On Algorithmic Shortcuts for Skeptical Preferred Reasoning in Abstract Argumentation

Lars Bengel, Julian Sander and Matthias Thimm

Artificial Intelligence Group, University of Hagen, Germany

Abstract

Skeptical acceptance under the preferred semantics is one of the most intricate reasoning problems in abstract argumentation. That is why efficient solving approaches are of utmost importance for this problem. In this work, we investigate algorithmic shortcuts that can be used to solve this reasoning problem with considerably less resources than sound and complete approaches. Besides an overview of the state-of-the-art, we also propose several novel shortcuts to improve the applicability of such techniques. Finally, we perform an extensive evaluation of all shortcuts regarding their potential to improve the runtime efficiency of argumentation solvers. As our results show, we can significantly improve the runtime of state-of-the-art argumentation solvers by utilizing a variety of shortcuts.

1. Introduction

Formal argumentation is one of the major areas of interest in the field of knowledge representation and reasoning [1, 2]. In particular, computational models of argumentation have found many applications, especially in the context of explainable and contestable artificial intelligence [3] or law and legal reasoning [4]. The area knows many well-researched formalism, for instance bipolar argumentation frameworks [5], assumption-based argumentation [6] or abstract dialectical frameworks [7].

Nevertheless, the *abstract argumentation frameworks* (AFs), first introduced by Dung [8], still remain the most prominent formalism in the argumentation literature. In an AF the arguments are modelled as abstract entities and directed attacks represent conflicts between them. Consequently, reasoning is then done via semantics that select jointly acceptable sets of arguments, according to different criteria [9]. A fundamental semantic property is *admissibility* which requires that the accepted arguments must be conflict-free and defend themselves against opposing arguments. The maximal admissible sets are then called *preferred extensions* and are one of the prevalent semantical approaches for AFs. Moreover, an argument is said to be *skeptically accepted* wrt. preferred semantics if it is accepted in every preferred extension. This reasoning problem will be the main focus of this work.

Many reasoning problems in argumentation, including skeptical preferred reasoning, are highly intractable [10], making the development of efficient algorithms a topic of significant interest [11]. 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. In recent years, state-of-the-art approaches are usually built on *satisfiability solving* (SAT) [12] and use the technique of *counter-example guided abstraction refinement* (CEGAR) [13].

In this work, we explore *shortcuts* for deciding skeptical preferred acceptance of an argument; that is, we identify conditions under which this problem becomes more tractable and can be solved more efficiently. More specifically, we examine several such conditions for both skeptical acceptance and non-acceptance. Subsequently, we conduct an experimental evaluation of these shortcuts using the ICCMA datasets. Our results demonstrate that, by first checking a combination of shortcut conditions, before resorting to the standard SAT-based CEGAR-style method, we can significantly improve the

24th International Workshop on Nonmonotonic Reasoning, July 17-19, 2026, Lisbon, Portugal

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

[1] https://argumentationcompetition.org

average runtime of state-of-the-art approaches by over 58 %. Furthermore, our findings offer valuable insight into the complexity of the ICCMA datasets and suggest ways in which the difficulty of problem instances could be increased. To summarise, the contributions of this work are:

1. We collect algorithmic shortcuts for solving the problem of skeptical preferred reasoning from the literature and propose several new shortcuts (Section 4),
2. We provide a systematic evaluation of all shortcuts in terms of applicability and runtime efficiency (Section 5).

In Section 2 we recall the necessary background on abstract argumentation, Section 3 gives an overview over state-of-the-art approaches and related work. Section 6 concludes the paper. Omitted proofs and SAT-encodings can be found in the extended version [14], available online[2].

2. Preliminaries

We consider abstract argumentation as introduced by Dung [8]. The central notion is the *abstract argumentation framework* (AF), which is a tuple $F = (A, R)$ where A is a finite set of arguments and R is the attack relation $R \subseteq A \times A$. For any two arguments $\mathbf{a}, \mathbf{b} \in A$, we say that $\mathbf{a}$ *attacks* $\mathbf{b}$ iff $(\mathbf{a}, \mathbf{b}) \in R$, sometimes also written as $\mathbf{a}R\mathbf{b}$. Furthermore, for a set $S \subseteq A$ we define the set of arguments attacked by S (attacking S) in F respectively as

$$S_F^+ = \{\mathbf{a} \in A \mid \exists \mathbf{b} \in S : \mathbf{b}R\mathbf{a}\}, \qquad S_F^- = \{\mathbf{a} \in A \mid \exists \mathbf{b} \in S : \mathbf{a}R\mathbf{b}\}.$$

If $S = \{\mathbf{a}\}$ is a singleton, we also write $\mathbf{a}_F^+$ (resp. $\mathbf{a}_F^-$). For convenience of notation, we consider in this work the labeling-based approach to semantics for AFs [15], instead of the original extension-based representation of Dung [8]. Note that both approaches are functionally equivalent [15].

For some AF $F = (A, R)$, a *labeling* is a total function $\mathscr{L} : A \mapsto \{\texttt{in}, \texttt{out}, \texttt{undec}\}$, that assigns to each argument $\mathbf{a} \in A$ one of three labels. The label `in` represents that the argument is accepted, `out` states that the argument is actively rejected and the label `undec` states that the argument is neither accepted nor rejected, i. e., its status remains undecided. For some labeling $\mathscr{L}$, we also use the notation $\mathscr{L}^{-1} : \{\texttt{in}, \texttt{out}, \texttt{undec}\} \mapsto A$, to represent the arguments assigned the given label by $\mathscr{L}$. For convenience we denote a labeling $\mathscr{L}$ as the tuple $(\mathscr{L}^{-1}(\texttt{in}), \mathscr{L}^{-1}(\texttt{out}), \mathscr{L}^{-1}(\texttt{undec}))$. We consider the following semantics in this work.

Definition 1. *Let $F = (A, R)$ be an AF and $\mathscr{L} : A \mapsto \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ is a labeling of F. Then $\mathscr{L}$ is called*

- complete *(CO) iff for every argument $\mathbf{a} \in A$ it holds that*
  1. *$\mathbf{a}$ is labeled* `in` *iff all $\mathbf{b} \in \mathbf{a}_F^-$ are labeled* `out`,
  2. *$\mathbf{a}$ is labeled* `out` *iff at least one $\mathbf{b} \in \mathbf{a}_F^-$ is labeled* `in`,
  3. *otherwise $\mathbf{a}$ is labeled* `undec`.
- grounded *(GR) iff $\mathscr{L}$ is complete and $\mathscr{L}^{-1}(\texttt{in})$ is $\subseteq$-minimal,*
- preferred *(PR) iff $\mathscr{L}$ is complete and $\mathscr{L}^{-1}(\texttt{in})$ is $\subseteq$-maximal.*

For some semantics $\sigma \in \{\text{CO}, \text{GR}, \text{PR}\}$, we denote with $\sigma(F)$ the set of all σ-labelings of the AF F. Since the grounded labeling is unique for every AF, we also denote this with $\mathscr{L}_{gr}$.

Example 1. *We consider the AF F_1 depicted in Figure 1. F_1 has three complete labelings: $\mathscr{L}_1 = (\{\mathbf{a}, \mathbf{d}\}, \{\mathbf{b}, \mathbf{c}\}, \emptyset)$, $\mathscr{L}_2 = (\{\mathbf{b}, \mathbf{d}\}, \{\mathbf{a}, \mathbf{c}\}, \emptyset)$ and $\mathscr{L}_3 = (\emptyset, \emptyset, \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\})$. For instance, in $\mathscr{L}_1$, the arguments $\mathbf{a}$ and $\mathbf{d}$ are accepted, while the remaining arguments $\mathbf{b}$ and $\mathbf{c}$ are rejected. $\mathscr{L}_1$ and $\mathscr{L}_2$ are the preferred labelings of F_1 while $\mathscr{L}_3$ is the unique grounded labeling of F_1.*

[2] https://doi.org/10.5281/zenodo.19629597

In this work, we are mainly concerned with reasoning problems wrt. the above semantics [16]. In particular, we consider the decision problem of skeptical acceptance wrt. some semantics σ, denoted as DS-σ. Formally, this is defined as follows:

DS-σ
Input: $F = (A, R), \mathbf{q} \in A$
Output: YES, iff $\mathscr{L}(\mathbf{q})$=in for all $\mathscr{L} \in \sigma(F)$, otherwise NO.

Example 2. *We continue Example 1 with the AF F_1 in Figure 1. For the preferred semantics, only the argument* **d** *is skeptically accepted. For the other semantics, no argument is skeptically accepted.*

The computational complexity of this problem has been studied extensively, we refer the interested reader to [16] for an overview. The prevalent strategy to solve these problems is reducing them to *satisfiability problems* (SAT) and use a dedicated SAT-solver for solving those, cf. [11, 17]. The particular focus of this work is the problem DS-PR, i. e., skeptical reasoning wrt. preferred semantics, which has been shown to be Π_2^P-complete [10]. 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, making it a very interesting candidate for utilizing shortcuts for solving the problem efficiently [18].

3. State-of-the-Art Approaches

Let us first take a look at the current state-of-the-art regarding algorithmic approaches for abstract argumentation, with a particular focus on skeptical preferred reasoning. As mentioned before, the prevalent strategy to solve this problem is to translate them to a SAT-problem. Notably, there also exist other approaches in the literature. Nofal et al. [19] use a depth-first backtracking procedure which searches through all labelings via binary search. Another approach is using answer set programming, used by the solver ASPARTIX [20].

In general, SAT-based approaches have performed significantly better at the recent ICCMA competitions. Specifically, approaches for DS-PR are typically based on *counter-example guided abstraction refinement* (CEGAR) [13]. That standard approach for skeptical preferred reasoning, given a problem instance $(F, \mathbf{q})$, works as follows:

(1) Search for a complete labeling $\mathscr{L}$ with $\mathscr{L}(\mathbf{q}) \neq \texttt{in}$,
   (1.1) Iteratively maximise $\mathscr{L}^{-1}(\texttt{in})$ to obtain a maximal complete labeling $\mathscr{L}'$ that satisfies $\mathscr{L}'(\mathbf{q}) \neq \texttt{in}$.
   (1.2) Check whether $\mathscr{L}'$ is $\subseteq$-maximally complete: If yes, we have a counter-example for skeptical preferred acceptance, otherwise ensure that previously encountered complete labelings can no longer be found and continue with (1).

(2) If no further complete labeling can be found, the query argument **q** is skeptically accepted.

Note that these steps are performed on the basis of SAT-encodings of the AF and the complete semantics [21]. The first argumentation solver to employ this technique is CEGARTIX [18], together

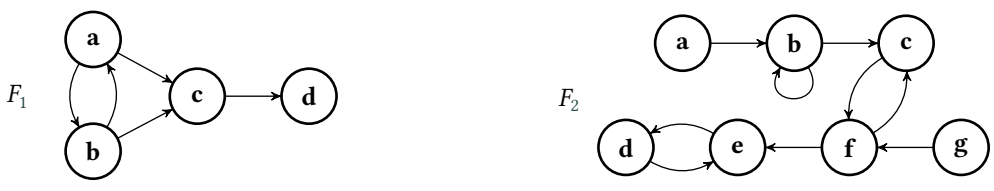

Figure 1: The AFs F_1 (left) and F_2 (right) from Examples 1 – 6.

with some shortcuts. In essence, the idea here is to search through preferred labelings for some counter-example of skeptical acceptance, by making repeated calls to the SAT-solver and using the answers to refine the problem instance with each step. Similarly, the solver μ-TOKSIA implements a pure CEGAR-style approach and explicitly uses no shortcuts to solve DS-PR [17]. Importantly, μ-TOKSIA was the fastest solver for DS-PR in the most recent ICCMA competitions. A different angle to the CEGAR-style approach is used by FUDGE [22], where the solver searches for admissible sets that are in conflict with the query argument and iteratively refines the desired counter-example based on that. Other CEGAR-style approaches for DS-PR are also used by ARGSEMSAT [23], CRUSTABRI [24] or REDUCTO [25].

Relevant is also the heuristics track at ICCMA, where the solvers are not required to be complete and measured based on the number of correctly solved instances. The approaches competing in this track make use of different shortcuts that we will also consider in our analysis in the following. In particular, the solver FARGO-LIMITED implements an algorithm that searches for an admissible set in which either the query argument or any of its attackers are contained [26]. This backtracking search is implemented as a variant of the Davis-Putnam-Logemann-Loveland (DPLL) algorithm [12]. FARGO-LIMITED implements a maximum search depth and thus does not necessarily find the correct solution. Another notable approach in that regard is HARPER++ [26], which uses the grounded labeling to predict the acceptance status of an argument regarding skeptical preferred reasoning. A similar approach is employed by the solver ARIPOTER [27]. It should be noted here, that by design these heuristic approaches may return incorrect answers, which is not in our interest here. Nevertheless, if used appropriately, the underlying ideas can still be very helpful as a shortcut for solving the reasoning problem. Ultimately, the main goal of the shortcuts that we investigate in this work is to aid sound and complete solvers and improve their runtime.

Finally, another related topic is preprocessing the AF before assembling the SAT-encoding [28]. Applying different preprocessing techniques to simplify the problem instance, before the actual computation, can be quite useful in improving the runtime, as recently illustrated by the solver REDUCTO [25].

4. Algorithmic Shortcuts for Skeptical Preferred Reasoning

Fundamentally, when deciding the problem DS-PR for some instance $(F, \mathbf{q})$, we have two scenarios.

(1) In order to conclude that the query argument $\mathbf{q}$ is skeptically accepted wrt. preferred semantics, we need to consider every preferred labeling of F and verify that $\mathbf{q}$ is labeled `in` in each of them.

(2) On the other hand, to show that the query argument $\mathbf{q}$ is not skeptically accepted wrt. preferred semantics, we require a preferred labeling of F in which $\mathbf{q}$ is either labeled `out` or `undec`.

In this section, we examine different simplifying circumstances and conditions which allow us to decide whether the query argument is skeptically accepted or not, without having to consider all preferred labelings necessarily.

Let us now consider shortcuts for solving the problem DS-PR. More specifically, we consider a *shortcut* to be a condition for DS-PR-instance $(F, \mathbf{q})$ that is sound *or* complete, but not necessarily both.

In other words, whenever the shortcut outputs YES or NO for a problem instance $(F, \mathbf{q})$ this answer is correct, but the shortcut is not guaranteed to be able to solve every problem instance. Another important aspect is, of course, that the shortcut should be significantly easier to decide than using a complete approach, which we will investigate in detail in Section 5. In total, we consider 10 such shortcuts, six of which have been known and used in the literature already. The remaining four shortcuts are newly introduced in this work. Table 1 gives an overview over these shortcuts and we discuss them in more detail in the remainder of this section.

Simple Shortcuts We start with some basic sufficient conditions for skeptical (non-)acceptance. A very simple sufficient condition for a negative instance of DS-PR is the fact that the query argument $\mathbf{q}$ is self-attacking (see S1). It is easy to see that in this case $\mathbf{q}$ cannot be accepted by any preferred

Table 1
The algorithmic shortcuts for DS-PR considered in this work. Each shortcut S poses a condition that, if satisfied, leads to the given answer for the corresponding DS-PR problem instance $(F, \mathbf{q})$. Highlighted rows are novel shortcuts, introduced in this work.

	Shortcut Condition	**Answer**
(S1)	$\mathbf{q} \in \mathbf{q}_F^-$	No
(S2)	$\mathbf{q}_F^- = \emptyset$	Yes
(S3)	$\mathscr{L}_{gr}(\mathbf{q}) = \texttt{in}$	Yes
(S4)	$\mathscr{L}_{gr}(\mathbf{q}) = \texttt{out}$	No
(S5)	$\neg\exists \mathscr{L} \in \mathrm{CO}(F) : \mathscr{L}(\mathbf{q}) = \texttt{in}$	No
(S6)	$\exists \mathscr{L} \in \mathrm{CO}(F) : \mathscr{L}(\mathbf{q}) = \texttt{out}$	No
(S7)	$\exists \mathscr{L} \in \mathrm{CO}(F) : \exists \mathbf{a} \in \mathbf{q}_F^+ : \mathscr{L}(\mathbf{a}) = \texttt{in}$	No
(S8)	$\exists \mathscr{L} \in \mathrm{CO}(F) : \exists \mathbf{a} \in \mathbf{q}_F^- : \forall \mathbf{b} \in \mathbf{a}_F^- \setminus \{\mathbf{a}\} : \mathscr{L}(\mathbf{b}) = \texttt{out}$	No
(S9)	$\exists \mathscr{L} \in \mathrm{CO}(F) : \mathscr{L}^{-1}(\texttt{in}) \neq \emptyset \wedge \neg\exists \mathscr{L} \in \mathrm{CO}(F) : \mathscr{L}^{-1}(\texttt{in}) \neq \emptyset \wedge \mathscr{L}(\mathbf{q}) \neq \texttt{in}$	Yes
(S10)	$\exists \mathscr{L} \in \mathrm{CO}(F) : \mathscr{L}^{-1}(\texttt{in}) \neq \emptyset \wedge \neg\exists \mathscr{L} \in \mathrm{CO}(F) : \mathscr{L}^{-1}(\texttt{in}) \neq \emptyset \wedge \mathscr{L}(\mathbf{q}) \neq \texttt{in} \wedge \forall \mathbf{a} \in \mathrm{sd}(F) : \mathscr{L}(\mathbf{a}) \neq \texttt{undec}$	Yes

labeling, since it would introduce a conflict to the labeling. This is a very simple polynomial time check and is used by many of the argumentation solvers discussed in Section 3. On the other hand, a simple condition for skeptical acceptance wrt. preferred semantics is whether the query argument $\mathbf{q}$ is unattacked in F (see S2). If this is the case, then the argument must be labeled `in` by every complete labeling, and thus also by every preferred labeling. This condition can easily be verified in linear time and is checked by any SAT-solver in the unit propagation step [12]. There is however the overhead of translating the instance to a SAT-encoding, which is reduced by checking this condition directly.

Example 3. *We consider the AF F_2 in Figure 1. Examining the argument* $\mathbf{b}$*, which attacks itself, it is clear that* $\mathbf{b}$ *cannot be accepted in any preferred labeling $\mathscr{L}$ of F_2, because that would directly violate the condition for* `in`*-labeled arguments. Since there is always at least one preferred labeling, we can immediately conclude that* $\mathbf{b}$ *is* not *skeptically accepted wrt. preferred semantics.*

For the arguments $\mathbf{a}$ *and* $\mathbf{g}$*, we have no attackers in F_2. That means, they may always be labeled* `in` *and since the preferred labelings maximise $\mathscr{L}^{-1}(\texttt{in})$ it is clear that they must be accepted in every preferred labeling of F_2. In all cases, we can make the decision, without actually considering the preferred labelings.*

These conditions are of course fairly trivial, so let us now consider some more general and well-known shortcuts, that can still be verified in polynomial time. In particular, we examine two shortcuts related to the grounded extension.

Example 4. *Let us again consider the AF F_2 shown in Figure 1, with the grounded labeling of F_2 given via $\mathscr{L}_{GR} = (\{\mathbf{a}, \mathbf{c}, \mathbf{g}\}, \{\mathbf{b}, \mathbf{f}\}, \{\mathbf{d}, \mathbf{e}\})$. Notice that we have two preferred labelings $\mathscr{L}_1, \mathscr{L}_2$ of F_2 with $\mathscr{L}_1^{-1}(\texttt{in}) = \{\mathbf{a}, \mathbf{c}, \mathbf{d}, \mathbf{g}\}$ and $\mathscr{L}_2^{-1}(\texttt{in}) = \{\mathbf{a}, \mathbf{c}, \mathbf{e}, \mathbf{g}\}$, respectively. As we can see, all arguments accepted by the grounded labeling $\mathscr{L}_{GR}$ are actually skeptically accepted wrt. preferred semantics. Equally relevant are the arguments labeled* `out` *by $\mathscr{L}_{GR}$, which are all not skeptically accepted wrt. preferred semantics.*

The shortcuts S3 and S4 are both concerned with the grounded labeling. Indeed, the observation from the above example holds in general [8], making it quite useful when considering the DS-PR problem in two ways. On the one hand, if the query argument $\mathbf{q}$ is labeled `in` in the grounded labeling $\mathscr{L}_{\mathrm{GR}}$, then it must be skeptically accepted wrt. preferred semantics (see S3). On the other hand, if the query argument labeled `out` (meaning it is attacked by the grounded extension), then this also holds for all preferred labelings, hence $\mathbf{q}$ is not skeptically accepted.

Note that this step is also covered in the unit propagation step of any SAT-solver before any search, but computing this directly does reduce the translation overhead and has been successfully used by various argumentation solvers, e. g., HARPER++ or REDUCTO.

Complex shortcuts While the shortcuts considered so far are very easy to compute, their applicability is usually quite limited, as we will also see later in Section 5. Consider the following example.

Example 5. *Consider again the AF F_1 in Figure 1. We have the grounded labeling $\mathscr{L}_{GR} = (\emptyset, \emptyset, \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\})$. Hence, the shortcuts S3 and S4 are of no use. The same applies to shortcuts S1 and S2, regardless of the query argument. In particular, we do actually have the argument* **d** *which is skeptically accepted in F_1 while all other arguments are not.*

As shown by the above example, preferred skeptical acceptance of an argument is not always determined by the grounded labeling. That means more sophisticated shortcuts are needed to cover these more complex cases of DS-PR. More specifically, we now consider conditions that are not computable in polynomial time, i. e., they will require a call to a SAT-solver.

It is important to note here that for these shortcuts we will consider complete labelings in all conditions, since those can be computed with a single SAT-call without having to maximise the accepted arguments. Recall that, by Definition 1, any preferred labeling is also a complete labeling. More importantly, for some complete labeling $\mathscr{L}$, we have that either $\mathscr{L}$ is a preferred labeling or there exists some preferred labeling $\mathscr{L}'$ with $\mathscr{L}^{-1}(\texttt{in}) \subsetneq \mathscr{L}'^{-1}(\texttt{in})$ and $\mathscr{L}^{-1}(\texttt{out}) \subsetneq \mathscr{L}'^{-1}(\texttt{out})$[3].

The shortcut S5 is concerned with the question whether there exists any complete labeling $\mathscr{L}$, such that $\mathscr{L}(\mathbf{q}) = \texttt{in}$. If no such labeling exists, then we can conclude that the query argument is not skeptically accepted wrt. preferred semantics (in fact, **q** would even be not credulously accepted). This idea has been used, in some form, for instance by FARGO-LIMITED or ARGSEMSAT.

Example 6. *We consider again the AF F_1 in Figure 1. The argument* **c** *is not accepted by any complete labeling. Hence, shortcut S5 is satisfied and we can conclude that* **c** *is not skeptically accepted wrt. the preferred semantics.*

Of course, this shortcut requires the argument to not be acceptable at all, which is a rather harsh restriction. Instead, one can also just check whether there is some complete labeling $\mathscr{L}$ with $\mathscr{L}(\mathbf{q}) = \texttt{out}$, see shortcut S6. If such a labeling exists, then it follows directly that there must be a preferred labeling $\mathscr{L}'$ for which we have $\mathscr{L}'(\mathbf{q}) = \texttt{out}$. The shortcut S6 is essentially a generalisation of the shortcut S4, where we only considered the minimal complete labeling, i. e., the grounded labeling. This idea has also been used by some solvers, for instance by CEGARTIX or ARGSEMSAT.

Example 7. *We consider again the AF F_1 in Figure 1. Like S5 the shortcut S6 is satisfied for the argument* **c**, *which is not skeptically accepted. Additionally, S6 is satisfied for the arguments* **a** *and* **b**, *since for both there is a complete labeling where the argument is labeled* out, *as discussed in Example 1. Note that S5 is not satisfied for* **a** *and* **b**.

Novel Shortcuts We turn now to some novel ideas for shortcuts, that have so far not been used in the literature. First of all, the shortcut S7 searches for a complete labeling for which some argument that is attacked by the query **q** is accepted. That then implies directly that **q** must be labeled out by Definition 1, which is the condition of S6. If such a labeling exists, then the query argument **q** is not skeptically accepted. While S6 and S7 are quite related, the shortcut conditions are not equivalent, since the satisfaction of S6 does not necessarily imply that S7 is satisfied, i. e., if **q** is labeled out, then there must not necessarily be an argument attacked by **q** that is labeled in (there may even be none).

More interesting is the shortcut S8, which is a proper generalisation of S6. For S8, we search for a complete labeling $\mathscr{L}$ for which **q** is not fully defended against one of its attackers. In other words, some attacker $\mathbf{a} \in \mathbf{q}^-$ of **q** must not be labeled out in $\mathscr{L}$. Note that with this approach, we do not require that the query is labeled out. In fact, this shortcut allows us to find a counter-example, even if **q** is never labeled out as illustrated by the following example.

Example 8. *We consider the AF F_3 in Figure 2 with the non-trivial complete labelings $\mathscr{L}_1 = (\{\mathbf{c}\}, \{\mathbf{b}\}, \{\mathbf{a}, \mathbf{q}\})$ and $\mathscr{L}_2 = (\{\mathbf{b}, \mathbf{q}\}, \{\mathbf{a}, \mathbf{c}\}, \emptyset)$. Notice that S6 would not be satisfied, since* **q** *is not labeled* out *by any complete labeling. However, S8 is satisfied for* **q**. *In particular,* **b** *is the only defender of* **q** *against* **a** *and in $\mathscr{L}_1$ the*

[3]Note that the same also holds if we consider admissible instead of complete labelings. However, preliminary experiments showed a slightly better performance when using complete semantics.

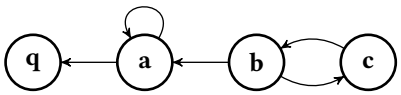

Figure 2: The AF F_3 from Example 8.

argument **b** *is labeled* out. *Hence, showing that S8 is a more general version of S6. Note also that S7 is not applicable to* **q**, *since* **q** *does not attack any arguments in* F_3.

So far, we have mostly seen shortcuts for proving that an argument is not skeptically accepted. Only S2 and S3 define conditions for positive instances of DS-PR, but these conditions are quite limited. If the query argument is not accepted in the grounded labeling, then it can be quite challenging to prove that it is skeptically accepted wrt. preferred semantics, especially if we want to avoid simply enumerating all preferred labelings. The final two shortcuts S9 and S10 in Table 1 formalise two ideas to achieve this. Note that both conditions can be verified with exactly two calls to a SAT-solver, hence there are still computationally easier than the default CEGAR-approach.

For shortcut S9, we first verify that the AF has at least one non-trivial complete labeling, i. e., a complete labeling $\mathscr{L}$ such that $\mathscr{L}^{-1}(\texttt{in}) \neq \emptyset$. If that is the case, we check whether there is a non-trivial complete labeling $\mathscr{L}$ such that $\mathscr{L}(\mathbf{q}) \neq \texttt{in}$. If no such labeling exists, then we can conclude that $\mathscr{L}(\mathbf{q}) = \texttt{in}$ holds for all non-trivial complete labelings of F and therefore **q** must be skeptically accepted wrt. preferred semantics.

Example 9. *We consider again the AF* F_1 *in Figure 1.* F_1 *has two non-trivial complete labelings* $\mathscr{L}_1, \mathscr{L}_2$ *and in both of them* **d** *is labeled* in. *Hence, S9 is satisfied for the argument* **d**.

For the final shortcut S10, we first define the notion of self-defending arguments $\texttt{sd}(F)$ of an AF F as $\texttt{sd}(F) = \{\mathbf{a} \in A \mid \mathbf{a}_F^- \setminus \mathbf{a}_F^+ = \emptyset \wedge (\mathbf{a}, \mathbf{a}) \notin R\}$. Note that the set of self-defending arguments can be computed in polynomial time. We can show that in every preferred labeling $\mathscr{L}$ of F the status of all self-defending arguments must be settled, i. e., they are either labeled in or out.

Proposition 1. *Let* $F = (A, R)$ *be an AF. Then it holds for every preferred labeling* $\mathscr{L} \in PR(F)$ *and every argument* $\mathbf{a} \in sd(F)$ *that* $\mathscr{L}(\mathbf{a}) \neq \texttt{undec}$.

For the shortcut S10, we check first that there exists at least one non-trivial complete labeling. If that is the case, we verify that in all complete labelings $\mathscr{L}$, where the status of all self-defending arguments is settled, **q** is labeled in. From that, we can conclude that **q** is skeptically accepted.

Example 10. *We consider the AF* F_4 *in Figure 3 with* $sd(F) = \{\mathbf{a}, \mathbf{d}\}$. *There is the complete labeling* $\mathscr{L}_1 = (\{\mathbf{a}\}, \{\mathbf{b}\}, \{\mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{q}\})$, *i. e., S9 is not satisfied. On the other hand, the status of* **d** *is not settled in* $\mathscr{L}_1$. *If we only consider labelings* $\mathscr{L}'$ *with both* **a** *and* **d** *settled, then the reader may verify that* $\mathscr{L}'(\mathbf{q}) = \texttt{in}$ *holds for all such labelings and therefore S10 is satisfied for* **q**.

The following example illustrates a scenario where neither S9 nor S10 are satisfied.

Example 11. *Consider the AF* F_5 *in Figure 3 with* $sd(F) = \{\mathbf{a}, \mathbf{b}\}$. **d** *and* **f** *are skeptically accepted wrt. preferred semantics. However, there is for instance the complete labeling* $\mathscr{L} = (\{\mathbf{a}\}, \{\mathbf{b}, \mathbf{c}\}, \{\mathbf{d}, \mathbf{e}, \mathbf{f}, \mathbf{g}\})$ *where both self-defending arguments are settled, and* $\mathscr{L}(\mathbf{d}) \neq \texttt{in}$. *Hence, neither S9 nor S10 are satisfied for* **d** *(or* **f***) even though they are skeptically accepted.*

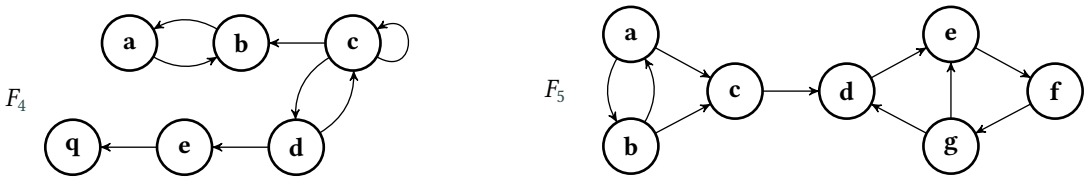

Figure 3: The AFs F_4 (left) and F_5 (right) from Examples 10 and 11.

Table 2
Statistics for all considered datasets. #AFs describes the number of instances in the dataset; $|A|$ describes the number of arguments per instance and the table displays average, median, and standard deviation; Avg. $|R|$ describes the average number of attacks in an instance; Avg. D describes the average node degree of the instances; #Yes (#No) represent the number of instances in which the query argument is (not) skeptically accepted. '*' denotes that not all reference solutions were obtainable for the dataset.

Dataset	#AFs	Avg. $\|A\|$	Med. $\|A\|$	Std. $\|A\|$	Avg. $\|R\|$	Avg. D	#Yes	#No
ICCMA'15	192	1,980	675	2,424	105,396	68	14	178
ICCMA'17	300 (350)	14,544	474	132,416	311,277	245	37*	301*
ICCMA'19	326	826	196	1,784	97,639	239	39	287
ICCMA'21	480	87,331	48,200	92,881	7,239,611	161	0*	470*
ICCMA'23	329	29,791	796	203,719	1,002,470	299	32*	281*
ICCMA'25	322	28,642	500	205,844	1,011,957	331	37*	276*

We conclude this section by demonstrating that all of the above defined shortcuts indeed constitute sufficient conditions for the skeptical (non-)acceptance of the query argument wrt. preferred semantics.

Theorem 1. *Let $F = (A, R)$ be an AF and $\mathbf{q} \in A$ is the query argument. Then, the following holds:*

1. *If any $S \in \{S2, S3, S9, S10\}$ holds for $(F, \mathbf{q})$, then $\mathbf{q}$ is skeptically accepted wrt. PR in F.*
2. *If any $S \in \{S1, S4, S5, S6, S7, S8\}$ holds for $(F, \mathbf{q})$, then $\mathbf{q}$ is not skeptically accepted wrt. PR in F.*

5. Empirical Evaluation

In the following, we systematically analyse the practicality of the aforementioned shortcuts for the problem of skeptical preferred reasoning. To that end, we have to essentially consider two dimensions:

(1) *Applicability*, i. e., for how many problem instances does the shortcut successfully provide a solution?

(2) *Performance*, i. e., how fast (in comparison to a complete solver) does the shortcut solve instances?

In general, it is clear that these two dimensions are somewhat contrary to each other, i. e., we can expect that higher applicability will lead to worse performance and vice versa. In Section 5.1 we describe the experimental setup and the considered datasets and subsequently, in Section 5.2 we present and discuss the results of our analysis.

5.1. Experimental Setup

The experimental evaluation has been conducted with the *probo2 benchmarking suite* for argumentation solvers [29]. 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 with the standard per-instance timeout of 1, 200 seconds. We use μ-toksia as the baseline solver, since it implements a pure CEGAR-style approach that explicitly uses no shortcuts [17]. For the experiments we used μ-toksia based on the Sat-solver Glucose [30][4].

Implementation We implemented each shortcut from Section 4 in C++ using standard datatypes. The shortcuts S1–S4 use simple iterative procedures, while for the remaining shortcuts S5–S10, we utilize Sat-encodings, listed in the extended version [14]. For the Sat-solver, we use Kissat 4.0.3 [31]. Notably, Kissat does not support iterative Sat-calls; therefore, it is not suitable for use by traditional CEGAR-style solvers. Our implementation is open source and available on GitHub[5].

[4]There also exist versions of μ-toksia based on CaDiCaL and CryptoMiniSat. However, preliminary experiments showed that the Glucose-version performs best overall.

[5]https://github.com/jlianSander/ascDSPR

Table 3
Overview over the applicability of shortcuts on the No-instances of all datasets. Each value is the percentage of No-instances of the dataset that can be solved by the shortcut within the timeout; I'15 – I'25 refers to the respective ICCMA datasets.

	I'15	I'17	I'19	I'21	I'23	I'25	Total
S1	14.6%	15.0%	19.9%	0.0%	0.0%	0.0%	7.1%
S4	47.2%	22.6%	41.5%	0.0%	22.1%	21.4%	21.9%
S5	98.3%	64.5%	51.9%	75.7%	57.3%	55.4%	66.3%
S6	69.1%	66.8%	99.0%	95.7%	76.9%	79.7%	83.3%
S7	39.9%	54.5%	78.1%	94.3%	53.7%	64.1%	68.6%
S8	70.8%	66.4%	99.0%	95.3%	76.5%	78.6%	77.7%
μ-T	100.0%	90.7%	100.0%	80.2%	95.0%	96.7%	92.0%
#No	178	301	287	470	281	276	1,793

Table 4
Overview over the applicability of shortcuts on the Yes-instances of all datasets. Each value is the percentage of Yes-instances of the dataset that can be solved by the shortcut within the timeout; I'15 – I'25 refers to the respective ICCMA datasets.

	I'15	I'17	I'19	I'21	I'23	I'25	Total
S2	35.7%	18.9%	41.0%	–	0.0%	0.0%	17.6%
S3	57.1%	86.5%	87.2%	–	59.4%	54.1%	71.1%
S9	64.3%	94.6%	92.3%	–	93.8%	89.2%	90.0%
S10	64.3%	100.0%	94.9%	–	100.0%	91.9%	93.7%
μ-T	100.0%	97.3%	100.0%	–	100.0%	100.0%	99.4%
#Yes	14	37	39	0	32	37	159

Datasets For our experiments, we consider the benchmark datasets of ICCMA, which is a well-established competition for argumentation solvers. These datasets are widely used and provide a broad variety of AFs, covering various graph generation approaches as well as AFs from real-world data. We consider all six datasets from ICCMA'15 to ICCMA'25 [32, 33, 34, 35]. Table 2 gives an overview over the considered datasets[6]. In general, it should be noted that all datasets are heavily biased towards negative instances (1,793 vs. 159 in total).

5.2. Results

Applicability Let us first take a look at the applicability of shortcuts, for each dataset individually. The following results also give interesting insights into the datasets and the difficulty of the included problem instances. Tables 3 and 4 show how many instances of each dataset were solved by the shortcuts, separately for the No- and Yes-instances.

For the No-instances, shown in Table 3, the simple shortcuts S1 and S4 are satisfied rarely. This is not surprising, especially for the latest competitions, where the problem instances have been designed to avoid self-attacking or unattacked query arguments [34, 35]. On the other hand, the shortcuts S5–S8 show a remarkably high applicability and are able to solve the majority of instances in all datasets. In particular, for ICCMA'19 and ICCMA'21, we can solve 99 % and over 95 % of No-instances with the shortcuts S6 and S8. Notably, for ICCMA'21 the shortcuts S6–S8 are even able to solve more instances

[6]Note that while the ICCMA'17 dataset contains 300 unique AFs, some of them are used twice, with a different query argument, so the total number of problem instances is 350 [33].

Table 5
Runtime comparison of all shortcuts over all 1999 instances. #AFs is the number of instances solved by both, the shortcut and μ-TOKSIA; RT is the cumulated runtime of the shortcut over these instances in seconds; RT (μ-T) is the runtime of μ-TOKSIA over the same set of instances; #VBS is the number of instances where the respective shortcut was faster than μ-TOKSIA.

Shortcut	#AFs	RT	RT (μ-T)	#VBS
S1	127	49.03	1,011.87	127
S2	28	1.35	2.58	23
S3	113	8.85	15.40	91
S4	383	294.70	385.23	361
S5	1,070	17,805.44	87,940.16	930
S6	1,400	60,235.92	99,513.48	1,095
S7	1,144	57,568.48	94,083.76	854
S8	1,299	60,477.12	95,209.53	978
S9	143	75.63	105.45	104
S10	148	79.20	106.09	120

than μ-TOKSIA. Overall, these numbers show that many of the datasets are actually not as difficult as one might expect, in particular the ICCMA'21 dataset where the instances are significantly larger.

For the YES-instances, see Table 4, we get an even more drastic picture. For the ICCMA'17 and ICCMA'23 datasets, S10 is able to solve all (known) positive instances and in the case of ICCMA'17 it even solves more instances than μ-TOKSIA. Similarly, S9 is satisfied for over 90 % of instances.

Runtime Performance Let us now look at whether the shortcuts actually improve on the runtime compared to baseline approach μ-TOKSIA. An overview over the runtime for each shortcut on solved instances in comparison to μ-TOKSIA is shown in Table 5. For each shortcut, the table shows the total number of instances solved within the time limit and, in comparison, the runtime of μ-TOKSIA on the same instances. We also consider the *virtual best solver* (VBS), which takes for each instance the runtime of the fastest approach. To dampen the impact of random variance, an approach gets attributed a contribution to the VBS if its runtime is within 5 % of the fastest approach. As we can see, all shortcuts are significantly faster than μ-TOKSIA. Particularly interesting here is the shortcut S5, which is able to solve 1070 of the 1, 999 instances and is in total over 81 % faster than μ-TOKSIA on these instances. Even the more sophisticated shortcuts S6, S7 and S8, which solve more instances than S5, are still significantly faster than μ-TOKSIA, with a runtime reduction of about 40 %. Notably, the trivial shortcut S1 allows for a runtime reduction of over 95 %, suggesting that this simple case of a self-attacking argument is not captured in the simplification of the SAT-solver. The other polynomial-time shortcuts are less impactful, likely because these cases are covered by the unit propagation step of the SAT-solver.

Finally, the shortcuts S9 and S10 are the most complex approaches, each requiring two SAT-calls. Nevertheless, both are still faster than μ-TOKSIA, by 28.3 % and 25.3 % respectively. Together with the fact that both shortcuts are able to solve nearly all of the 159 (known) YES-instances, this is actually a surprisingly good result.

Combination with μ-TOKSIA To get a proper view on how the shortcuts perform in practice, we need to also take into consideration the instances where the shortcut is not able to solve the problem instance. For that, we consider a combination of a shortcut and the baseline solver μ-TOKSIA. Meaning, if the shortcut does not successfully solve the problem instance, we add, in addition to that, the runtime of μ-TOKSIA for solving this instance. This approach penalises shortcuts that may be very fast, but have low applicability and allows for a fairer comparison between the shortcuts.

Table 6 (a) shows the results for each shortcut applied in combination with μ-TOKSIA. We consider the

Table 6
Runtime comparison for shortcuts in combination with μ-TOKSIA. RT is the total runtime for solved instances; #TO is the number of timeouts; PAR-2 is the average runtime per instance where timeouts are counted double; #VBS is the number of instances contributed to the VBS.

(a) Comparison of individual shortcuts.

Approach	RT	#TO	PAR-2	#VBS
(S1, μ-T)	348,059.87	191	288.14	213
(S2, μ-T)	350,268.55	192	289.85	120
(S3, μ-T)	353,553.30	193	292.09	182
(S4, μ-T)	342,395.56	184	281.10	410
(S5, μ-T)	**232,854.97**	**85**	**138.44**	**686**
(S6, μ-T)	337,699.51	103	178.25	605
(S7, μ-T)	366,543.95	117	195.72	460
(S8, μ-T)	359,660.48	111	192.50	621
(S9, μ-T)	511,168.56	225	339.99	91
(S10, μ-T)	508,628.18	224	338.45	104
μ-TOKSIA	347,389.17	192	289.04	491
VBS	105,102.80	48	81.39	-

(b) Comparison of selected cascading combinations.

Solver Combination	RT	#TO	PAR-2
(S5, S7, S8, S10, μ-T)	354,848.62	64	118.58
(S5, S7, S8, μ-T)	333,289.17	**65**	**119.79**
(S5, S7, S6, μ-T)	332,326.15	67	121.24
(S5, S7, μ-T)	276,839.00	67	121.42
(S5, S6, μ-T)	279,524.88	72	124.04
(S5, S8, μ-T)	287,288.56	72	124.61
(S5, μ-T)	**232,854.97**	85	138.44
(S5, S10, μ-T)	287,523.05	89	149.04
μ-TOKSIA	347,389.17	192	289.04

total runtime on solved instances, the number of timeouts, the PAR-2 score, i. e., the average runtime with timeouts counted double, and the number of instances contributed to the VBS. Surprisingly, the polynomial-time shortcuts are not really effective and only the shortcut S4 does yield a slight improvement compared to pure μ-TOKSIA. Most remarkable is the result for S5, where we can improve the total runtime of μ-TOKSIA by 33 %, while even solving 107 more instances than μ-TOKSIA on its own. Consequently, the PAR-2 score improves by over 52 %. While S6–S8 do not perform as well as S5, they still showcase a significant improvement over μ-TOKSIA. Note that the total runtime on solved instances of these shortcuts is fairly close to that of μ-TOKSIA, however in that time they solve between 75–89 more instances. The results suggest that solving the conditions of S6–S8 is more difficult for the SAT-solver, compared to S5.

The shortcuts S9 and S10 are an interesting case. While their results are technically worse than pure μ-TOKSIA, this does not paint the full picture. The problem here is simply that the datasets are heavily biased towards negative instances, which means trying to verify that the query argument is skeptically accepted is usually a waste of time. These shortcuts can however still be very useful, if utilized correctly. We will get into that in more detail in the following paragraph. In regard of this, note also that the VBS shows a theoretical best PAR-2 score of 81.39, showing that there is a lot of potential for improvement by choosing the right shortcuts and when to apply them, for each instance.

Cascading Combinations So far, we have considered applying a single shortcut for a problem instance before falling back to the complete solver to solve the instance. However, we can of course consider multiple different shortcuts, and even use them in different order, before reverting back to μ-TOKSIA. Basically, we consider a sequence $(S_1, \dots, S_n)$ of shortcuts, called a *cascading combination*. When given a problem instance, we check each shortcut S_i, in the prescribed order, and immediately return the result, if one of the shortcuts is satisfied. In case none of the conditions is satisfied, we continue with the next and finally revert back to μ-TOKSIA to compute the result.

Since there are over 9.8 million different variations for the given shortcuts, we only consider a few selected variants here. In general, there are multiple factors to take into account when thinking about reasonable combinations of shortcuts. For simplicity, we will disregard the trivial shortcuts S1 and S2 in the following, since their performance improvement were marginal with very limited applicability. To attain a suitable combination of shortcuts, we will pursue the following objectives. First of all, we want

to include shortcuts for both YES- and NO-instances. Secondly, we obviously want to give preference to shortcuts with high applicability and good runtime performance. Lastly, we want to maximise the coverage of instances by including shortcuts that pose different types of conditions.

Given the fact that there are significantly more NO- than YES-instances, it is only reasonable to check the NO-shortcuts first, to minimize the amount of times the more costly YES-shortcuts are to be computed. Now, S5 is clearly the fastest shortcut for any NO-instances and, in addition to that, also showed a high applicability, making it an ideal candidate for the NO-instances. Notably, S6, S7 and S8 also performed reasonably well, while posing conditions quite different from S5. Regarding the YES-instances, S10 turned out to be the best performing shortcut.

Table 6 (b) shows a runtime comparison for a selection of cascading shortcut combinations. The approaches are ordered by their PAR-2 score. We consider the combination (S5, μ-TOKSIA) as the baseline to improve upon here, since that was the best single-shortcut approach, cf. Table 6 (a). The first observation we can make is that simply combining the best NO-shortcut (S5) with the best YES-shortcut (S10) does actually yield a worse PAR-2 score. On the other hand, combining S5 with some other NO-shortcut does increase the coverage and leads to a significant improvement of the PAR-2 score, compared to the S5-baseline. In particular, the combination of S5 and S7 does yield the best result in that regard. By adding the shortcuts S8 and S10 to that, we can then achieve the best PAR-2 score and thus reduce the number of timeouts even further to just 64 instances.

To summarise, utilizing a variety of the above described shortcuts, in particular the more complex shortcuts that cannot be computed in polynomial time, significantly improves the runtime of a solver for skeptical preferred reasoning. Notably, these shortcuts have not really been used by any solvers that participated in the recent ICCMA competitions, but our results here show clearly that integrating them into a solver outperforms state-of-the-art approaches.

6. Conclusion

In this work, we considered the problem of skeptical preferred reasoning, i. e., the question whether some argument is accepted in every preferred labeling of an AF. More specifically, we examined algorithmic shortcuts that are computationally easier to solve and still allow us to answer this question. We considered a total of 10 shortcuts, including four new ones, that characterise conditions for both positive as well as negative instances of skeptical acceptance. As our experimental evaluation shows, many of these shortcuts have a surprisingly high applicability in the standard ICCMA datasets. Most importantly, we showed that using a selection of these shortcuts, before falling back to a sound and complete solver, significantly improves the runtime performance and allows us to solve significantly more problem instances within the time limit. In particular, a combination of three of our novel shortcuts and one of the existing shortcuts showed the best performance and significantly improves the PAR-2 score by over 58 %, compared to state-of-the-art approaches.

It should also be noted that many of the shortcuts discussed here can of course be directly applied or adjusted to other semantics or reasoning problems. We leave a detailed analysis of this for future work and only note here that, for instance, the semi-stable semantics [36] would be an interesting candidate for utilizing shortcuts. Regarding future work, there are multiple interesting directions to explore. In our analysis, we only considered a sequential application of shortcuts. It would however also make sense to check these conditions in parallel, to improve the runtime performance even more. Alternatively, one could investigate efficient heuristics for determining which shortcut to apply to a given problem instance. Our results give direct insight into the composition of the ICCMA datasets, showing that they contain many problem instances that are actually not that hard to solve. Dissecting the shortcut conditions may be helpful in developing new graph generation approaches that specifically construct harder problem instances, similar to the work of Kuhlmann and Thimm [37]. Finally, algorithmic shortcuts of different sorts are also used in other field, for instance in SAT-solving [12]. It would therefore be interesting to investigate whether some of the shortcuts considered in this work can be extrapolated to these fields, or vice versa.

Acknowledgments

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

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] K. Atkinson, P. Baroni, M. Giacomin, A. Hunter, H. Prakken, C. Reed, G. R. Simari, M. Thimm, S. Villata, Towards artificial argumentation, AI Mag. 38 (2017) 25–36.
[2] P. Baroni, D. Gabbay, M. Giacomin, L. van der Torre (Eds.), Handbook of Formal Argumentation, Vol.1, College Publications, 2018.
[3] F. Leofante, H. Ayoobi, A. Dejl, G. Freedman, D. Gorur, J. Jiang, G. Paulino-Passos, A. Rago, A. Rapberger, F. Russo, X. Yin, D. Zhang, F. Toni, Contestable AI needs computational argumentation, in: P. Marquis, M. Ortiz, M. Pagnucco (Eds.), Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, 2024.
[4] T. J. M. Bench-Capon, H. Prakken, G. Sartor, Argumentation in legal reasoning, in: G. R. Simari, I. Rahwan (Eds.), Argumentation in Artificial Intelligence, Springer, 2009, pp. 363–382.
[5] C. Cayrol, M. Lagasquie-Schiex, Bipolarity in argumentation graphs: Towards a better understanding, Int. J. Approx. Reason. 54 (2013) 876–899.
[6] P. M. Dung, R. A. Kowalski, F. Toni, Assumption-based argumentation, in: G. R. Simari, I. Rahwan (Eds.), Argumentation in Artificial Intelligence, Springer, 2009, pp. 199–218.
[7] G. Brewka, S. Ellmauthaler, H. Strass, J. P. Wallner, S. Woltran, Abstract dialectical frameworks, in: Handbook of Formal Argumentation, Vol.1, College Publications, 2018, pp. 237–286.
[8] 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.
[9] P. Baroni, M. Caminada, M. Giacomin, Abstract argumentation frameworks and their semantics, in: P. Baroni, D. Gabbay, M. Giacomin, L. van der Torre (Eds.), Handbook of Formal Argumentation, Vol.1, College Publications, 2018, pp. 159–236.
[10] P. E. Dunne, T. J. M. Bench-Capon, Coherence in finite argument systems, Artif. Intell. 141 (2002) 187–203.
[11] F. Cerutti, S. A. Gaggl, M. Thimm, J. P. Wallner, Foundations of implementations for formal argumentation, in: P. Baroni, D. Gabbay, M. Giacomin, L. van der Torre (Eds.), Handbook of Formal Argumentation, Vol.1, College Publications, 2018, pp. 689–767.
[12] A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2009.
[13] 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.
[14] L. Bengel, J. Sander, M. Thimm, On Algorithmic Shortcuts for Skeptical Preferred Reasoning in Abstract Argumentation (Extended Version), Zenodo, 2026.
[15] M. W. A. Caminada, D. M. Gabbay, A logical account of formal argumentation, Stud Logica 93 (2009) 109–145.
[16] W. Dvorák, P. E. Dunne, Computational problems in formal argumentation and their complexity, in: P. Baroni, D. Gabbay, M. Giacomin, L. van der Torre (Eds.), Handbook of Formal Argumentation, Vol.1, College Publications, 2018, pp. 631–688.
[17] A. Niskanen, M. Järvisalo, -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.

[18] W. Dvorák, M. Järvisalo, J. P. Wallner, S. Woltran, Complexity-sensitive decision procedures for abstract argumentation, Artif. Intell. 206 (2014) 53–78.
[19] S. Nofal, K. Atkinson, P. E. Dunne, Algorithms for decision problems in argument systems under preferred semantics, Artificial Intelligence 207 (2014) 23–51.
[20] 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.
[21] 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.
[22] 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.
[23] 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.
[24] J. Lagniez, E. Lonca, J. Mailly, A sat-based approach for argumentation dynamics, in: M. Dastani, J. S. Sichman, N. Alechina, V. Dignum (Eds.), Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, ACM, 2024, pp. 2351–2353.
[25] L. Bengel, J. Sander, M. Thimm, A reduct-based approach to skeptical preferred reasoning in abstract argumentation, in: Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning, KR 2025, 2025.
[26] M. Thimm, Heuristic algorithms for credulous and sceptical reasoning problems in abstract argumentation, Journal of Logic and Computation 35 (2025).
[27] J. Delobelle, J.-G. Mailly, J. Rossit, Aripoter: Solvers for approximate reasoning based on grounded semantics, International Journal of Approximate Reasoning 189 (2026) 109599.
[28] 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.
[29] 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.
[30] G. Audemard, L. Simon, On the glucose SAT solver, Int. J. Artif. Intell. Tools 27 (2018) 1840001:1–1840001:25.
[31] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, F. Pollitt, CaDiCaL, Gimsatul, IsaSAT and Kissat entering the SAT Competition 2024, in: M. Heule, M. Iser, M. Järvisalo, M. Suda (Eds.), Proc. of SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions, volume B-2024-1 of *Department of Computer Science Report Series B*, University of Helsinki, 2024, pp. 8–10.
[32] M. Thimm, S. Villata, The first international competition on computational models of argumentation: Results and analysis, Artificial Intelligence 252 (2017) 267–294.
[33] 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).
[34] S. Bistarelli, L. Kotthoff, J. Lagniez, E. Lonca, J. Mailly, J. Rossit, F. Santini, C. Taticchi, The third and fourth international competitions on computational models of argumentation: Design, results and analysis, Argument Comput. 16 (2025) 236–299.
[35] M. Järvisalo, T. Lehtonen, A. Niskanen, ICCMA 2023: 5th international competition on computational models of argumentation, Artif. Intell. 342 (2025) 104311.
[36] M. Caminada, Semi-stable semantics, in: P. E. Dunne, T. J. M. Bench-Capon (Eds.), Computational Models of Argument: Proceedings of COMMA 2006, volume 144 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2006, pp. 121–130.
[37] I. Kuhlmann, M. Thimm, A discussion of challenges in benchmark generation for abstract argumentation, in: Proceedings of the First International Workshop on Argumentation and Applications, 2023, volume 3472 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 78–84.