reducto – A Reduct-based Solver for Skeptical Preferred Reasoning

Lars Bengel
Artificial Intelligence Group
University of Hagen
Germany
lars.bengel@fernuni-hagen.de

Julian Sander
Artificial Intelligence Group
University of Hagen
Germany
julian.sander@fernuni-hagen.de

Matthias Thimm
Artificial Intelligence Group
University of Hagen
Germany
matthias.thimm@fernuni-hagen.de

***Abstract*—We present `reducto`, a SAT-based solver for reasoning problems in abstract argumentation. The solver is mainly built on standard SAT-encodings with some improvements. In particular, for skeptical reasoning wrt. preferred semantics it contributes a non-standard approach. `reducto` refrains from iterative maximisation and instead utilises a reduct-based characterisation of preferred semantics to efficiently find counterexamples for the skeptical acceptance of an argument.**

I. Background

An *(abstract) argumentation framework* (AF) is a tuple $F = (A, R)$ where A is a finite set of arguments and R is a relation $R \subseteq A \times A$ [1]. For two arguments $a, b \in A$, the relation aRb means that a attacks b. For a set $S \subseteq A$ we define $S_F^+ = \{a \in A \mid \exists b \in S : bRa\}$. We say that a set $S \subseteq A$ is *conflict-free* iff for all $a, b \in S$ we do not have aRb. A set S *defends* an argument $b \in A$ iff for all a with aRb there is $c \in S$ with cRa. Furthermore, a set S is called *admissible* (ad) iff it is conflict-free and S defends all $a \in S$.

We define different semantics by imposing constrains on admissible sets [2]. In particular, an admissible set $S \subseteq A$ is called a

- *complete* (CO) extension iff for every $a \in A$, if S defends a then $a \in S$,
- *preferred* (PR) extension iff there exists no admissible S' with $S \subsetneq S'$,
- *stable* (ST) extension iff $S \cup S_F^+ = A$,
- *grounded* (GR) extension iff S is complete and there is no complete S' with $S' \subsetneq S$.

For a given argumentation framework $F = (A, R)$ and a semantics $\sigma \in \{\mathsf{CO}, \mathsf{PR}, \mathsf{ST}\}$, we denote with $\sigma(F)$ the set of σ-extensions of F. We consider the following reasoning problems for the above semantics [3]:

DC$-\sigma$	Given an argument $a \in A$, decide whether there exists some σ-extension $E \in \sigma(F)$ with $a \in E$,
DS$-\sigma$	Given an argument $a \in A$, decide whether a is contained in all σ-extensions of F,
SE$-\sigma$	Return a σ-extension of F.

II. System Overview

The `reducto` solver supports the following problems: DC-CO, DC-ST, DS-PR, DS-ST, SE-PR, SE-ST. In general, `reducto` uses the classical SAT-reduction technique to solve these problems [4]–[6]. Per default, `reducto` uses CaDiCaL 2.1.3 [7], but it can be used with any SAT-solver via the IPASIR interface. The source code is open source and available on Github[1].

For the problems DC-CO, DC-ST, DS-ST and SE-ST the solver uses mostly standard SAT-encodings and one-shot queries to the SAT-solver. For DS-PR `reducto` uses novel approaches built on a reduct-based characterisation of preferred semantics which we will outline in the following.

III. Reduct-based Approach to Skeptical Preferred Reasoning

The central notion behind our approach is the S-*reduct* [8].

Definition 1. *Let $F = (A, R)$ be an AF and $S \subseteq A$. We define the S-reduct of F as the AF $F^S = (A', R')$ with*

$$A' = A \setminus (S \cup S_F^+), \qquad R' = R \cap (A' \times A').$$

Essentially, the reduct allows us to remove the part of the AF F that is already "solved" by S. Based on this concept, the notion of *vacuous reduct semantics* has been introduced [9].

Definition 2. *Let σ, τ be argumentation semantics and $F = (A, R)$ is an AF. A set $S \subseteq A$ is a σ^τ-extension iff S is a σ-extension and it holds that $\tau(F^S) \subseteq \{\emptyset\}$.*

Intuitively, a set S is a σ^τ extension of F iff it is σ-extension of F and in the reduct F^S there exists no non-empty τ-extension. We denote with $\sigma^\tau(F)$ the set of all σ^τ-extensions of F. It has been shown, that the preferred semantics can be characterised as a vacuous reduct semantics [9], [10].

Theorem 1. *For any AF $F = (A, R)$. It holds that*

$$\mathit{PR}(F) = \mathit{ad}^{\mathit{ad}}(F) = \mathit{CO}^{\mathit{CO}}(F).$$

That means, in order to verify whether a complete extension S is preferred, we construct the reduct F^S and verify that it has no non-empty complete set. Related to that is also the concept of *Modularization* of argumentation semantics [11] which is satisfied by both complete and preferred semantics.

Theorem 2. *Let $F = (A, R)$ be an AF and $\sigma \in \{\mathit{CO}, \mathit{PR}\}$. If $S \in \sigma(F)$ and $S' \in \sigma(F^S)$, then $S \cup S' \in \sigma(F)$.*

[1] `https://github.com/aig-hagen/reducto`

In our algorithm, we employ the above properties to simplify the argumentation framework during the computation and then, if necessary, to combine the partial results in the end to obtain the required witness for non-acceptance.

A. Algorithm

The algorithm employed by `reducto` for the DS-PR problem is shown in Algorithm 1. Before starting the main routine, we perform some pre-processing on the AF. That includes restricting the AF to the arguments "relevant" to the query a, similar to how it was outlined in [12]. We also explicitly compute the grounded extension of the AF (line 1), which can be done in polynomial time [3], and remove it from the AF (line 6).

We use a standard SAT-encoding for complete semantics, denoted as Ψ_F^{CO}, whose models correspond to complete extensions [13], [14]. Furthermore, we add a clause to ensure that any model is non-empty, defined as $\Psi_F^{ne} = \bigvee_{a \in A} \mathtt{in}_a$. We write $\textsc{Witness}(\Psi)$ for a call to the SAT-solver that returns the set $\{a \in A \mid \omega(\mathtt{in}_a) = \textsc{True}\}$, where ω is a model of Ψ, if Ψ is satisfiable, otherwise it returns FALSE.

Essentially, our algorithm iterates over the non-empty complete extensions of F that do not include the query. If such an extension S attacks the query it represents a witness for its non-acceptance. Otherwise, we consider the reduct F^S and verify with one additional SAT-call whether there is a non-empty complete extension in F^S. In case there is none, we have found a witness for non-acceptance, otherwise we add a complement clause to the encoding and ask the SAT-solver for another complete extension of F. This complement clause is defined as $\mathcal{C}_F(S) = \bigvee_{a \in A \setminus S} \mathtt{in}_a$ for some set S. This clause, for each found complete extension S, ensures not only that the SAT-solver does not find S again, but also that any S' with $S' \subseteq S$ is no valid witness in any following SAT-call.

In most cases of non-acceptance, `reducto` will return an admissible set S that attacks the query argument as a witness. Note that this implies immediately that there exists a preferred extension S' such that $S \subseteq S'$ with $a \in S_F^+$. Thus such an admissible set is sufficient as a witness for the non-acceptance of a. In case such a set does not exist the solver simply returns a preferred extension S such that $a \notin S$.

Algorithm 1 Algorithm for DS-PR.

Input: $F = (A, R)$, $a \in A$
Output: $S \subseteq A$, otherwise YES

1: $S_{\mathsf{GR}} \leftarrow \textsc{Grounded}(F)$
2: **if** $a \in S_{\mathsf{GR}}$ **then**
3: **return** YES
4: **if** $a \in S^+_{\mathsf{GR},F}$ **then**
5: **return** S_{GR}
6: $F \leftarrow F^{S_{\mathsf{GR}}}$
7: $\Psi \leftarrow \Psi_F^{\mathsf{CO}} \wedge \Psi_F^{ne}$
8: $i \leftarrow 0$
9: **while** TRUE **do**
10: $S \leftarrow \textsc{Witness}(\Psi \wedge \neg \mathtt{in}_a)$
11: **if** $S = \textsc{False}$ **then**
12: **if** $i{+}{+} = 0$ **then**
13: **if** $\textsc{Witness}(\Psi \wedge \mathtt{in}_a)$ **then**
14: **return** YES
15: **else**
16: **return** S_{GR}
17: **return** YES
18: **if** $a \in S_F^+$ **then**
19: **return** $S_{\mathsf{GR}} \cup S$
20: $S' \leftarrow \textsc{Witness}(\Psi_{F^S}^{\mathsf{CO}} \wedge \Psi_{F^S}^{ne})$
21: **if** $S' = \textsc{False}$ **then**
22: **return** $S_{\mathsf{GR}} \cup S$
23: **if** $a \in S'^+_F$ **then**
24: **return** $S_{\mathsf{GR}} \cup S \cup S'$
25: **if** $a \in S'$ **then**
26: $\Psi \leftarrow \Psi \wedge \mathcal{C}_F(S \cup S')$
27: **else**
28: $\Psi \leftarrow \Psi \wedge \mathcal{C}_F(S)$

REFERENCES

[1] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artif. Intell.*, vol. 77, no. 2, pp. 321–358, 1995.

[2] P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre, Eds., *Handbook of Formal Argumentation*. College Publications, 2018.

[3] W. Dvorák and P. E. Dunne, "Computational problems in formal argumentation and their complexity," *Handbook of Formal Argumentation*, vol. 1, 2017.

[4] G. Charwat, W. Dvorák, S. A. Gaggl, J. P. Wallner, and S. Woltran, "Methods for solving reasoning problems in abstract argumentation - A survey," *Artif. Intell.*, vol. 220, pp. 28–63, 2015.

[5] F. Cerutti, S. A. Gaggl, M. Thimm, and J. P. Wallner, "Foundations of implementations for formal argumentation," *Handbook of Formal Argumentation*, vol. 1, 2017.

[6] A. Niskanen and M. Järvisalo, "μ-toksia: An efficient abstract argumentation reasoner," in *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020*, D. Calvanese, E. Erdem, and M. Thielscher, Eds., 2020, pp. 800–804.

[7] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, and F. Pollitt, "Cadical 2.0," in *Computer Aided Verification - 36th International Conference, CAV 2024*, ser. Lecture Notes in Computer Science, A. Gurfinkel and V. Ganesh, Eds., vol. 14681. Springer, 2024, pp. 133–152.

[8] R. Baumann, G. Brewka, and M. Ulbricht, "Revisiting the foundations of abstract argumentation - semantics based on weak admissibility and weak defense," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. AAAI Press, 2020, pp. 2742–2749.

[9] M. Thimm, "On undisputed sets in abstract argumentation," in *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*, B. Williams, Y. Chen, and J. Neville, Eds. AAAI Press, 2023, pp. 6550–6557.

[10] L. Blümel and M. Thimm, "Revisiting vacuous reduct semantics for abstract argumentation," in *ECAI 2024 - 27th European Conference on Artificial Intelligence, 2024*. IOS Press, 2024, pp. 3517–3524.

[11] R. Baumann, G. Brewka, and M. Ulbricht, "Shedding new light on the foundations of abstract argumentation: Modularization and weak admissibility," *Artif. Intell.*, vol. 310, p. 103742, 2022.

[12] B. Liao and H. Huang, "Partial semantics of argumentation: basic properties and empirical," *J. Log. Comput.*, vol. 23, no. 3, pp. 541–562, 2013.

[13] P. Besnard and S. Doutre, "Checking the acceptability of a set of arguments," in *10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, J. P. Delgrande and T. Schaub, Eds., 2004, pp. 59–64.

[14] F. Cerutti, M. Giacomin, and M. Vallati, "How we designed winning algorithms for abstract argumentation and which insight we attained," *Artif. Intell.*, vol. 276, pp. 1–40, 2019.