On Scaling the Enumeration of the Preferred Extensions of Abstract Argumentation Frameworks

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

34th Annual ACM Symposium on Applied Computing

April 8-12, 2019

Limassol, Cyprus

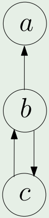
Argumentation in AI

- 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)

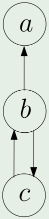


Computing preferred extensions is hard

- Several semantics have been proposed to identify “reasonable” sets of arguments, called *extensions*
- We focus on the *preferred* semantics, whose extensions are maximal sets of “acceptable” 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)



- The preferred extensions are $\{a, c\}$ and $\{b\}$ corresponding to the two “possible worlds”

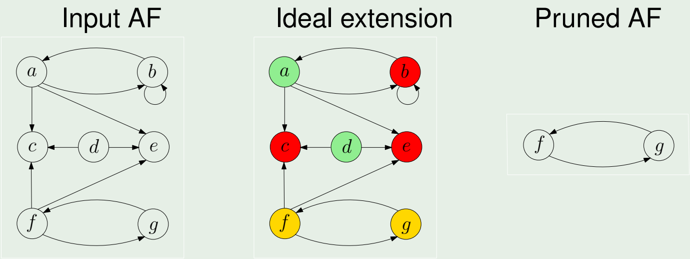
- However, enumerating preferred extensions (i.e., solving the ICCMA [1] EE-`pr` problem) is computationally **intractable**

[1] `http://argumentationcompetition.org`

Pruned AF & Algorithm (1/2)

- We show that the set of preferred extensions can be computed by looking only at a small part of the AF, called the *Pruned* AF
- The *Pruned* AF is obtained by "pruning" arguments whose status is entailed by the ideal extension of the input AF

Example (From the input AF to the Pruned AF)


Input AF

Ideal extension

Pruned AF

Pruned AF & Algorithm (2/2)

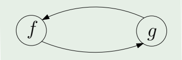
- We compute the preferred extensions of the Pruned AF, and
- then combine them with the ideal extension of the input AF to get the set of extensions of the input AF

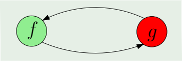
Example (From the extensions of the Pruned AF to those of the input AF)

Pruned AF

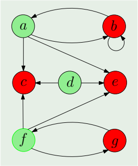
Extensions of the Pruned AF

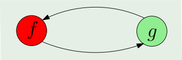
Extensions of the initial AF



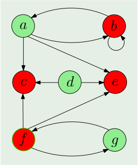


$\Longrightarrow$





$\Longrightarrow$



Pruned AF , Algorithm & Experiments

We propose an approach for scaling up the computation of the EE-$\mathtt{pr}$ problem, i.e, the problem of enumerating the preferred extensions of an AF

- We formally defined the *Pruned AF*, a smaller AF for local computation of the preferred extensions—it uses information provided by the ideal extension
- We introduce an efficient algorithm for computing all the preferred extensions, by focusing only on the Pruned AFs and incorporating state-of-the-art AF solvers
- We provide a thorough experimental analysis showing the effectiveness of our approach: two orders of magnitude faster than the solver that won the ICCMA'17 competition for the computational task EE-$\mathtt{pr}$

Outline

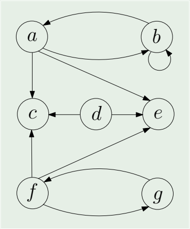
1. Introduction
   - Motivation
   - Contributions
2. Enumerating Preferred Extensions
   - Semantics of Abstract Argumentation Frameworks
   - Pruned AF
   - Algorithm
3. Implementation and Experiments
4. Conclusions and future work

Basic concepts: conflict-freeness and admissibility

- An *(abstract) argumentation framework* (*AF*) is a pair $\langle A, \Sigma \rangle$, where A is a set of *arguments* and $\Sigma \subseteq A \times A$ is a set of *attacks*.
- A set $S \subseteq A$ is *conflict-free* if there are no $a, b \in S$ such that *a attacks b*
- *S defends a* iff $\forall b \in A$ that *attacks a* there is $c \in S$ that *attacks b*
- S is *admissible* if it is conflict-free and it defends all its arguments.

Example (Admissible sets)

- $A = \{a, b, \ldots, g\}$
  $\Sigma = \{(a, b), (b, a), (b, b), \ldots\}$
- $\{a, d\}$ is conflict-free
- $\{a, d\}$ defends a since it attacks b (the attacker of a)
- $\{a, d\}$ defends d (d has no attacker)
- $\{a, d\}$ is admissible

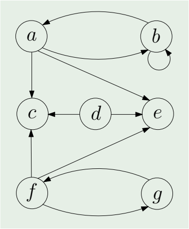


Basic concepts: conflict-freeness and admissibility

- An *(abstract) argumentation framework* (*AF*) is a pair $\langle A, \Sigma \rangle$, where A is a set of *arguments* and $\Sigma \subseteq A \times A$ is a set of *attacks*.
- A set $S \subseteq A$ is *conflict-free* if there are no $a, b \in S$ such that *a attacks b*
- *S defends a* iff $\forall b \in A$ that *attacks a* there is $c \in S$ that *attacks b*
- *S* is *admissible* if it is conflict-free and it defends all its arguments.

Example (Admissible sets)

- $A = \{a, b, \dots, g\}$
  $\Sigma = \{(a, b), (b, a), (b, b), \dots\}$
- $\{a, d\}$ is conflict-free
- $\{a, d\}$ defends *a* since it attacks *b* (the attacker of *a*)
- $\{a, d\}$ defends *d* (*d* has no attacker)
- $\{a, d\}$ is admissible



Preferred, grounded, and ideal semantics

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

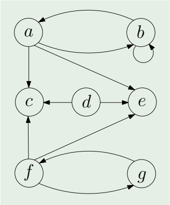
- A *complete extension* is an admissible set that contains all the arguments that it defends

A complete extension S is said to be:

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

Example (Preferred, ideal, and grounded semantics)

- Complete extensions: $\{d\}, \{a, d\}, \{d, f\}, \{d, g\}, \{a, d, f\}, \{a, d, g\}$
- The set of preferred extensions is $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}_0) = \{\{a, d, f\}, \{a, d, g\}\}$
- The grounded extension $E_{gr} = \{d\}$
- The ideal extension is $E_{id} = \{a, d\}$



For every preferred extension $E \in \mathcal{E}_{\mathrm{pr}}(\mathcal{A})$, it holds that $E_{gr} \subseteq E_{id} \subseteq E$

Preferred, grounded, and ideal semantics

A semantics identifies "reasonable" sets of arguments, called *extensions*

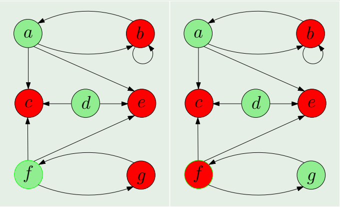
- A *complete extension* is an admissible set that contains all the arguments that it defends

A complete extension S is said to be:

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

Example (Preferred, ideal, and grounded semantics)

- Complete extensions: $\{d\}, \{a, d\}, \{d, f\}, \{d, g\}, \{a, d, f\}, \{a, d, g\}$
- The set of preferred extensions is $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}_0) = \{\{a, d, f\}, \{a, d, g\}\}$
- The grounded extension $E_{gr} = \{d\}$
- The ideal extension is $E_{id} = \{a, d\}$



For every preferred extension $E \in \mathcal{E}_{\mathrm{pr}}(\mathcal{A})$, it holds that $E_{gr} \subseteq E_{id} \subseteq E$

Preferred, grounded, and ideal semantics

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

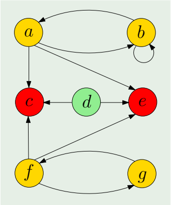
- A *complete extension* is an admissible set that contains all the arguments that it defends

A complete extension S is said to be:

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

Example (Preferred, ideal, and grounded semantics)

- Complete extensions: $\{d\}, \{a, d\}, \{d, f\}, \{d, g\}, \{a, d, f\}, \{a, d, g\}$
- The set of preferred extensions is $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}_0) = \{\{a, d, f\}, \{a, d, g\}\}$
- The grounded extension $E_{gr} = \{d\}$
- The ideal extension is $E_{id} = \{a, d\}$



For every preferred extension $E \in \mathcal{E}_{\mathrm{pr}}(\mathcal{A})$, it holds that $E_{gr} \subseteq E_{id} \subseteq E$

Preferred, grounded, and ideal semantics

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

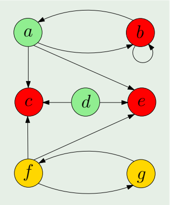
- A *complete extension* is an admissible set that contains all the arguments that it defends

A complete extension S is said to be:

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

Example (Preferred, ideal, and grounded semantics)

- Complete extensions: $\{d\}, \{a, d\}, \{d, f\}, \{d, g\}, \{a, d, f\}, \{a, d, g\}$
- The set of preferred extensions is $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}_0) = \{\{a, d, f\}, \{a, d, g\}\}$
- The grounded extension $E_{gr} = \{d\}$
- The ideal extension is $E_{id} = \{a, d\}$



For every preferred extension $E \in \mathcal{E}_{\mathrm{pr}}(\mathcal{A})$, it holds that $E_{gr} \subseteq E_{id} \subseteq E$

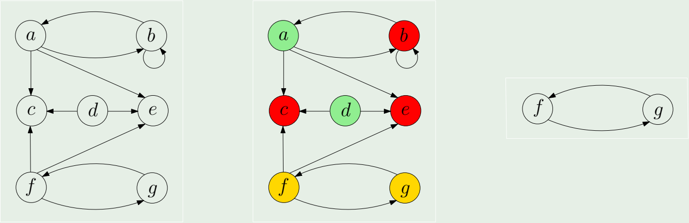
Definition of Pruned AF

The Pruned AF for $\mathcal{A}$, denoted as *Pruned*$(\mathcal{A})$, is obtained by removing from $\mathcal{A}$:

- all the arguments belonging to the ideal extension E_{id} of $\mathcal{A}$
- all the arguments in E_{id}^{+}, i.e., attacked by some argument in the ideal extension
- all the attacks towards or from the arguments in $E_{id} \cup E_{id}^{+}$

Example (From the input AF to the Pruned AF)

Input AF — Ideal extension — Pruned AF



How to use the Pruned AF

- Every preferred extension E of an AF $\mathcal{A}$ one-to-one corresponds to a preferred extension of the AF *Pruned*$(\mathcal{A})$
- A preferred extension of the whole AF can be obtained by joining a preferred extension of the Pruned AF with the ideal extension of $\mathcal{A}$

Theorem (Obtaining the preferred extensions by using the Pruned-AF)

Let $\mathcal{A} = \langle A, \Sigma \rangle$ be an AF, E_{id} the ideal extension for $\mathcal{A}$, and Pruned$(\mathcal{A}) = \langle A_p, \Sigma_p \rangle$ the Pruned AF for $\mathcal{A}$.
Then, $E \in \mathcal{E}_{\mathrm{pr}}(\mathcal{A})$ iff $E = E_{id} \cup E_p$ where $E_p \in \mathcal{E}_{\mathrm{pr}}(Pruned(\mathcal{A}))$.

Example

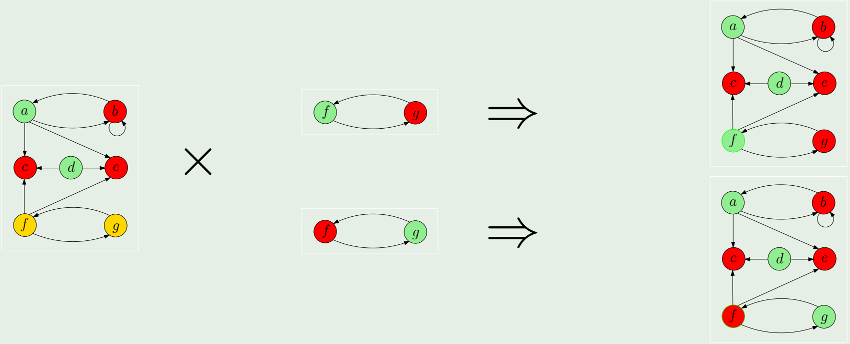
In our example, set of preferred extensions of the Pruned AF is $\mathcal{E}_{\mathrm{pr}}(Pruned(\mathcal{A})) = \{\{f\}, \{g\}\}$.
We obtain that the preferred extension of the whole AF as $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}) = \{\{a, d, f\}, \{a, d, g\}\} = \{\{f\} \cup E_{id}, \{g\} \cup E_{id}\}$,
where $E_{id} = \{a, d\}$.

How to use the Pruned AF

- Every preferred extension E of an AF $\mathcal{A}$ one-to-one corresponds to a preferred extension of the AF *Pruned*$(\mathcal{A})$
- A preferred extension of the whole AF can be obtained by joining a preferred extension of the Pruned AF with the ideal extension of $\mathcal{A}$

Example (From the extensions of the Pruned AF to those of the input AF)

Ideal extension Extensions of the Pruned AF Extensions of the initial AF



Algorithm for computing the set of preferred extensions

Algorithm ScaleEE($\mathcal{A}, k$)

```
Input: AF $\mathcal{A} = \langle A, \Sigma \rangle$,
       A percentage value $k$. // $k$ is used to decide if the Pruned AF should be used or not
Output: Set $\mathcal{E}_{\mathrm{pr}}(\mathcal{A})$ of preferred extensions of $\mathcal{A}$.
begin
  $E_{gr} =$ GR-Solver($\mathcal{A}$)  // Compute the grounded extension
  if $|E_{gr}| \geq k \cdot |A|$ then
      // If the grounded extension if “sufficiently large” then so is the ideal extension;
      thus compute the ideal extension and use it for pruning
      $E_{id} =$ ID-Solver($\mathcal{A}$)  // compute the ideal extension
      $\mathcal{A}_p = Pruned(\mathcal{A})$  // compute the Pruned AF (using $E_{id}$)
      $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}_p) =$ PR-Solver($\mathcal{A}_p$)  // compute the preferred extensions of the Pruned AF
      $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}) = \{E \mid E = E_{id} \cup E_p$, where $E_p \in \mathcal{E}_{\mathrm{pr}}(\mathcal{A}_p)\}$  // getting the output
  else
      $\mathcal{E}_{\mathrm{pr}}(\mathcal{A}) =$ PR-Solver($\mathcal{A}$) // Otherwise, directly compute the preferred extensions
  return $\mathcal{E}_{\mathrm{pr}}(\mathcal{A})$
end.
```

Theorem

Given an AF $\mathcal{A}$, if GR-Solver, ID-Solver, and PR-Solver are sound and complete, then ScaleEE computes the set of preferred extensions of $\mathcal{A}$.

Outline

1. Introduction
   - Motivation
   - Contributions
2. Enumerating Preferred Extensions
   - Semantics of Abstract Argumentation Frameworks
   - Pruned AF
   - Algorithm
3. Implementation and Experiments
4. Conclusions and future work

Competitor and external solvers used

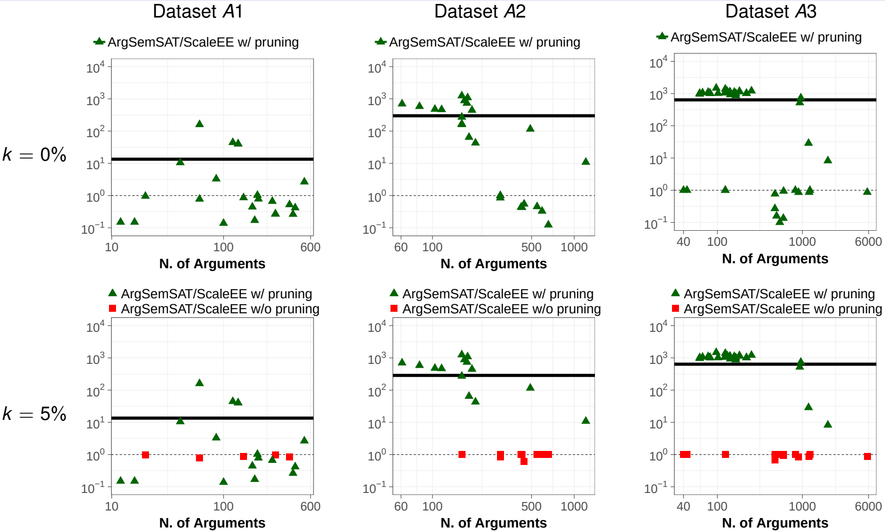
- We compared *ScaleEE* with *ArgSemSAT* [Cerutti et al. 2017]
- It is the winner the last ICCMA competition for the task EE-`pr` (i.e., computing all the preferred extensions of a given AF)
- We used the following external solvers:
  - GR-Solver: CoQuiAAS [Lagniez et al. 2015], the winner of ICCMA'17 track for computing the grounded extension
  - ID-Solver: *pyglaf* [Alviano 2017], the winner of ICCMA'17 track for computing the ideal extension
  - PR-Solver: *ArgSemSAT* for the direct computation when the Pruned AF is not used

Datasets

- We used benchmark AFs from the EE-`pr` track of ICCMA'17.
- AFs in the datasets named $A1$, $A2$, and $A3$ having more than one preferred extension
- Some statistics below

	Dataset		
	A1	A2	A3
Number of AFs	23	25	43
Min number of arguments	12	61	40
Max number of arguments	528	1.200	5.700
Min number of attacks	18	97	72
Max number of attacks	3.300	184.000	690.000
Average degree	4	21	22
Average density	0.04	0.05	0.04

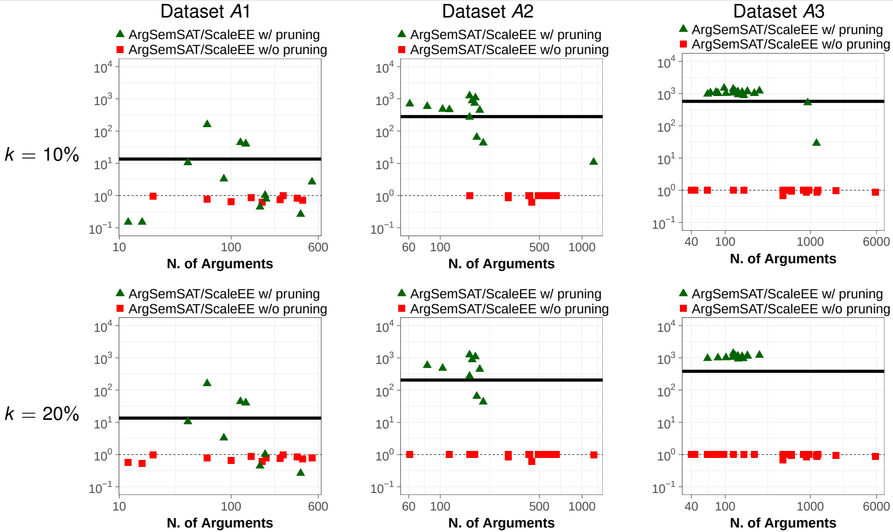
Improvement (i.e., run time of *ArgSemSAT* over that of *ScaleEE*) (1/2)



Triangular points (green): $|E_{gr}| \geq k \cdot |A|$, i.e., the Pruned AF is computed.

Squared points (red): $|E_{gr}| \ngeq k \cdot |A|$, the Pruned AF is not computed.

Improvement (i.e., run time of *ArgSemSAT* over that of *ScaleEE*) (2/2)



Triangular points (green): $|E_{gr}| \geq k \cdot |A|$, i.e., the Pruned AF is computed.

Squared points (red): $|E_{gr}| \not\geq k \cdot |A|$, the Pruned AF is not computed.

Results (1/2)

- *ScaleEE* is at least 10, 200, and 380 times faster than *ArgSemSAT* over the datasets *A*1, *A*2, and *A*3. Detailed improvements for different values of *k*:

	Dataset		
Percentage *k*	A1	A2	A3
0%	13.43	299	637.28
5%	13.51	286	637.35
10%	13.57	281	572
20%	13.52	205	384
Average degree	4	21	22

- The larger the average degree of the AFs, the bigger the (average) improvement obtained.
- For the datasets *A*2 and *A*3, the amount of time required decreases from dozens of minutes (direct computation) to a few seconds (our algorithm).
- The average improvement remains high for $k = 0\%$, that is, when computing both the ideal extension and the Pruned AF irrespectively of the size of the grounded extension.

Results (2/2)

- However, the number of AFs for which the improvement is too lower than 1 decreases if $k > 0\%$.
- Thus, using *k* greater than zero allows us to reduce the overhead due to the computation of the ideal extension and the Pruned AF.
- Using too high values of *k* deteriorates performances on average because the Pruned AF is not built even when it would be helpful.
- All in all, the best trade-off between paying the cost of computing the ideal extension along with the Pruned AF and risking to have the overhead of the computation of the ideal extension is choosing *k* greater than zero but no more than 10%.

Outline

1. Introduction
   - Motivation
   - Contributions
2. Enumerating Preferred Extensions
   - Semantics of Abstract Argumentation Frameworks
   - Pruned AF
   - Algorithm
3. Implementation and Experiments
4. Conclusions and future work

Conclusions and Future Work

- We introduced a technique for efficiently enumerating the preferred extensions of abstract argumentation frameworks.
- Our approach is modular with respect to the external solvers used
- We have experimentally investigated the behaviour of our technique
- We analysed the conditions under which computing the ideal extension (which is costly) is convenient for building the Pruned AF and then computing the preferred extensions using the Pruned AF.
- It is worth paying the cost of computing the ideal extension if is not empty—this can be easily checked by looking at the size of the grounded extension
- The computation of the preferred extensions over the Pruned AF yields significant improvements over the direct computation.
- Future work #1: applying the technique to other argumentation semantics
- Future work #2: considering dynamics, i.e., updates

Thank you!

... questions?

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.

Federico Cerutti, Massimiliano Giacomin, Mauro Vallati.
ArgSemSAT: Solving Argumentation Problems Using SAT.
Proceedings of International Conference on Computational Models of Argument (COMMA), 455–456, 2014

Mario Alviano.
The Pyglaf Argumentation Reasoner.
Proceedings of International Conference on Logic Programming (ICLP), 2:1–2:3, 2017.

Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly.
CoQuiAAS: A constraint-based quick abstract argumentation solver.
Proceeding of IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 928–935, 2015.