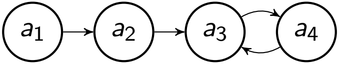
Revisiting SAT Techniques for Abstract Argumentation

Jonas Klein, Matthias Thimm

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

COMMA 2020

Overview



- Many existing solvers for abstract argumentation rely on reductions to satisfiability (SAT) problems
- They employ existing SAT solvers for solving these subproblems
- Different SAT solvers use different search strategies

$\rightarrow$ what is the impact of the choice of SAT solver?

Our contributions:

- miniAF: a lightweight and customizable AF solver based on SAT solving
- A general experimental evaluation on the impact of SAT solvers for solving abstract argumentation problems

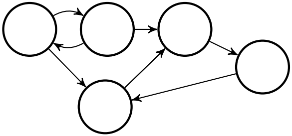
See also [Gning, Mailly; SAFA20] for a very similar work

Overview

Definition (Dung, 1995)

An *abstract argumentation framework* (AA) $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ is a directed graph with nodes A and directed edges $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$.

An extension E is a set $E \subseteq \mathsf{A}$ and is supposed to model a "plausible and jointly acceptable" set of arguments.



Definition

E is *admissible* iff

1. for all $a, b \in E$ it is not the case that $a\mathsf{R}b$,
2. for all $a \in E$, if $b\mathsf{R}a$ then there is $c \in E$ with $c\mathsf{R}b$

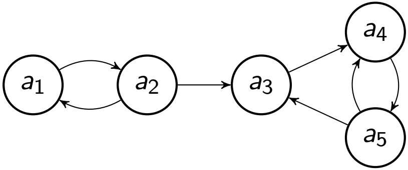
and it is *complete* if additionally

3. every argument c that is *defended* by E, belongs to E

Definition

- E is *grounded* if and only if E is minimal (wrt. set inclusion).
- E is *preferred* if and only if E is maximal (wrt. set inclusion).
- E is *stable* if and only if E attacks all arguments $\mathsf{A} \setminus E$.

Example



$E = \{a_1, a_5\}$ is admissible, complete, preferred, and stable.

$E' = \emptyset$ is admissible, complete, and grounded.

Overview

1. Abstract argumentation
2. The reduction-based approach via SAT
3. Experiments
4. Summary

Computational tasks

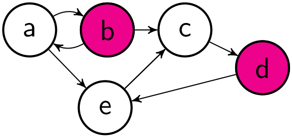
Let $\sigma \in \{\text{complete}, \text{preferred}, \ldots\}$

- Given $\mathrm{AF} = (\mathrm{A}, \mathrm{R})$, determine some σ-extension (**SE**)
- Given $\mathrm{AF} = (\mathrm{A}, \mathrm{R})$, determine all σ-extensions (**EE**)
- Given $\mathrm{AF} = (\mathrm{A}, \mathrm{R})$ and some $a \in \mathrm{A}$, decide whether a is contained in some σ-extension (**DC**)
- Given $\mathrm{AF} = (\mathrm{A}, \mathrm{R})$ and some $a \in \mathrm{A}$, decide whether a is contained in all σ-extension (**DS**)

Note on complexity (see e. g. [Dvořák, Dunne; 2018]):

- problems related to grounded semantics are tractable
- most other decision problems are on the first level of the polynomial hierarchy
- preferred-**DS** is even Π_2^P-complete

Task: Find some stable extension



Argumentation framework $\mathrm{AF} = (\mathrm{A}, \mathrm{R})$

Definition

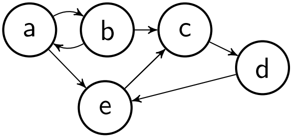
A stable extension $E \subseteq \mathrm{A}$ is conflict-free and attacks every $a \in \mathrm{A} \setminus E$.

Idea: model graph and semantics in propositional logic:

- Arguments: in_a, in_b, in_c, ...
- Conflict-freeness: $\bigwedge_{(X,Y)\in R} \neg(\mathrm{in}_X \wedge \mathrm{in}_Y)$
- Stability: $\bigwedge_{Y\in \mathrm{A}}(\neg \mathrm{in}_Y \Leftrightarrow \bigvee_{(X,Y)\in R} \mathrm{in}_X)$

Reduction to SAT 2/3

$$T_{\mathrm{AF}} = \{\bigwedge_{(X,Y)\in R} \neg(\mathrm{in}_X \wedge \mathrm{in}_Y), \bigwedge_{Y\in \mathrm{A}}(\neg \mathrm{in}_Y \Leftrightarrow \bigvee_{(X,Y)\in R} \mathrm{in}_X)\}$$



Theorem (Besnard, Doutre; 2004)

AF *has a stable extension iff* T_{AF} *has a model; every stable extension of* AF *corresponds to a model of* T_{AF}*.*

Reduction approach to AF reasoning:

$$\boxed{\mathrm{AF}} \longrightarrow \boxed{T_{\mathrm{AF}}} \longrightarrow \boxed{\mathrm{CNF}} \longrightarrow \boxed{\text{SAT solver}} \longrightarrow \boxed{\text{Extract extension}}$$

- Similar encodings work for other semantics
- Can be used to solve tasks SE, EE, DS, DC
- Solvers: CoQuiAAS [Lagniez, Lonca, Mailly; 2019], ArgSemSAT [Cerutti et al.; 2017], μ-toksia [Niskanen, Järvisalo; 2019], . . .

Reduction to SAT 2/3

AF → T_{AF} → CNF → SAT solver → Extract extension

- 2 parameters:
  1. Encoding/encoding strategy
  2. SAT solver
- SAT instances of abstract argumentation problems have a very similar structure
- Different search strategies of different SAT solvers behave differently on
  - different domains,
  - different instance structures
  - different hardness categories

→ what is the impact of the choice of SAT solver?

→ what is the "best" search strategy for abstract argumentation?

miniAF

- Lightweight SAT-based argumentation solver written in C
  `https://github.com/jklein94/miniAF`
- parameterisable with any SAT solver following the interface of the SAT competition
- supports
  - grounded semantics
  - complete semantics
  - preferred semantics
  - stable semantics

  and tasks EE, SE, DC, DS

Overview

Setup

Aim:

- Compare performance of miniAF using different SAT solvers

Details:

- tasks EE, SE, DC, DS, all four semantics
- ICCMA'17 benchmark data set and setting
- 12 SAT solvers (default and non-parallel configuration):
  - CaDiCal
  - Glucose
  - MapleLCMDistChronoBT-DL-v2.1
  - MapleLCMDistChronoBT-DL-v2.2
  - MapleLCMDistChronoBT-DL-v3
  - MapleLCMdistCBTcoreFirst
  - MergeSAT
  - PADC_Maple_LCM_Dist
  - PSIDS_MapleLCMDistChronoBT
  - PicoSAT
  - Relaxed_LCMDistChronoBT
  - optsat

	CO							
SAT	EE		SE		DS		DC	
	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.
CaDiCal	3009.22	50.29	**35.83**	**99.43**	**41.75**	**99.33**	882.95	85.43
Glucose	3084.61	49.14	131.47	98.29	197.61	97.33	857.38	86.00
MapleLCMDistChronoBT-DL-v2.1	2974.07	51.14	229.29	96.57	230.37	96.67	678.69	89.14
MapleLCMDistChronoBT-DL-v2.2	2988.36	50.86	244.73	96.29	230.38	96.67	679.64	89.14
MapleLCMDistChronoBT-DL-v3	3022.13	50.29	245.54	96.29	249.69	96.33	822.01	86.57
MapleLCMdistCBTcoreFirst	2968.79	51.14	269.04	96.00	330.52	95.00	693.15	88.86
MergeSAT	2965.29	51.14	197.32	97.14	230.39	96.67	709.02	88.57
PADC_Maple_LCM_Dist	**2947.17**	**51.43**	246.77	96.29	230.77	96.67	**660.84**	**89.43**
PSIDS_MapleLCMDistChronoBT	2950.45	**51.43**	228.95	96.57	248.92	96.33	677.13	89.14
PicoSAT	3212.29	46.86	35.90	**99.43**	41.90	**99.33**	934.61	84.57
Relaxed_LCMDistChronoBT	3121.88	48.86	213.51	96.86	230.26	96.67	819.41	86.86
optsat	3183.14	47.43	258.05	97.14	306.84	96.67	757.36	87.71

Too Hard								
SAT	**EE**		**SE**		**DS**		**DC**	
	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.
CaDiCal	6000.0	0.0	2949.47	51.0	2387.0	60.67	2555.19	57.67
Glucose	6000.0	0.0	3010.92	50.0	2627.89	56.67	2766.28	54.0
MapleLCMDistChronoBT-DL-v2.1	6000.0	0.0	2837.28	53.0	2498.8	58.67	2055.83	66.67
MapleLCMDistChronoBT-DL-v2.2	6000.0	0.0	2731.88	55.0	2464.64	59.33	2112.71	65.67
MapleLCMDistChronoBT-DL-v3	6000.0	0.0	2891.96	52.0	2653.66	56.0	2519.44	58.33
MapleLCMdistCBTcoreFirst	6000.0	0.0	2673.72	56.0	2538.8	58.0	2109.2	65.67
MergeSAT	6000.0	0.0	2901.41	52.0	2466.64	59.33	2126.66	65.33
PADC_Maple_LCM_Dist	6000.0	0.0	2719.46	55.0	2497.63	58.67	**2014.59**	**67.33**
PSIDS_MapleLCMDistChronoBT	6000.0	0.0	2897.68	52.0	2581.88	57.33	2052.36	66.67
PicoSAT	6000.0	0.0	3066.47	49.0	3011.66	50.0	2800.98	53.33
Relaxed_LCMDistChronoBT	6000.0	0.0	**2538.58**	**58.0**	**2136.9**	**65.0**	2447.71	59.6
optsat	6000.0	0.0	2950.19	51.0	2447.11	60.0	2332.47	61.67

Insights

- Performance is generally comparable between hardness categories, but we observe some differences between categories and tasks
- CaDiCal is significantly better than all other solvers for stable semantics
- PicoSAT solver exhibit good results for the Very Easy, Easy and the DC instances of the Medium set, but is in the lower range for the Hard and Too Hard instances
- Relaxed_LCMDistChronoBT system tends to achieve higher scores for the Hard and Too Hard instances, but lower scores for the Very Easy, Easy and Medium sets

Overview

Contributions:

- miniAF: a lightweight and customizable AF solver based on SAT solving
- A general experimental evaluation on the impact of SAT solvers for solving abstract argumentation problems

Future work:

- Configuration options of SAT solvers
- Comparison with other AF solvers
- Portfolio solvers

Thank you for your attention