# A Reduct-based Approach to Skeptical Preferred Reasoning in Abstract Argumentation

**Lars Bengel, Julian Sander, Matthias Thimm**

Artificial Intelligence Group, University of Hagen, Germany

{lars.bengel, julian.sander, matthias.thimm}@fernuni-hagen.de

### Abstract

We consider abstract argumentation frameworks and, in particular, the problem of skeptical reasoning wrt. preferred semantics, i. e., deciding whether a given argument is contained in every preferred extension of the argumentation framework. We introduce a novel SAT-based approach, building on recent results from the literature, that searches through complete extensions to efficiently decide this problem. It also employs effective simplification procedures to shorten computation times. As our experimental evaluation shows, our algorithm significantly outperforms state-of-the-art approaches.

## 1 Introduction

Formal argumentation is a research field within the area of knowledge representation and reasoning that offers a great variety of formalisms (Brewka, Polberg, and Woltran 2014), and the *(abstract) argumentation framework* (AF) introduced by Dung (1995) is a core area of research. In an AF, arguments are modelled as abstract entities and we consider directed attacks between them as the only relation. Reasoning in AFs is via acceptability semantics, which are functions that return sets of arguments, called *extensions*, considered jointly acceptable. A fundamental property of acceptable sets is admissibility, which requires that a set of arguments is conflict-free and also defends all of its members against attacks from the other arguments. For instance, the *preferred extensions* are then simply defined as the $\subseteq$-maximal admissible sets (Dung 1995). One can then define different reasoning problems based on these semantics (Dvorák and Dunne 2017). One of the most prominent ones is the problem of *skeptical reasoning* wrt. some semantics, i. e., deciding whether a given argument is contained in every extension wrt. the given semantics. In general, most of these reasoning problems are non-tractable (Dvorák and Dunne 2017). Especially because of that, algorithms to efficiently compute these problems are of great importance in order to apply argumentation in practice.

In this work, we introduce a SAT-based algorithm for solving the problem of skeptical reasoning wrt. preferred semantics. Our algorithm is built upon recent results on the characterisation of preferred semantics as a vacuous-reduct semantics (Thimm 2023). Essentially, our algorithm searches through complete extensions and looks for one that disproves the skeptical acceptance of the query argument by being *incompatible* with it. In contrast to existing work, our algorithm does not maximise each complete extension and instead searches for new complete extensions containing unvisited arguments. Our algorithm also employs effective preprocessing measures to simplify computation. Thus, our algorithm is able to solve the problem of skeptical preferred reasoning without having to actually compute the preferred extensions. Moreover, we show that our algorithm is sound and complete and our experiments show that it significantly outperforms current state-of-the-art solvers. To summarise, the contribution of this work is twofold:

- We introduce a novel algorithm for skeptical reasoning wrt. preferred semantics and show that it is sound and complete (Section 3),
- We implement our algorithm and evaluate it against state-of-the-art argumentation solvers (Section 4).

In Section 2 we introduce the necessary background on abstract argumentation and Section 5 concludes the paper. Omitted proofs, the underlying SAT-encoding and an extended evaluation can be found in the extended version (Bengel, Sander, and Thimm 2025)[1].

## 2 Preliminaries

We consider abstract argumentation (Dung 1995). The central notion is the *abstract argumentation framework* (AF), which is a tuple $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is a finite set of arguments and $\mathcal{R}$ is the attack relation $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$. For any two arguments $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}$, we say that $\boldsymbol{a}$ *attacks* $\boldsymbol{b}$ iff $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}$, sometimes also written as $\boldsymbol{a}\mathcal{R}\boldsymbol{b}$. For a set of arguments $S \subseteq \mathcal{A}$ we denote with $\mathcal{F}|_S = (S, \mathcal{R} \cap (S \times S))$ the *restriction* of $\mathcal{F}$ to $S$. For a set $S \subseteq \mathcal{A}$ we define the set of arguments attacked by $S$ (attacking $S$) in $\mathcal{F}$ respectively as $S^+_\mathcal{F} = \{\boldsymbol{a} \in \mathcal{A} \mid \exists \boldsymbol{b} \in S : \boldsymbol{b}\mathcal{R}\boldsymbol{a}\}$ and $S^-_\mathcal{F} = \{\boldsymbol{a} \in \mathcal{A} \mid \exists \boldsymbol{b} \in S : \boldsymbol{a}\mathcal{R}\boldsymbol{b}\}$. Moreover, we say that $S$ is *conflict-free* iff we have $S \cap S^+_\mathcal{F} = \emptyset$. The set $S$ *defends* an argument $\boldsymbol{a} \in \mathcal{A}$ iff for all $\boldsymbol{b} \in \{\boldsymbol{a}\}^-_\mathcal{F}$ there is some $\boldsymbol{c} \in S$ such that $\boldsymbol{c}\mathcal{R}\boldsymbol{b}$. Furthermore, $S$ is called *admissible* iff it is conflict-free and $S$ defends all $\boldsymbol{a} \in S$, i. e., we have that $S \cap S^+_\mathcal{F} = \emptyset$ and $S^-_\mathcal{F} \subseteq S^+_\mathcal{F}$.

Consequently, an admissible set $E \subseteq \mathcal{A}$ is called a *complete* ($\mathsf{CO}$) extension iff $E$ includes every argument $\boldsymbol{a} \in \mathcal{A}$

[1] https://doi.org/10.5281/zenodo.16022973

that it defends, and it is called a *preferred* (PR) extension iff there exists no admissible $E'$ with $E \subsetneq E'$. The *grounded* (GR) extension is then the $\subseteq$-minimal complete extension. For a given AF $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ and a semantics $\sigma \in \{\mathsf{CO}, \mathsf{PR}\}$, we denote with $\sigma(\mathcal{F})$ the set of $\sigma$-extensions of $\mathcal{F}$.

The focus of this work is the reasoning problem of skeptical acceptance wrt. some semantics $\sigma$ (Dvorák and Dunne 2017), defined as:

DS-$\sigma$ Given an argument $\boldsymbol{a} \in \mathcal{A}$, decide whether $\boldsymbol{a}$ is contained in all $\sigma$-extensions of $\mathcal{F}$.

The computational complexity of this problem has been well studied (Dvorák and Dunne 2017) and the prevalent strategy to solve it is a reduction to *satisfiability problems* (SAT) and using a dedicated SAT-solver for solving those, cf. (Cerutti et al. 2017). In particular, the problem of skeptical reasoning wrt. preferred semantics (DS-PR) is $\Sigma_2^P$-complete. Most importantly, that means it cannot be solved by a single SAT-call and it is one of the more difficult decision problems in abstract argumentation.

## 3 The Vacuous Reduct-based Approach to Skeptical Preferred Reasoning

In this section, we present our main contribution, a novel algorithm for skeptical reasoning wrt. preferred semantics. Our approach is built on the concept of *counterexample guided abstraction refinement* (CEGAR) (Clarke et al. 2003). This concept is widely used by argumentation solvers (Niskanen and Järvisalo 2020; Thimm, Cerutti, and Vallati 2021) and has been pioneered by the CEGARTIX system (Dvorák et al. 2012) for the domain of abstract argumentation. Given an AF $\mathcal{F}$ and a query argument $\boldsymbol{a}$, the general idea is to find complete extensions of sub-frameworks of $\mathcal{F}$ attacking the query argument $\boldsymbol{a}$. The general procedure of our algorithm consists of the following steps:

(1) Simplify the AF by removing irrelevant arguments and "resolving" the grounded extension,

(2) Iterate through complete extensions of the remaining AF in search of a counterexample for the skeptical acceptance of the query,

(3) Combine partial results of the previous steps to a proper counterexample.

During each step, we actively check whether the query argument is attacked by the current (partial) counterexample, which can allow us to terminate sooner.

**Simplifying the Argumentation Framework** The first step in our approach is simplifying the problem instance, without affecting the result, to accelerate the subsequent problem solving process. This kind of preprocessing is an important part of many problem solving paradigms, for instance SAT-solving (Biere, Järvisalo, and Kiesl 2021).

The first simplification step is based on the *Directionality* of argumentation semantics and has already been outlined by Liao and Huang (2013). For some AF $\mathcal{F} = (\mathcal{A}, \mathcal{R})$, we define the set of *unattacked sets* of $\mathcal{F}$ as $\mathsf{UA}(\mathcal{F}) = \{S \subseteq \mathcal{A} \mid \nexists \boldsymbol{a} \in (\mathcal{A} \setminus S) : \boldsymbol{a} \in S^-_{\mathcal{F}}\}$. Based on that, the *Directionality* principle has been defined (Baroni and Giacomin 2007).

**Principle 1.** Let $\sigma$ be a semantics. We say that $\sigma$ satisfies *Directionality* if and only if for all AFs $\mathcal{F}$ and every set $U \in \mathsf{UA}(\mathcal{F})$ it holds that $\sigma(\mathcal{F}|_U) = \{E \cap U \mid E \in \sigma(\mathcal{F})\}$.

Essentially, the above principle states that the computation of an extension for a semantics $\sigma$ should only depend on its attackers (and in turn on their attackers and so on). As has been shown by Baroni and Giacomin (2007), both the complete and preferred semantics satisfy *Directionality*.

**Proposition 1.** *Complete and preferred semantics satisfy* Directionality.

Now we can determine an unattacked set $U \in \mathsf{UA}(\mathcal{F})$ that contains the query argument $\boldsymbol{a}$ and restrict the AF to $U$ to simplify the computation without affecting the acceptance status of the query argument.

A simple but effective way to achieve this is to consider the arguments *relevant* for $\boldsymbol{a}$. For two arguments $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{A}$ we say that $\boldsymbol{b}$ is *relevant* for $\boldsymbol{a}$ iff there exists a directed path from $\boldsymbol{b}$ to $\boldsymbol{a}$. We then define the set of arguments *relevant* for $\boldsymbol{a}$ in $\mathcal{F}$ as follows.

$$\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a}) = \{\boldsymbol{a}\} \cup \{\boldsymbol{b} \in \mathcal{A} \mid \boldsymbol{b} \text{ is relevant for } \boldsymbol{a}\} \quad (1)$$

For convenience, we explicitly define that $\boldsymbol{a}$ is always relevant for itself. Notably, this notion of relevance has already been used in (Liao and Huang 2013), but also recently been defined in the context of acceptance explanations by Borg and Bex (2024). It is then easy to see that $\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})$ is an unattacked set of $\mathcal{F}$ for any argument $\boldsymbol{a}$.

**Corollary 1.** *For all AFs $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ and arguments $\boldsymbol{a} \in \mathcal{A}$ it holds that* $\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a}) \in \mathsf{UA}(\mathcal{F})$.

This allows us to restrict the input AF to just the arguments relevant for the query $\boldsymbol{a}$ before performing further calculations, which not only allow us to ignore arguments that are only attacked by the query, but also enables us to disregard unrelated components of the AF entirely.

The second simplification we perform is explicitly computing the grounded extension of the AF, which can be done in polynomial time (Dvorák and Dunne 2017). For that we utilise the simple iterative procedure for computing the grounded extension that has been outlined by (Dung 1995).

**Iterating Complete Extensions** As shown by Thimm, Cerutti, and Vallati (2021), it is not necessary to explicitly consider preferred extensions to solve the DS-PR problem. Instead, we will primarily consider complete extensions.

The main notion underlying our algorithm is the *S-reduct* introduced by Baumann, Brewka, and Ulbricht (2020).

**Definition 1.** Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an AF and $S \subseteq \mathcal{A}$. We define the *S-reduct* of $\mathcal{F}$ as the AF $\mathcal{F}|_{\mathcal{A} \setminus (S \cup S^+_{\mathcal{F}})}$.

Essentially, the reduct allows us to remove the part of the AF $\mathcal{F}$ that is already "resolved" by $S$. Based on this concept, the notion of *vacuous reduct semantics* has been introduced (Thimm 2023).

**Definition 2.** Let $\sigma$ be a semantics and $\mathcal{F}$ is an AF. We say that $\mathcal{F}$ is *$\sigma$-vacuous* iff $\sigma(\mathcal{F}) \subseteq \{\emptyset\}$.

**Definition 3.** Let $\sigma, \tau$ be argumentation semantics and $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ is an AF. A set $S \subseteq \mathcal{A}$ is a $\sigma^\tau$-extension iff $S$ is a $\sigma$-extension and it holds that $\mathcal{F}^S$ is $\tau$-vacuous.

A set $S$ is a $\sigma^\tau$ extension of $\mathcal{F}$ iff it is $\sigma$-extension of $\mathcal{F}$ and in the reduct $\mathcal{F}^S$ there exists no non-empty $\tau$-extension. Denote with $\sigma^\tau(\mathcal{F})$ the set of all $\sigma^\tau$-extensions of $\mathcal{F}$.

Of particular interest to us, is the fact that the preferred semantics can be characterised as a vacuous reduct semantics, as shown by (Thimm 2023; Blümel and Thimm 2024).

**Proposition 2.** *For any AF $\mathcal{F} = (\mathcal{A}, \mathcal{R})$. It holds that*

$$\mathsf{PR}(\mathcal{F}) = \mathsf{CO}^{\mathsf{CO}}(\mathcal{F}).$$

We will utilise this in our algorithm to verify whether some complete extension $E$ is preferred by checking whether the reduct $\mathcal{F}^E$ is $\mathsf{CO}$-vacuous.

**Constructing the Counterexample** During the algorithm we construct different complete extensions in different restrictions and reducts of the original AF. The concept of *Modularisation* of argumentation semantics (Baumann, Brewka, and Ulbricht 2022) allows us to combine them to construct a proper counterexample for skeptical acceptance.

**Principle 2.** Let $\sigma$ be a semantics. We say that $\sigma$ satisfies *Modularisation* if and only if for all AFs $\mathcal{F}$ it holds that, if $E_1 \in \sigma(\mathcal{F})$ and $E_2 \in \sigma(\mathcal{F}^{E_1})$ then $E_1 \cup E_2 \in \sigma(\mathcal{F})$.

*Modularisation* is satisfied by complete and preferred semantics (Baumann, Brewka, and Ulbricht 2022).

**Proposition 3.** *Complete and preferred semantics satisfy* Modularisation.

**Algorithm for Skeptical Preferred Reasoning** Our algorithm for skeptical preferred reasoning, like most state-of-the-art approaches, utilises a reduction to SAT. The atom $\mathtt{in}_a$ represents that argument $a$ is contained in the corresponding extension. We denote with $\Psi^{\mathsf{CO}}_{\mathcal{F}}$ the SAT-encoding of the complete semantics, cf. (Besnard and Doutre 2004; Cerutti, Giacomin, and Vallati 2019), for some AF $\mathcal{F}$, such that each model of $\Psi^{\mathsf{CO}}_{\mathcal{F}}$ corresponds to a complete extension of $\mathcal{F}$. We write WITNESS($\Psi$) for a call to the SAT-solver that returns a witness $E$ of $\Psi$, if $\Psi$ is satisfiable, otherwise it returns FALSE. SAT($\Psi$) simply returns TRUE iff $\Psi$ is satisfiable and FALSE otherwise. GROUNDED($\mathcal{F}$) denotes the iterative algorithm computing the grounded extension.

Our algorithm for deciding skeptical acceptance wrt. preferred semantics is shown in Algorithm 1. For the input $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ and some argument $\boldsymbol{a} \in \mathcal{A}$, the algorithm returns either YES iff $\boldsymbol{a}$ is skeptically accepted wrt. preferred semantics in $\mathcal{F}$, otherwise it returns a complete extension $E$ of $\mathcal{F}$ that serves as a witness for the non-acceptance of $\boldsymbol{a}$. In detail, the procedure of our algorithm works as follows:

(1) Restrict $\mathcal{F}$ to the arguments relevant for $\boldsymbol{a}$ and compute grounded extension $E_{\mathsf{GR}}$ of $\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})}$ (lines 1-2). If $\boldsymbol{a} \in E_{\mathsf{GR}}$ terminate with YES, if $\boldsymbol{a} \in E^+_{\mathsf{GR},\mathcal{F}}$ terminate with $E_{\mathsf{GR}}$. Otherwise move to $(\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})})^{E_{\mathsf{GR}}}$ (lines 3-7).

(2) If $(\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})})^{E_{\mathsf{GR}}}$ possesses no non-empty complete extensions at all, $E_{\mathsf{GR}}$ is a counterexample (lines 9-10).

**Algorithm 1** Algorithm for $\mathsf{DS\text{-}PR}$.

**Input:** $\mathcal{F} = (\mathcal{A}, \mathcal{R})$, $\boldsymbol{a} \in \mathcal{A}$
**Output:** $E \subseteq \mathcal{A}$, otherwise YES
1: $\mathcal{F} \leftarrow \mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})}$
2: $E_{\mathsf{GR}} \leftarrow$ GROUNDED($\mathcal{F}$)
3: **if** $\boldsymbol{a} \in E_{\mathsf{GR}}$ **then**
4:     **return** YES
5: **if** $\boldsymbol{a} \in E^+_{\mathsf{GR},\mathcal{F}}$ **then**
6:     **return** $E_{\mathsf{GR}}$
7: $\mathcal{F} \leftarrow \mathcal{F}^{E_{\mathsf{GR}}}$
8: $\Psi \leftarrow \Psi^{\mathsf{CO}}_{\mathcal{F}} \wedge \bigvee_{a \in \mathcal{A}} \mathtt{in}_{\boldsymbol{a}}$
9: **if** SAT($\Psi$) = FALSE **then**
10:     **return** $E_{\mathsf{GR}}$
11: **while** TRUE **do**
12:     $E \leftarrow$ WITNESS($\Psi \wedge \neg \mathtt{in}_a$)
13:     **if** $E$ = FALSE **then**
14:         **return** YES
15:     **if** $\boldsymbol{a} \in E^+_{\mathcal{F}}$ **then**
16:         **return** $E_{\mathsf{GR}} \cup E$
17:     $E' \leftarrow$ WITNESS($\Psi^{\mathsf{CO}}_{\mathcal{F}^E} \wedge \bigvee_{\boldsymbol{a} \in \mathcal{A}^E} \mathtt{in}_{\boldsymbol{a}}$)
18:     **if** $E'$ = FALSE **then**
19:         **return** $E_{\mathsf{GR}} \cup E$
20:     **if** $\boldsymbol{a} \in E'^+_{\mathcal{F}}$ **then**
21:         **return** $E_{\mathsf{GR}} \cup E \cup E'$
22:     **if** $\boldsymbol{a} \in E'$ **then**
23:         $\Psi \leftarrow \Psi \wedge \bigvee_{\boldsymbol{a} \in \mathcal{A} \setminus (E \cup E')} \mathtt{in}_{\boldsymbol{a}}$
24:     **else**
25:         $\Psi \leftarrow \Psi \wedge \bigvee_{\boldsymbol{a} \in \mathcal{A} \setminus E} \mathtt{in}_{\boldsymbol{a}}$

(3) Compute a non-empty complete extension $E$ of $\mathcal{F}$ that does not contain $\boldsymbol{a}$ (line 12).

(4) If no further complete extension $E$ is found, terminate with YES (lines 13-14).

(5) If $E$ attacks $\boldsymbol{a}$, then $E_{\mathsf{GR}} \cup E$ is a counterexample for skeptical acceptance of $\boldsymbol{a}$ (lines 15-16).

(6) Otherwise, check whether there exists a non-empty complete extension $E'$ in the reduct $\mathcal{F}^E$ (lines 17-25).
   (a) If not, $E_{\mathsf{GR}} \cup E$ is a preferred extension of $\mathcal{F}$ and a counterexample for skeptical acceptance (lines 18-19).
   (b) If there is a complete extension $E'$ and $E'$ attacks $\boldsymbol{a}$, then $E_{\mathsf{GR}} \cup E \cup E'$ is a counterexample (lines 20-21).
   (c) Otherwise, add a complement clause and continue with (2) (lines 22-25).

The algorithm is sound and complete. So, for some input $(\mathcal{F}, \boldsymbol{a})$, it returns YES if and only if the query argument $\boldsymbol{a}$ is skeptically accepted wrt. preferred semantics in $\mathcal{F}$.

**Theorem 1.** *Algorithm 1 is sound and complete for the problem $\mathsf{DS\text{-}PR}$.*

## 4 Empirical Evaluation

To evaluate the performance of our algorithm for skeptical preferred reasoning, we conducted an evaluation and compared its runtime to that of current state-of-the-art solvers.

| **Solver** | #TO | RT | PAR2 | #VBS |
|---|---|---|---|---|
| VBS | 18 | 3,281.75 | 141.28 | – |
| `reducto` | 23 | 5,198.47 | 183.58 | 182 |
| $\mu$-TOKSIA (GLUCOSE) | 30 | 6,795.78 | 239.50 | 44 |
| CRUSTABRI | 33 | 11,295.75 | 275.06 | 17 |
| $\mu$-TOKSIA (CMSAT) | 39 | 12,202.34 | 321.59 | 21 |
| FUDGE | 59 | 10,885.23 | 463.48 | 45 |
| PORTSAT | 171 | 11,620.18 | 1,282.74 | 2 |

Table 1: Results for the ICCMA'23 dataset (329 instances). #TO gives the number of time-outs; RT gives the total runtime on all correctly solved instances; PAR2 gives the average runtime where time-outs are counted double, i. e., 2,400 seconds; #VBS gives the number of instances contributed to the virtual best solver (VBS).

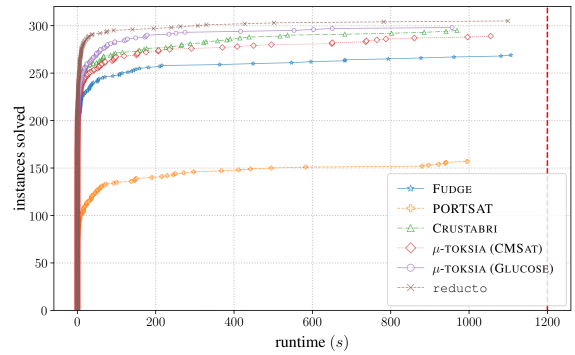


Figure 1: Number of solved instances given the per-instance runtime by each solver for skeptical reasoning wrt. preferred semantics on the ICCMA'23 dataset.

**Experimental Setup** We implemented Algorithm 1 in C++ as part of an argumentation solver which we called `reducto`. For all calls of the form SAT$(\cdot)$, WITNESS$(\cdot)$ `reducto` uses the SAT-solver CADICAL 2.1.3 (Biere et al. 2024). The computation of $\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})}$ and the function GROUNDED$(\mathcal{F})$ are implemented directly in C++ via simple iterative procedures. The implementation is open source and available on GitHub[2].

For the evaluation of our algorithm, we consider the benchmark datasets of the *International Competition on Computational Models of Argumentation*[3] (ICCMA). We consider the decision problem DS-PR and make use of the appropriate dataset from ICCMA'23 (Järvisalo, Lehtonen, and Niskanen 2025), which consists of 329 instances.

We consider the runtime per instance and compare it to that of current state-of-the-art argumentation solvers. Beside our own solver `reducto` (v2.13), we consider all competitors from the latest ICCMA'23 for the evaluation:

$\mu$-TOKSIA (Niskanen and Järvisalo 2020): written in C++, iterative SAT-based CEGAR approach. Available are two versions, one with GLUCOSE (Audemard and Simon 2018) and one with CRYPTOMINISAT (Soos, Nohl, and Castelluccia 2009) as the SAT-solver.

FUDGE (Thimm, Cerutti, and Vallati 2021): written in C++, iterative SAT-based approach with CADICAL (Biere et al. 2024) as the SAT-solver that solves the problem by computing admissible sets attacking admissible sets that contain the query argument.

CRUSTABRI (Lagniez, Lonca, and Mailly 2024): written in Rust, iterative SAT-based approach with CADICAL (Biere et al. 2024) as the SAT-solver.

PORTSAT (Declercq et al. 2023): written in Rust, enumerates preferred extensions with the help of a portfolio of different SAT-solvers.

The experimental evaluation has been conducted with the *probo2 benchmarking suite* for argumentation solvers (Klein and Thimm 2022). All experiments where executed on a machine running Ubuntu 20.04 with an Intel Xeon E5 3.4 GHz CPU and 192 GB of RAM. We used a per-instance time-out of 1200 seconds.

**Results** The results of our experiments are summarised in Table 1 and Figure 1. In general, `reducto` solves the most instances out of all the considered solvers and has the lowest total runtime. Moreover, `reducto` also has the best PAR2 score and contributes the most instances to the VBS, i. e., it has the fastest runtime of any solver on the most instances. The simplification steps outlined in Section 3 allow us to reduce the size $|\mathcal{A}|$ of an instance by $58.2\%$ on average. More specifically, restricting the AF to the arguments relevant for the query removes on average $38.2\%$ of the arguments, and "resolving" the grounded extension removes another $31.7\%$ of the remaining arguments. Figure 1 shows the number of solved instances of each solver given the per-instance runtime for the ICCMA'23 dataset and we can see that `reducto` performs best out of all competitors.

## 5 Conclusion

In this work, we considered the problem of skeptical reasoning wrt. preferred semantics, i. e., deciding whether every preferred extension of an AF contains a query argument. We introduced a novel algorithm that first simplifies the problem instance and subsequently searches through non-empty complete extensions of the simplified AF. Instead of maximising these extensions, our approach checks whether they directly attack the query argument and continues searching for extensions that contain unvisited arguments. We implemented this approach in the solver `reducto`. As our experimental results show, the combination of these simplifications and the search procedure allows `reducto` to outperform current state-of-the-art solvers.

Regarding future work, the simplification steps offer an interesting point for further research. First of all, there are other possibilities for more sophisticated preprocessing (Dvorák et al. 2019) that could be of use. Moreover, preprocessing is not used at all by many of the existing solvers and thus it would be interesting to study its effectiveness in the context of the other applicable algorithms.

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

[3] `https://argumentationcompetition.org`

## Acknowledgements

The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grant 550735820).

## References

Audemard, G., and Simon, L. 2018. On the glucose SAT solver. *Int. J. Artif. Intell. Tools* 27(1):1840001:1–1840001:25.

Baroni, P., and Giacomin, M. 2007. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence* 171(10-15):675–700.

Baumann, R.; Brewka, G.; and Ulbricht, M. 2020. 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*, 2742–2749. AAAI Press.

Baumann, R.; Brewka, G.; and Ulbricht, M. 2022. Shedding new light on the foundations of abstract argumentation: Modularization and weak admissibility. *Artificial Intelligence* 310:103742.

Bengel, L.; Sander, J.; and Thimm, M. 2025. *A Reduct-based Approach to Skeptical Preferred Reasoning in Abstract Argumentation (Extended Version)*. Zenodo.

Besnard, P., and Doutre, S. 2004. Checking the acceptability of a set of arguments. In Delgrande, J. P., and Schaub, T., eds., *10th International Workshop on Non-Monotonic Reasoning (NMR 2004)*, 59–64.

Biere, A.; Faller, T.; Fazekas, K.; Fleury, M.; Froleyks, N.; and Pollitt, F. 2024. Cadical 2.0. In Gurfinkel, A., and Ganesh, V., eds., *Computer Aided Verification - 36th International Conference, CAV 2024*, volume 14681 of *Lecture Notes in Computer Science*, 133–152. Springer.

Biere, A.; Järvisalo, M.; and Kiesl, B. 2021. Preprocessing in SAT solving. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 391–435.

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

Borg, A., and Bex, F. 2024. Minimality, necessity and sufficiency for argumentation and explanation. *Int. J. Approx. Reason.* 168:109143.

Brewka, G.; Polberg, S.; and Woltran, S. 2014. Generalizations of dung frameworks and their role in formal argumentation. *IEEE Intelligent Systems* 29(1):30–38.

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

Cerutti, F.; Giacomin, M.; and Vallati, M. 2019. How we designed winning algorithms for abstract argumentation and which insight we attained. *Artificial Intelligence* 276:1–40.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM* 50(5):752–794.

Declercq, S.; Capellini, Q. J.; Yang, C.; Delobelle, J.; and Mailly, J.-G. 2023. Portsat: A portfolio of sat solvers for abstract argumentation. *ICCMA 2023* 32.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77(2):321–358.

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

Dvorák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2012. Cegartix: A sat-based argumentation system. In *Pragmatics of SAT Workshop (POS)*.

Dvorák, W.; Järvisalo, M.; Linsbichler, T.; Niskanen, A.; and Woltran, S. 2019. Preprocessing argumentation frameworks via replacement patterns. In Calimeri, F.; Leone, N.; and Manna, M., eds., *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Proceedings*, volume 11468 of *LNCS*, 116–132. Springer.

Järvisalo, M.; Lehtonen, T.; and Niskanen, A. 2025. Iccma 2023: 5th international competition on computational models of argumentation. *Artificial Intelligence* 342:104311.

Klein, J., and Thimm, M. 2022. probo2: A benchmark framework for argumentation solvers. In Toni, F.; Polberg, S.; Booth, R.; Caminada, M.; and Kido, H., eds., *Computational Models of Argument - Proceedings of COMMA 2022*, volume 353 of *Frontiers in Artificial Intelligence and Applications*, 363–364. IOS Press.

Lagniez, J.; Lonca, E.; and Mailly, J. 2024. A sat-based approach for argumentation dynamics. In Dastani, M.; Sichman, J. S.; Alechina, N.; and Dignum, V., eds., *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024*, 2351–2353. ACM.

Liao, B., and Huang, H. 2013. Partial semantics of argumentation: basic properties and empirical. *Journal of Logic and Computation* 23(3):541–562.

Niskanen, A., and Järvisalo, M. 2020. $\mu$-toksia: An efficient abstract argumentation reasoner. In Calvanese, D.; Erdem, E.; and Thielscher, M., eds., *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020*, 800–804.

Soos, M.; Nohl, K.; and Castelluccia, C. 2009. Extending SAT solvers to cryptographic problems. In Kullmann, O., ed., *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, 244–257. Springer.

Thimm, M.; Cerutti, F.; and Vallati, M. 2021. Skeptical reasoning with preferred semantics in abstract argumentation without computing preferred extensions. In Zhou, Z., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*, 2069–2075. ijcai.org.

Thimm, M. 2023. On undisputed sets in abstract argumentation. In Williams, B.; Chen, Y.; and Neville, J., eds., *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*, 6550–6557. AAAI Press.