework.

Algorithm 1 Incr-Alg(AF_0, u, σ, E_0, Solver_σ) [5]

Input: $AF_0 = \langle Ar_0, att_0 \rangle$,
update $u = \pm(a, b)$,
semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$,
extension $E_0 \in \mathcal{E}_\sigma(AF_0)$,
function $\mathsf{Solver}_\sigma(AF)$ returning a σ-extension of AF if it exists, $\perp$ otherwise;
Output: A σ-extension $E \in \mathcal{E}_\sigma(u(AF_0))$ if it exists, $\perp$ otherwise;

1: $S = INF(u, AF_0, E_0)$;
2: **if** $(S = \emptyset)$ **then**
3: **return** E_0;
4: **end if**
5: $AF_d = RAF(u, AF_0, E_0)$;
6: Let $E_d = \mathsf{Solver}_\sigma(AF_d)$;
7: **if** $(E_d \neq \perp)$ **then**
8: **return** $E = (E_0 \setminus S) \cup E_d$;
9: **else**
10: **return** $\mathsf{Solver}_\sigma(u(AF_0))$;
11: **end if**

3.1 Incremental Algorithm

Algorithm 1 computes an extension of an updated argumentation framework [5]. Besides taking as input an initial argumentation framework AF_0, an update u, a semantics $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$, and an extension $E_0 \in \mathcal{E}_\sigma(AF_0)$, it also takes as input a function that computes a σ-extension for an argumentation framework, if any. In particular, function $\mathsf{Solver}_\sigma(AF)$ will be used to compute an extension of the reduced argumentation framework, which will be then combined with the portion of the initial extension that does not change in order to obtain an extension for the updated argumentation framework (as stated in Theorem 3.7).

More in detail, Algorithm 1 works as follows. First, the influenced set of AF_0 w.r.t. update u and the given initial extension E_0 is computed (Line 1). If it is empty, then E_0 will be still an extension of the updated argumentation framework under the given semantics σ, and thus it is returned (Line 3). Otherwise, the reduced argumentation framework AF_d is computed at Line 5, and function Solver_σ is invoked to compute a σ-extension of AF_d, if any. If $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{GR}\}$, then AF_d will have an extension E_d, which is combined with $E_0 \setminus S$ at Line 8 to get an extension for the updated argumentation framework. For the stable semantics, if $\mathcal{E}_{\mathcal{ST}}(RAF(u, AF_0, E_0))$ is not empty, then the algorithm proceeds as for the other

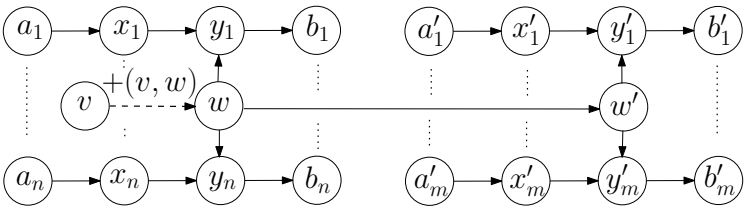

Figure 5: Simulating multiple updates by a single one.

semantics (Line 8). Otherwise, function Solver_σ is invoked to compute a stable extension of the whole updated argumentation framework $u(AF_0)$, if any.

The soundness and completeness of the algorithm follows from the result of Theorem 3.7 and the soundness and completeness of function $\mathsf{Solver}_\mathcal{S}$ used.

Theorem 3.10 (Soundness and Completeness [5])**.** *Let AF_0 be an argumentation framework, $u = \pm(a, b)$, and $E_0 \in \mathcal{E}_\sigma(AF_0)$ an extension for AF_0 under $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$. If Solver_σ is sound and complete then Algorithm 1 computes $E \in \mathcal{E}_\sigma(u(AF_0))$ if $\mathcal{E}_\sigma(u(AF_0))$ is not empty, otherwise it returns $\perp$.*

3.2 Applying Multiple Updates Simultaneously

The approach described in the previous section extends to the case of multiple updates, *i.e.,* set of updates performed simultaneously. In fact, performing a set of updates $U = \{+(a_1, b_1), \ldots, +(a_n, b_n), -(a'_1, b'_1), \ldots, -(a'_m, b'_m)\}$ on AF_0 can be reduced to performing a single update $+(v, w)$ on an argumentation framework $AF^U_{E_0}$ whose definition depends on both the set of updates U and the initial σ-extension E_0, as explained in what follows.

Given a set U of updates for an argumentation framework AF_0, and a σ-extension E_0 for AF_0, we use U^* to denote the subset of U consisting of the relevant updates (that is, the updates in U for which the conditions of Proposition 3.1 do not hold).

The argumentation framework $AF^U_{E_0}$ for applying a set U^* of relevant updates is obtained from AF_0 by (*i*) adding arguments x_i, y_i and the chain of attacks between a_i and b_i as shown in Figure 5, for each update $+(a_i, b_i) \in U^*$; (*ii*) replacing each attack (a'_j, b'_j) in AF_0 with the chain of attacks between a'_j and b'_j as shown in Figure 5, for each update $-(a_j, b_j) \in U^*$; and (*iii*) adding the new arguments v, w, w' and the attacks involving them as shown in Figure 5. The following definition considers a general set of updates which includes also irrelevant updates.

Definition 3.11 (AF for applying a set of updates [75])**.**
Let $AF_0 = \langle Ar_0, att_0 \rangle$ be an argumentation framework, and E_0 a σ-extension for AF_0. Let

- $att^+ = \{(a_1, b_1), \ldots, (a_n, b_n)\} \subseteq (Ar_0 \times Ar_0) \setminus att_0$, *and*
- $att^- = \{(a'_1, b'_1), \ldots, (a'_m, b'_m)\} \subseteq att_0$

such that $att^+ \cap att^- = \emptyset$ *be two sets of attacks.*

Let $U = \{+(a_i, b_i) \mid (a_i, b_i) \in att^+\} \cup \{-(a_j, b_j) \mid (a_j, b_j) \in att^-\}$ *be a set of updates, and* $U^* \subseteq U$ *be the set of relevant updates w.r.t.* E_0 *and* σ*. Then,* $AF^U_{E_0} = \langle Ar^U, att^U \rangle$ *denotes the argumentation framework obtained from* AF_0 *as follows:*

- $Ar^U = Ar_0 \cup \{x_i, y_i \mid +(a_i, b_i) \in U^*\} \cup \{x'_j, y'_j \mid -(a_j, b_j) \in U^*\} \cup \{v, w, w'\}$, *where all* x_i, y_i, x'_j, y'_j, w, w', *and* v *are new arguments not occurring in* Ar_0, *and*
- $att^U = (att_0 \setminus att^-) \cup \{(a_i, b_i) \mid +(a_i, b_i) \in (U \setminus U^*)\} \cup$
  $\{(a_i, x_i), (x_i, y_i), (y_i, b_i) \mid +(a_i, b_i) \in U^*\} \cup$
  $\{(a_j, x'_j), (x'_j, y'_j), (y'_j, b_j) \mid -(a_j, b_j) \in U^*\} \cup$
  $\{(w, y_i) \mid +(a_i, b_i) \in U^*\} \cup$
  $\{(w', y'_j) \mid -(a_j, b_j) \in U^*\} \cup \{(w, w')\}$.

It is worth noting that, in the definition above, each argument x_i, y_i, x'_i, and y'_i is assumed to be unique and non-identical for every attack (a_i, b_i).

The following theorem states that every extension of the argumentation framework AF obtained by performing on AF_0 all the updates in U corresponds to an extension of $+(v, w)(AF^U_{E_0})$, where $+(v, w)$ is a single attack update.

Theorem 3.12 ([75; 5])**.** *Let* $AF_0 = \langle Ar_0, att_0 \rangle$ *be an argumentation framework,* E_0 *a* σ*-extension for* AF_0*, and* U *a set of updates. Let* AF *be the argumentation framework obtained from* AF_0 *by performing all updates in* U *on it. Then, for any semantics* $\sigma \in \{\mathcal{CO}, \mathcal{PR}, \mathcal{ST}, \mathcal{GR}\}$ $E \in \mathcal{E}_\sigma(AF)$ *iff there is* $E^U \in \mathcal{E}_\sigma(+(v, w)(AF^U_{E_0}))$ *such that* $E^U \cap Ar_0 = E$.

3.3 Dealing with the Ideal Semantics

Algorithm 1 can be extended to deal with the ideal semantics. The only difference is that we need a new definition of reduced argumentation framework since, as illustrated in the following example, that of Definition 3.5 does not work for the ideal semantics.

Example 3.13. *Consider the argumentation framework* $AF_0 = \langle \{a, b, c, d\}, \{(a, b), (b, a), (c, d), (d, c), (a, c), (b, c)\} \rangle$ *and the update* $u = -(b, c)$*. The ideal extension of* AF_0 *is* $E_0 = \{d\}$ *(i.e., arguments* a *and* b *are both labeled as* `undec`*). The influenced set is* $INF(u, AF_0, E) = \{c, d\}$*. However, the RAF obtained by applying*

Definition 3.5 is $\langle\{c,d\}, \{(c,c), (c,d), (d,c)\}\rangle$*, its ideal extension is* $\{d\}$*, and applying the result of Theorem 3.7 we would obtain that* $\{d\}$ *is still the ideal extension for* $u(AF_0)$*. But this is not correct, as the ideal extension for* $u(AF_0)$ *is the empty set.*

Before defining the reduced framework for the ideal semantics, we define the paths providing the information on the "context" outside the influenced set $INF(u, AF, E)$ that needs to be added to determine the new status of the arguments influenced by update u w.r.t. the ideal extension E.

Given an argumentation framework $AF = \langle Ar, att\rangle$ with ideal extension E and a set $S \subseteq Ar$, we use $Node(AF, S, E)$ (resp. $Edge(AF, S, E)$) to denote a set of arguments $x_1, \ldots, x_n$ (resp. attacks $(x_1, x_2), \ldots, (x_{n-1}, x_n)$) in AF such that there is a path $x_1 \ldots x_n$ in AF with $x_n \in S$, $x_1, \ldots, x_{n\text{-}1} \notin S$ and $x_1, \ldots, x_{n\text{-}1} \notin E \cup E^+$ (*i.e.,* $x_1, \ldots, x_{n\text{-}1}$ are `undec`). Essentially, if S is the influenced set of an update, to determine the status of nodes in S we must also consider all nodes and attacks occurring in paths (of any length) ending in S whose nodes outside S are all labeled as `undec`. The motivation to also consider the paths ending in S is that some of the undecided arguments occurring in these paths could be labelled `in` or `out` in some preferred labelling and, therefore, together they could determine a change in the status of nodes in S.

Definition 3.14. (Reduced Argumentation Framework for Ideal Semantics [75; 8])
Let $AF_0 = \langle Ar_0, att_0\rangle$ *be an argumentation framework,* E_0 *be the ideal extension for* AF_0*, and* $u = \pm(a, b)$ *an update. Let* $AF = \langle Ar, att\rangle$ *be the argumentation framework updated by using* u*. The reduced argumentation framework for* AF_0 *w.r.t.* E_0 *and* u *(denoted as* $RAF_{\mathcal{ID}}(u, AF_0, E_0)$*) is as follows.*

- $RAF_{\mathcal{ID}}(u, AF_0, E_0)$ *is empty if* $INF(u, AF_0, E_0)$ *is empty.*
- $RAF_{\mathcal{ID}}(u, AF_0, E_0) = AF{\downarrow}_{INF(u,AF_0,E)} \sqcup AF_1 \sqcup AF_2$ *where:*
  - *(i)* AF_1 *is the union of the frameworks* $\langle\{a, b\}, \{(a, b)\}\rangle$ *such that* $(a, b) \in att$ *and* $a \notin INF(u, AF_0, E_0)$*,* $a \in E_0$*, and* $b \in INF(u, AF_0, E_0)$*;*
  - *(ii)* AF_2 *is the union of the frameworks* $\langle Node(AF, INF(u, AF_0, E_0), E_0)$ *and* $Edge(AF, INF(u, AF_0, E_0), E_0)\rangle$*.*

Example 3.15. *For the argumentation framework* AF_0 *of running example (see Figures 1 and 3), where the initial ideal extension is* $E_0 = \{f, g\}$ *and* $u = +(c, f)$*, the reduced argumentation framework* $RAF_{\mathcal{ID}}(+(c, f), AF_0, E_0)$ *consists of the subgraph induced by* $INF(u, AF_0, E_0) = \{f, e\}$ *plus the sub-graph consisting of the paths of undecided arguments ending in the influenced set, that is,* $AF_2 = \langle\{a, b, c\}, \{(a, b), (b, a),$

$(b, c), (c, c),$
$(c, f)\}\rangle$. *Hence,* $RAF_{\mathcal{ID}}(+(c, f), AF_0, E_0) = \langle\{a, b, c, e, f\}, \{(a, b), (b, a), (b, c), (c, c),$
$(c, f), (f, e)\}\rangle$. *The ideal extension of the reduced framework is the empty set.*

It can be shown that the result of Theorem 3.7 also holds for the case of the ideal semantics [8]. By applying that result, we obtain that the (updated) ideal extension for the updated argumentation framework of Example 3.15 is $(\{f, g\} \setminus \{f, e\}) \cup \emptyset = \{g\}$ (see Table 1).

Example 3.16. *Consider again the argumentation framework* AF_0 *and the update* u *of Example 3.13, where the ideal extension of* AF_0 *is* $E_0 = \{d\}$ *and* $INF(u, AF_0, E) = \{c, d\}$.

Thus, $RAF_{\mathcal{ID}}(u, AF_0, E_0) = AF{\downarrow}_{INF(u,AF_0,E_0)} \sqcup AF_1 \sqcup AF_2$ *where:*

- $AF{\downarrow}_{INF(u,AF_0,E)} = \langle\{c, d\}, \{(c, d), (d, c)\}\rangle$,
- $AF_1 = \langle\emptyset, \emptyset\rangle$ *and*
- $AF_2 = \langle\{a, b, c\}, \{(a, b), (b, a), (a, c)\}\rangle$.

That is, $RAF_{\mathcal{ID}}(u, AF_0, E_0) = \langle\{a, b, c, d\}, \{(a, b), (b, a), (c, d), (d, c), (a, c)\}\rangle$, *and its ideal extension is* $\emptyset$. *Thus, using the result of Theorem 3.7, we obtain that the ideal extension for the updated argumentation framework* $u(AF_0)$ *is the empty set.*

Finally, Algorithm 1 can be used to compute the updated ideal extension of a given argumentation framework by using $AF_d = RAF_{\mathcal{ID}}(u, AF_0, E_0)$ at Line 5 and an external solver that computes the ideal extension of the reduced argumentation framework.

In the next two sections, we will deal with other possible ways to apply the incremental algorithm in other approaches to formal (computational) argumentation. First, Section 4 deals with bipolarity and extended argumentation frameworks, while Section 5 centers on Defeasible Logic Programming (DeLP) as a structured argumentation formalism.

4 Bipolarity and Second-Order Attacks

Dung's framework has been extended along several dimensions; for instance, see [22; 83; 96]. The proposed incremental approach can be applied to different kinds of abstract argumentation frameworks that extend Dung's model. The main idea is that of using meta-argumentation approaches, which provide ways to transform a more general abstract framework into a Dung framework, and apply the incremental technique on the meta argumentation framework [4; 6; 7].

Bipolarity in argumentation is discussed in [17], where a survey of the use of bipolarity is given, as well as a formal definition of bipolar argumentation frameworks, which extend Dung's concept of argumentation framework by also including the relation of support between arguments. The notion of support has been found to be useful in many application domains, including decision making [16]. Several interpretations of the notion of support have been proposed in the literature [17; 46; 47; 48; 38; 96] (see [53] for a comprehensive survey). In this work, we focus on *deductive* support [38; 96] which is intended to capture the following intuition: if argument a supports argument b then the acceptance of a implies the acceptance of b, and thus the non-acceptance of b implies the non-acceptance of a. However, the approach presented in this section can be adapted to work also with *necessary* support [89; 88; 23] due to the duality between these two kinds of interpretations of the support relation [53]. The acceptability of arguments in the presence of a support relation was first investigated in [46]. Later on, to handle bipolarity in argumentation, [47; 48] proposed an approach based on using the concept of *coalition* of arguments, where sets of arguments are considered as a group that plays the role of an argument and defeats then occur between coalitions. However, when considering a deductive interpretation of support [38; 96], coalitions may lead to counter-intuitive results [53]; nevertheless, they are useful in contexts where support is interpreted differently.

Furthermore, other abstract argumentation frameworks have been considered, such as Extended Argumentation Frameworks, which extend bipolar argumentation frameworks by modelling (apart from attacks/supports between arguments) also attacks towards an attack or a support (called second-order attacks). Thanks to a meta argumentation approach, an extended argumentation framework can be converted into an abstract argumentation framework by using additional meta-arguments as well as attacks between them to model supports and second-order attacks.

In the following, we discuss how to extend the incremental technique to deal with extended argumentation frameworks. An *Extended Argumentation Framework* [38] is a quadruple $\langle Ar, att, sup, s\text{-}att\rangle$, where where (i) $Ar \subseteq \mathcal{A}rgs$, (ii) $att \subseteq Ar \times Ar$, (iii) $sup \subseteq Ar \times Ar$ is a binary relation over Ar whose elements are called *supports*, (iv) $att \cap sup = \emptyset$,and (v) $s\text{-}att$ is a binary relation over $Ar \times (att \cup sup)$ whose elements are called *second-order attacks*.

In the following, a second-order attack from an argument a to an attack (b, c) will be denoted as $(a \twoheadrightarrow (b \rightarrow c))$, while an attack from an argument a to a support (b, c) will be denoted as $(a \twoheadrightarrow (b \Rightarrow c))$. Thus, a Dung argumentation framework is an extended argumentation framework of the form $\langle Ar, att, \emptyset, \emptyset\rangle$, while a bipolar argumentation framework is extended argumentation framework of the form $\langle Ar, att, sup, \emptyset\rangle$.

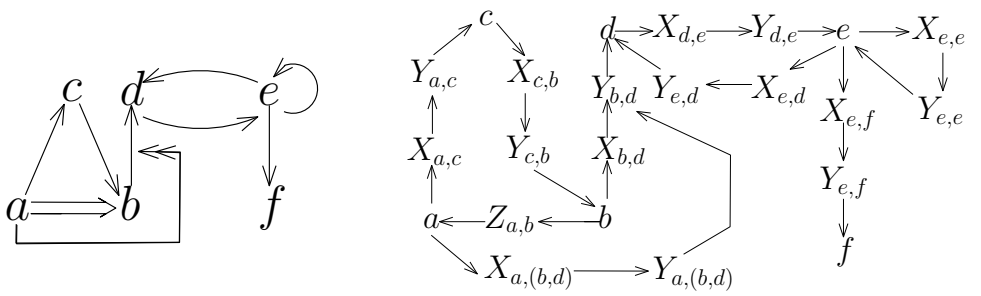

Figure 7: Meta framework for EF_0 of Example 4.1.

Example 4.1. *Consider the extended argumentation framework* $EF_0 = \langle Ar_0, att_0, sup_0, s\text{-}att_0 \rangle$ *where:*

- $Ar_0 = \{a, b, c, d, e, f\}$ *is the set of arguments;*
- $att_0 = \{(a, c),\ (c, b),\ (b, d),\ (d, e), (e, d),\ (e, e),\ (e, f)\}$ *is the set of attacks;*
- $sup_0 = \{(a, b)\}$ *is the set of supports; and*
- $s\text{-}att_0 = \{(a, (b, d))\}$ *is the set of second-order attacks.*

The corresponding graph is shown in Figure 6, where second-order attacks are drawn using double-headed arrows.

The semantics of an extended argumentation framework can be given by means of the following meta argumentation framework.

Definition 4.2 (Meta Argumentation Framework [38])**.** *The meta argumentation framework for* $EF = \langle Ar, att, sup, s\text{-}att \rangle$ *is* $MF = \langle Ar^m, att^m \rangle$ *where:*

- $Ar^m = Ar \cup \{X_{a,b}, Y_{a,b} \mid (a, b) \in att\} \cup \{Z_{a,b} \mid (a, b) \in sup\} \cup \{X_{a,(b,c)}, Y_{a,(b,c)} \mid (a, (b, c)) \in s\text{-}att, (b, c) \in att\}$
- $att^m = \{(a, X_{a,b}), (X_{a,b}, Y_{a,b}), (Y_{a,b}, b) | (a, b) \in att\} \cup \{(b, Z_{a,b}), (Z_{a,b}, a) \mid (a, b) \in sup\} \cup \{(a, X_{a,(b,c)}), (X_{a,(b,c)}, Y_{a,(b,c)}), (Y_{a,(b,c)}, Y_{b,c}) \mid (a, (b, c)) \in s\text{-}att, (b, c) \in att\} \cup \{(a, X_{a,(b,c)}), (X_{a,(b,c)}, Y_{a,(b,c)}), (Y_{a,(b,c)}, Z_{b,c}) \mid (a, (b, c)) \in s\text{-}att, (b, c) \in sup\}$

The meaning of meta-arguments $X_{a,b}$, $Y_{a,b}$ and $Z_{a,b}$ is as follows. $X_{a,b}$ represents the fact that the corresponding attack (a, b) is "negligible" in the extended argumentation framework—it belongs to an extension of the meta argumentation framework iff a does not belong to an extension of the extended argumentation framework. On the other hand, $Y_{a,b}$ represents the fact that (a, b) is "significant" in the extended argumentation framework, and it belongs to an extension of the meta argumentation framework iff argument b does not. Finally, meta-argument $Z_{a,b}$ represents a support relation between a and b: it does not belong to an extension for the meta argumentation framework iff the supported argument b is accepted in the deductive model of support.

Moreover, a second order attack of the form $(a \twoheadrightarrow (b \rightarrow c))$ is encoded as an attack towards the meta-argument $Y_{b,c}$ (that represents the fact that (b, c) is "significant"), while an attack of the form $(a \twoheadrightarrow (b \Rightarrow c))$ is encoded as an attack toward the meta-argument $Z_{b,c}$. The meta argumentation framework for the extended argumentation framework of Example 4.1 is shown in Figure 7.

Extensions for an extended argumentation framework *EF* are obtained from extensions for its meta argumentation framework *MF*: E is an σ-extension for *EF* iff $E^m \in \mathcal{E}_\sigma(MF)$ and $E = E^m \cap Ar$, where Ar is the set of arguments of *EF*. Using this relationship, the notion of labelling can be extended to extended argumentation frameworks as well. As done in [38], in the following we will focus on the preferred and stable semantics. However, the technique can be also applied to grounded, ideal, and complete semantics by means of meta argumentation approach.

Example 4.3. *For the meta argumentation framework MF of Figure 7, we have the following preferred extensions (which are also stable extensions): (i)* $\{a, b, d, f, Y_{a,c}, X_{c,b}, Y_{d,e}, Y_{a,(b,d)}, X_{e,e}, X_{e,d}, X_{e,f},\}$*, which corresponds to the extension* $\{a, b, d, f\}$ *of the extended argumentation framework of Example 4.1, and (ii)* $\{c, d, f, X_{a,c}, Y_{c,b}, Z_{a,b}, X_{b,d}, Y_{d,e}\ X_{e,e}, X_{e,d}, X_{e,f}, X_{a,(b,d)}\}$*, which corresponds to the extension* $\{c, d, f\}$ *of the extended argumentation framework.*

Updates over Extended Argumentation Frameworks For extended argumentation frameworks we also consider updates consisting of additions and deletions of support relations and second-order attacks, in addition to the attack updates considered for Dung's frameworks. Specifically, the addition (resp., deletion) of a support relation from an argument a to an argument b will be denoted as $+(a \Rightarrow b)$ (resp. $-(a \Rightarrow b)$). Analogously, the addition (resp., deletion) of a second-order attack from an argument a to an attack (b, c) will be denoted as $+(a \twoheadrightarrow (b \rightarrow c))$ (resp., $-(a \twoheadrightarrow (b \rightarrow c))$). Finally, if (b, c) is a support, then the update will be denoted as $+(a \twoheadrightarrow (b \Rightarrow c))$ (resp., $-(a \twoheadrightarrow (b \Rightarrow c))$). We use $u(EF_0)$ to denote the

extended argumentation framework resulting from the application of update u to an initial extended framework EF_0.

We introduce the compact argumentation framework for performing an update on extended argumentation frameworks—it will be used in a variant of Algorithm 1 for the incremental computation. The definition builds on (the compact version of) that proposed in [38] and considers additional meta-arguments and attacks that will allow us to simulate addition updates to be performed on the extended argumentation framework by means of single updates performed on the corresponding (compact) meta argumentation framework.

Definition 4.4 (Compact Argumentation Framework [7])**.** *Let $EF = \langle Ar, att, sup, s\text{-}att\rangle$ be an extended argumentation framework, and u an update of one of the following forms:*

$$\begin{array}{ll} \bullet\ u = \pm(e \to f) & \bullet\ u = \pm(e \Rightarrow f) \\ \bullet\ u = \pm(e \twoheadrightarrow (g \to h)) & \bullet\ u = \pm(e \twoheadrightarrow (g \Rightarrow h)). \end{array}$$

The compact argumentation framework for EF w.r.t. u is $CF(EF, u) = \langle Ar^m, att^m\rangle$ where:

- $Ar^m = A \cup \{Z_{a,b} \mid (a,b) \in sup\} \cup$
  $\{X_{c,d}, Y_{c,d} \mid (e,(c,d)) \in s\text{-}att, (c,d) \in att\} \cup$
  $\{Z_{e,f} \mid u = +(e \Rightarrow f)\} \cup$
  $\{X_{g,h}, Y_{g,h} \mid u = +(e \twoheadrightarrow (g \to h))\}$
- $att^m = att \setminus \{(g,h) \mid u = +(e \twoheadrightarrow (g \to h))\} \cup$
  $\{(g, X_{g,h}), (X_{g,h}, Y_{g,h}), (Y_{g,h}, h) \mid u = +(e \twoheadrightarrow (g \to h))\} \cup$
  $\{(b, Z_{a,b}), (Z_{a,b}, a) \mid (a,b) \in sup\} \cup$
  $\{(e, Z_{a,b}) \mid (e,(a,b)) \in s\text{-}att, (a,b) \in sup\} \cup$
  $\{(c, X_{c,d}), (X_{c,d}, Y_{c,d}), (Y_{c,d}, d), (e, Y_{c,d}) \mid$
  $(e,(c,d)) \in s\text{-}att, (c,d) \in att\} \cup$
  $\{(f, Z_{e,f}) \mid u = +(e \Rightarrow f)\}.$

Besides the meta-arguments $Z_{a,b}$ of Definition 4.2, and the attacks involving those arguments, the above meta argumentation framework contains meta-arguments $X_{c,d}, Y_{c,d}$ for encoding second order attacks in $s\text{-}att$ toward attacks $(c,d) \in att$. In fact, an attack $e \twoheadrightarrow (a \Rightarrow b)$ in $s\text{-}att$ toward a support is encoded as an attack from e toward $Z_{a,b}$ in the meta argumentation framework, while $e \twoheadrightarrow (c \to d)$ in $s\text{-}att$ is encoded as an attack from e toward $Y_{c,d}$ in the meta argumentation framework (which contains also the attacks $(c, X_{c,d}), (X_{c,d}, Y_{c,d}), (Y_{c,d}, d)$). Moreover, meta-arguments $Z_{e,f}$ and $X_{g,h}, Y_{g,h}$, are added to the meta argumentation framework for encoding,

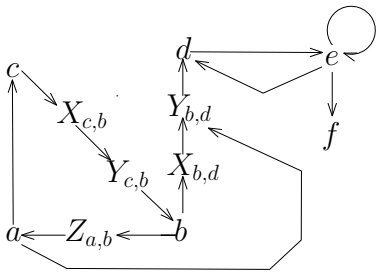

Figure 8: Compact argumentation framework for the extended argumentation framework EF_0 of Figure 6 w.r.t. the update $u = +(d \twoheadrightarrow (c \to b))$.

respectively, the addition of a second order attack toward a support $(e, f) \in sup$ or toward an attack $(g, h) \in att$. In the latter case, meta-arguments $X_{g,h}$ and $Y_{g,h}$ along with the set of attacks $\{(g, X_{g,h}), (X_{g,h}, Y_{g,h}), (Y_{g,h}, h)\}$ are used to simulate the attack $g \to h$ which is attacked by e in the extended argumentation framework. This enables the definition of simple attack updates to simulate second-order attack updates.

Example 4.5. *The compact argumentation framework for the EAF EF_0 of Figure 6 w.r.t. the update $u = +(d \twoheadrightarrow (c \to b))$ is shown in Figure 8. Herein, the attacks involving the meta-arguments $X_{b,d}$ and $Y_{b,d}$ allow us to simulate the second order attack $a \twoheadrightarrow (b \to d)$. Moreover, the attacks involving the meta-arguments $X_{c,b}$ and $Y_{c,b}$ are added to enable the simulation of the second-order update u by a single attack update on the meta argumentation framework.*

We now define updates on the meta argumentation framework.

Definition 4.6 (Updates for the meta argumentation framework [7])**.** *Let $EF = \langle Ar, att, sup, s\text{-}att\rangle$ be an extended argumentation framework, and u an update for EF. The corresponding update u^m for the compact argumentation framework $CF(EF, u)$ is as follows:*

$u^m =$	$u^m =$
$+(Z_{e,f} \to e)$ *if* $u = +(e \Rightarrow f)$	$-(Z_{e,f} \to e)$ *if* $u = -(e \Rightarrow f))$
$+(c \to d)$ *if* $u = +(c \to d)$	$-(c \to d))$ *if* $u = -(c \to d))$
$+(e \to Y_{g,h})$ *if* $u = +(e \twoheadrightarrow (g \to h))$	$-(e \to Y_{g,h})$ *if* $u = -(e \twoheadrightarrow (g \to h))$
$+(e \to Z_{a,b})$ *if* $u = +(e \twoheadrightarrow (a \Rightarrow b))$	$-(e \to Z_{a,b})$ *if* $u = -(e \twoheadrightarrow (a \Rightarrow b))$

For instance, continuing with Example 4.5, given the extended argumentation framework EF_0 of Figure 6 and the update $u = +(d \twoheadrightarrow (c \to b))$, we have that

update u^m for the compact argumentation framework $CF(EF_0, u)$ shown in Figure 8 is $u^m = +(d \rightarrow Y_{c,b})$.

Finally, given an initial extension for an extended argumentation framework and an update, we define the initial labelling for the corresponding compact argumentation framework as follows.

Definition 4.7 (Corresponding initial labelling [7])**.** *Given an extended argumentation framework $EF_0 = \langle Ar, att, sup, s\text{-}att\rangle$ and a initial labelling $\mathcal{L}_0$, the corresponding initial labelling $\mathcal{L}_0^m$ for the compact argumentation framework $CF(EF_0, u) = \langle Ar^m, att^m\rangle$ is as follows:*

$\forall a \in Ar \cap Ar^m$:	$\mathcal{L}_0^m(a) = \mathcal{L}_0(a)$;	
$\forall\ X_{a,b} \in Ar^m$:	$\mathcal{L}_0^m(X_{a,b}) = \texttt{in}$	*if* $\mathcal{L}_0(a) = \texttt{out}$
	$\mathcal{L}_0^m(X_{a,b}) = \texttt{out}$	*if* $\mathcal{L}_0(a) = \texttt{in}$
	$\mathcal{L}_0^m(X_{a,b}) = \texttt{undec}$	*if* $\mathcal{L}_0(a) = \texttt{undec}$
$\forall\ Y_{a,b} \in Ar^m$:	$\mathcal{L}_0^m(Y_{a,b}) = \texttt{in}$	*if* (i) $\mathcal{L}_0^m(X_{a,b}) = \texttt{out}$ *and* (ii) $\forall c \in Ar$ *s.t.* $(c,(a,b)) \in s\text{-}att$, $\mathcal{L}_0(c) = \texttt{out}$
	$\mathcal{L}_0^m(Y_{a,b}) = \texttt{out}$	*if* (i) $\mathcal{L}_0^m(X_{a,b}) = \texttt{in}$ *or* (ii) $\exists\ c \in Ar \mid (c,(a,b)) \in s\text{-}att$ *and* $\mathcal{L}_0(c) = \texttt{in}$
	$\mathcal{L}_0^m(Y_{a,b}) = \texttt{undec}$,	*otherwise.*

$\forall Z_{a,b} \in Ar^m$:	$\mathcal{L}_0^m(Z_{a,b}) = \texttt{in}$	*if* (i) $\mathcal{L}_0(b) = \texttt{out}$ *and* (ii) $\forall\ c \in Ar$ *s.t.* $(c,(a,b)) \in s\text{-}att$, $\mathcal{L}_0(c) = \texttt{out}$
	$\mathcal{L}_0^m(Z_{a,b}) = \texttt{out}$	*if* (i) $\mathcal{L}_0(b) = \texttt{in}$ *or* (ii) $\exists\ c \in Ar \mid (c,(a,b)) \in s\text{-}att$ *and* $\mathcal{L}_0(c) = \texttt{in}$
	$\mathcal{L}_0^m(Z_{a,b}) = \texttt{undec}$,	*otherwise.*

For instance, given the initial preferred extension $E_0 = \{a, b, d, f\}$ of the extended argumentation framework EF_0 of Example 4.1, the initial labelling for the compact argumentation framework $CF(EF_0, +(d \rightarrow Y_{c,b}))$ of Figure 8 is such that $\mathcal{L}_0^m(a) = \mathcal{L}_0(a) = \texttt{in}$, $\mathcal{L}_0^m(c) = \mathcal{L}_0(c) = \texttt{out}$, $\mathcal{L}_0^m(X_{c,b}) = \texttt{in}$, and $\mathcal{L}_0^m(Y_{c,b}) = \texttt{out}$. Also, we have that $\mathcal{L}_0^m(b) = \mathcal{L}_0(b) = \texttt{in}$, $\mathcal{L}_0^m(X_{b,d}) = \texttt{out}$, $\mathcal{L}_0^m(Y_{b,d}) = \texttt{out}$ since $\mathcal{L}_0^m(a) = L_0(a) = \texttt{in}$, and $\mathcal{L}_0^m(d) = \mathcal{L}_0(d) = \texttt{in}$.

Algorithm 2 Incr-EAF(EF_0, u, E_0, σ, Solver_σ)

Input: Extended argumentation framework $EF_0 = \langle Ar_0, att_0, sup_0, satt_0 \rangle$,
update u over EF_0,
an initial σ-extension E_0 for EF_0,
semantics $\sigma \in \{\mathcal{PR}, \mathcal{ST}\}$,
function $\mathsf{Solver}_\sigma(AF)$ that returns an σ-extension of AF if it exists, and $\perp$ otherwise;
Output: An σ-extension E for $u(EF_0)$ if it exists, $\perp$ otherwise;

1: **if** $checkProp(EF_0, u, E_0, \sigma)$ **then**
2: **return** E_0**;**
3: **end if**
4: Let $CF_0 = CF(EF_0, u)$ be the compact argumentation framework for EF_0 w.r.t. u (cf. Definition 4.4);
5: Let u^m be the update for CF_0 corresponding to u (cf. Definition 4.6);
6: Let E_0^m be the initial σ-extension for CF_0 corresponding to E_0;
7: Let $E^m = \mathsf{Incr\text{-}Alg}(CF_0, u^m, \sigma, E_0^m, \mathsf{Solver}_\sigma)$;
8: **if** $(E^m \neq \perp)$ **then**
9: **return** $E = (E^m \cap Ar_0)$;
10: **else**
11: **return** $\perp$;
12: **end if**

Incremental Algorithm for Extended Argumentation Frameworks We are now ready to present the algorithm for extending the incremental technique to the case of extended argumentation frameworks. Given an extended argumentation framework EF_0, a semantics $\sigma \in \{\mathcal{PR}, \mathcal{ST}\}$, an extension $E_0 \in \mathcal{E}_\sigma(EF_0)$, and an update u of the form $u = \pm(a \Rightarrow b)$, $u = \pm(a \rightarrow b)$, $u = \pm(e \twoheadrightarrow (c \Rightarrow d))$, or $u = \pm(e \twoheadrightarrow (c \rightarrow d))$, Algorithm 2 computes an extension E of the updated extended argumentation framework $u(EF_0)$, if it exists [7]. The algorithm works as follows. It first checks if the initial extension E_0 is still an extension of the updated extended argumentation framework at Line 1, where $checkProp(EF_0, u, E_0, \sigma)$ is a function returning *true* iff the update is irrelevant—the interested reader can find the conditions under which an update for an extended argumentation framework is irrelevant in [7]. If this is the case, it immediately returns the initial extension. Otherwise, it computes the compact argumentation framework CF_0 (Line 4), the update u^m for CF_0 (Line 5), and the initial σ-extension E_0^m for CF_0 (Line 6). Next, it invokes function Incr-Alg (*i.e.,* Algorithm 1). Incr-Alg takes as input the parameters CF_0, u^m, σ, E_0^m, and Solver_σ, where Solver_σ is an external solver that can compute

an σ-extension for the input argumentation framework. Finally, the extension of the updated extended argumentation framework (if any) is obtained by projecting out the extension E^m returned by Incr-Alg over the set of arguments Ar_0 of the initial extended argumentation framework (Line 9).

From a computational point of view, in the worst case (that is, when every argument is influenced, and thus the RAF collapses to be the updated framework), Algorithm 1 and Algorithm 2 have the same computational complexity as the corresponding task in the static setting under the considered AF semantics. It is worth noting that the overhead of computing the influenced set and the RAF is polynomial in the input framework's size.

The use of the incremental techniques discussed in this section and the previous one become significant in practice. In fact, in [5] it is shown that Algorithm 1 outperforms state-of-the-art solvers that compute the extensions from scratch for single updates by two orders of magnitude on average, and it remains faster than the competitors even when recomputing an extension after performing updates simultaneously. Moreover, [7] reports on an experimental analysis showing that Algorithm 2 also outperforms by two orders of magnitude the computation from scratch on EAFs, where solvers from scratch taking as input the (compact) Dung argumentation frameworks resulting from the transformation of the candidate EAF (cf. Definition 4.4) are used. Finally, the experimental results concerning the use of both Algorithm 1 and Algorithm 2 also revealed that the improvements of using incremental techniques become larger as the computation from scratch becomes more challenging.

5 Incremental Computation in Defeasible Logic Programming

In [30], four frameworks that consider the structure of arguments were presented. Two of them—ASPIC$^+$ [85] and ABA [94]—build the set of all possible arguments from the knowledge base and then rely on using one of the possible Dung semantics to decide on the acceptance of arguments. The other two—Logic-Based Deductive Argumentation [32] and DeLP [70]—only build the arguments involved in answering the query. These last two frameworks exhibit several differences [30]—among them is the base logic used as a knowledge representation language: [32] relies on propositional logic, requiring a theorem prover to solve queries; on the other hand, DeLP [70] adopts an extension of logic programming, which is a computational framework per se. To better understand the differences among the frameworks mentioned above, we refer the interested reader to [68], where a variant of DeLP using the grounded semantics is also discussed.

A fundamental distinction between DeLP and the other three frameworks, which significantly affects a query's resolution, rests on how attacks between arguments are described. DeLP considers two forms of defeat: *proper* and *blocking*; the former is akin to Dung's attack [55], whereas the latter presents a different behavior since the two arguments that are part of the blocking defeat relation, attacker and attackee, are defeated (hence the use of the term *blocking defeater*). Of course, this could be modeled in Dung's graphs as a mutual attack, but the DeLP mechanism forbids, in a properly formed dialogue, the use of two blocking defeaters successively because the introduction of another blocking defeater is unnecessary since the first two are already defeated. Moreover, to find the answers required by the query, other considerations of dialogical nature are taken into account, strengthening the reasoning process by forbidding common dialogical fallacies; these characteristics have been reflected in the development of a game-based semantics [95].

In this section, we focus on the incremental computation in the context of structured argumentation. Particularly, we discuss an incremental technique [12; 11] for *Defeasible Logic Programming (DeLP)* [69; 70] which shares the same underlying ideas and the goal of avoiding wasted effort as in the (incremental) technique previously discussed for AFs. Given that our primary focus is on the changes in the structure of the arguments used to answer a query, we have considered the DeLP language; however, the ideas here developed can inspire similar techniques for other structured argumentation frameworks such as ABA and ASPIC$^+$. Next, we will summarize the necessary elements to develop the updating techniques in DeLP's structured argumentation; see [11] for an extended presentation.

5.1 Defeasible Logic Programming and Updates

A DeLP program $\mathcal{P} = (\Pi, \Delta)$ consists of sets Π and Δ of *strict* and *defeasible* rules defined using elements of a set *Lit* of literals, that are ground atoms obtained from a set **At** of atoms. $Lit_{\mathcal{P}}$ denotes the set of literals occurring in a rule of $\mathcal{P}$, and the symbol "$\sim$" represents strong negation; for any literal $\alpha \in Lit$ the formula $\sim\sim\alpha$ is considered equivalent to α and can be used for denoting it. Particularly, given the literals $\alpha_0, \alpha_1, \ldots, \alpha_n$, a strict rule $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_n$ (with $n \geq 0$) represents non-defeasible information, while *defeasible* rules $\alpha_0 \multimap \alpha_1, \ldots, \alpha_n$ (with $n > 0$) represent tentative information, *i.e.,* information that can be used if nothing can be posed against it. Given a strict or defeasible rule r, we use $head(r)$ to denote α_0, and $body(r)$ to denote the set of literals $\{\alpha_1, \ldots, \alpha_n\}$. Strict rules with empty body will also be called *facts*.[1]

[1]With a little abuse of notation, in the following we will denote a fact $(\alpha \leftarrow)$ simply by α.

As an example of DeLP program, let us consider $\mathcal{P}_1 = (\Pi_1, \Delta_1)$, where:

$\Pi_1 = \{x, y, z, (w \leftarrow y)\}$ is the set of strict rules (and facts), and

$\Delta_1 = \{(a \prec\!\!\!- w), (a \prec\!\!\!- z), (\sim a \prec\!\!\!- z), (b \prec\!\!\!- a), (b \prec\!\!\!- z), (c \prec\!\!\!- b, x),$

$(\sim c \prec\!\!\!- b), (d \prec\!\!\!- \sim c)\}$ is the set of defeasible rules.

Given a DeLP program $\mathcal{P} = (\Pi, \Delta)$ and a literal $\alpha \in Lit_{\mathcal{P}}$, a *(defeasible) derivation* for α w.r.t. $\mathcal{P}$ is a finite sequence $\alpha_1, \alpha_2, \ldots, \alpha_n = \alpha$ of literals such that (i) each literal α_i is in the sequence because there exists a (strict or defeasible) rule $r \in \mathcal{P}$ with head α_i and body $\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_k}$ such that $i_j < i$ for all $j \in [1, k]$, and (ii) there do not exist two literals α_i and α_j such that $\alpha_j = \sim\alpha_i$. A derivation is said to be a *strict derivation* if only strict rules are used.

A program $\mathcal{P}$ is *contradictory* if and only if there exist defeasible derivations for at least two complementary literals α and $\sim\alpha$ from $\mathcal{P}$. We assume that Π (the strict part of $\mathcal{P}$) is not contradictory. However, complementary literals can be derived from $\mathcal{P}$ when defeasible rules are used in the derivation. Two literals α and β are said to be *contradictory* if (i) neither $\Pi \cup \{\alpha\}$ nor $\Pi \cup \{\beta\}$ strictly derive a pair of complementary literals, whereas (ii) $\Pi \cup \{\alpha, \beta\}$ does. Pairs of complementary literals are clearly contradictory; a set of literals is said to be contradictory if it contains two contradictory literals.

Considering the program $\mathcal{P}_1$, the literal c can be derived using the following sets of rules and facts: $\{(c \prec\!\!\!- x, b), (x), (b \prec\!\!\!- a), (a \prec\!\!\!- w), (w \leftarrow y), (y)\}$; the derivation $(y, w, a, b, x,$
$c)$ describes how rules can be applied to derive c. However, the set of rules $\Pi_1 \cup \Delta_1$ is contradictory since also $\sim c$ can be derived using the rules: $\{(\sim c \prec\!\!\!- b), (b \prec\!\!\!- a), (a \prec\!\!\!- w), (w \leftarrow y), (y)\}$. The non-contradictory set of literals that can be derived from Π_1 is $\{x, y, w, z\}$.

DeLP incorporates a defeasible argumentation formalism for the treatment of contradictory knowledge, allowing the identification of conflicting pieces of knowledge, and a *dialectical process* is used for deciding which information prevails as warranted. This process involves the construction and evaluation of arguments that either support or interfere with a user-issued query. An *argument* $\mathcal{A}$ for a literal α is a couple $\langle \mathcal{A}, \alpha \rangle$ where $\mathcal{A}$ is a set of defeasible rules representing a derivation that is (i) supported by facts, (ii) non-contradictory, and (iii) $\subseteq$-minimal (*i.e.,* there is no proper subset of $\mathcal{A}$ satisfying both (i) and (ii)). As an example, $\langle \mathcal{A}_1, c \rangle = \langle \{(c \prec\!\!\!- x, b), (b \prec\!\!\!- a), (a \prec\!\!\!- w), (w \leftarrow y)\}, c \rangle$ and $\langle \mathcal{A}_2, \sim a \rangle = \langle \{(\sim a \prec\!\!\!- z)\}, \sim a \rangle$ are two arguments that can be obtained from the program $\mathcal{P}_1$. An argument $\langle \mathcal{A}, \alpha \rangle$ is said to be a *sub-argument* of $\langle \mathcal{A}', \alpha' \rangle$ if $\mathcal{A} \subseteq \mathcal{A}'$.

The main task of DeLP is establishing *warranted* literals. A literal α is said to be warranted if there exists an undefeated argument $\langle \mathcal{A}, \alpha \rangle$. To determine if an argument $\langle \mathcal{A}, \alpha \rangle$ is undefeated, *defeaters* for $\langle \mathcal{A}, \alpha \rangle$ are considered, and since reinstatement could happen when all of $\mathcal{A}$'s possible defeaters are defeated, the process continues considering defeaters for $\mathcal{A}$'s defeaters, and so on. To define defeaters, to decide when an attack is successful, we require a comparison criterion $\succ$ over arguments, which is irreflexive and asymmetric. As the comparison criterion is a modular part of the argumentation inference engine, we will abstract away from this criterion and simply assume the existence of a comparison criterion $\succ$ between arguments: $\langle \mathcal{A}, \alpha \rangle \succ \langle \mathcal{B}, \beta \rangle$ meaning that argument $\langle \mathcal{A}, \alpha \rangle$ is preferred to $\langle \mathcal{B}, \beta \rangle$. Intuitively, an argument $\langle \mathcal{A}, \alpha \rangle$ *attacks* an argument $\langle \mathcal{B}, \beta \rangle$ when there is a sub-argument $\langle \mathcal{C}, \gamma \rangle$ of $\langle \mathcal{B}, \beta \rangle$, such that α and γ are contradictory. When the attacker satisfies that $\langle \mathcal{C}, \gamma \rangle$ is not preferred to $\langle \mathcal{A}, \alpha \rangle$ (*i.e.,* $\langle \mathcal{C}, \gamma \rangle \not\succ \langle \mathcal{A}, \alpha \rangle$), the attacker is called a *defeater*. A defeater $\langle \mathcal{A}, \alpha \rangle$ for $\langle \mathcal{B}, \beta \rangle$ will be referred to as a *proper defeater* if $\langle \mathcal{A}, \alpha \rangle \succ \langle \mathcal{C}, \gamma \rangle$; otherwise, it will be called a *blocking defeater*.

An other part of the dialectical process is the construction of the so called *dialectical tree*, which is used to decide the warrant status of a literal. A dialectical tree contains all the possible acceptable argumentation lines (namely, sequences of defeating arguments) that can be constructed from the given argument that sits on the root of that tree as paths from the root to the leaves. (see [52] for a discussion). More in detail, a dialectical tree for an argument $\langle \mathcal{A}, \alpha \rangle$ is a tree-like structure where nodes are arguments and the root node is $\langle \mathcal{A}, \alpha \rangle$. Each root-to-leaf path in the tree is an *acceptable argumentation line*, which is a finite sequence of arguments that satisfy the following four constraints: (*i*) every argument of the sequence defeats its predecessor; (*ii*) the arguments in odd (resp., even) positions of the sequence does not contradict the strict part of the program; (*iii*) two blocking defeaters cannot appear one immediately after the other in the sequence; and (*iv*) arguments cannot appear twice in the sequence (also when appearing as sub-arguments).

Therefore, it is interesting to note that a dialectical tree for an argument represents the exhaustive dialectical analysis for that argument. Each dialectical tree is then marked to obtain the status of the literal α in the argument at its root through a bottom-up marking procedure, consisting in *i*) marking all leaves of the tree as UNDEFEATED; then, *ii*) every non-leaf node is marked as DEFEATED if and only if at least one of its children is marked as UNDEFEATED, otherwise it is marked as UNDEFEATED. Thus, if there exists a marked dialectical tree whose root contains an argument for α, which is marked as UNDEFEATED, we will say that α is *warranted*[2].

[2]The system available at the following link allows us to compare the abstract semantics with that of DeLP: `https://hosting.cs.uns.edu.ar/~daqap/client/index.html`; see [78] for a description.

Considering the program $\mathcal{P}_1$, only $x,\ y,\ z,\ w$, and b are warranted.

Given a DeLP program $\mathcal{P}$, we define a total function $\mathcal{S}_\mathcal{P} : Lit \rightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ assigning a *status* to each literal w.r.t. $\mathcal{P}$ as follows: $\mathcal{S}_\mathcal{P}(\alpha) = \texttt{in}$ if α is warranted; $\mathcal{S}_\mathcal{P}(\alpha) = \texttt{out}$ if $\mathcal{S}_\mathcal{P}(\sim\alpha) = \texttt{in}$; $\mathcal{S}_\mathcal{P}(\alpha) = \texttt{undec}$ if neither $\mathcal{S}_\mathcal{P}(\alpha) = \texttt{in}$ nor $\mathcal{S}_\mathcal{P}(\alpha) = \texttt{out}$. For literals not occurring in the program we also say that their status is unknown.

Updates. An *update* for a DeLP program $\mathcal{P} = \langle \Pi, \Delta \rangle$ modifies $\mathcal{P}$ into a new program $\mathcal{P}' = \langle \Pi', \Delta' \rangle$ by adding or removing a strict or a defeasible rule r. In particular, we allow the removal of any rule r of $\mathcal{P}$ through an update, and consider the addition of a rule r such that $body(r) \subseteq Lit_\mathcal{P}$ and $head(r) \subseteq Lit$, thus allowing also the addition of a rule whose head is a literal not belonging to $Lit_\mathcal{P}$. Given a DeLP program $\mathcal{P}$ and a strict or defeasible rule r, we use $u = +r$ (resp., $u = -r$) to denote a rule addition (resp., deletion) update to be performed on $\mathcal{P}$, obtaining the DeLP-program $u(\mathcal{P})$ resulting from the application of update u to $\mathcal{P}$. In the following, we assume that any update u is feasible, meaning that i) we only remove (resp. add) strict or defeasible rules appearing (resp., not appearing) in the given program $\mathcal{P}$, and ii) guaranteeing that the strict part of the updated program $u(\mathcal{P})$ will not be contradictory.

5.2 Incremental Computation of Warranted Literals

We first introduce the concept of labeled directed hypergraph associated with a DeLP program, which is central to our incremental approach.

Given a program $\mathcal{P}$, the corresponding labelled hypergraph $G(\mathcal{P}) = \langle N, H \rangle$ consists of a set N of nodes and a set H of labelled hyper-edges (Src, t, l), where Src is a possibly empty set called the *source set*, t is called the *target node*, and $l \in \{\texttt{def}, \texttt{str}, \texttt{cfl}\}$ is a label associated to the hyper-edge. Literals for which there exists a strict derivation in Π are immediately added to the set N of nodes of $G(\mathcal{P})$. Then, for each (strict or defeasible) rule whose body is in N, the head is added to N, and a (`str` or `def`) labelled hyper-edge corresponding to the (strict or defeasible) rule is added to the set H of hyper-edges. Finally, there is a pair of (`cfl`) labelled hyper-edges for each pair of complementary literals appearing as nodes in the hypergraph.

The hypergraph $G(\mathcal{P}_1)$ for the DeLP program $\mathcal{P}_1$ is shown in Figure 9(a) where $\leftrightarrow$ (resp. $\leftarrow$ and $\blacktriangleleft\!\!-$) denotes hyper-edges labeled as `cfl` (resp. `def` and `str`).

We say that there is a path from a literal β to a literal α, if either (i) there exists a hyper-edge whose source set contains β and whose target is α, or (ii) there exists a literal γ and also there exist paths from β to γ and from γ to α. Moreover, we say

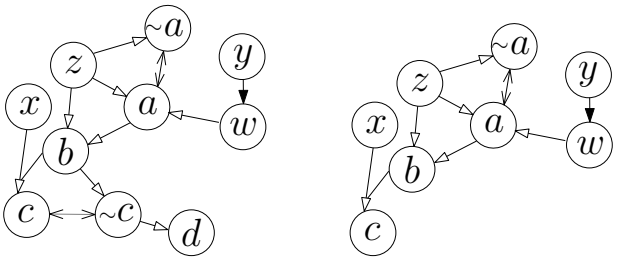

Figure 9: (a) $G(\mathcal{P}_1)$ for program $\mathcal{P}_1$; (b) $G(\mathcal{P}_1')$ for program $\mathcal{P}_1' = -(\sim c \multimap b)(\mathcal{P}_1)$.

that a node y is *reachable* from a set X of nodes if there exists a path from some x in X to y.

Given an update u, we denote with $G(u, \mathcal{P})$ the labeled hypergraph $G(u^+(\mathcal{P}))$ or $G(u^-(\mathcal{P}))$, depending on whether u consists of an insertion or deletion, respectively. The reason of this difference is that, to determine the set of literals whose status may change by deleting a rule r, we need to consider the hypergraph also containing the hyper-edge derived from r.

Given a DeLP-program $\mathcal{P}$ and an update u, our incremental approach for recomputing the status of the literals after performing u consists of the following steps.

- Firstly, it is checked whether the update u is *irrelevant*, that is all literals in *Lit* are preserved. In such a case the initial status $\mathcal{S}_\mathcal{P}$ is returned.
- If u is not irrelevant, we need to:
  - (i) compute the set of literals that are "influenced" by the update;
  - (ii) among the influenced literals determine the subset of literals (called *core literals*) whose status may change after performing the update. The status of uninfluenced literals does not change after the update.
  - (iii) compute the updated status of the core literals; and
  - (iv) determine the updated status of the *inferable* literals, *i.e.,* the literals whose status can be immediately determined from the status of the core literals.

The identification of relevant and irrelevant updates, as well influenced, preserved, core, and inferable literals is discussed below. In [12; 11], it is shown that, in practice, the algorithm resulting from applying the above-mentioned steps turns out to be much more efficient than recomputing everything from scratch.

Irrelevant updates. Sufficient conditions guaranteeing that the status of each literal in the updated program is the same as that of the initial program are investigated in [11]. In these cases we say that the update u is *irrelevant*. One of these conditions holds whenever we add (resp. remove) a defeasible rule whose head's status is `in` (resp. `out`) w.r.t. the initial program. However, this does not hold for updates concerning strict rules. In these cases, we need to makes use of the hypergraph associated with a DeLP program, as well as the status of the literals *related* to an update.

A literal is said to be related to a given update $u = \pm r$ and program $\mathcal{P}$ if it can be reached from $head(r)$ in the labelled hypergraph $G(u, \mathcal{P})$ by navigating forward each rules and backward strict rules only, until no new related literals can be found. We call *deductive closure* of facts and strict rules of a program $\mathcal{P}$ the set of literals that are facts in $\mathcal{P}$ or can be derived from the strict part Π of $\mathcal{P}$. Given this, an update $u = \pm r$ is irrelevant if either (i) $head(r)$ does not belong to $G(u, \mathcal{P})$; or (ii) either $head(r)$ or $\sim head(r)$ appears in the deductive closure of facts and strict rules of both programs $\mathcal{P}$ and $u(\mathcal{P})$; or (iii) at least one literal in the body of r is either `out` or not related to u. Recomputing the status of the updated program's literals can be avoided if an irrelevant update is performed.

Relevant updates and influenced set. We now consider the computation of the status of literals for updates which have not been identified as irrelevant. An update is *relevant* whenever it causes the status of at least one literal to change. That is, even if for relevant updates the status of some literals may not change, and therefore for those literals, their status does not need to be recomputed when the update is performed. To avoid wasted effort, we determine the subset of literals whose status needs to be recomputed after an update. Towards this end, we discuss the concept of *influenced set*, which consists of the set of literals that are related to a given update u and program $\mathcal{P}$ but using only labeled hyper-edges whose corresponding rules are such that (i) the head (or its complement) is not in the deductive closures of both $\mathcal{P}$ and $u(\mathcal{P})$, and (ii) the body does not contain a literal that is not related to u and $\mathcal{P}$ and such that its status is `out`—intuitively, the other hyper-edges can be ignored as they correspond to rules whose head does not change status. For instance, for the program $\mathcal{P}_1$ and update $u = -(\sim c \multimap\!\!\prec b)$, the influenced set consists only of the literals b, c, $\sim c$, and d.

The notion of influenced set for DeLP programs is conceptually similar to the influenced set of Definition 3.3 for abstract argumentation frameworks (where arguments have no internal structure). Although the aim is analogous, here we deal with incremental computation of the status of *structured* arguments, and consider a notion of influenced set w.r.t. an update for a DeLP program and its status that we

then apply to (hyper)graphs representing DeLP programs, from which structured arguments are derived. A significant difference is that here we have both strict and defeasible rules meaning that to determine a portion of the hypergraph that contains nodes corresponding to literals whose status may change, we need to navigate strict edges both forward and backward. As an example, consider the DeLP program $\mathcal{P}_\chi = \langle \Pi_\chi, \Delta_\chi \rangle$ where $\Pi_\chi = \{f_1,\ f_2,\ a \leftarrow b,\ \sim a \leftarrow c\}$ and $\Delta_\chi = \{b \prec\!\!\!- f_1\}$, and let $u = +(c \prec\!\!\!- f_2)$ be an update yielding the updated DeLP program $\mathcal{P}'_\chi$. The influenced set is $\{c,\ \sim a,\ a, b\}$. Observe that b is included in the influenced set by navigating backward via the (hyper)edge corresponding to the strict rule $a \leftarrow b$, while the other literals are reached by forward reachability. Note that including b is important as its status changes (it is `undec` w.r.t. $\mathcal{P}'_\chi$, it was `in` w.r.t. $\mathcal{P}_\chi$).

Preserved, core, and inferable literals. Using the influenced set we can identify the preserved literals, *i.e.,* the literals whose status does not change after performing a relevant update. This set consists of the literals (*i.e.,* nodes) of the updated hypergraph that are not influenced. The status of a literal for which there is no argument in the (updated) program may depend only on the status of its complementary literal—we call such literals *inferable* and use them to define what we call *core literals*. The core literals for an update u are those in $Lit_{\mathcal{P}'}$ that are influenced but are not inferable, where $\mathcal{P}'$ is the updated program. The status of an inferred literal w.r.t. the updated program can be either `out` or `undec`, and if it is `out` it is entailed by the status of a core or preserved literal that is `in`. Finally, the status of the literals not in $Lit_{\mathcal{P}'}$ can be readily determined to be `undec`.

Considering the program $\mathcal{P}_1$ and update $u = -(\sim c \prec\!\!\!- b)$, $\sim c$ and d are the only inferable literals, while b and c are core literals. The (updated) status of the inferable literal $\sim c$ is `out` as it is entailed by the (updated) status of its complementary literal, which is `in`; the status of the inferable literal d remains `undec`.

Efficiency. The incremental technique discussed in this section has been the subject of analysis in [11; 12], which report on a set of experiments comparing the incremental approach with full recomputation from scratch (that is, the direct computation of the status of all the literals in an updated DeLP program using the DeLP-Solver). It turned out that the incremental approach significantly outperforms computation from scratch. Specifically, the incremental algorithm takes only a few seconds for DeLP programs, while the approach from scratch takes almost 2 minutes.

6 Related Work

Overviews of key concepts in argumentation theory and formal models of argumentation in the field of Artificial Intelligence are presented in [29; 31; 91; 19]. Further discussion regarding uses of computational argumentation as an Agreement Technology can be found in [86].

A comprehensive introduction to the semantics of static abstract argumentation frameworks can be found in [21]. Although the idea underlying abstract argumentation frameworks is intuitive and straightforward, most of the semantics proposed so far suffer from a high computational complexity [58; 57; 59; 60; 64; 65; 66; 67]. Complexity bounds and evaluation algorithms for abstract argumentation frameworks have been intensely studied in the literature, but most of this research focused on static frameworks, whereas, in practice, argumentation frameworks are dynamic systems [42; 62; 25; 81; 24; 51]. In fact, in general, an AF represents a temporary situation, and new arguments and attacks can be added/retracted to model new available knowledge. For instance, for disputes among users of online social networks [76], arguments/attacks are continuously added or removed by users to express their point of view in response to the last move made by the adversaries (often disclosing as few arguments/attacks as possible).

There have been several significant efforts aimed at coping with the dynamic aspects of abstract argumentation. In [39; 40], the authors have investigated the principles according to which a grounded extension of a Dung abstract argumentation framework does not change when the set of arguments/attacks is changed. Meanwhile, in [35] a synthesis is presented concerning the characterization of changes based on the work presented in [44; 45; 33; 34] where the evolution of the set of extensions after performing a change operation is studied; here, a change operation can be about adding or removing one interaction or adding or removing one argument and a set of interactions.

Dynamic argumentation has been applied to the decision-making mechanisms of an autonomous agent by [18], where it is studied how the acceptability of arguments evolves when a new argument is added to the decision system. Other relevant works on dynamic aspects of Dung's argumentation frameworks include the following. [25] has proposed an approach exploiting the concept of the splitting of logic programs to deal with dynamic argumentation. The technique considers weak expansions of the initial argumentation framework, where added arguments never attack previous ones. [28] have investigated whether and how it is possible to modify a given argumentation framework so that a desired set of arguments becomes an extension, whereas [90] have studied equivalence between two argumentation frameworks when further information (another argumentation framework) is added to

both argumentation frameworks. [26] has focused on expansions where new arguments and attacks may be added, but the attacks among the old arguments remain unchanged, while [27] have characterized update and deletion equivalence, where adding/deleting arguments/attacks is allowed (deletions were not considered by [90; 26]).

Several approaches for dividing argumentation frameworks into subgraphs have been explored in the context of dynamic argumentation frameworks. The division-based method, proposed in [81] and then refined in [24], divides the updated framework into two parts: *affected* and *unaffected*, where only the status of affected arguments is recomputed after updates. Using the results introduced in [81], the work presented in [80] investigated the efficient evaluation of the justification status of a subset of arguments in an argumentation framework (instead of the whole set of arguments), and proposed an approach based on answer-set programming for *local* computation. In [79], an argumentation framework is decomposed into a set of strongly connected components, yielding sub-argumentation frameworks located in layers, which are then used for incrementally computing the semantics of the given argumentation framework by proceeding layer by layer. Then, [97] introduced a matrix representation of argumentation frameworks and proposed a matrix reduction that, when applied to dynamic argumentation frameworks, resembles the division-based method in [81].

Changes in bipolar argumentation frameworks have been studied in the work [49], where it is shown how the addition of one argument together with one support that involves that argument (and without introducing any attack) impacts the extensions of the updated bipolar argumentation framework. However, these works do not address the incremental computation in dynamic bipolar argumentation frameworks, nor in extended argumentation frameworks modeling attacks towards the attack relation [82; 22] and defeasible support [38].

There have been fewer attempts to consider the dynamics of the defeasible argumentation in the field of structured argumentation [30]. As in the abstract argumentation case, there have been some works following the belief revision approach. In [63], the issue of modifying strict rules to become defeasible was analyzed in the context of revisions effected over a knowledge base, while in [87] the authors thoroughly explored the different cases that may occur when a DeLP program is modified by adding, deleting, or changing its elements. Neither of these works explored the implementation issues related to the problems studied here. Regarding implementations of approaches focusing on improving the tractability of determining the status of pieces of knowledge, in [42; 43], the authors consider several alternatives to avoid recomputing warrants. In [54], the authors focus on challenges arising in the development of recommender systems, addressing them via the design of novel

architectures that improve the computation of answers. Finally, [73] makes use of heuristics designed to improve efficiency, and [92] deals with the computational complexity of performing recalculations in a structured argumentation setting by relying on an approximation algorithm.

We believe that the set of ideas proposed in this work may be a forerunner of similar techniques for the optimization of other structured argumentation frameworks such as, for example, ABA and ASPIC$^+$. Regarding ABA, the construction of deductions is very similar to that of arguments for DeLP, although the way arguments attack each other is different. Therefore, similarly, the ABA framework could be represented using hypergraphs (where assumptions may be modeled as defeasible facts) to identify irrelevant updates and restrict the hypergraph to compute the semantics of updated programs efficiently. The similarities between DeLP and ASPIC$^+$ are even more substantial: both have a distinction between strict and defeasible inference rules, and both use comparison criteria to resolve attacks into defeats; however, while ASPIC$^+$ evaluates arguments with the standard AF semantics, DeLP has a special-purpose definition of argument evaluation [71]. Therefore, the ideas developed here can be of inspiration to optimize the incremental computation of the semantics of ASPIC$^+$ programs.

7 Conclusions and Future Work

We have reviewed techniques for the incremental and efficient computation in dynamic abstract argumentation and defeasible knowledge bases. In the case of abstract argumentation, we have presented a technique enabling any non-incremental algorithm to be used as an incremental one for computing some extension of dynamic argumentation frameworks. The algorithm identifies a tighter portion of the updated argumentation framework to be examined for recomputing the semantics. The incremental algorithm proposed for Dung's frameworks enables a technique for the incremental computation of extensions of dynamic frameworks incorporating supports and second-order attacks (that we called extended argumentation frameworks). Recently, in [3], we have investigated incremental techniques for the ASAF framework [72], where attacks and support relations of any order are considered. For the case of structured argumentation, we have discussed an algorithm able to incrementally solve the problem determining the warrant status of literals in a DeLP program which is updated by adding or deleting strict or defeasible rules. The experimental analysis performed in [5; 7; 12; 11] showed that, in practice, the incremental approach, for both the cases of abstract and structured frameworks, turns out to be much more efficient than recomputing everything from scratch.

The notions behind the use of an incremental approach can be extended further, as done in [8; 1], where an incremental technique was recently proposed aimed at determining whether a given argument is skeptically preferred accepted in dynamic argumentation frameworks by exploiting the concept of influenced and reduced argumentation frameworks presented here in Section 3. Future work will be devoted to extending our technique to cope with other argumentation frameworks [13; 14] and other computational problems [2; 9; 15]. It would be interesting to deal with different interpretations of the support relation, *e.g.,* that one in [47; 48] where a meta argumentation approach is also adopted to deal with bipolarity. We plan to address the problem of incrementally enumerating *all* the extensions of an abstract argumentation framework. Following [8; 10], devising an incremental computation approach for the skeptical/credulous acceptance in dynamic argumentation frameworks, and its extensions (*e.g.,* bipolar argumentation frameworks and ASAFs), is another intriguing direction for future work. Finally, we believe the basic ideas presented for the case of structured argumentation could carry over to other frameworks, *e.g.,* ASPIC$^+$ or ABA; this is another research direction we are planning to pursue in future work.

Acknowledgments

The authors are grateful to the participants of the workshop *Current Trends in Formal Argumentation* (held at the University Centre of Bertinoro from November 3rd to 6th, 2019), and in particular to Matthias Thimm, for discussions on the limitations of the technique previously presented in [11] that improved the characterization of the irrelevant updates, and of the set of influenced literals, which are on the basis of the incremental approach for DeLP. Moreover, the authors wish to thank the reviewers for their comments and suggestions that helped us in improving the paper. Finally, this work was partly supported in Argentina by Universidad Nacional del Sur (UNS) under grant PGI 24/ZN34, Consejo Nacional de Investigaciones Cientificas y Técnicas (CONICET), and Agencia Nacional de Promoción Cientifica y Tecnológica under grant PICT-2018-0475.

References

[1] G. Alfano and S. Greco. Incremental skeptical preferred acceptance in dynamic argumentation frameworks. *IEEE Intelligent Systems*, 2021.

[2] Gianvincenzo Alfano, Marco Calautti, Sergio Greco, Francesco Parisi, and Irina Trubitsyna. Explainable acceptance in probabilistic abstract argumentation: Complexity and approximation. In *Proceedings of the 17th International Conference on Principles of*

Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020, pages 33–43, 2020.

[3] Gianvincenzo Alfano, Andrea Cohen, Sebastian Gottifredi, Sergio Greco, Francesco Parisi, and Guillermo Ricardo Simari. Dynamics in abstract argumentation frameworks with recursive attack and support relations. In *Proc. of European Conference on Artificial Intelligence (ECAI)*, pages 577–584, 2020.

[4] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. Computing stable and preferred extensions of dynamic bipolar argumentation frameworks. In *Proc. of Workshop on Advances In Argumentation In Artificial Intelligence co-located with XVI International Conference of the Italian Association for Artificial Intelligence (AI*IA)*, pages 28–42, 2017.

[5] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 49–55, 2017.

[6] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. Computing extensions of dynamic abstract argumentation frameworks with second-order attacks. In *IDEAS*, pages 183–192, 2018.

[7] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. A meta-argumentation approach for the efficient computation of stable and preferred extensions in dynamic bipolar argumentation frameworks. *Intelligenza Artificiale*, 12(2):193–211, 2018.

[8] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. An efficient algorithm for skeptical preferred acceptance in dynamic argumentation frameworks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 18–24, 2019.

[9] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. On scaling the enumeration of the preferred extensions of abstract argumentation frameworks. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, pages 1147–1153. ACM, 2019.

[10] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. Computing skeptical preferred acceptance in dynamic argumentation frameworks with recursive attack and support relations. In *Proceedings of COMMA*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, pages 67–78. IOS Press, 2020.

[11] Gianvincenzo Alfano, Sergio Greco, Francesco Parisi, Gerardo I. Simari, and Guillermo R. Simari. An incremental approach to structured argumentation over dynamic knowledge bases. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR*, pages 78–87, 2018.

[12] Gianvincenzo Alfano, Sergio Greco, Francesco Parisi, Gerardo I. Simari, and Guillermo R. Simari. Incremental computation of warranted arguments in dynamic defeasible argumentation: The rule addition case. In *Proc. of the Symposium on Applied Computing (SAC)*, pages 911–917, 2018.

[13] Gianvincenzo Alfano, Sergio Greco, Francesco Parisi, and Irina Trubitsyna. On the se-

mantics of abstract argumentation frameworks: A logic programming approach. *Theory Pract. Log. Program.*, 20(5):703–718, 2020.

[14] Gianvincenzo Alfano, Sergio Greco, Francesco Parisi, and Irina Trubitsyna. On the semantics of recursive bipolar afs and partial stable models. In *Proceedings of the Workshop on Advances In Argumentation In Artificial Intelligence 2020 co-located with the 19th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2020), Online, November 25-26, 2020*, volume 2777 of *CEUR Workshop Proceedings*, pages 16–30. CEUR-WS.org, 2020.

[15] Gianvincenzo Alfano, Sergio Greco, Francesco Parisi, and Irina Trubitsyna. Argumentation frameworks with strong and weak constraints: Semantics and complexity. In *Proc. of AAAI*, page (to appear), 2021.

[16] Leila Amgoud, Jean-François Bonnefon, and Henri Prade. An argumentation-based approach to multiple criteria decision. In *Proc. of European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 269–280, 2005.

[17] Leila Amgoud, Claudette Cayrol, and Marie-Christine Lagasquie-Schiex. On the bipolarity in argumentation frameworks. In *Proc. of International Workshop on Non-Monotonic Reasoning (NMR)*, pages 1–9, 2004.

[18] Leila Amgoud and Srdjan Vesic. Revising option status in argument-based decision systems. *Journal of Logic and Computation*, 22(5):1019–1058, 2012.

[19] Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo R. Simari, Matthias Thimm, and Serena Villata. Towards artificial argumentation. *AI Magazine*, 38(3):25–36, 2017.

[20] Pietro Baroni, Guido Boella, Federico Cerutti, Massimiliano Giacomin, Leendert W. N. van der Torre, and Serena Villata. On the input/output behavior of argumentation frameworks. *Artificial Intelligence*, 217:144–197, 2014.

[21] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410, 2011.

[22] Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Giovanni Guida. Encompassing attacks to attacks in abstract argumentation frameworks. In *Proc. of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 83–94, 2009.

[23] Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Giovanni Guida. AFRA: Argumentation Framework with Recursive Attacks. *IJAR*, 52(1):19–37, 2011.

[24] Pietro Baroni, Massimiliano Giacomin, and Beishui Liao. On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: A division-based method. *Artificial Intelligence*, 212:104–115, 2014.

[25] Ringo Baumann. Splitting an argumentation framework. In *Proc. of International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR)*, pages 40–53, 2011.

[26] Ringo Baumann. Normal and strong expansion equivalence for argumentation frameworks. *Artificial Intelligence*, 193:18–44, 2012.

[27] Ringo Baumann. Context-free and context-sensitive kernels: Update and deletion equivalence in abstract argumentation. In *Proc. of European Conference on Artificial Intelligence (ECAI)*, pages 63–68, 2014.

[28] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. In *Proc. of Third International Conference on Computational Models of Argument (COMMA)*, pages 75–86, 2010.

[29] Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10 - 15):619 – 641, 2007.

[30] Philippe Besnard, Alejandro J. Garcia, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo R. Simari, and Francesca Toni. Introduction to structured argumentation. *Argument & Computation – Special Issue: Tutorials on Structured Argumentation*, 5(1):1–4, 2014.

[31] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[32] Philippe Besnard and Anthony Hunter. Constructing argument graphs with deductive arguments: A tutorial. *Argument & Computation*, 5(1):5–30, 2014.

[33] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Change in argumentation systems: Exploring the interest of removing an argument. In *Proceedings of the 5th International Conference on Scalable Uncertainty Management (SUM)*, pages 275–288, 2011.

[34] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Changement dans un système d'argumentation : suppression d'un argument. *Rev. d'Intelligence Artif.*, 26(3):225–253, 2012.

[35] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Characterizing change in abstract argumentation systems. In *Trends in Belief Revision and Argumentation Dynamics*, volume 48, pages 75–102. 2013.

[36] Stefano Bistarelli, Francesco Faloci, Francesco Santini, and Carlo Taticchi. Studying dynamics in argumentation with Rob. In *COMMA*, pages 451–452, 2018.

[37] Stefano Bistarelli, Lars Kotthoff, Francesco Santini, and Carlo Taticchi. Containerisation and dynamic frameworks in iccma'19. In *Proceedings of the Second International Workshop on Systems and Algorithms for Formal Argumentation (SAFA 2018) co-located with the 7th International Conference on Computational Models of Argument (COMMA 2018), Warsaw, Poland, September 11, 2018.*, pages 4–9, 2018.

[38] Guido Boella, Dov M. Gabbay, Leendert W. N. van der Torre, and Serena Villata. Support in abstract argumentation. In *Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010.*, pages 111–122, 2010.

[39] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre. Dynamics in argumentation with single extensions: Abstraction principles and the grounded extension. In *Proc. of European Conference on Symbolic and Quantitative Approaches to Reasoning*

and Uncertainty (ECSQARU), pages 107–118, 2009.

[40] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre. Dynamics in argumentation with single extensions: Attack refinement and the grounded extension. In *Proc. of Sixth Int. Workshop on Argumentation in Multi-Agent Systems (ArgMAS)*, pages 150–159, 2009.

[41] Martin Caminada. Semi-stable semantics. In *Proc. of 1st International Conference on Computational Models of Argument (COMMA)*, pages 121–130, 2006.

[42] Marcela Capobianco, Carlos I. Chesñevar, and Guillermo R. Simari. Argumentation and the dynamics of warranted beliefs in changing environments. *Autonomous Agents and Multi-Agent Systems*, 11(2):127–151, 2005.

[43] Marcela Capobianco and Guillermo R. Simari. A proposal for making argumentation computationally capable of handling large repositories of uncertain data. In *Proc. of Scalable Uncertainty Management*, pages 95–110, 2009.

[44] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Revision of an argumentation system. In *Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 124–134, 2008.

[45] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Change in abstract argumentation frameworks: Adding an argument. *Journal of Artificial Intelligence Research*, 38:49–84, 2010.

[46] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *Proc. of European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 378–389, 2005.

[47] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolar abstract argumentation systems. In *Argumentation in Artificial Intelligence*, pages 65–84. 2009.

[48] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Coalitions of arguments: A tool for handling bipolar argumentation frameworks. *International Journal of Intelligent System*, 25(1):83–109, 2010.

[49] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Change in abstract bipolar argumentation systems. In *Proc. of International Conference on Scalable Uncertainty Management (SUM)*, pages 314–329, 2015.

[50] Laura A. Cecchi, Pablo R. Fillottrani, and Guillermo R. Simari. On the complexity of DeLP through game semantics. In *Proc. of the 11th International Workshop on Nonmonotonic Reasoning, Lake District, GBP*, pages 386–394, 2006.

[51] Günther Charwat, Wolfgang Dvorák, Sarah A. Gaggl, Johannes P. Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation - A Survey. *Artificial Intelligence*, 220:28–63, 2015.

[52] Carlos I. Chesñevar and Guillermo R. Simari. Modelling inference in argumentation through labelled deduction: Formalization and logical properties. *Logica Universalis*, 1(1):93–124, 2007.

[53] Andrea Cohen, Sebastian Gottifredi, Alejandro J. Garcia, and Guillermo R. Simari. A

survey of different approaches to support in argumentation systems. *The Knowledge Engineering Review*, 29(5):513–550, 2014.

[54] Cristhian A. D. Deagustini, Santiago E. Fulladoza Dalibón, Sebastian Gottifredi, Marcelo A. Falappa, Carlos I. Chesñevar, and Guillermo R. Simari. Relational databases as a massive information source for defeasible argumentation. *Knowledge-Based Systems*, 51:93–109, 2013.

[55] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[56] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.

[57] Paul E. Dunne. The computational complexity of ideal semantics. *Artificial Intelligence*, 173(18):1559–1591, 2009.

[58] Paul E. Dunne and Michael Wooldridge. Complexity of abstract argumentation. In *Argumentation in Artificial Intelligence*, pages 85–104. 2009.

[59] Wolfgang Dvorak, Reinhard Pichler, and Stefan Woltran. Towards fixed-parameter tractable algorithms for argumentation. In *Proc. of European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, 2010.

[60] Wolfgang Dvorak and Stefan Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Information Processing Letters*, 110(11):425–430, 2010.

[61] Thomas Eiter, Hannes Strass, Miroslaw Truszczynski, and Stefan Woltran, editors. *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation*, volume 9060. Springer, 2015.

[62] Marcelo A. Falappa, Alejandro J. Garcia, Gabriele Kern-Isberner, and Guillermo R. Simari. On the evolving relation between belief revision and argumentation. *The Knowledge Engineering Review*, 26(1):35–43, 2011.

[63] Marcelo A. Falappa, Gabriele Kern-Isberner, and Guillermo R. Simari. Explanations, belief revision and defeasible reasoning. *Artif. Intell.*, 141(1/2):1–28, 2002.

[64] Bettina Fazzinga, Sergio Flesca, and Francesco Parisi. Efficiently estimating the probability of extensions in abstract argumentation. In *Proc. of International Conference on Scalable Uncertainty Management (SUM)*, pages 106–119, 2013.

[65] Bettina Fazzinga, Sergio Flesca, and Francesco Parisi. On the complexity of probabilistic abstract argumentation frameworks. *ACM Transactions on Computational Logic*, 16(3):22, 2015.

[66] Bettina Fazzinga, Sergio Flesca, and Francesco Parisi. On efficiently estimating the probability of extensions in abstract argumentation frameworks. *International Journal of Approximate Reasoning*, 69:106–132, 2016.

[67] Bettina Fazzinga, Sergio Flesca, Francesco Parisi, and Adriana Pietramala. PARTY: A mobile system for efficiently assessing the probability of extensions in a debate. In *Proc. of International Conference on Database and Expert Systems Applications (DEXA)*, pages 220–235, 2015.

[68] Alejandro J. Garcia, Henry Prakken, and Guillermo R. Simari. A comparative study of some central notions of ASPIC$^+$ and DeLP. *Theory and Practice of Logic Programming*, 20(3):358–390, 2020.

[69] Alejandro J. Garcia and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming (TPLP)*, 4(1-2):95–138, 2004.

[70] Alejandro J. Garcia and Guillermo R. Simari. Defeasible logic programming: DeLP-servers, contextual queries, and explanations for answers. *Argument & Computation*, 5(1):63–88, 2014.

[71] Alejandro Javier Garcia, Henry Prakken, and Guillermo Ricardo Simari. A comparative study of some central notions of ASPIC+ and delp. *Theory Pract. Log. Program.*, 20(3):358–390, 2020.

[72] Sebastian Gottifredi, Andrea Cohen, Alejandro J. Garcia, and Guillermo R. Simari. Characterizing acceptability semantics of argumentation frameworks with recursive attack and support relations. *Artif. Intell.*, 262:336–368, 2018.

[73] Sebastian Gottifredi, Nicolas D. Rotstein, Alejandro J. Garcia, and Guillermo R. Simari. Using argument strength for building dialectical bonsai. *Annals of Mathematics and Artificial Intelligence*, 69(1):103–129, 2013.

[74] Sergio Greco and Francesco Parisi. Efficient computation of deterministic extensions for dynamic abstract argumentation frameworks. In *Proc. of European Conference on Artificial Intelligence (ECAI)*, pages 1668–1669, 2016.

[75] Sergio Greco and Francesco Parisi. Incremental computation of deterministic extensions for dynamic argumentation frameworks. In *Proc. of European Conference On Logics In Artificial Intelligence (JELIA)*, pages 288–304, 2016.

[76] Nadin Kökciyan, Nefise Yaglikci, and Pinar Yolum. Argumentation for resolving privacy disputes in online social networks: (extended abstract). In *Proc. of International Conference on Autonomous Agents and Multiagent Sytems (AAMAS)*, pages 1361–1362, 2016.

[77] Markus Kröll, Reinhard Pichler, and Stefan Woltran. On the complexity of enumerating the extensions of abstract argumentation frameworks. In *Proc. of IJCAI*, pages 1145–1152, 2017.

[78] Mario A. Leiva, Gerardo I. Simari, Sebastian Gottifredi, Alejandro J. Garcia, and Guillermo R. Simari. DAQAP: defeasible argumentation query answering platform. In *Proceedings of the 13th International Conference on Flexible Query Answering Systems (FQAS)*, pages 126–138, 2019.

[79] Beishui Liao. Toward incremental computation of argumentation semantics: A decomposition-based approach. *Annals of Mathematics and Artificial Intelligence*, 67(3-4):319–358, 2013.

[80] Beishui Liao and Huaxin Huang. Partial semantics of argumentation: basic properties and empirical results. *Journal of Logic and Computation*, 23(3):541–562, 2013.

[81] Beishui Liao, Li Jin, and Robert C. Koons. Dynamics of argumentation systems: A

division-based method. *Artificial Intelligence*, 175(11):1790–1814, 2011.

[82] Sanjay Modgil. An abstract theory of argumentation that accommodates defeasible reasoning about preferences. In *Proc. of European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 648–659, 2007.

[83] Sanjay Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9-10):901–934, 2009.

[84] Sanjay Modgil and Henry Prakken. Revisiting preferences and argumentation. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1021–1026, 2011.

[85] Sanjay Modgil and Henry Prakken. The ASPIC$^+$ framework for structured argumentation: A tutorial. *Argument & Computation*, 5(1):31–62, 2014.

[86] Sanjay Modgil, Francesca Toni, Floris Bex, Ivan Bratko, Carlos I. Chesnevar, Wolfgang Dvorak, Marcelo A. Falappa, Xiuyi Fan, Sarah Alice Gaggl, Alejandro J. Garcia, Maria P. Gonzalez, Thomas F. Gordon, Joao Leite, Martin Mouzina, Chris Reed, Guillermo R. Simari, Stefan Szeider, Paolo Torroni, and Stefan Woltran. *Agreement Technologies*, volume 8 of *Law, Governance and Technology*, chapter 21: The Added Value of Argumentation: Examples and Challenges, pages 357–404. Springer, 2013.

[87] Martin O. Moguillansky, Nicolas D. Rotstein, Marcelo A. Falappa, Alejandro J. Garcia, and Guillermo R. Simari. Dynamics of knowledge in DeLP through argument theory change. *TPLP*, 13(6):893–957, 2013.

[88] Farid Nouioua and Vincent Risch. Bipolar argumentation frameworks with specialized supports. In *Proc. of ICTAI*, pages 215–218, 2010.

[89] Farid Nouioua and Vincent Risch. Argumentation frameworks with necessities. In *Proc. of SUM*, pages 163–176, 2011.

[90] Emilia Oikarinen and Stefan Woltran. Characterizing strong equivalence for argumentation frameworks. *Artificial Intelligence*, 175(14-15):1985–2009, 2011.

[91] Iyad Rahwan and Guillermo R. Simari. *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[92] Bas Testerink, Daphne Odekerken, and Floris Bex. A method for efficient argument-based inquiry. In *Proceedings of the 13th International Conference on Flexible Query Answering Systems (FQAS)*, pages 114–125, 2019.

[93] Matthias Thimm and Serena Villata. The first international competition on computational models of argumentation: Results and analysis. *Artificial Intelligence*, 252:267–294, 2017.

[94] Francesca Toni. A tutorial on assumption-based argumentation. *Argument & Computation*, 5(1):89–117, 2014.

[95] Ignacio D. Viglizzo, Fernando A. Tohmé, and Guillermo R. Simari. *Annals of Mathematics and Artificial Intelligence*, 57(2):181–204, 2009.

[96] Serena Villata, Guido Boella, Dov M. Gabbay, and Leendert W. N. van der Torre. Modelling defeasible and prioritized support in bipolar argumentation. *Annals of Mathematics and Artificial Intelligence*, 66(1-4):163–197, 2012.

[97] Yuming Xu and Claudette Cayrol. The matrix approach for abstract argumentation frameworks. In *Proc. of International Workshop on Theory and Applications of Formal Argumentation (TAFA)*, pages 243–259, 2015.

Received May 2021