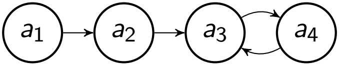
On Computing the Set of Acceptable Arguments in Abstract Argumentation

Matthias Thimm, Federico Cerutti, Mauro Vallati

Institute for Web Science and Technologies,
University of Koblenz-Landau

COMMA 2020



- E is *complete* iff
  1. for all $a, b \in E$ it is not the case that $a\mathrm{R}b$,
  2. for $a\mathrm{R}b$ and $b \in E$ then there is $c \in E$ and $c\mathrm{R}a$,
  3. for every argument a defended by E, $a \in E$.
- Acceptance of arguments:

$$\mathrm{Acc}^s(\mathrm{AF}) = \{a \in \mathrm{A} \mid a \text{ is in all complete } E\}$$
$$\mathrm{Acc}^c(\mathrm{AF}) = \{a \in \mathrm{A} \mid a \text{ is in some complete } E\}$$

For $x \in \{s, c\}$ we consider the following decision problem:

ACC^x	**Input**:	$\mathrm{AF} = (\mathrm{A}, \mathrm{R})$ and $E \subseteq \mathrm{A}$
	Output:	TRUE iff $E = \mathrm{Acc}^x(\mathrm{AF})$.

Complexity of Computing the Set of Acceptable Arguments

	ACC^s	ACC^c
complete semantics	P	DP-complete
grounded semantics	P	P
stable semantics	DP-complete	DP-complete
preferred semantics	DP2-complete	DP-complete

Question: How to compute ACC^c?

- There exist already algorithms for deciding $a \in \mathrm{ACC}^c$
  - For each a run such an algorithm, collect all positive instances (IAQ)
- There exist already algorithms to enumerate all complete extensions
  - Determine ACC^c by taking the union of all extensions (EEE)

Can we do better?

$\textsc{Sat}$ reductions

Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ be and abstract argumentation framework.

- For $a \in \mathsf{A}$ introduce variables $\mathtt{in}_a, \mathtt{out}_a, \mathtt{undec}_a$
- Define

$$\Phi_a = \left(\mathtt{out}_a \Leftrightarrow \bigvee_{bRa} \mathtt{in}_b\right) \wedge \left(\mathtt{in}_a \Leftrightarrow \bigwedge_{bRa} \mathtt{out}_b\right) \wedge (\mathtt{in}_a \dot{\vee} \mathtt{out}_a \dot{\vee} \mathtt{undec}_a)$$

and

$$\Psi_{\mathsf{AF}} = \bigwedge_{a \in \mathsf{A}} \Phi_a$$

Proposition (Besnard, Doutre; 2004)

1. *If* $\omega \in Mod(\Psi_{\mathsf{AF}})$ *then* $E(\omega)$ *is a complete extension of* AF*.*
2. *If* E *is a complete extension of* AF *then there is* $\omega \in Mod(\Psi_{\mathsf{AF}})$ *with* $E(\omega) = E$*.*
3. $a \in Acc^c(\mathsf{AF})$ *if and only if* $\Psi_{\mathsf{AF}} \wedge \mathtt{in}_a$ *is satisfiable.*

where $\mathrm{Mod}(\Phi)$ is the set of models of Φ and $E(\omega) = \{a \mid \omega(\mathtt{in}_a) = \textsc{true}\}$.

New algorithm SEE

Idea SEE (selective extension enumeration):

- Enumerate only those extensions that give new information and take their union
- $\textsc{Witness}(X)$ returns a model of X (or $\textsc{false}$)

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^{c}_{\mathsf{CO}}(\mathsf{AF})$

1: $S = \emptyset$
2: $D \leftarrow \mathsf{A}$
3: **while** $\textsc{false} \neq \omega = \textsc{Witness}(\Psi_{\mathsf{AF}} \wedge \bigvee_{a \in D} \mathtt{in}_a)$ **do**
4: $\quad S \leftarrow S \cup E(\omega)$
5: $\quad D \leftarrow D \setminus E(\omega)$
6: **return** S

New algorithm SEEM

Idea SEEM (selective extension enumeration via $\textsc{MaxSat}$*)*:

- Enumerate only those extensions that give *maximally* new information and take their union
- $\textsc{MaxSat}(S, H)$ returns a model of H that satisfies a maximum number of formulas from S (or $\textsc{false}$)

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^{c}_{\mathsf{CO}}(\mathsf{AF})$

1: $S = \emptyset$
2: $D \leftarrow \mathsf{A}$
3: **while** $\textsc{false} \neq \omega = \textsc{MaxSat}(\{\mathtt{in}_a \mid a \in D\}, \Psi_{\mathsf{AF}})$ **do**
4: $\quad S \leftarrow S \cup E(\omega)$
5: $\quad D \leftarrow D \setminus E(\omega)$
6: **return** S

Experiments: setup

- Implementations available in the TweetyProject[1]
- We used the Open-WBO MaxSat solver[2] for all calls of the form $\textsc{Sat}(\cdot)$, $\textsc{Witness}(\cdot)$, and $\textsc{MaxSat}(\cdot,\cdot)$.
- We considered the following sets of benchmarks:
  1. All ICCMA'15 benchmarks (N=192 instances)
  2. All ICCMA'17 benchmarks (N=1050)
  3. All ICCMA'19 benchmarks (N=326)
- Timeout 20 minutes
- Measured number of timeouts, runtime, and PAR10 score (=average runtime where runtime of timeouts is counted as 10*20min)

[1] http://tweetyproject.org/r/?r=acc_reasoner

[2] http://sat.inesc-id.pt/open-wbo/

Experiments: results

ICCMA'15

No.	Algorithm	N	#TO	RT	PAR10
1	SEE	192	58	19947.35	3728.89
2	EEE	192	72	27061.65	4640.95
3	SEEM	192	79	21247.51	5048.16
4	IAQ	192	149	20285.96	9418.16

ICCMA'17

No.	Algorithm	N	#TO	RT	PAR10
1	SEE	1050	558	60866.95	6435.11
2	SEEM	1050	579	55810.53	6670.29
3	IAQ	1050	742	54504.04	8531.91
4	EEE	1050	791	50607.98	9088.2

ICCMA'19

No.	Algorithm	N	#TO	RT	PAR10
1	SEE	326	81	9775.1	3011.58
2	SEEM	326	82	14574.94	3063.11
3	EEE	326	109	11717.29	4048.21
4	IAQ	326	130	20257.52	4847.42

Summary and Future Work

Summary:

- We considered the computational task of computing the set of acceptable arguments in abstract argumentation wrt. credulous and skeptical reasoning and grounded, complete, stable, and preferred semantics.
- We presented four different SAT-based algorithms for computing the set of credulously accepted arguments wrt. complete semantics and our evaluation showed that the two optimised algorithms significantly outperform the baseline algorithms.

Future work:

- Stable and preferred semantics (and maybe others)
- Skeptical reasoning

Thank you for your attention