Computing Extensions of Dynamic Abstract Argumentation Frameworks with Second-Order Attacks

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

22nd International Database Engineering & Applications Symposium

June 18-20, 2018

Villa San Giovanni, Italy

Argumentation in AI

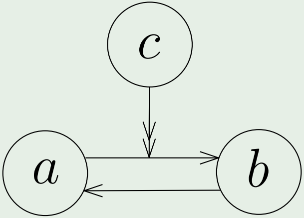
- A general way for representing arguments and relationships between them
- It allows representing dialogues, making decisions, and handling inconsistency and uncertainty
- **Extended Abstract Argumentation Framework** (EAF)

Example (a simple EAF)

a = The week-end will be dry in Rome since AccuWeather forecasts sunshine

b = The week-end will be wet in Rome since The Weather Channel forecasts rain

c = The Weather Channel is more trustworthy than AccuWeather



Semantics for Extended Argumentation Frameworks: "reasonable" sets of arguments, called *extensions*. We focus on preferred and stable semantics.

Argumentation in AI

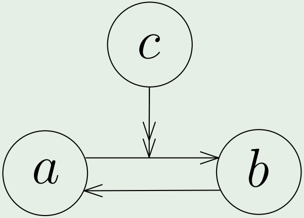
- A general way for representing arguments and relationships between them
- It allows representing dialogues, making decisions, and handling inconsistency and uncertainty
- **Extended Abstract Argumentation Framework** (EAF)

Example (a simple EAF)

a = The week-end will be dry in Rome since AccuWeather forecasts sunshine

b = The week-end will be wet in Rome since The Weather Channel forecasts rain

c = The Weather Channel is more trustworthy than AccuWeather



Semantics for Extended Argumentation Frameworks: “reasonable” sets of arguments, called *extensions*. We focus on preferred and stable semantics.

Dynamic Argumentation Frameworks

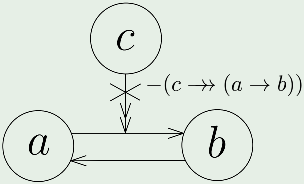
Many argumentation frameworks are highly dynamic in practice.

Example (a simple EAF)

a = The week-end will be dry in Rome since AccuWeather forecasts sunshine

b = The week-end will be wet in Rome since The Weather Channel forecasts rain

c = The Weather Channel is more trustworthy than AccuWeather



Should we recompute the semantics from scratch?

Contributions

1) We identify early-termination conditions.
2) Following the meta-argumentation approach proposed in [BoellaGTV10], we define a reduction of the problem of determining an extension of an updated EAF to that of determining an extension of a corresponding updated Dung's argumentation framework.
3) We define an incremental algorithm for computing extensions of dynamic EAFs by leveraging on the incremental technique proposed in [Alfano,Greco,Parisi IJCAI 2017].
4) Experimental analysis showing that our incremental approach for EAFs outperforms by two orders of magnitude the computation from scratch, where the fastest solvers from the last edition of the ICCMA are used.

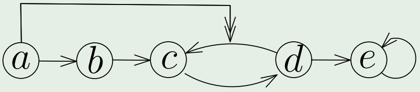
Outline

Extended Abstract Argumentation Frameworks

- An *Extended Argumentation Framework* (*EAF* for short) is a triple $\langle A, \Sigma, \Delta \rangle$, where
  - $A \subseteq Arg$ is a (finite) set whose elements are referred to as *arguments*,
  - $\Sigma \subseteq A \times A$ is a binary relation over A whose elements are called *attacks*,
  - Δ is a binary relation over $A \times \Sigma$ whose elements are called *second-order attacks*, and
- A Dung's argumentation framework (AF) [Dung 1995] is an EAF of the form $\langle A, \Sigma, \emptyset \rangle$.

Example (EAF)

$A = \{a, b, c, d, e\}$
$\Sigma = \{(a, b), (b, c), (c, d), (d, c), (d, e), (e, e)\}$
$\Delta = \{(a, (d, c))\}$



Semantics for Extended Abstract Argumentation(1/2)

A semantics identifies “reasonable” sets of arguments, called *extensions*.

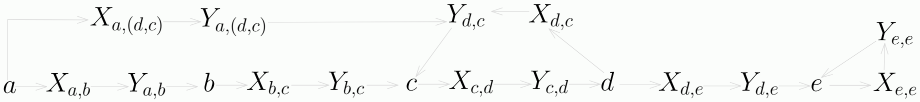
The semantics of EAFs can be given in terms of meta-argumentation frameworks (i.e., Dung’s AFs) where additional (meta-)arguments and attacks are considered to model second-order attacks.

Definition (Meta-AF)

The meta-AF for $\mathcal{EA} = \langle A, \Sigma, \Delta \rangle$ is $\mathcal{M} = \langle A^m, \Sigma^m \rangle$ where:

- $A^m = A \cup \{X_{a,b}, Y_{a,b} \mid (a,b) \in \Sigma\} \cup \{X_{a,(b,c)}, Y_{a,(b,c)} \mid (a,(b,c)) \in \Delta\}$
- $\Sigma^m = \{(a, X_{a,b}), (X_{a,b}, Y_{a,b}), (Y_{a,b}, b) \mid (a,b) \in \Sigma\} \cup \{(a, X_{a,(b,c)}), (X_{a,(b,c)}, Y_{a,(b,c)}), (Y_{a,(b,c)}, Y_{b,c}) \mid (a,(b,c)) \in \Delta\}$

Example (Meta-AF of our running example)



Semantics for Extended Abstract Argumentation(1/2)

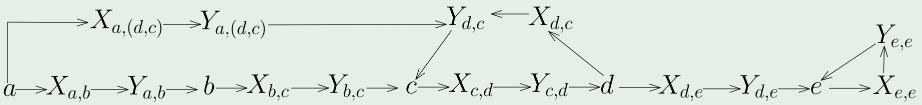
A semantics identifies “reasonable” sets of arguments, called *extensions*. The semantics of EAFs can be given in terms of meta-argumentation frameworks (i.e., Dung’s AFs) where additional (meta-)arguments and attacks are considered to model second-order attacks.

Definition (Meta-AF)

The meta-AF for $\mathcal{EA} = \langle A, \Sigma, \Delta \rangle$ is $\mathcal{M} = \langle A^m, \Sigma^m \rangle$ where:

- $A^m = A \cup \{X_{a,b}, Y_{a,b} \mid (a,b) \in \Sigma\} \cup \{X_{a,(b,c)}, Y_{a,(b,c)} \mid (a,(b,c)) \in \Delta\}$
- $\Sigma^m = \{(a, X_{a,b}), (X_{a,b}, Y_{a,b}), (Y_{a,b}, b) \mid (a,b) \in \Sigma\} \cup$ $\{(a, X_{a,(b,c)}), (X_{a,(b,c)}, Y_{a,(b,c)}), (Y_{a,(b,c)}, Y_{b,c}) \mid (a,(b,c)) \in \Delta\}$

Example (Meta-AF of our running example)

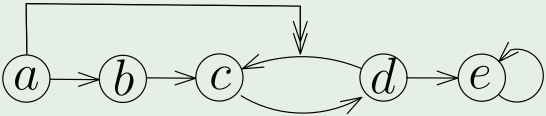


Semantics for Extended Abstract Argumentation(2/2)

A semantics identifies “reasonable” sets of arguments, called *extensions*.
The semantics of EAFs can be given in terms of meta-argumentation frameworks (i.e., Dung’s AFs) where additional (meta-)arguments and attacks are considered to model second-order attacks.

Example

preferred extensions: $\{\{a, c\}\}$ *stable extension*: $\{\{a, c\}\}$

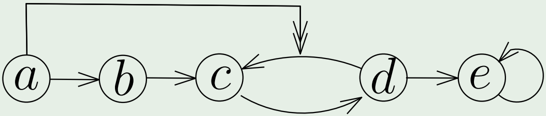


Semantics for Extended Abstract Argumentation(2/2)

A semantics identifies “reasonable” sets of arguments, called *extensions*.
The semantics of EAFs can be given in terms of meta-argumentation frameworks (i.e., Dung’s AFs) where additional (meta-)arguments and attacks are considered to model second-order attacks.

Example

preferred extensions: $\{\{a, c\}\}$ *stable extension*: $\{\{a, c\}\}$

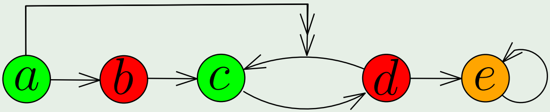


Extensions and labellings

- Semantics can be also defined in terms of *labelling*.
- Function $L : A \to \{\text{IN}, \text{OUT}, \text{UN}\}$ assigns a label to each argument
  - $L(a) = \text{IN}$ means a is accepted
  - $L(a) = \text{OUT}$ means a is rejected
  - $L(a) = \text{UN}$ means that a is undecided

Example (Preferred extension and labelling)

Preferred extension: $\{a, c\}$

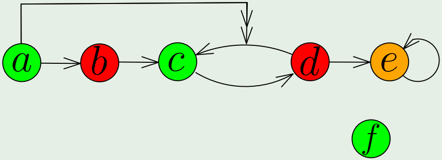


Preferred labelling:
$\{a, c\}$ are IN (green nodes)
$\{b, d\}$ are OUT (red nodes)
$\{e\}$ are UN (orange nodes)

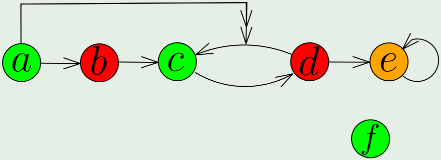
Updates

- An *update u* for an EAF $\mathcal{EA}_0$ allows us to change $\mathcal{EA}_0$ into an EAF $\mathcal{EA}$ by adding or removing an argument, an attack, or a second-order attack.
- If E_0 is an extension for $\mathcal{EA}_0$ and $\mathcal{EA}$ is obtained by adding (resp. removing) the set S of isolated arguments, then $E = E_0 \cup S$ (resp. $E = E_0 \setminus S$)
- We focus on the addition (+) and deletion (−) of an attack ($a \rightarrow b$) or a second-order attack ($a \twoheadrightarrow (b \rightarrow c)$).
- $u(\mathcal{EA}_0)$ denotes the application of update $u = \pm(a \rightarrow b)$ *or* $\pm(a \twoheadrightarrow (b \rightarrow c))$ to $\mathcal{EA}_0$.

Example (Extensions/labellings after adding the isolated argument g)



preferred extension:
$\{a, c\} \cup \{f\}$

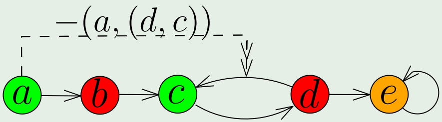


stable extension:
$\{a, c\} \cup \{f\}$

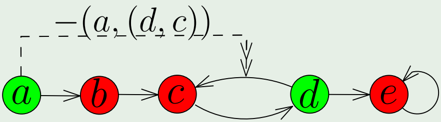
Updates

- An *update* u for an EAF $\mathcal{EA}_0$ allows us to change $\mathcal{EA}_0$ into an EAF $\mathcal{EA}$ by adding or removing an argument, an attack, or a second-order attack.
- If E_0 is an extension for $\mathcal{EA}_0$ and $\mathcal{EA}$ is obtained by adding (resp. removing) the set S of isolated arguments, then $E = E_0 \cup S$ (resp. $E = E_0 \setminus S$)
- We focus on the addition $(+)$ and deletion $(-)$ of an attack $(a \to b)$ or a second-order attack $(a \twoheadrightarrow (b \to c))$.
- $u(\mathcal{EA}_0)$ denotes the application of update $u = \pm(a \to b)$ *or* $\pm(a \twoheadrightarrow (b \to c))$ to $\mathcal{EA}_0$.

Example (Extensions/labellings after removing $-(a \twoheadrightarrow (d \to c))$)



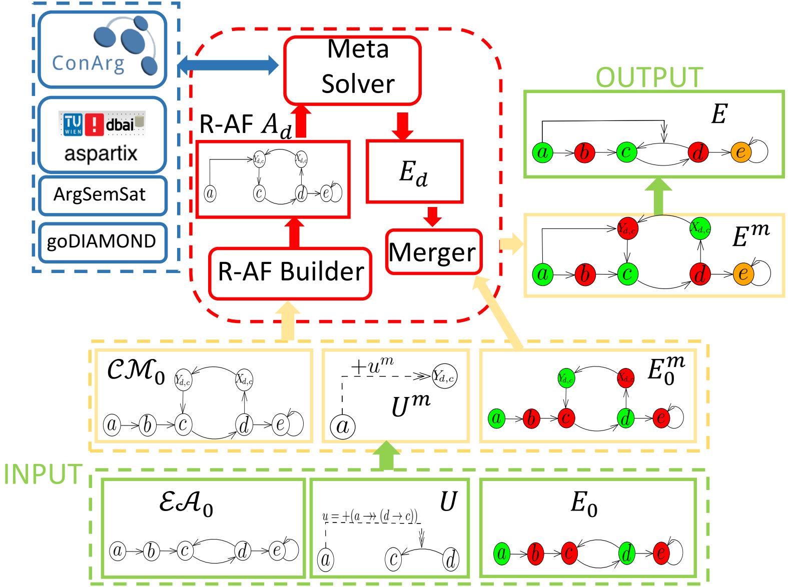
preferred extension: $\{\{a, c\}, \{a, d\}\}$



stable extension: $\{a, d\}$

Outline

Overview of the approach



Extension preservation for addition/deletion of an attack

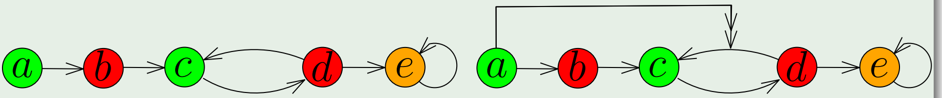
- Cases for which E_0 is still an extension of the updated EAF after a positive update.

update $+(a \to b)$		$L_0(b)$ IN	$L_0(b)$ UN	$L_0(b)$ OUT
$L_0(a)$	IN			pr, st
	UN			pr
	OUT	pr,st		pr,st

update $+(a \twoheadrightarrow (b \to c))$		$L_0(b)$ IN	$L_0(b)$ UN	$L_0(b)$ OUT
$L_0(a)$	IN			pr, st
	UN			pr
	OUT	pr,st		pr,st

Example (For $u = +(a \twoheadrightarrow (d \to c))$ the initial preferred extension $E_0 = \{a, c\}$ is preserved ($L_0(a) = \text{IN}$ and $L_0(d) = \text{OUT}$))

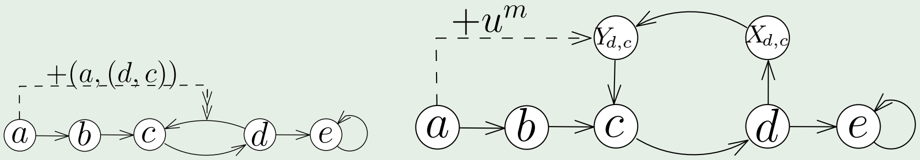
Preferred extension: $\{a, c\}$



The Compact Meta-Argumentation Framework

Our definition of meta-AF builds on that proposed in [BoellaGTV10] and considers additional meta-arguments that will allow us to simulate addition updates to be performed on EAF $\mathcal{EA}_0$ by means of updates performed on the corresponding meta-AF $\mathcal{CM}(\mathcal{EA}_0, u)$. In particular, the meta-AF contains meta-arguments $X_{c,d}$, $Y_{c,d}$ for encoding second-order attacks in Δ toward attacks $(c, d) \in \Sigma$.

Example (Compact Meta-AF $\mathcal{CM}_0$ for the BAF $\mathcal{EA}_0$ w.r.t. the update $u = +(a \twoheadrightarrow (d \rightarrow c))$.)

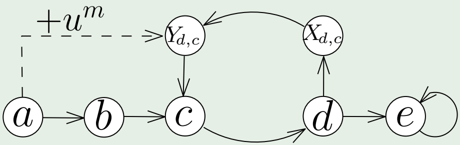


Updates for the Compact Meta-AF

Let $\mathcal{EA} = \langle A, \Sigma, \Delta \rangle$ be an EAF, and u an update for $\mathcal{EA}$. Update u^m for the meta-AF $\mathcal{CM}(\mathcal{EA}, u) = \langle A^m, \Sigma^m \rangle$ is as follows:

$$u^m = \begin{cases} +(c \to d) \;\; \textit{if } u = +(c \to d) \\ -(c \to d)) \;\; \textit{if } u = -(c \to d)) \\ +(e \to Y_{g,h}) \textit{ if } u = +(e \twoheadrightarrow (g \to h)) \\ -(e \to Y_{g,h}) \textit{ if } u = -(e \twoheadrightarrow (g \to h)) \end{cases}$$

Example (for $u = +(a \twoheadrightarrow (d \to c))$ is $u^m = +(a, Y_{d,c})$)



Incremental Algorithm

Algorithm Boost-EAF($\mathcal{EA}_0, u, E_0, \mathcal{S}, Solver_{\mathcal{S}}$)

Input: EAF $\mathcal{EA}_0 = \langle A_0, \Sigma_0\ \Delta_0 \rangle$,
update u of the form $u = \pm(a \to b)$ or $u = \pm(c \twoheadrightarrow (d \to e))$,
an initial $\mathcal{S}$-extension E_0 for $\mathcal{EA}_0$,
semantics $\mathcal{S} \in \{\mathtt{pr}, \mathtt{st}\}$,
function Solver$_{\mathcal{S}}(\mathcal{A})$ returning an $\mathcal{S}$-extension for AF $\mathcal{A}$ if it exists, $\bot$ otherwise;

Output: An $\mathcal{S}$-extension E for $u(\mathcal{EA}_0)$ if it exists, $\bot$ otherwise;

```
if checkProp(EA_0, u, E_0, S) then
    return E_0; // Extension preserved
Let M_0 = CM(EA_0, u) be the compact meta-AF for EA_0 w.r.t. u; // Build the compact meta-AF
Let u^m be the update for M_0 corresponding to u;
Let E_0^m be the initial S-extension for M_0 corresponding to E_0;
Let E^m = Incr-Alg(M_0, u^m, S, E_0^m, Solver_S); // Compute an S-extension for the meta-AF by calling Incr-Alg;
if (E^m ≠ ⊥) then
    return E = (E^m ∩ A_0); // The final extension will exclude meta arguments
else
    return ⊥; // A stable extension not always exists
```

Outline

Methodology

Datasets: Generated EAFs by starting from AFs used as benchmarks at ICCMA'17 for the tracks SE-$\texttt{pr}$ and SE-$\texttt{st}$. Specifically, we used :

- $B1$ consisting in AFs with : $|A| \in [2, 50K]$ and $|\Sigma| \in [1, 1.6M]$.
- $B2$ consisting in AFs with : $|A| \in [35, 200K]$ and $|\Sigma| \in [73, 4M]$.

Generated set of EAFs $\mathcal{EA}_0 = \langle A, \Sigma, \Delta \rangle$ from AF used as ICCMA'17 benchmarks, given a percentage $s \in \{0\%, 10\%, 20\%\}$ of second-order attacks as follows. We selected $s \times |\Sigma|$ attacks in Σ in a random way, and for each attack (x, y) selected, we added in Δ a second-order attack from a randomly selected argument in A to (x, y).

Methodology

The average run time of our Algorithm *Boost-EAF* to compute an $\mathcal{S}$-extension was compared with the average run time of *ArgSemSAT* if $\mathcal{S} = \texttt{pr}$ (*goDIAMOND* if $\mathcal{S} = \texttt{st}$) to compute an $\mathcal{S}$-extension for $u^m(\mathcal{CM}(\mathcal{EA}_0, u))$ from scratch.

Methodology

Datasets: Generated EAFs by starting from AFs used as benchmarks at ICCMA'17 for the tracks SE-$\mathtt{pr}$ and SE-$\mathtt{st}$. Specifically, we used :

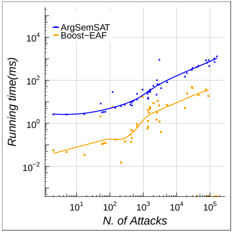
- $B1$ consisting in AFs with : $|A| \in [2, 50K]$ and $|\Sigma| \in [1, 1.6M]$.
- $B2$ consisting in AFs with : $|A| \in [35, 200K]$ and $|\Sigma| \in [73, 4M]$.

Generated set of EAFs $\mathcal{EA}_0 = \langle A, \Sigma, \Delta \rangle$ from AF used as ICCMA'17 benchmarks, given a percentage $s \in \{0\%, 10\%, 20\%\}$ of second-order attacks as follows. We selected $s \times |\Sigma|$ attacks in Σ in a random way, and for each attack (x, y) selected, we added in Δ a second-order attack from a randomly selected argument in A to (x, y).

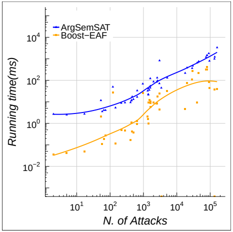
Methodology

The average run time of our Algorithm *Boost-EAF* to compute an $\mathcal{S}$-extension was compared with the average run time of *ArgSemSAT* if $\mathcal{S} = \mathtt{pr}$ (*goDIAMOND* if $\mathcal{S} = \mathtt{st}$) to compute an $\mathcal{S}$-extension for $u^m(\mathcal{CM}(\mathcal{EA}_0, u))$ from scratch.

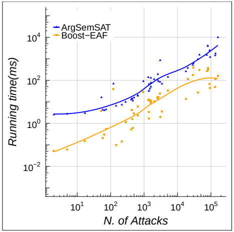
B1 Dataset



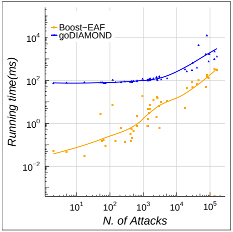
(a) $\mathcal{S} = \mathtt{pr}$, $s = 0\%$.



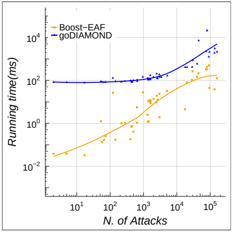
(b) $\mathcal{S} = \mathtt{pr}$, $s = 10\%$.



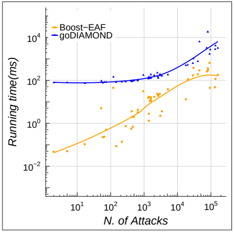
(c) $\mathcal{S} = \mathtt{pr}$, $s = 20\%$.



(d) $\mathcal{S} = \mathtt{st}$, $s = 0\%$.

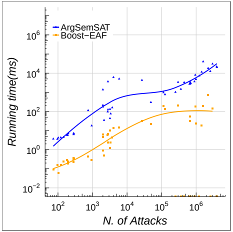


(e) $\mathcal{S} = \mathtt{st}$, $s = 10\%$.

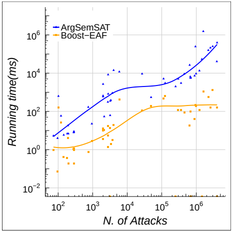


(f) $\mathcal{S} = \mathtt{st}$, $s = 20\%$.

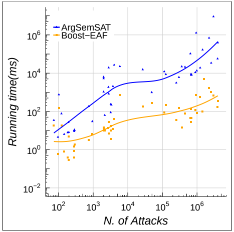
B2 Dataset



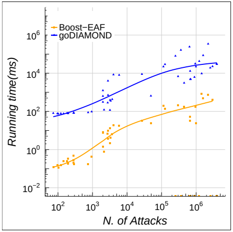
(a) $\mathcal{S} = \texttt{pr},\ s = 0\%$.



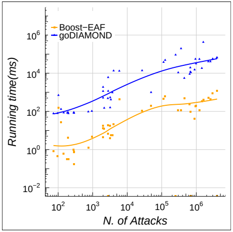
(b) $\mathcal{S} = \texttt{pr},\ s = 10\%$.



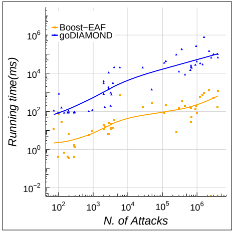
(c) $\mathcal{S} = \texttt{pr},\ s = 20\%$.



(d) $\mathcal{S} = \texttt{st},\ s = 0\%$.



(e) $\mathcal{S} = \texttt{st},\ s = 10\%$.



(f) $\mathcal{S} = \texttt{st},\ s = 20\%$.

Results

- The incremental algorithm outperforms the competitors that compute extensions from scratch by two orders of magnitude.
- The time saved by the incremental computation is higher for the dataset B2(s), where solvers takes much more time due to the complexer structures of the AFs in $B2$.
- Improvements obtained for the stable semantics are larger than preferred one due to different external solvers used.
- Improvements slightly decrease when increasing the percentage s of second-order attacks.
- However the incremental technique remains much faster than the computation from scratch in all cases.

Outline

Conclusions and Future Work

- We introduced a technique for the incremental computation of extensions of dynamic EAFs.
- We introduced a translation where updates and initial extensions of EAFs are taken into account.
- We exploited the incremental algorithm recently proposed in [Alfano,Greco,Parisi IJCAI 2017] and computed extensions of the meta-AFs, from which the updated extensions of EAFs are obtained.
- Experiments showed that our incremental technique is on average 100 times faster than the computation from scratch.

FW) We plan to investigate on extending our technique to deal with sets of updates performed simultaneously.

FW) Also, we plan to extend our technique to deal with other approaches that make use of meta-argumentation to deal with second-order attacks.

FW) Finally, we envisage the use of approaches based on incremental computation also in the context of *structured argumentation*.

Conclusions and Future Work

- We introduced a technique for the incremental computation of extensions of dynamic EAFs.
- We introduced a translation where updates and initial extensions of EAFs are taken into account.
- We exploited the incremental algorithm recently proposed in [Alfano,Greco,Parisi IJCAI 2017] and computed extensions of the meta-AFs, from which the updated extensions of EAFs are obtained.
- Experiments showed that our incremental technique is on average 100 times faster than the computation from scratch.

FW) We plan to investigate on extending our technique to deal with sets of updates performed simultaneously.

FW) Also, we plan to extend our technique to deal with other approaches that make use of meta-argumentation to deal with second-order attacks.

FW) Finally, we envisage the use of approaches based on incremental computation also in the context of *structured argumentation*.

Thank you!

... any ~~question~~ argument?

Selected References

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.

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.

Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran.
Complexity-sensitive decision procedures for abstract argumentation.
Artificial Intelligence, 206:53–78, 2014.

Guido Boella and Dov M. Gabbay and Leendert W. N. van der Torre and Serena Villata.
Support in Abstract Argumentation.
In *COMMA*, 2010, 111–122.

Gianvincenzo Alfano and Sergio Greco and Francesco Parisi.
Efficient Computation of Extensions for Dynamic Abstract Argumentation Frameworks: An Incremental Approach.
In *IJCAI*, pages 49–55, 2017.

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