

# Efficient Computation of Extensions for Dynamic Abstract Argumentation Frameworks: An Incremental Approach

Gianvincenzo Alfano, Sergio Greco, **Francesco Parisi**

{g.alfano, greco, fparisi}@dimes.unical.it

Department of Informatics, Modeling, Electronics and System Engineering  
University of Calabria  
Italy

26<sup>th</sup> International Joint Conference on Artificial Intelligence

August 19-25, 2017

Melbourne, Australia

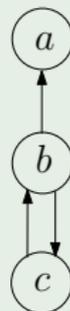
# Argumentation in AI

- A general way for representing arguments and relationships (rebuttals) between them
- It allows representing dialogues, making decisions, and handling inconsistency and uncertainty

**Abstract Argumentation Framework (AF)** [Dung 1995]: arguments are abstract entities (no attention is paid to their internal structure) that may attack and/or be attacked by other arguments

## Example (a simple AF)

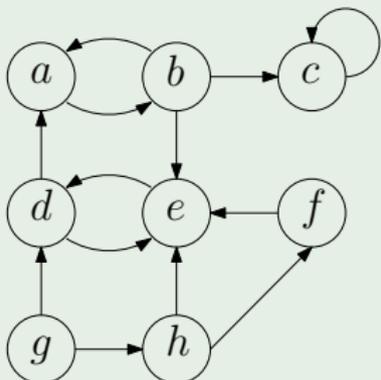
- a = Our friends will have great fun at our party on Saturday  
b = Saturday will rain (according to the weather forecasting service 1)  
c = Saturday will be sunny (according to the weather forecasting service 2)



# Argumentation Semantics

- Several semantics have been proposed to identify “reasonable” sets of arguments, called *extensions*

## Example (AF $\mathcal{A}_0$ )



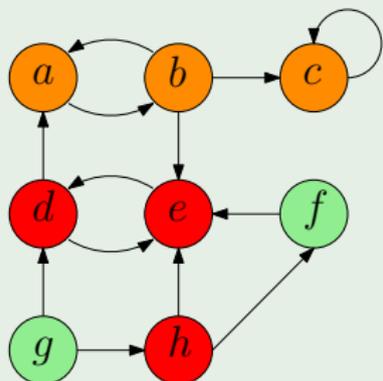
Semantic $\mathcal{S}$	Set of extensions $\mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$
complete (co)	$\{\{f, g\}, \{a, f, g\}, \{b, f, g\}\}$
preferred (pr)	$\{\{a, f, g\}, \{b, f, g\}\}$
stable (st)	$\{\{b, f, g\}\}$
grounded (gr)	$\{\{f, g\}\}$

- Argumentation semantics can be also defined in terms of *labelling*
- Function  $L : A \rightarrow \{\text{IN}, \text{OUT}, \text{UN}\}$  assigns a label (**accepted**, **rejected**, **undecided**) to each argument

# Argumentation Semantics

- Several semantics have been proposed to identify “reasonable” sets of arguments, called *extensions*

## Example (AF $\mathcal{A}_0$ )



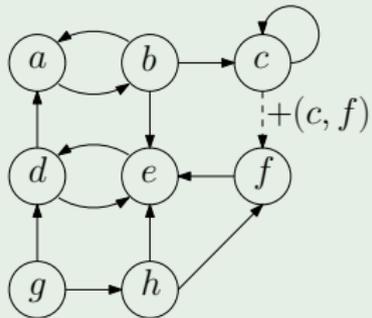
Semantic $\mathcal{S}$	Set of extensions $\mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$
complete (co)	$\{\{f, g\}, \{a, f, g\}, \{b, f, g\}\}$
preferred (pr)	$\{\{a, f, g\}, \{b, f, g\}\}$
stable (st)	$\{\{b, f, g\}\}$
<b>grounded (gr)</b>	<b><math>\{\{f, g\}\}</math></b>

- Argumentation semantics can be also defined in terms of *labelling*
- Function  $L : A \rightarrow \{\text{IN}, \text{OUT}, \text{UN}\}$  assigns a label (**accepted**, **rejected**, **undecided**) to each argument

# Dynamic Abstract Argumentation Frameworks

- Most argumentation frameworks are dynamic systems, which are often updated by adding/removing arguments/attacks.
- For each semantics, extensions/labellings change if we update the initial AF by adding/removing arguments/attacks

## Example (Updated AF $\mathcal{A} = +(c, f)(\mathcal{A}_0)$ )



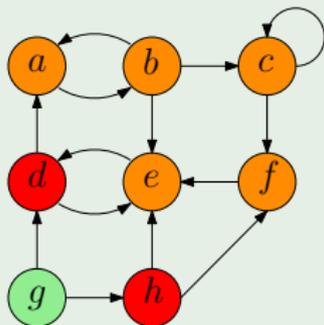
$S$	$\mathcal{E}_S(\mathcal{A}_0)$	$\mathcal{E}_S(\mathcal{A})$
co	$\{\{f, g\}, \{a, f, g\}, \{b, f, g\}\}$	?
pr	$\{\{a, f, g\}, \{b, f, g\}\}$	?
st	$\{\{b, f, g\}\}$	?
gr	$\{\{f, g\}\}$	?

- Should we recompute the semantics of updated AFs from scratch?

# Dynamic Abstract Argumentation Frameworks

- Most argumentation frameworks are dynamic systems, which are often updated by adding/removing arguments/attacks.
- For each semantics, extensions/labellings change if we update the initial AF by adding/removing arguments/attacks

## Example (Updated AF $\mathcal{A} = +(c, f)(\mathcal{A}_0)$ )



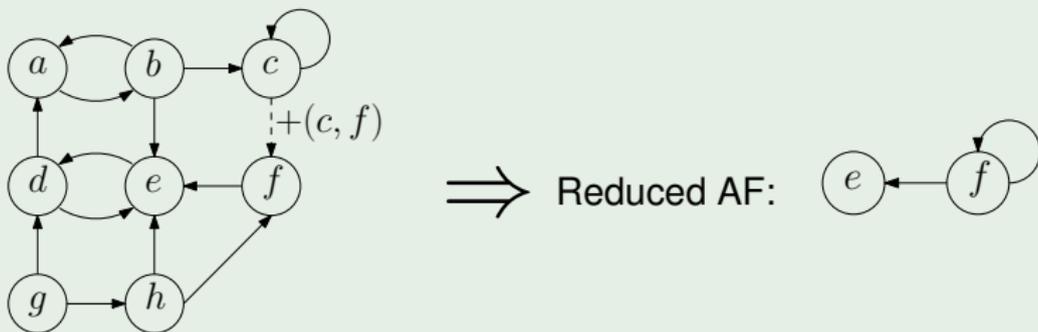
$S$	$\mathcal{E}_S(\mathcal{A}_0)$	$\mathcal{E}_S(\mathcal{A})$
co	$\{\{f, g\}, \{a, f, g\}, \{b, f, g\}\}$	$\{\{g\}, \{a, g\}, \{b, f, g\}\}$
pr	$\{\{a, f, g\}, \{b, f, g\}\}$	$\{\{a, g\}, \{b, f, g\}\}$
st	$\{\{b, f, g\}\}$	$\{\{b, f, g\}\}$
gr	$\{\{f, g\}\}$	$\{\{g\}\}$

- Should we recompute the semantics of updated AFs from scratch?

# Reduced AF

- We show that for four well-known semantics (i.e., *grounded*, *complete*, *preferred*, and *stable*) an extension of the updated AF can be efficiently computed by looking only at a small part of the AF, called the *Reduced AF*, which is “influenced by” the update operation

## Example (From the updated AF to the Reduced AF)



- Once computed an extension for the reduced AF, it can be combined with the initial extension of the given AF to get an extension of the updated AF

# Incremental Algorithm

- 1 We formally define the *Reduced AF*
  - Sub-AF consisting of the arguments whose status could change after an update
  - It depends on both the update and the initial extension  $E_0$  (and thus the semantics)
- 2 We present an incremental algorithm for recomputing an extension of an updated AF for the *grounded, complete, preferred, and stable* semantics
  - It calls a non-incremental solver to compute an extension of the reduced AF
  - It obtains the final extension by merging a portion of the initial extension with that computed for the reduced AF.
- 3 A thorough experimental analysis showing the effectiveness of our approach for all the four semantics
  - Our technique outperforms the computation from scratch of the best solvers by two orders of magnitude

# Outline

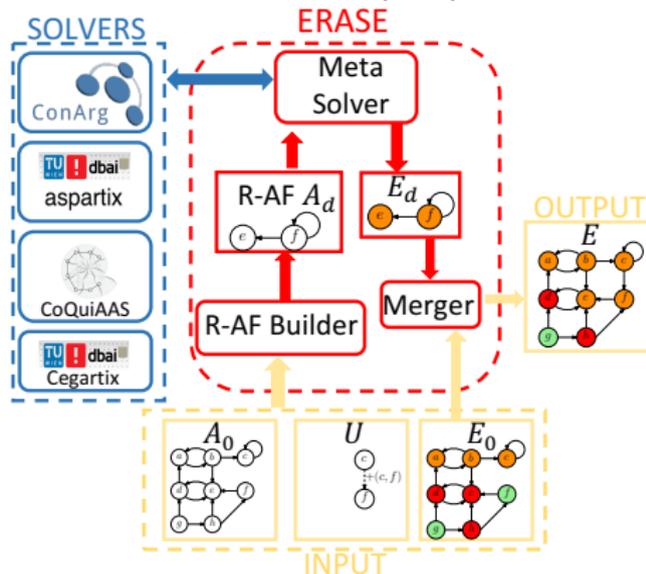
- 1 Introduction
  - Motivation
  - Contributions
- 2 **Incremental Computation**
  - Influenced Arguments
  - Reduced Argumentation Framework
  - Incremental Algorithm
- 3 Experiments
- 4 Conclusions and future work
  - References

# Overview of the approach

Given an initial AF  $\mathcal{A}_0$ , an extension  $E_0$ , and an update  $u = \pm(a, b)$

Three main steps/modules:

- 1) Identify a sub-AF  $\mathcal{A}_d = \langle A_d, \Sigma_d \rangle$ , called *reduced* AF (R-AF) on the basis of the updates in  $U$  and additional information provided by the initial extension  $E_0$
- 2) Compute an  $\mathcal{S}$ -extension  $E_d$  of the reduced AF  $\mathcal{A}_d$  by using an external (non-incremental) solver
- 3) Merge  $E_d$  with the portion ( $E_0 \setminus A_d$ ) of the initial extension that does not change



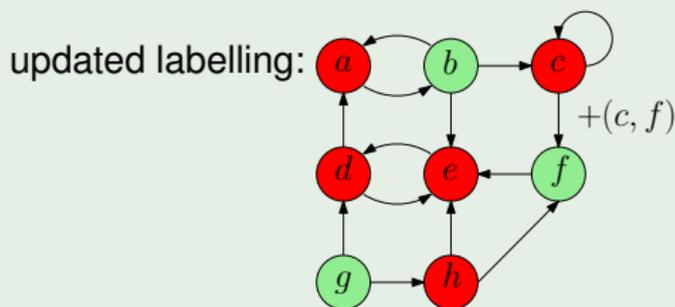
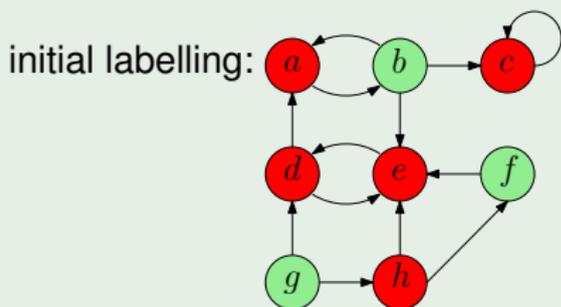
Architecture of ERASE, our system for Efficiently Recomputing Argumentation Semantics.

# Irrelevant updates (1/2)

- Updates preserving a given initial extension/labelling
- Cases for which  $E_0$  is still an extension of the updated AF after a *positive* update

update $+(a, b)$		$L_0(b)$		
		IN	UN	OUT
$L_0(a)$	IN			co, pr, st, gr
	UN		co, gr	co, pr, gr
	OUT	co, <b>pr</b> , st	co, gr	co, pr, st, gr

Example (For the update  $+(c, f)$  the initial preferred extension  $E_0 = \{b, f, g\}$  is preserved, as  $L_0(c) = \text{OUT}$  and  $L_0(f) = \text{IN}$ )



# Irrelevant updates (2/2)

- Similar result for *negative* updates
- Cases for which  $E_0$  is still an extension of the updated AF after a *negative* update

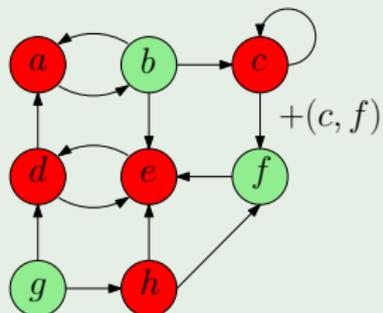
		update			
		$-(a, b)$			
$L_0(a)$		$L_0(b)$			
		IN	UN	OUT	
		IN	NA	NA	
		UN	NA		co, pr, gr
	OUT	co, pr, st, gr	co, pr, gr	co, pr, st, gr	

- In these cases we do not need to recompute the semantics of the updated AF: just return the initial extension

# Influenced set: Intuition

- $\mathcal{I}(u, \mathcal{A}_0, E_0)$  denotes the *influenced set* of  $u = \pm(a, b)$  w.r.t.  $\mathcal{A}_0$  and  $E_0$
- 1)  $\mathcal{I}(u, \mathcal{A}_0, E_0) = \emptyset$  if  $u$  is irrelevant w.r.t.  $E_0$  and the considered semantics.
  - 2) The status of an argument can change only if it is reachable from  $b$ :  
 $\mathcal{I}(u, \mathcal{A}_0, E_0) \subseteq \text{Reach}_{\mathcal{A}}(b)$
  - 3) If argument  $z$  is not reachable from  $b$  and  $z \in E_0$ , then also the status of the arguments attacked by  $z$  cannot change: their status remain OUT

## Example (Set of arguments influenced by an update operation)



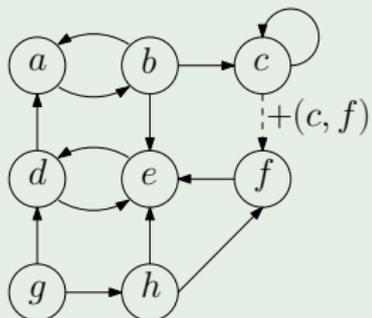
Update  $+(c, f)$  is irrelevant w.r.t. the preferred extension  $E_0 = \{b, f, g\}$

$$\Rightarrow \mathcal{I}(+(c, f), \mathcal{A}_0, \{b, f, g\}) = \emptyset$$

# Influenced set: Intuition

- $\mathcal{I}(u, \mathcal{A}_0, E_0)$  denotes the *influenced set* of  $u = \pm(a, b)$  w.r.t.  $\mathcal{A}_0$  and  $E_0$
- 1)  $\mathcal{I}(u, \mathcal{A}_0, E_0) = \emptyset$  if  $u$  is irrelevant w.r.t.  $E_0$  and the considered semantics.
- 2) The status of an argument can change only if it is reachable from  $b$ :  
 $\mathcal{I}(u, \mathcal{A}_0, E_0) \subseteq \text{Reach}_{\mathcal{A}}(b)$
- 3) If argument  $z$  is not reachable from  $b$  and  $z \in E_0$ , then also the status of the arguments attacked by  $z$  cannot change: their status remain OUT

## Example (Set of arguments influenced by an update operation)



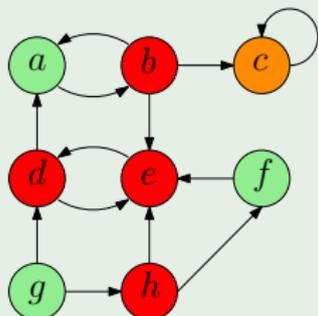
$$\mathcal{I}(+(c, f), \mathcal{A}_0, E_0) \subseteq \text{Reach}_{\mathcal{A}}(f) = \{e, d, a, b, c\}$$

$$\Rightarrow g, h \notin \mathcal{I}(+(c, f), \mathcal{A}_0, E_0)$$

# Influenced set: Intuition

- $\mathcal{I}(u, \mathcal{A}_0, E_0)$  denotes the *influenced set* of  $u = \pm(a, b)$  w.r.t.  $\mathcal{A}_0$  and  $E_0$
- 1)  $\mathcal{I}(u, \mathcal{A}_0, E_0) = \emptyset$  if  $u$  is irrelevant w.r.t.  $E_0$  and the considered semantics.
  - 2) The status of an argument can change only if it is reachable from  $b$ :  
 $\mathcal{I}(u, \mathcal{A}_0, E_0) \subseteq \text{Reach}_{\mathcal{A}}(b)$
  - 3) If argument  $z$  is not reachable from  $b$  and  $z \in E_0$ , then also the status of the arguments attacked by  $z$  cannot change: their status remain OUT

## Example (Set of arguments influenced by an update operation)



$d \notin \mathcal{I}(+(d, f), \mathcal{A}_0, E_0)$  since it is attacked by  $g \in E_0$  and  $g$  is not reachable from  $f$ .

Thus the arguments that can be reached only using  $d$  cannot belong to  $\mathcal{I}(+(c, f), \mathcal{A}_0, E_0)$ .

⇒ **The influenced set is**  $\mathcal{I}(+(c, f), \mathcal{A}_0, E_0) = \{f, e\}$

# Influenced set: Definition

- $\mathcal{I}(\pm(a, b), \mathcal{A}_0, E_0)$  is the set of arguments 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  $z \in E_0$  which is not reachable from  $b$

## Definition (Influenced set)

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

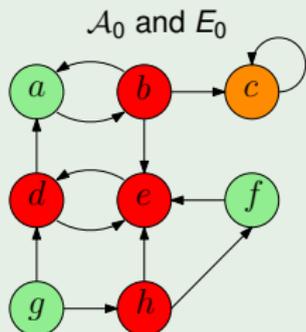
- $\mathcal{I}_0(u, \mathcal{A}, E) = \begin{cases} \emptyset & \text{if } u \text{ is irrelevant w.r.t. } E \text{ and } S \text{ or} \\ \exists(z, b) \in \Sigma \text{ s.t. } z \in E \wedge z \notin \text{Reach}_{\mathcal{A}}(b); & \\ \{b\} & \text{otherwise;} \end{cases}$
- $\mathcal{I}_{i+1}(u, \mathcal{A}, E) = \mathcal{I}_i(u, \mathcal{A}, E) \cup \{y \mid \exists(x, y) \in \Sigma \text{ s.t. } x \in \mathcal{I}_i(u, \mathcal{A}, E) \wedge \nexists(z, y) \in \Sigma \text{ s.t. } z \in E \wedge z \notin \text{Reach}_{\mathcal{A}}(b)\}.$

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

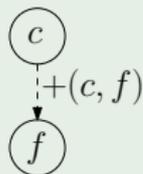
# Reduced AF

- Given an AF  $\mathcal{A}_0$ , an extension  $E_0$ , and an update  $u = \pm(a, b)$ , an extension for the updated AF is recomputed for a small part of the updated AF, called *reduced AF* and denoted  $\mathcal{R}(u, \mathcal{A}_0, E_0)$
- $\mathcal{R}(u, \mathcal{A}_0, E_0)$  consists of the subgraph of  $u(\mathcal{A}_0)$  induced by  $\mathcal{I}(u, \mathcal{A}_0, E_0)$
- plus additional nodes/edges representing the “external context”:
  - if there is in  $u(\mathcal{A}_0)$  an edge from a node  $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$  to a node  $b \in \mathcal{I}(u, \mathcal{A}_0, E_0)$ , we add edge  $(a, b)$  if the status of  $a$  is IN,
  - if there is in  $u(\mathcal{A}_0)$  an edge from a node  $e \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$  to a node  $c \in \mathcal{I}(u, \mathcal{A}_0, E_0)$  such that  $e$  is UN, we add edge  $(c, e)$  to  $\mathcal{R}_{\text{ext}}(u, \mathcal{A}_0, E_0)$

## Example (Reduced AF)



$u = +(c, f)$



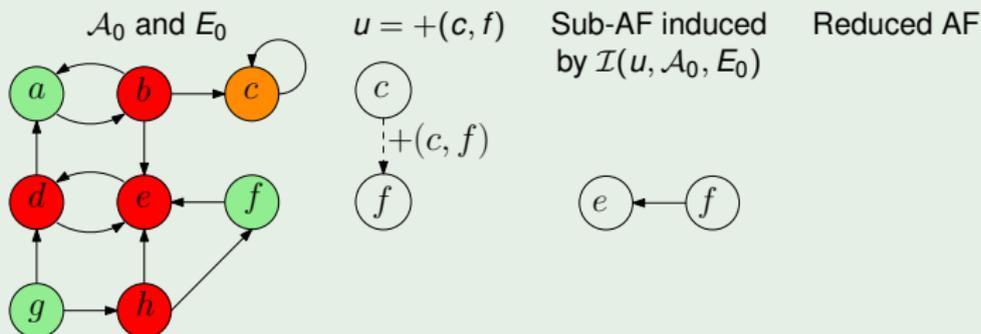
Sub-AF induced  
by  $\mathcal{I}(u, \mathcal{A}_0, E_0)$

Reduced AF

# Reduced AF

- Given an AF  $\mathcal{A}_0$ , an extension  $E_0$ , and an update  $u = \pm(a, b)$ , an extension for the updated AF is recomputed for a small part of the updated AF, called *reduced AF* and denoted  $\mathcal{R}(u, \mathcal{A}_0, E_0)$
- $\mathcal{R}(u, \mathcal{A}_0, E_0)$  consists of the subgraph of  $u(\mathcal{A}_0)$  induced by  $\mathcal{I}(u, \mathcal{A}_0, E_0)$
- plus additional nodes/edges representing the “external context”:
  - if there is in  $u(\mathcal{A}_0)$  an edge from a node  $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$  to a node  $b \in \mathcal{I}(u, \mathcal{A}_0, E_0)$ , we add edge  $(a, b)$  if the status of  $a$  is IN,
  - if there is in  $u(\mathcal{A}_0)$  an edge from a node  $e \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$  to a node  $c \in \mathcal{I}(u, \mathcal{A}_0, E_0)$  such that  $e$  is UN, we add edge  $(c, e)$  to  $\mathcal{R}_{rr}(u, \mathcal{A}_0, E_0)$

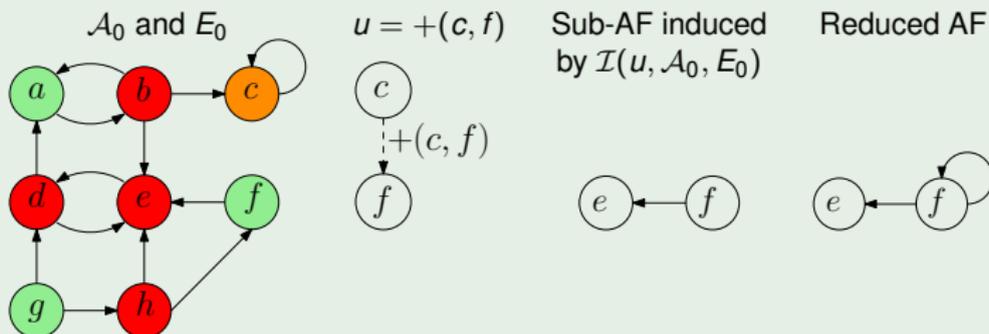
## Example (Reduced AF)



# Reduced AF

- Given an AF  $\mathcal{A}_0$ , an extension  $E_0$ , and an update  $u = \pm(a, b)$ , an extension for the updated AF is recomputed for a small part of the updated AF, called *reduced AF* and denoted  $\mathcal{R}(u, \mathcal{A}_0, E_0)$
- $\mathcal{R}(u, \mathcal{A}_0, E_0)$  consists of the subgraph of  $u(\mathcal{A}_0)$  induced by  $\mathcal{I}(u, \mathcal{A}_0, E_0)$
- plus additional nodes/edges representing the “external context”:
  - if there is in  $u(\mathcal{A}_0)$  an edge from a node  $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$  to a node  $b \in \mathcal{I}(u, \mathcal{A}_0, E_0)$ , we add edge  $(a, b)$  if the status of  $a$  is IN,
  - if there is in  $u(\mathcal{A}_0)$  an edge from a node  $e \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$  to a node  $c \in \mathcal{I}(u, \mathcal{A}_0, E_0)$  such that  $e$  in UN, we add edge  $(c, c)$  to  $\mathcal{R}_{gr}(u, \mathcal{A}_0, E_0)$

## Example (Reduced AF)

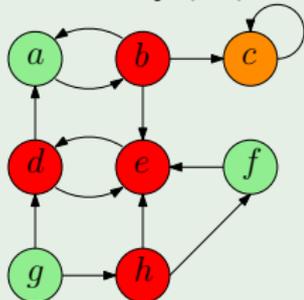
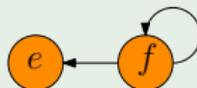
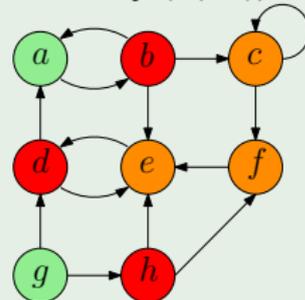


# Using extensions of the reduced AF

## Theorem (Merging extensions)

Let  $\mathcal{A}_0$  be an AF, and  $\mathcal{A} = u(\mathcal{A}_0)$  be the AF resulting from performing update  $u = \pm(a, b)$  on  $\mathcal{A}_0$ . Let  $E_0 \in \mathcal{E}_S(\mathcal{A}_0)$  be an extension for  $\mathcal{A}_0$  under a semantics  $S \in \{co, pr, st, gr\}$ . Then, if  $\mathcal{E}_S(\mathcal{R}(u, \mathcal{A}_0, E_0))$  is not empty, then there is an extension  $E \in \mathcal{E}_S(\mathcal{A})$  for the updated AF  $\mathcal{A}$  such that  $E = (E_0 \setminus \mathcal{I}(u, \mathcal{A}_0, E_0)) \cup E_d$  where  $E_d$  is an  $S$ -extension for reduced AF  $\mathcal{R}(u, \mathcal{A}_0, E_0)$ .

## Example (Merging an initial extension with that of the reduced AF)

 $E_0 \in \mathcal{E}_{pr}(\mathcal{A}_0)$ 

 $E_d \in \mathcal{E}_{pr}(\mathcal{R}(u, \mathcal{A}_0, E_0))$ 

 $E \in \mathcal{E}_{pr}(u(\mathcal{A}_0))$ 


# Incremental Algorithm

## Algorithm Incr-Alg( $\mathcal{A}_0, u, \mathcal{S}, E_0, \text{Solver}_{\mathcal{S}}$ )

**Input:** AF  $\mathcal{A}_0 = \langle \mathcal{A}_0, \Sigma_0 \rangle$ , update  $u = \pm(a, b)$ ,  
semantics  $\mathcal{S} \in \{\text{co}, \text{pr}, \text{st}, \text{gr}\}$ , extension  $E_0 \in \mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$ ,  
function  $\text{Solver}_{\mathcal{S}}(\mathcal{A})$  returning an  $\mathcal{S}$ -extension for AF  $\mathcal{A}$  if it exists,  $\perp$  otherwise;

**Output:** An  $\mathcal{S}$ -extension  $E \in \mathcal{E}_{\mathcal{S}}(u(\mathcal{A}_0))$  if it exists,  $\perp$  otherwise;

- 1:  $S = \mathcal{I}(u, \mathcal{A}_0, E_0)$ ; // Compute the influenced set
- 2: **if** ( $S = \emptyset$ ) **then**
- 3:     **return**  $E_0$ ; // If the influenced set is empty, return the initial extension  $E_0$
- 4:  $\mathcal{A}_d = \mathcal{R}(u, \mathcal{A}_0, E_0)$ ; // Otherwise, compute the reduced AF
- 5: Let  $E_d = \text{Solver}_{\mathcal{S}}(\mathcal{A}_d)$ ; // Compute an extension for the reduced AF using an external solver
- 6: **if** ( $E_d \neq \perp$ ) **then**
- 7:     **return**  $E = (E_0 \setminus S) \cup E_d$ ; // Merge  $E_0$  with extension  $E_d$  of the reduced AF
- 8: **else**
- 9:     **return**  $\text{Solver}_{\mathcal{S}}(u(\mathcal{A}_0))$ ; // If an extension for the reduced AF doesn't exist (it can happen for stable semantics only), compute an extension from scratch

## Theorem (The algorithm is sound and complete)

*Let  $\mathcal{A}_0$  be an AF,  $u = \pm(a, b)$ , and  $E_0 \in \mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$  an extension for  $\mathcal{A}_0$  under  $\mathcal{S} \in \{\text{co}, \text{pr}, \text{st}, \text{gr}\}$ . If  $\text{Solver}_{\mathcal{S}}$  is sound and complete then the algorithm computes  $E \in \mathcal{E}_{\mathcal{S}}(u(\mathcal{A}_0))$  if  $\mathcal{E}_{\mathcal{S}}(u(\mathcal{A}_0)) \neq \emptyset$ , otherwise it returns  $\perp$ .*

# Outline

- 1 Introduction
  - Motivation
  - Contributions
- 2 Incremental Computation
  - Influenced Arguments
  - Reduced Argumentation Framework
  - Incremental Algorithm
- 3 Experiments
- 4 Conclusions and future work
  - References

# Datasets and algorithms

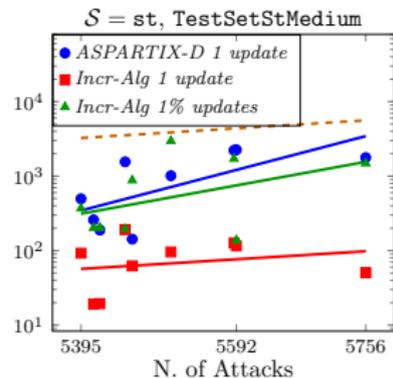
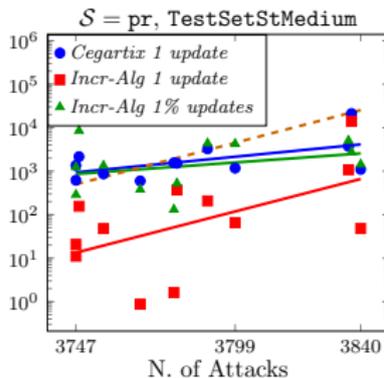
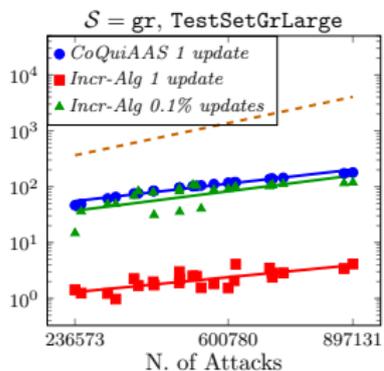
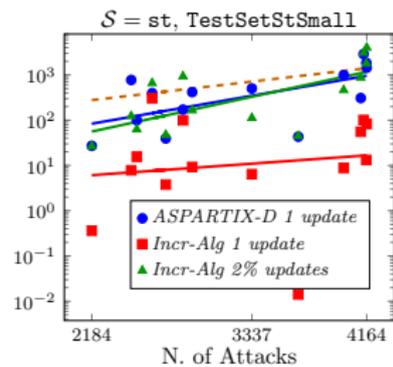
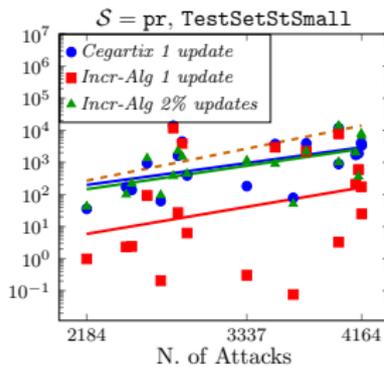
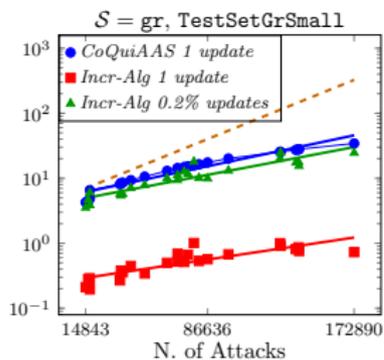
## Datasets: ICCMA'15 benchmarks

- `TestSetGr` consists of AFs with a very large grounded extension and many arguments in general
- `TestSetSt` consists of AFs with many complete/preferred/stable extensions
- `TestSetSCC` consists of AFs with a rich structure of strongly connected components
- for each of these test sets, three classes of AFs of different sizes: `Small`, `Medium`, and `Large`.

**Methodology:** the average run time of our algorithm to compute an  $\mathcal{S}$ -extension was compared with the average run time of the best ICCMA solver to compute an  $\mathcal{S}$ -extension for  $u(\mathcal{A}_0)$  from scratch

- As `SolverS` for computing an  $\mathcal{S}$ -extension for the reduced AF we used the solver that won the ICCMA'15 competition for the task  $\mathcal{S}$ -SE
- `CoQuiAAS` [Lagniez et al. 2015] for  $\mathcal{S} = \text{co}$  and  $\mathcal{S} = \text{gr}$
- `Cegartix` [Dvorák et al. 2014] for  $\mathcal{S} = \text{pr}$
- `ASPARTIX-D` [Gaggl and Manthey 2015] for  $\mathcal{S} = \text{st}$ .

# Experimental Results



# Results

- The size of the reduced AF w.r.t. that of the input AF is about 9% for single updates and 52% for multiple updates with about 1% of the attacks updated.
- Two orders of magnitude faster than the best ICCMA solvers for single updates on average.
- The harder the computation from scratch, the larger the improvements
- Faster even when performing updates simultaneously (green lines) — include the time needed to reduce the application of multiple updates to single attack update
- For sets of updates regarding a relevant portion of the input AF, recomputing extensions after applying them simultaneously is faster than recomputing extensions after applying them sequentially (dashed orange lines)

# Outline

- 1 Introduction
  - Motivation
  - Contributions
- 2 Incremental Computation
  - Influenced Arguments
  - Reduced Argumentation Framework
  - Incremental Algorithm
- 3 Experiments
- 4 Conclusions and future work
  - References

# Conclusions and Future Work

- Our technique enables any non-incremental algorithm to be used as an incremental one for computing some extension of dynamic AFs
- The technique can be used for general (multiple) updates
- We identified a tighter portion of the updated AF to be examined for recomputing the semantics
- Our algorithm exploits the initial extension of an AF for computing an extension of the updated AF
- The experiments showed that the incremental computation outperforms that of the base (non-incremental) computation
- Future work #1: applying the technique to other argumentation semantics (good results for ideal semantics, using ConArg [Bistarelli et al. 2016])
- Future work #2: enumerating all the extensions and deciding credulous/sceptical acceptance

see you at the poster!

## Efficient Computation of Extensions for Dynamic Abstract Argumentation Frameworks: An Incremental Approach

GIANVINCENZO ALFANO, SERGIO GRECO, FRANCESCO PARESÌ

Department of Informatic, Modeling, Electronics and System Engineering, University of Calabria, ITALY

g.alfano, greg, fparesi@unical.it

Thank you!

... questions?

## ABSTRACT ARGUMENTATION

An abstract argumentation framework (AF) is a pair  $(A, \mathcal{A})$ , where  $A$  is a set of arguments and  $\mathcal{A} \subseteq A \times A$  is a set of attacks.

It allows representing dialogues, making decisions, and handling inconsistency.

An AF can be viewed as a directed graph, whose nodes are arguments and whose edges are attacks.

## DYNAMIC ARGUMENTATION FRAMEWORKS

An argumentation framework models a temporary situation as new arguments and attacks can be added/removed to take into account new available knowledge.

For each semantics  $S$ , the sets of extensions change if we update an initial AF  $A_0$  by adding/removing arguments/attacks. For instance,  $E_S(A_0) = \{\{f, g\}\}$  becomes  $E_S(A_1) = \{\{f\}\}$  for the updated AF  $A_1 = (A_0 \cup \{h\}, \mathcal{A})$  obtained from  $A_0$  by adding attack  $h, f$ .



Argument	Attacks
A	B, C, D, E, F, G, H
B	C, D, E, F, G, H
C	D, E, F, G, H
D	E, F, G, H
E	F, G, H
F	G, H
G	H
H	

Should we recompute the semantics of updated AFs from scratch?

## SEMANTICS FOR AFS

An argumentation semantics specifies the criteria for identifying "reasonable" sets of arguments, called *extensions*.

A complete extension  $S$  is an admissible set that contains all the arguments that it defends.

A complete extension  $S$  is said to be preferred (w.r.t.  $S'$ ) iff it is maximal (w.r.t.  $\subseteq$ ).

It is stable (w.r.t.  $S'$ ) iff it attacks all the arguments  $A \notin S$ .

It is grounded (w.r.t.  $S'$ ) iff it is minimal (w.r.t.  $\subseteq$ ).

## UPDATES

An update  $u$  for an AF  $A_0$  consists in modifying  $A_0$  into an AF  $A_1$  by adding or removing arguments or attacks.

$(+a, b)$  (resp.  $(-a, b)$ ) denotes the addition (resp. deletion) of an attack  $(a, b)$ .

$(+A_1)$  means applying  $u = (+a, b)$  to  $A_0$ .

multiple (un)safe updates can be simulated by a single attack update.

## CONTRIBUTIONS

We show that an extension of the updated AF can be efficiently computed by looking only at a small part of the AF, called the *reduced AF*, which is "influenced" by the update operation.

For the example above, the reduced AF is:



We present an incremental technique for computing an extension of an updated AF for the grounded, complete, preferred, and stable semantics.

It consists of the following three main steps:

1) Identify a sub- $AF$   $A_r = (A_r, \mathcal{A}_r)$ , called *reduced AF* (R-AF) on the basis of the updates  $u$  and additional information provided by the initial extension  $E_0$ .

2) Give R-AF  $A_r$  as input to an external (non-incremental) solver to compute an  $S$ -extension  $E_r$  of the reduced AF.

3) Merge  $E_r$  with the portion  $(E_0 \setminus A_r)$  of the initial extension that does not change.

A thorough experimental analysis showing the effectiveness of our approach.

## EXPERIMENTS

For each semantics  $S \in \{\text{pr}, \text{st}, \text{gr}, \text{cp}\}$ , we compared the performance of our technique with that of the solver that won the last ICCMA competition for the computational task  $S$ -SE. Given an AF, determine some  $S$ -extension.

Dataset: ICCMA18 benchmarks.

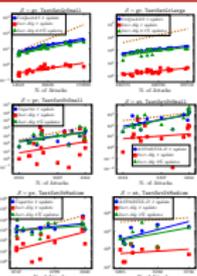
Results: The figure reports the average run times (ms) of ICCMA solvers and our algorithm (Inv-Arg) for different semantics  $S$  over different datasets versus the number of attacks.

Our algorithm significantly outperforms the competitors that compute the extensions from scratch for single updates. In fact, on average, our technique is two orders of magnitude faster than them. Moreover, the harder the computation from scratch is, the larger the improvements are: the improvements obtained for  $S \in \{\text{pr}, \text{st}\}$  are beyond those for  $S \in \{\text{gr}, \text{cp}\}$ .

Our algorithm remains faster than the competitors even when computing an extension after performing a quite large number of updates simultaneously. In particular, the graphs we show the threshold percentages of updated attacks (green lines) up to which the incremental approach for multiple updates is faster than the computation from scratch.

For sets of updates regarding a relevant portion of the input AF (on average at least 1% of the attacks for  $S \in \{\text{st}, \text{pr}\}$  and 0.1% of the attacks for  $S \in \{\text{gr}, \text{cp}\}$ ) computing extensions after applying them simultaneously is faster than computing extensions after applying them sequentially. Indeed, the green lines in the graphs are mostly below the (dashed) orange lines representing the run times of computing extensions after applying the updates sequentially.

The experiments also showed that, on average, the size of the reduced AF w.r.t. that of the input AF is about 9% for single updates and 52% for multiple updates with about 1% of the attacks updated.



# Selected References



Phan Minh Dung.

On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.  
*Artif. Intell.*, 77(2):321–358, 1995.



Bei Shui Liao, Li Jin, Robert C. Koons.

Dynamics of argumentation systems: A division-based method.  
*Artif. Intell.*, 175(11), 1790–1814, (2011).



Baroni, P., Giacomin, M., Liao, B.

On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: A division-based method.  
*Artificial Intelligence* 212, 104–115 (2014)



Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly.

CoQuiAAS: A constraint-based quick abstract argumentation solver.  
In *ICTAI*, pages 928–935, 2015.



Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran.

Complexity-sensitive decision procedures for abstract argumentation.  
*AI*, 206:53–78, 2014.



Sarah Alice Gaggl and Norbert Manthey.

ASPARTIX-D ready for the competition, 2015.



Stefano Bistarelli, Fabio Rossi, and Francesco Santini.

ConArg: A tool for classical and weighted argumentation.  
In *COMMA*, 2016.



Sergio Greco, Francesco Parisi.

Efficient Computation of Deterministic Extensions for Dynamic Abstract Argumentation Frameworks.  
In *ECAI*, 2016.