Skeptical Preferred Reasoning via Reducts in Abstract Argumentation*

Lars Bengel, Julian Sander and Matthias Thimm

Artificial Intelligence Group, University of Hagen, Germany

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 current state-of-the-art approaches in most instances.

Keywords

abstract argumentation, reasoning, preferred semantics, algorithms

1. Introduction

Formal argumentation is a research field within the area of knowledge representation and reasoning that offers a great variety of formalisms [2], and the *(abstract) argumentation framework* (AF) introduced by Dung [3] is a core area of research. In an argumentation framework, arguments are modelled as abstract entities and we consider directed attacks between them as the only relation. Reasoning in argumentation frameworks 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 [3]. One can then define different reasoning problems based on these semantics [4]. 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 [4]. Especially because of that, algorithms to efficiently compute these problems are of great importance in order to apply argumentation in practice.

A rally point for this field of research is the *International Competition on Computational Models of Argumentation* (ICCMA)[1], a biennial competition that evaluates solvers based on different computational problems in abstract and assumption-based argumentation. This topic has received a lot of attention in the literature over the years and many different problem solving paradigms have been utilised, see Cerutti et al. [5] for a survey. The most prominent approach is based on a reduction to a *satisfiability* (SAT) problem [6, 5] as is widely used by many argumentation solvers [7, 8, 9]. Another relevant approach is using *answer set programming* which is, for instance, used by the ASPARTIX system [10] to solve various tasks for abstract argumentation and other related formalisms. Lastly, there also exist some direct approaches [11, 12], mainly based on the labeling-based characterisation of semantics for argumentation frameworks [13]. In general however, the SAT-based approaches have proven to be the fastest, in particular for the problem of skeptical preferred reasoning [14].

23rd International Workshop on Nonmonotonic Reasoning (NMR'25), November 11-13, 2025, Melbourne, Australia

*A short version of this paper has been accepted at KR'25 [1].

lars.bengel@fernuni-hagen.de (L. Bengel); julian.sander@fernuni-hagen.de (J. Sander); matthias.thimm@fernuni-hagen.de (M. Thimm)

[1] https://argumentationcompetition.org

In this work, we introduce a novel SAT-based algorithm for solving the problem of skeptical reasoning wrt. preferred semantics. This algorithm is built upon recent results on the characterisation of preferred semantics as a vacuous-reduct semantics [15]. 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, the algorithm does not maximise each complete extension and instead continues searching for new complete extensions containing unvisited arguments. Our algorithm also employs effective preprocessing measures, outlined by Liao and Huang [16], to simplify computation. Thus, our algorithm is able to solve the problem of skeptical preferred reasoning without having to actually compute the preferred extensions, similar to the approach of Thimm et al. [9]. Moreover, we show that our algorithm is sound and complete and our experiments show that it significantly outperforms current state-of-the-art solvers on most benchmarks.

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, in Section 5 we discuss related work and Section 6 concludes the paper.

2. Preliminaries

We consider abstract argumentation as introduced by Dung [3]. 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. Furthermore, for a set $S \subseteq \mathcal{A}$ we define the set of arguments attacked by S (attacking S) in $\mathcal{F}$ respectively as

$$S^+_\mathcal{F} = \{\boldsymbol{a} \in \mathcal{A} \mid \exists \boldsymbol{b} \in S : \boldsymbol{b}\mathcal{R}\boldsymbol{a}\}, \qquad S^-_\mathcal{F} = \{\boldsymbol{a} \in \mathcal{A} \mid \exists \boldsymbol{b} \in S : \boldsymbol{a}\mathcal{R}\boldsymbol{b}\}.$$

If $S = \{\boldsymbol{a}\}$ is a singleton set, we also write $\boldsymbol{a}^+_\mathcal{F}$ (respectively $\boldsymbol{a}^-_\mathcal{F}$) for readability. 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 $S^-_\mathcal{F} \subseteq S^+_\mathcal{F}$. We denote with $\mathrm{ad}(\mathcal{F})$ the admissible sets of $\mathcal{F}$.

We can now impose further constraints on admissible sets to obtain different argumentation semantics [17]. In particular, an admissible set $E \subseteq \mathcal{A}$ is called a

- *complete* (CO) extension iff for every $\boldsymbol{a} \in \mathcal{A}$, if E defends $\boldsymbol{a}$ then $\boldsymbol{a} \in E$,
- *preferred* (PR) extension iff there exists no admissible E' with $E \subsetneq E'$,
- *grounded* (GR) extension iff E is complete and there is no complete E' with $E' \subsetneq E$.

For a given argumentation framework $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ and a semantics $\sigma \in \{\mathrm{CO}, \mathrm{PR}, \mathrm{GR}\}$, we denote with $\sigma(\mathcal{F})$ the set of σ-extensions of $\mathcal{F}$.

Example 1. Consider the argumentation framework $\mathcal{F}$ depicted in Figure 1. The complete extensions of $\mathcal{F}$ are $\{\boldsymbol{a}\}$, $\{\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{f}\}$ and $\{\boldsymbol{a}, \boldsymbol{d}, \boldsymbol{f}, \boldsymbol{h}\}$. Note that, for instance, $\{\boldsymbol{a}, \boldsymbol{c}\}$ is admissible, but not complete since it also defends $\boldsymbol{f}$. Furthermore, $\{\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{f}\}$ and $\{\boldsymbol{a}, \boldsymbol{d}, \boldsymbol{f}, \boldsymbol{h}\}$ are the preferred extensions of $\mathcal{F}$. Finally, $\{\boldsymbol{a}\}$ is the minimal complete extension and thus the unique grounded extension of $\mathcal{F}$.

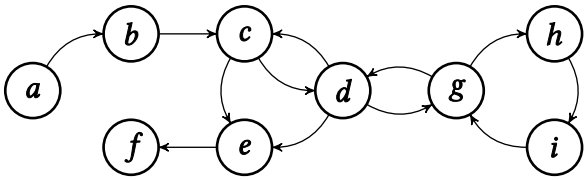


Figure 1: The argumentation framework $\mathcal{F}$ from Examples 1 – 6.

For each of the above introduced semantics we consider different reasoning problems known in the literature that are relevant when working with argumentation frameworks [4]. In particular, we consider the problems of credulous and skeptical acceptance wrt. some semantics σ, denoted as DC-σ and DS-σ respectively. We also consider the problem of computing a single extension wrt. some semantics σ, denoted as SE-σ. These problems are formally defined as follows:

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

The following example illustrates the above problems.

Example 2. Consider again the argumentation framework $\mathcal{F}$ in Figure 1. For complete and preferred semantics the arguments $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{f}$ and $\boldsymbol{h}$ are credulously accepted, while $\boldsymbol{b}, \boldsymbol{e}, \boldsymbol{g}$ and $\boldsymbol{i}$ are not credulously accepted wrt. any semantics. For the grounded semantics $\boldsymbol{a}$ is both the only credulously and skeptically accepted argument.

For complete and preferred semantics, $\boldsymbol{a}$ is also skeptically accepted. Interestingly, while $\boldsymbol{f}$ is *not* skeptically accepted wrt. complete semantics, it is however skeptically accepted wrt. preferred semantics, since it is contained in both preferred extensions, cf. Example 1.

The computational complexity of these problems has been well studied, we refer the interested reader to [4] for an overview. The prevalent strategy to solve many of these problems is reducing them to *satisfiability problems* (SAT) and using a dedicated SAT-solver for solving those, cf. [5, 8].

In particular, we want to highlight two results which are of importance for our work. The problem SE-GR is in P, i. e., computing the grounded extension of an AF can be done in polynomial time. Secondly, the problem of skeptical reasoning wrt. preferred semantics (DS-PR) is Π_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. In particular, the problem DS-PR will be the main focus of this work.

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. We first outline in Section 3.1 the underlying ideas of our algorithm, which are based on a combination of results and characterisations from the literature. Afterwards we describe the SAT-encoding used by our approach (Section 3.2). Finally, we present our algorithm for skeptical reasoning wrt. preferred semantics (Section 3.3) and show that it is sound and complete in Section 3.4.

Our approach is built on the concept of *counterexample guided abstraction refinement* (CEGAR) [18]. This concept is widely used by argumentation solvers [8, 9] and has been pioneered by the CEGARTIX system [7] for the domain of abstract argumentation. Given an argumentation framework $\mathcal{F}$ 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 argumentation framework by removing irrelevant arguments and "resolving" the grounded extension,

(2) Iterate through complete extensions of the remaining argumentation framework 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.

3.1. Theoretical Background

Let us start with considering the relevant notions and results from the literature that our algorithm is built upon. Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an arbitrary argumentation framework and $\boldsymbol{a} \in \mathcal{A}$ is the query argument in question.

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 [19]. In the context of abstract argumentation, this can mean replacing parts of the argumentation framework with simplified structures [20] or removing parts of the framework altogether [16]. In this work, we consider only the latter approach.

The simplification performed by our algorithm consists essentially of two steps:

(1) Restrict the argumentation framework to only those arguments relevant for the query $\boldsymbol{a}$.

(2) Compute the grounded extension E_{GR} of the restricted argumentation framework and remove all arguments contained in E_{GR} and those attacked by E_{GR}.

The first step is based on the principle of *Directionality* for argumentation semantics and has already been outlined by Liao and Huang [16]. For that, for some argumentation framework $\mathcal{F} = (\mathcal{A}, \mathcal{R})$, we first define the set of *unattacked sets* of $\mathcal{F}$.

$$\mathrm{UA}(\mathcal{F}) = \{S \subseteq \mathcal{A} \mid \nexists \boldsymbol{a} \in (\mathcal{A} \setminus S) \,:\, \boldsymbol{a} \in S^{-}_{\mathcal{F}}\} \tag{1}$$

Based on that, the *Directionality* principle has been defined [21].

Principle 1. Let σ be a semantics. We say that σ satisfies *Directionality* if and only if for all argumentation frameworks $\mathcal{F}$ and every set $U \in \mathrm{UA}(\mathcal{F})$ 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 σ should only depend on its attackers (and in turn on their attackers and so on). As has been shown by Baroni and Giacomin [21], both the complete and preferred semantics satisfy *Directionality*.

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

Now we can determine an unattacked set $U \in \mathrm{UA}(\mathcal{F})$ that contains the query argument $\boldsymbol{a}$ and restrict the argumentation framework 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.

$$\mathrm{Rel}_{\mathcal{F}}(\boldsymbol{a}) = \{\boldsymbol{a}\} \cup \{\boldsymbol{b} \in \mathcal{A} \mid \boldsymbol{b} \text{ is relevant for } \boldsymbol{a}\} \tag{2}$$

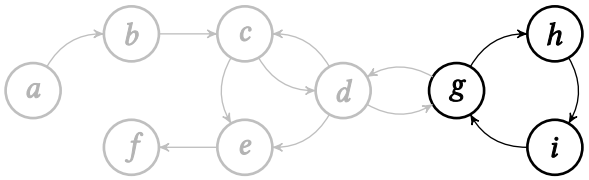


Figure 2: The reduct $\mathcal{F}^{\{a,c,f\}}$ of the argumentation framework $\mathcal{F}$.

Example 3. Consider again the argumentation framework $\mathcal{F}$ in Figure 1. We examine the argument d. It is easy to observe that $\mathsf{Rel}_{\mathcal{F}}(d) = \{a, b, c, d, g, h, i\}$. Note that for both e and f there exists no directed path to d thus they are not relevant for d. Similarly, we also have $\mathsf{Rel}_{\mathcal{F}}(b) = \{a, b\}$ and $\mathsf{Rel}_{\mathcal{F}}(f) = \mathcal{A}_1$.

For convenience, we explicitly define that a is always relevant for itself. Notably, this notion of relevance has already been used in [16], but also recently been defined in the context of acceptance explanations by Borg and Bex [22]. It is then easy to see that $\mathsf{Rel}_{\mathcal{F}}(a)$ is an unattacked set of $\mathcal{F}$ for any argument a.

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

According to the directionality property, this allows us to restrict the input argumentation framework to just the arguments relevant for the query a before performing further calculations. This does not only allow us to ignore arguments that are only attacked by the query, but also enables us to disregard unrelated components of the argumentation framework entirely.

The second simplification we perform is explicitly computing the grounded extension of the AF, which can be done in polynomial time [4]. For that we utilise the simple iterative procedure for computing the grounded extension that has been outlined by [3]. As we will show in the following paragraphs, we can then remove the grounded extension and everything attacked by it from the argumentation framework, to simplify the subsequently created SAT-encoding.

Iterating Complete Extensions We are concerned with the problem of skeptical reasoning wrt. preferred semantics. This, however, does not mean, that we have to explicitly compute preferred extensions to solve this problem [9]. Instead, we will primarily consider complete extensions.

The main notion underlying our algorithm is the *S*-reduct introduced by Baumann et al. [23].

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 $\mathcal{F}^S = (\mathcal{A}', \mathcal{R}')$ with

$$\mathcal{A}' = \mathcal{A} \setminus (S \cup S^+_{\mathcal{F}}),$$
$$\mathcal{R}' = \mathcal{R} \cap (\mathcal{A}' \times \mathcal{A}').$$

Essentially, the reduct allows us to remove the part of the AF $\mathcal{F}$ that is already "resolved" by S.

Example 4. Consider again the argumentation framework $\mathcal{F}$ in Figure 1. Let $S = \{a, c, f\}$, then the S-reduct of $\mathcal{F}$ is the argumentation framework $\mathcal{F}^{\{a,c,f\}}$ depicted in Figure 2.

Based on this concept, the notion of *vacuous reduct semantics* has been introduced [15].

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

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

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

Example 5. Consider again the argumentation framework $\mathcal{F}$ in Figure 1. Let us consider the vacuous reduct semantics cf^{ad}, i. e., the conflict-free sets S such that the reduct $\mathcal{F}^S$ contains no non-empty admissible sets[2]. Examine, for instance, the set $S_1 = \{a, c, f\}$. Clearly S_1 is conflict-free in $\mathcal{F}$. The reduct $\mathcal{F}^{S_1}$ is depicted in Figure 2. It is easy to verify that $\emptyset$ is the only admissible set of $\mathcal{F}^{S_1}$, thus S_1 is a cf^{ad}-extension of $\mathcal{F}$. On the other hand, the conflict-free set $\{a, d, f\}$ is not a cf^{ad}-extension, because there is the admissible set $\{h\}$ in the reduct $\mathcal{F}^{\{a,d,f\}}$.

Of particular interest to us, is the fact that the preferred semantics can be characterised as a vacuous reduct semantics, as shown by [15, 24].

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

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

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

Constructing the Counterexample In the course of executing the algorithm we construct different complete extensions in different restrictions and reducts of the original argumentation framework. What allows us to combine them to construct a proper counterexample for skeptical acceptance is the concept of *Modularisation* of argumentation semantics introduced by Baumann et al. [25].

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

As shown by Baumann et al. [25], *Modularisation* is satisfied by both complete and preferred semantics.

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

This concept will prove useful for our algorithm and allows us to combine partial results into a proper counterexample for skeptical acceptance.

Example 6. Consider again the argumentation framework $\mathcal{F}$ depicted in Figure 1. We have the complete extension $\{a\}$ and in the reduct $\mathcal{F}^{\{a\}}$, there is for instance the complete extension $\{c, f\}$ and, as already seen in Example 1, the union $\{a\} \cup \{c, f\}$ is a complete extension of $\mathcal{F}$. Notably, this combined set then also serves as a counterexample for skeptical acceptance of the argument d in $\mathcal{F}$.

3.2. SAT-Encoding for Complete Semantics

Our algorithm for skeptical preferred reasoning, like most state-of-the-art approaches, utilises a reduction to SAT. In the following, we will briefly describe the SAT-encoding used for computing complete extensions. The encodings are built on standard SAT-encodings for AFs [6, 26], used in similar fashion by many argumentation solvers [8, 14].

Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework. Let us first introduce basic notion. We consider propositional logic and we model the acceptance (rejection) of an argument $a \in \mathcal{A}$ wrt. to some extension $E \subseteq \mathcal{A}$ with the propositional variable $\texttt{in}_a$ ($\texttt{out}_a$). We will then define formulae over these atoms with the usual connectives: $\neg, \wedge, \vee$. Let $\omega : \{\texttt{in}_a\}_{a \in \mathcal{A}} \cup \{\texttt{out}_a\}_{a \in \mathcal{A}} \mapsto \{\text{TRUE}, \text{FALSE}\}$ be some interpretation over the arguments of $\mathcal{F}$ representing some extension $E \subseteq \mathcal{A}$. Then, if $\texttt{in}_a$ is TRUE in ω, then $a \in E$, otherwise $a \notin E$. Analogously, if $\texttt{out}_a$ is TRUE in ω, then $a \in E^+_{\mathcal{F}}$, otherwise $a \notin E^+_{\mathcal{F}}$.

Our encoding for complete semantics then consists of the following components. Equation (3) models basic conditions for acceptance: an argument cannot be accepted and rejected at the same time, if an attacker b of a is accepted, then a is rejected and if a is rejected, then one of its attackers must be accepted. Recall that a complete extension is conflict-free, defends all its members and includes all

[2]Note that the cf^{ad}-extensions have also been coined *undisputed sets* in [15].

arguments that it defends. Conflict-Freeness is modelled by $\Psi^{cf}_{\mathcal{F}}$ in Equation (4) via the condition if $\boldsymbol{b}$ attacks $\boldsymbol{a}$ then if $\boldsymbol{b}$ is accepted $\boldsymbol{a}$ must not be accepted. Defense is modelled in Equation (5), stating that, if an argument $\boldsymbol{a}$ is accepted then every attacker $\boldsymbol{b}$ must be rejected. Finally, completeness is described in Equation (6) via the condition if $\boldsymbol{a}$ is not accepted, then one of its attackers must not be rejected.

$$\Psi^{re}_{\mathcal{F}} = \bigwedge_{\boldsymbol{a}\in\mathcal{A}} \left((\neg \mathtt{in}_{\boldsymbol{a}} \vee \neg \mathtt{out}_{\boldsymbol{a}}) \wedge \bigwedge_{\boldsymbol{b}\in\boldsymbol{a}^-_{\mathcal{F}}} (\mathtt{out}_{\boldsymbol{a}} \vee \neg \mathtt{in}_{\boldsymbol{b}}) \wedge \left(\neg \mathtt{out}_{\boldsymbol{a}} \vee \bigvee_{\boldsymbol{c}\in\boldsymbol{a}^-_{\mathcal{F}}} \mathtt{in}_{\boldsymbol{c}}\right) \right) \tag{3}$$

$$\Psi^{cf}_{\mathcal{F}} = \Psi^{re}_{\mathcal{F}} \wedge \bigwedge_{\boldsymbol{a}\in\mathcal{A}} \bigwedge_{\boldsymbol{b}\in\boldsymbol{a}^-_{\mathcal{F}}} \neg \mathtt{in}_{\boldsymbol{a}} \vee \neg \mathtt{in}_{\boldsymbol{b}} \tag{4}$$

$$\Psi^{ad}_{\mathcal{F}} = \Psi^{cf}_{\mathcal{F}} \wedge \bigwedge_{\boldsymbol{a}\in\mathcal{A}} \bigwedge_{\boldsymbol{b}\in\boldsymbol{a}^-_{\mathcal{F}}} \neg \mathtt{in}_{\boldsymbol{a}} \vee \mathtt{out}_{\boldsymbol{b}} \tag{5}$$

$$\Psi^{\mathrm{CO}}_{\mathcal{F}} = \Psi^{ad}_{\mathcal{F}} \wedge \bigwedge_{\boldsymbol{a}\in\mathcal{A}} \left(\mathtt{in}_{\boldsymbol{a}} \vee \bigvee_{\boldsymbol{b}\in\boldsymbol{a}^-_{\mathcal{F}}} \neg \mathtt{out}_{\boldsymbol{b}} \right) \tag{6}$$

As has been shown in [6], every model of $\Psi^{\mathrm{CO}}_{\mathcal{F}}$ corresponds to a complete extension of $\mathcal{F}$. In particular, for some argumentation framework $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ and a model ω of $\Psi^{\mathrm{CO}}_{\mathcal{F}}$ we define the corresponding complete extension E_ω of $\mathcal{F}$ as follows:

$$E_\omega = \{\boldsymbol{a} \in \mathcal{A} \mid \omega(\mathtt{in}_{\boldsymbol{a}}) = \textsc{True}\} \tag{7}$$

Beyond that, we also utilise the following clause in every SAT-call which ensures that the computed complete extension is non-empty.

$$\Psi^{ne}_{\mathcal{F}} = \bigvee_{\boldsymbol{a}\in\mathcal{A}} \mathtt{in}_{\boldsymbol{a}} \tag{8}$$

3.3. Algorithm for Skeptical Reasoning wrt. Preferred Semantics

First, we introduce some notation. GROUNDED($\mathcal{F}$) denotes the iterative algorithm computing the grounded extension of Dung [3]. We write SAT(Ψ) for a call to the SAT-solver that returns TRUE iff Ψ is satisfiable and FALSE otherwise. Furthermore, we write WITNESS(Ψ) for a call to the SAT-solver that returns the set E_ω for some model ω of Ψ, if Ψ is satisfiable, otherwise it returns FALSE. We also utilise a complement clause $\mathcal{C}_{\mathcal{F}}(E)$ defined as

$$\mathcal{C}_{\mathcal{F}}(E) = \bigvee_{\boldsymbol{a}\in\mathcal{A}\setminus E} \mathtt{in}_{\boldsymbol{a}} \tag{9}$$

for some set of arguments $E \subseteq \mathcal{A}$. This clause, for each found complete extension S, ensures not only that the SAT-solver does not find E as a solution again, but also that any E' with $E' \subseteq E$ is no valid witness in any following SAT-call.

Our novel algorithm for deciding skeptical acceptance wrt. preferred semantics is shown in Algorithm 1. For the input $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ and some $\boldsymbol{a} \in \mathcal{A}$, the algorithm returns 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 the grounded extension E_{GR} of $\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})}$ (lines 1-2). If $\boldsymbol{a} \in E_{\mathrm{GR}}$ terminate with YES, if $\boldsymbol{a} \in E^+_{\mathrm{GR},\mathcal{F}}$ terminate with E_{GR} as the witness for non-acceptance. Otherwise move to $(\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(\boldsymbol{a})})^{E_{\mathrm{GR}}}$ (lines 3-7).

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

(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 $E_{\mathrm{GR}} \cup E$ is a counterexample for skeptical acceptance of $\boldsymbol{a}$ (lines 15-16).

(6) Otherwise, check if there is a non-empty complete extension E' in the reduct $\mathcal{F}^E$ (lines 17-25).

a) If not, then $E_{\mathrm{GR}} \cup E$ is a preferred extension of $\mathcal{F}$ and thus a counterexample for the skeptical acceptance of $\boldsymbol{a}$ (lines 18-19).

b) If there is a non-empty complete extension E' and E' attacks $\boldsymbol{a}$, then $E_{\mathrm{GR}} \cup E \cup E'$ is a counterexample (lines 20-21).

c) Otherwise, add a complement clause for $E \cup E'$ and continue with (2) (lines 22-25).

Algorithm 1 Algorithm for DS-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: $E_{\mathrm{GR}} \leftarrow \textsc{Grounded}(\mathcal{F})$
3: **if** $\boldsymbol{a} \in E_{\mathrm{GR}}$ **then**
4: **return** Yes
5: **if** $\boldsymbol{a} \in E^{+}_{\mathrm{GR},\mathcal{F}}$ **then**
6: **return** E_{GR}
7: $\mathcal{F} \leftarrow \mathcal{F}^{E_{\mathrm{GR}}}$
8: $\Psi \leftarrow \Psi^{\mathrm{CO}}_{\mathcal{F}} \wedge \Psi^{ne}_{\mathcal{F}}$
9: **if** $\textsc{Sat}(\Psi) = \textsc{False}$ **then**
10: **return** E_{GR}
11: **while** True **do**
12: $E \leftarrow \textsc{Witness}(\Psi \wedge \neg \mathtt{in}_a)$
13: **if** $E = \textsc{False}$ **then**
14: **return** Yes
15: **if** $\boldsymbol{a} \in E^{+}_{\mathcal{F}}$ **then**
16: **return** $E_{\mathrm{GR}} \cup E$
17: $E' \leftarrow \textsc{Witness}(\Psi^{\mathrm{CO}}_{\mathcal{F}^E} \wedge \Psi^{ne}_{\mathcal{F}^E})$
18: **if** $E' = \textsc{False}$ **then**
19: **return** $E_{\mathrm{GR}} \cup E$
20: **if** $\boldsymbol{a} \in E'^{+}_{\mathcal{F}}$ **then**
21: **return** $E_{\mathrm{GR}} \cup E \cup E'$
22: **if** $\boldsymbol{a} \in E'$ **then**
23: $\Psi \leftarrow \Psi \wedge \mathcal{C}_{\mathcal{F}}(E \cup E')$
24: **else**
25: $\Psi \leftarrow \Psi \wedge \mathcal{C}_{\mathcal{F}}(E)$

3.4. Soundness and Completeness

As the following result shows, our algorithm is indeed sound and complete. 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 $\mathcal{F}$. The full proof of Theorem 1 can be found in the extended version of this paper [27], available online[3].

Theorem 1. *Algorithm 1 is sound and complete for the problem DS-PR.*

Note that the algorithm is still sound and complete if we use the encoding $\Psi^{\mathrm{ad}}_{\mathcal{F}}$ for admissibility instead of the encoding for complete semantics. This follows easily via Proposition 2[4].

[3] https://doi.org/10.5281/zenodo.16022972

[4] Early experiments showed however that using $\Psi^{\mathrm{CO}}_{\mathcal{F}}$ is slightly faster in practice, see also [26] for a detailed performance analysis of different types of semantical encodings.

In case the query argument a is not skeptically accepted wrt. preferred semantics, the algorithm returns a witness that serves as a counterexample for the non-acceptance of a. As already mentioned before, this output is not necessarily a preferred extension that does not contain the query a. Instead, Algorithm 1 returns three different types of witnesses, depending on the situation:

(1) an admissible set of $\mathcal{F}$ attacking the query (lines 6, 16, 21),

(2) the grounded extension of $\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(a)}$ not containing the query (line 10),

(3) a preferred extension of $\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(a)}$ not containing the query (line 19).

Importantly, as follows from the proof of Theorem 1, the witness produced by Algorithm 1 is in any case sufficient to prove that the query is not skeptically accepted wrt. preferred semantics.

Corollary 2. *Let $\mathcal{F} = (\mathcal{A}, \mathcal{R})$ be an argumentation framework and $a \in \mathcal{A}$ and $a \notin \bigcap_{E \in PR(\mathcal{F})} E$. Let $S \subseteq \mathcal{A}$ be the output of Algorithm 1 for the input $\mathcal{F} = (\mathcal{A}, \mathcal{R}), a \in \mathcal{A}$. Then, there exists a preferred extension $E \in PR(\mathcal{F})$ with $S \subset E$ and $a \notin E$.*

The following section will show the feasibility of our approach in practice and that it significantly outperforms state-of-the-art approaches in most cases.

4. Empirical Evaluation

To evaluate the performance of our novel algorithm for skeptical preferred reasoning, we conducted an evaluation and compared its runtime to that of current state-of-the-art argumentation solvers. In the following, we will briefly describe the implementation of our algorithm as well as the experimental setup. Finally, we will present the results of the evaluation and provide an ablation study wrt. SAT-solvers.

4.1. System Overview

We implemented Algorithm 1 in C++ as part of an argumentation solver which we called REDUCTO. For all calls of the form SAT(·), WITNESS(·) REDUCTO uses the SAT-solver CADICAL 2.1.3 [28]. The computation of $\mathcal{F}|_{\mathsf{Rel}_{\mathcal{F}}(a)}$ and the function GROUNDED($\mathcal{F}$) are implemented directly in C++ via simple iterative procedures. The implementation is open source and available on GitHub[5].

4.2. Experimental Setup

For the evaluation of our algorithm, we consider the benchmark datasets of the *International Competition on Computational Models of Argumentation* (ICCMA). We consider the decision problem DS-PR and make use of the appropriate datasets from all competitions, i. e., ICCMA'15 to ICCMA'23 [29, 30, 31, 14].

Table 1
Statistics for all considered benchmark datasets. #AFs is the number of instances in the dataset; $|\mathcal{A}|$ is the number of arguments per instance, for which we consider average, median and standard deviation; $|\mathcal{R}|$ is the number of attacks in an instance; Avg. D is the average node degree of the whole dataset.

Dataset	#AFs	Avg. $\lvert\mathcal{A}\rvert$	Med. $\lvert\mathcal{A}\rvert$	Std. $\lvert\mathcal{A}\rvert$	Avg. $\lvert\mathcal{R}\rvert$	Avg. D
ICCMA'15	192	1,980	675	2,424	105,396	68
ICCMA'17	300	14,544	474	132,416	311,277	245
ICCMA'19	326	826	196	1,784	97,639	239
ICCMA'21	480	87,331	48,200	92,881	7,239,611	161
ICCMA'23	329	29,791	796	203,719	1,002,470	299

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

Table 1 summarises some key statistics for the considered benchmark sets. Note that the ICCMA'21 dataset is comprised of particularly large instances compared to all other benchmark sets.

To evaluate the performance of our algorithm for skeptical preferred reasoning, we consider the runtime per instance and compare it to that of current state-of-the-art argumentation solvers. Beside our solver REDUCTO (version 2.13), we consider all competitors from the latest ICCMA'23 for the evaluation:

μ-TOKSIA [8]: written in C++, uses an iterative SAT-based CEGAR approach [7]. Available are two versions, one with GLUCOSE [32] and one with CRYPTOMINISAT [33] as the SAT-solver.

FUDGE [9]: written in C++, SAT-based approach with CADICAL [28] as the SAT-solver that solves the problem by computing admissible sets attacking admissible sets that contain the query argument.

CRUSTABRI [34]: written in Rust, iterative SAT-based approach with CADICAL [28] as the SAT-solver.

PORTSAT [35]: 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 [36]. 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.

4.3. Results

The results of our experiments are summarised in Table 2. For each benchmark set, we also consider the *virtual best solver* (VBS), which is computed by taking the best runtime for each instance from all competing solvers. Solvers are ranked in increasing order according to the number of unsolved instances and in case of ties by total runtime.

In general, REDUCTO solves the most instances out of all the considered solvers for the ICCMA'15, ICCMA'17, ICCMA'19 and ICCMA'23 benchmark sets. On all of these datasets 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 total runtime is also generally the lowest, only for ICCMA'17 is it higher than that of μ-TOKSIA (GLUCOSE) and CRUSTABRI.

This is simply due to the fact that REDUCTO solves significantly more instances of the dataset. Especially on the easier benchmark sets ICCMA'15 and ICCMA'19, where most solvers are able to solve all instances, the total runtime and PAR2 score of REDUCTO is significantly lower than that of all competitors.

Only on the ICCMA'21 dataset, μ-TOKSIA (GLUCOSE) solves more instances than REDUCTO and is also faster on almost all instances. The ICCMA'21 dataset generally consists of only very large instances, compared to the other benchmark sets (cf. Table 1). In addition to that, for all of the instances the simplification steps employed by our approach are not applicable, i. e., the AFs have the empty set as the grounded extension and all arguments in the AF are relevant for the query argument. We presume this is the reason for the worse performance of REDUCTO, in combination with a less efficient SAT-encoding compared to μ-TOKSIA. Notably, all other solvers, apart from μ-TOKSIA (GLUCOSE), including μ-TOKSIA (CMSAT), still perform significantly worse than REDUCTO on the ICCMA'21 dataset.

Let us take a closer look at the most recent ICCMA'23 dataset. The simplification steps outlined in Section 3.1 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 3a visualises the performance of each solver on the ICCMA'23 dataset. In particular, it shows the number of solved instances of each solver given the per-instance runtime. As one can see, REDUCTO performs significantly better than all other solvers on this dataset. Moreover, Figure 3b gives a direct comparison of REDUCTO and μ-TOKSIA (GLUCOSE), the two best ranked solvers, on the ICCMA'23 dataset. Markers above the diagonal line are instances where REDUCTO is faster and below μ-TOKSIA is faster. As we can see, the performance of both solvers is within one order of magnitude of each other for the majority of

Table 2
Results for all ICCMA datasets. #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).

(a) Results for ICCMA'23 (329 instances).

Solver	#TO	RT	PAR2	#VBS
VBS	18	3,281.75	141.28	–
REDUCTO	23	5,198.47	183.58	182
μ-TOKSIA (GLUCOSE)	30	6,795.78	239.50	44
CRUSTABRI	33	11,295.75	275.06	17
μ-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

(b) Results for ICCMA'21 (480 instances).

Solver	#TO	RT	PAR2	#VBS
VBS	100	93,733.47	695.28	-
μ-TOKSIA (GLUCOSE)	102	92,458.95	702.62	361
REDUCTO	190	86,159.50	1,129.50	19
CRUSTABRI	246	87,110.60	1,411.48	0
μ-TOKSIA (CMSAT)	344	51,107.46	1,826.47	0
FUDGE	390	41,360.11	2,036.17	0
PORTSAT	480	0.00	2,400.00	0

(c) Results for ICCMA'19 (326 instances).

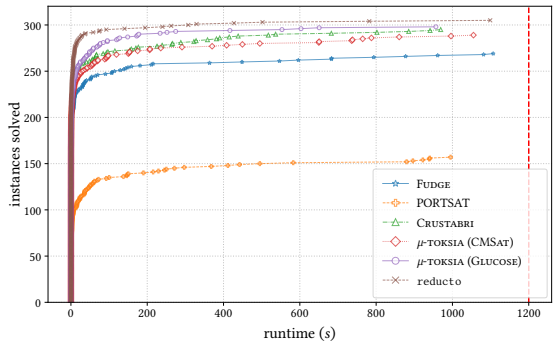
Solver	#TO	RT	PAR2	#VBS
VBS	0	54.66	0.17	–
REDUCTO	0	85.71	0.26	126
μ-TOKSIA (GLUCOSE)	0	169.91	0.52	72
CRUSTABRI	0	190.89	0.59	6
μ-TOKSIA (CMSAT)	0	264.18	0.81	48
FUDGE	4	343.72	30.50	68
PORTSAT	23	7,826.34	193.33	6

(d) Results for ICCMA'17 (350 instances).

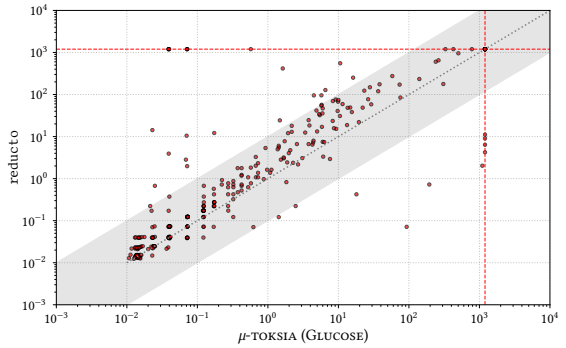
Solver	#TO	RT	PAR2	#VBS
VBS	24	12,777.54	201.08	–
REDUCTO	26	13,583.26	217.10	188
μ-TOKSIA (GLUCOSE)	41	11,884.86	315.10	49
μ-TOKSIA (CMSAT)	42	14,875.78	330.50	32
CRUSTABRI	50	12,681.05	379.09	3
FUDGE	52	17,769.32	407.34	54
PORTSAT	212	15,869.49	1,499.06	0

(e) Results for ICCMA'15 (192 instances).

Solver	#TO	RT	PAR2	#VBS
VBS	0	291.15	1.52	–
REDUCTO	0	309.23	1.61	106
μ-TOKSIA (GLUCOSE)	0	831.21	4.33	18
CRUSTABRI	0	1,233.27	6.42	1
μ-TOKSIA (CMSAT)	0	1,805.81	9.41	10
FUDGE	3	1,738.81	46.56	57
PORTSAT	5	9,883.08	113.97	0



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



(b) Comparison of per-instance runtimes (*s*) for REDUCTO and μ-TOKSIA (GLUCOSE) on the ICCMA'23 dataset. The diagonal line indicates equal runtime.

Figure 3: Evaluation results for the ICCMA'23 dataset.

instances. However, REDUCTO is clearly faster for almost all instances. That is also consistent with the fact that REDUCTO contributes the most instances to the VBS for all datasets except ICCMA'21.

4.4. Ablation study wrt. SAT-Solvers

To evaluate the impact of the utilised SAT-solver we conducted a small ablation study on the ICCMA'23 benchmark set. Besides CADICAL 2.1.3 [28], we also considered the SAT-solvers GLUCOSE 4.2.1 [32],

Table 3
Results for REDUCTO with different SAT-Solvers on the ICCMA'23 dataset.

Solver	#TO	RT	PAR2	#VBS
REDUCTO (CADICAL)	23	5,198.47	183.58	94
REDUCTO (CMSAT)	25	9,313.60	210.68	89
REDUCTO (MERGESAT)	26	7,599.64	212.76	64
REDUCTO (GLUCOSE)	29	6,101.97	230.10	82

CRYPTOMINISAT (CMSAT) 5.11.21 [33] and MERGESAT 4.0 [37]. In Table 3 we summarise the results for REDUCTO based on the different SAT-solvers on the ICCMA'23 dataset. As we can see, CADICAL significantly outperforms the other SAT-solvers. While the CMSAT and MERGESAT versions of REDUCTO perform quite similarly, the GLUCOSE version clearly performs worse than the others. This is in contrast to μ-TOKSIA, where GLUCOSE performs clearly better than CMSAT on all benchmarks. Interestingly, all versions contribute a significant amount of instances to the VBS, however CADICAL contributes the most. It should also be noted that all versions of REDUCTO are faster than all other solvers, cf. Table 2a.

5. Discussion

The problem of skeptical reasoning wrt. preferred semantics is a focal point of argumentation research, and there exist many different approaches in the literature to tackle it. Like our approach, many of them are based on CEGAR [18]. The μ-TOKSIA solver [8] uses the same approach as CEGARTIX [7], without shortcuts. Given an argumentation framework $\mathcal{F}$ and some argument $\boldsymbol{a}$, it searches for a complete extension E of $\mathcal{F}$ that does not contain $\boldsymbol{a}$. It then iteratively maximises E to try and obtain a preferred extension E' with $E' \supseteq E$ and $a \notin E'$. If no such E' exists, a complement clause is added to the SAT-encoding and the search for complete extensions without $\boldsymbol{a}$ continues. Simplifications are not applied in the latest version.

While our approach is fairly similar, there are some important differences. First of all, we employ simplifications, as outlined in Section 3.1. We also utilise shortcuts by checking whether the query argument $\boldsymbol{a}$ is attacked by some found non-empty complete extension E, similar to CEGARTIX. We also additionally use a clause to ensure non-emptiness, which is not used in other approaches. However, the key difference is that for each found complete extension E, we only check once whether there is a larger complete extension (by searching for a non-empty complete extension in the reduct $\mathcal{F}^E$). Meaning, we effectively search through different complete extensions that each contain at least one never before visited argument, instead of trying to maximise each one. Thus, we narrow down the search field with each iteration but grant more freedom for the complete extensions to be computed.

A quite different approach is employed by the FUDGE solver [9]. It uses a conflict-driven approach and directly searches for admissible sets that attack admissible set that contain the query argument. As already mentioned before, this is different to our approach, since we do not actively search for acceptable sets that attack the query, but rather check if that is the case during the search.

6. Conclusion

In this work, we considered the problem of skeptical reasoning wrt. preferred semantics, i. e., deciding whether every preferred extension of an argumentation framework contains some specific argument. We introduced a novel algorithm that first simplifies the problem instance and subsequently searches through non-empty complete extensions of the simplified argumentation framework. 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 on most benchmarks.

Regarding future work, we intent to further refine the underlying SAT-encoding, cf. the work of [26]. Furthermore, the simplification steps offer an interesting point for further research. First of all, there are other possibilities for more sophisticated preprocessing [20] 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 algorithms for skeptical preferred reasoning.

Acknowledgments

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

References

[1] L. Bengel, J. Sander, M. Thimm, A reduct-based approach to skeptical preferred reasoning in abstract argumentation, in: Proceedings of the 22th International Conference on Principles of Knowledge Representation and Reasoning, KR 2025, 2025.

[2] G. Brewka, S. Polberg, S. Woltran, Generalizations of dung frameworks and their role in formal argumentation, IEEE Intelligent Systems 29 (2014) 30–38.

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

[4] W. Dvorák, P. E. Dunne, Computational problems in formal argumentation and their complexity, Handbook of Formal Argumentation 1 (2017).

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

[6] P. Besnard, S. Doutre, Checking the acceptability of a set of arguments, in: 10th International Workshop on Non-Monotonic Reasoning (NMR 2004), 2004, pp. 59–64.

[7] W. Dvorák, M. Järvisalo, J. P. Wallner, S. Woltran, Cegartix: A sat-based argumentation system, in: Pragmatics of SAT Workshop (POS), 2012.

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

[9] M. Thimm, F. Cerutti, M. Vallati, Skeptical reasoning with preferred semantics in abstract argumentation without computing preferred extensions, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, ijcai.org, 2021, pp. 2069–2075.

[10] W. Dvorák, S. A. Gaggl, A. Rapberger, J. P. Wallner, S. Woltran, The ASPARTIX system suite, in: Computational Models of Argument - Proceedings of COMMA 2020, volume 326 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 461–462.

[11] S. Doutre, J. Mengin, Preferred extensions of argumentation frameworks: Query answering and computation, in: Automated Reasoning, First International Joint Conference, IJCAR 2001, Proceedings, volume 2083 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 272–288.

[12] S. Nofal, K. Atkinson, P. E. Dunne, Algorithms for decision problems in argument systems under preferred semantics, Artificial Intelligence 207 (2014) 23–51.

[13] M. W. A. Caminada, D. M. Gabbay, A logical account of formal argumentation, Stud Logica 93 (2009) 109–145.

[14] M. Järvisalo, T. Lehtonen, A. Niskanen, Iccma 2023: 5th international competition on computational models of argumentation, Artificial Intelligence 342 (2025) 104311.

[15] M. Thimm, On undisputed sets in abstract argumentation, in: Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, AAAI Press, 2023, pp. 6550–6557.

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

[17] P. Baroni, D. Gabbay, M. Giacomin, L. van der Torre (Eds.), Handbook of Formal Argumentation, College Publications, 2018.
[18] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, Counterexample-guided abstraction refinement for symbolic model checking, J. ACM 50 (2003) 752–794.
[19] A. Biere, M. Järvisalo, B. Kiesl, Preprocessing in SAT solving, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability - Second Edition, volume 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2021, pp. 391–435.
[20] W. Dvorák, M. Järvisalo, T. Linsbichler, A. Niskanen, S. Woltran, Preprocessing argumentation frameworks via replacement patterns, in: Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Proceedings, volume 11468 of *LNCS*, Springer, 2019, pp. 116–132.
[21] P. Baroni, M. Giacomin, On principle-based evaluation of extension-based argumentation semantics, Artificial Intelligence 171 (2007) 675–700.
[22] A. Borg, F. Bex, Minimality, necessity and sufficiency for argumentation and explanation, Int. J. Approx. Reason. 168 (2024) 109143.
[23] R. Baumann, G. Brewka, 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.
[24] L. Blümel, M. Thimm, Revisiting vacuous reduct semantics for abstract argumentation, in: ECAI 2024 - 27th European Conference on Artificial Intelligence, IOS Press, 2024, pp. 3517–3524.
[25] R. Baumann, G. Brewka, M. Ulbricht, Shedding new light on the foundations of abstract argumentation: Modularization and weak admissibility, Artificial Intelligence 310 (2022) 103742.
[26] F. Cerutti, M. Giacomin, M. Vallati, How we designed winning algorithms for abstract argumentation and which insight we attained, Artificial Intelligence 276 (2019) 1–40.
[27] L. Bengel, J. Sander, M. Thimm, A Reduct-based Approach to Skeptical Preferred Reasoning in Abstract Argumentation (Extended Version), Zenodo, 2025.
[28] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, F. Pollitt, Cadical 2.0, in: A. Gurfinkel, V. Ganesh (Eds.), Computer Aided Verification - 36th International Conference, CAV 2024, volume 14681 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 133–152.
[29] M. Thimm, S. Villata, The first international competition on computational models of argumentation: Results and analysis, Artificial Intelligence 252 (2017) 267–294.
[30] S. A. Gaggl, T. Linsbichler, M. Maratea, S. Woltran, Design and results of the second international competition on computational models of argumentation, Artificial Intelligence 279 (2020).
[31] S. Bistarelli, L. Kotthoff, J.-M. Lagniez, E. Lonca, J.-G. Mailly, J. Rossit, F. Santini, C. Taticchi, The third and fourth international competitions on computational models of argumentation: Design, results and analysis, Argument & Computation 0 (2024) 1–73.
[32] G. Audemard, L. Simon, On the glucose SAT solver, Int. J. Artif. Intell. Tools 27 (2018) 1840001:1–1840001:25.
[33] M. Soos, K. Nohl, C. Castelluccia, Extending SAT solvers to cryptographic problems, in: Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Proceedings, volume 5584 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 244–257.
[34] J. Lagniez, E. Lonca, J. Mailly, A sat-based approach for argumentation dynamics, in: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, ACM, 2024, pp. 2351–2353.
[35] S. Declercq, Q. J. Capellini, C. Yang, J. Delobelle, J.-G. Mailly, Portsat: A portfolio of sat solvers for abstract argumentation, ICCMA 2023 (2023) 32.
[36] J. Klein, M. Thimm, probo2: A benchmark framework for argumentation solvers, in: Computational Models of Argument - Proceedings of COMMA 2022, volume 353 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2022, pp. 363–364.
[37] N. Manthey, The mergesat solver, in: Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Proceedings, volume 12831 of *LNCS*, Springer, 2021, pp. 387–398.