Algorithms for Computing the Set of Acceptable Arguments

Lars Bengel[a], Matthias Thimm[a,*], Federico Cerutti[b], Mauro Vallati[c]

[a] *University of Hagen, Germany*
[b] *University of Brescia, Italy*
[c] *University of Huddersfield, United Kingdom*

Abstract

We investigate the computational problem of determining the set of acceptable arguments in abstract argumentation wrt. credulous and skeptical reasoning under grounded, complete, stable, and preferred semantics. In particular, we investigate the computational complexity of that problem and its verification variant, and develop several algorithms for all problem variants, including two baseline approaches based on iterative acceptability queries and extension enumeration, and some optimised versions. We experimentally compare the runtime performance of these algorithms: our results show that our newly optimised algorithms significantly outperform the baseline algorithms in most cases.

1. Introduction

In abstract argumentation [1], an argument a is skeptically (credulously) accepted wrt. some semantics σ, if it belongs to all (at least one) σ-extensions, respectively. Work on algorithms for solving reasoning problems in abstract argumentation—see e. g. the survey [2]—so far focused on deciding acceptability for a single query argument, or determining a single or all σ-extensions. However, the computational problem of directly computing the set of all acceptable arguments (wrt. either credulous or skeptical reasoning) has not been considered yet explicitly in the literature. Of course, this problem can be solved by reducing

*Corresponding author
Email addresses: lars.bengel@fernuni-hagen.de (Lars Bengel), matthias.thimm@fernuni-hagen.de (Matthias Thimm), federico.cerutti@unibs.it (Federico Cerutti), m.vallati@hud.ac.uk (Mauro Vallati)

it to the problems mentioned above. For example, one can determine the set of all credulously accepted arguments by first computing all σ-extensions and then taking their union. In this paper, we ask whether this approach is appropriate for the problem and whether other approaches provide superior performance.

Having efficient algorithms for computing the set of credulously or skeptically accepted arguments is of practical importance. For instance, consider CISpaces [3], an argumentation-based research-grade prototype for supporting intelligence analysts in their sense-making process, under consideration for transitioning into a commercial product. It supports intelligence analysts in sense-making in assessing competing hypotheses, where each hypothesis is a preferred extension. Knowing whether specific arguments are not in any possible extensions—the dual problem of credulous acceptance—or knowing whether arguments are skeptically justified is of great service as also discussed in [4]. It allows human analysts to reduce their cognitive burden by consciously deciding whether or not to look more into a specific argument they made in their sense-making process.

In this paper, we first look at the theoretical complexity of the problem of verifying whether a given set of arguments is exactly the set of acceptable arguments wrt. both credulous and skeptical reasoning under grounded, complete, stable, and preferred semantics. Our results mirror similar previous results [5] in that, for example, the verification problem for grounded semantics under both credulous and skeptical reasoning is in P, while the verification problem for skeptical reasoning for preferred semantics is DP2-complete (see Section 3 for definitions of the complexity classes). While the proofs of membership follow easily from existing results [5], the hardness proofs require some novel reduction techniques and insights.

In addition to the theoretical analysis, we present and analyse concrete algorithms for determining the set of acceptable arguments wrt. preferred and stable semantics and both reasoning modes.[1] We first consider two baseline algorithms. The first one computes the set of acceptable arguments by simply iterating over all arguments and solving the corresponding decision problem for each argument. To be comparable with our other algorithms, we use simple SAT-solver based algorithms in the spirit of μ-toksia [6] for these decision problems. Our second baseline method simply enumerates all extensions and then takes their union (for

[1] We do not consider grounded semantics and skeptical reasoning with complete semantics, since these problems are polynomial; we also do not consider credulous reasoning with complete semantics explicitly, since this is equivalent to credulous reasoning with preferred semantics.

credulous reasoning) or intersection (for skeptical reasoning). We improve upon this second algorithm by defining an optimised version that enumerates only a subset of all extensions that already cover the whole set of acceptable arguments. Finally, we describe a fourth algorithm that uses a MAXSAT-solver to maximise the number of newly discovered acceptable arguments in each call. We provide an extensive experimental evaluation of these four algorithms on all benchmarks from all ICCMA[2] competitions. Our results consistently show that the two optimised algorithms significantly outperform the baseline algorithms.

To summarise, the contributions of this paper are as follows.

1. We characterise the computational complexity of the verification problem of checking whether a given set is exactly the set of acceptable arguments wrt. both credulous and skeptical reasoning and the grounded, complete, stable, and preferred semantics (Section 3).
2. We present four (SAT-based) algorithms for solving the problem of determining the set of acceptable arguments wrt. credulous reasoning and the stable and preferred semantics (Section 4)
3. We present four (SAT-based) algorithms for solving the problem of determining the set of acceptable arguments wrt. skeptical reasoning and the stable and preferred semantics (Section 5)
4. We report on an experimental evaluation of the runtime performance of the above four algorithms (Section 6).

We provide necessary preliminaries in Section 2 and conclude in Section 7. All proofs of technical results can be found in the appendix.

This paper is an extended version of the conference paper [7]. This version contains all proofs of technical results, an extensive presentation of the contributions, algorithms for both credulous and skeptical reasoning wrt. grounded, complete, stable, and preferred semantics (the previous version only considered credulous reasoning wrt. to complete semantics), and a thorough experimental evaluation of algorithms for both credulous and skeptical reasoning wrt. the grounded, complete, stable, and preferred semantics on data sets from ICCMA 2015–2023 (the previous version only covered credulous reasoning wrt. to complete semantics and the data sets from ICCMA 2015–2019).

[2] `http://argumentationcompetition.org`

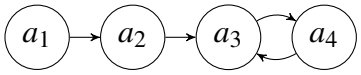


Figure 1: The abstract argumentation framework AF_1 from Example 1.

2. Preliminaries

An *abstract argumentation framework* AF is a tuple $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ where A is a set of arguments and R is a relation $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$. For two arguments $a, b \in \mathsf{A}$ the relation $a\mathsf{R}b$ means that argument a attacks argument b. For $a \in \mathsf{A}$ define $a^- = \{b \mid b\mathsf{R}a\}$ and $a^+ = \{b \mid a\mathsf{R}b\}$. We say that a set $S \subseteq \mathsf{A}$ *defends* an argument $b \in \mathsf{A}$ if for all a with $a\mathsf{R}b$ then there is $c \in S$ with $c\mathsf{R}a$.

Semantics are given to abstract argumentation frameworks by means of extensions [1]. An extension E is a set of arguments $E \subseteq \mathsf{A}$ intended to represent a coherent point of view on the argumentation modelled by AF. Arguably, the most important property of an extension is its admissibility. An extension E is called *admissible* if and only if (1) E is *conflict-free*, i. e., there are no arguments $a, b \in E$ with $a\mathsf{R}b$ and (2) E *defends* every $a \in E$, and it is called *complete* (CO) if, additionally, it satisfies (3) if E defends a then $a \in E$.

Different types of classical semantics can be phrased by imposing further constraints. In particular, a complete extension E

- is *grounded* (GR) if and only if E is minimal;
- is *preferred* (PR) if and only if E is maximal; and
- is *stable* (ST) if and only if $\mathsf{A} = E \cup \{b \mid \exists a \in E : a\mathsf{R}b\}$.

All statements on minimality/maximality are meant to be with respect to set inclusion. Note that the grounded extension is uniquely determined and that stable extensions may not exist [1].

Example 1. Consider the abstract argumentation framework AF_1 depicted as a directed graph in Figure 1. In AF_1 there are three complete extensions E_1, E_2, E_3 defined via $E_1 = \{a_1\}$, $E_2 = \{a_1, a_3\}$, and $E_3 = \{a_1, a_4\}$. E_1 is also grounded and E_2 and E_3 are both stable and preferred.

Let $\sigma \in \{\mathsf{CO}, \mathsf{GR}, \mathsf{ST}, \mathsf{PR}\}$ be some semantics and $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ an abstract argumentation framework. Then, an argument $a \in \mathsf{A}$ is *skeptically accepted* in AF, denoted by $\mathsf{AF} \models^{\mathrm{s}}_{\sigma} a$, if a is contained in *every* σ-extension. An argument $a \in \mathsf{A}$

is *credulously accepted* in AF, denoted by $\mathsf{AF} \models^{\mathrm{c}}_{\sigma} a$, if a is contained in *some* σ-extension. Define $\mathsf{Acc}^{s}_{\sigma}(\mathsf{AF}) = \{a \in \mathsf{A} \mid \mathsf{AF} \models^{s}_{\sigma} a\}$ and $\mathsf{Acc}^{c}_{\sigma}(\mathsf{AF}) = \{a \in \mathsf{A} \mid \mathsf{AF} \models^{c}_{\sigma} a\}$ to be the sets of skeptically and credulously accepted arguments in AF, respectively. Observe that $\mathsf{Acc}^{s}_{\sigma}(\mathsf{AF}) \subseteq \mathsf{Acc}^{c}_{\sigma}(\mathsf{AF})$ for all semantics and abstract argumentation frameworks, except for $\sigma = \mathsf{ST}$ and an argumentation framework AF' that possesses no stable extension. In the latter case $\mathsf{Acc}^{s}_{\sigma}(\mathsf{AF}') = \mathsf{A}$ and $\mathsf{Acc}^{c}_{\sigma}(\mathsf{AF}') = \emptyset$ by definition.

In the remainder of the paper, we consider the computational problem of determining the sets $\mathsf{Acc}^{s}_{\sigma}(\mathsf{AF})$ and $\mathsf{Acc}^{c}_{\sigma}(\mathsf{AF})$, respectively. Note that these exact problems have not been investigated before, to the best of our knowledge, in terms of computational complexity and algorithms. Previous studies and algorithms either focus on a single acceptability problem, such as deciding whether $\mathsf{AF} \models^{\mathrm{x}}_{\sigma} a$ is true for $x \in \{s, c\}$ and some argument $a \in \mathsf{A}$, or computing one or all extensions (as done in the ICCMA series of argumentation competitions).

3. Complexity of Computing the Set of Acceptable Arguments

We assume familiarity with basic concepts of computational complexity and basic complexity classes such as P, NP and coNP, see [8] for an introduction. Recall that every decision problem can be represented as a language L that contains exactly those instances to the problem with answer "yes." A complexity class can then be represented by the languages of those problems it contains. We will make use of the complexity class DP, which is defined as $\mathsf{DP} = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP}, L_2 \in \mathsf{coNP}\}$. So DP contains the intersections of a language in NP and a language in coNP. We also need the following class $\mathsf{DP2} = \{L_1 \cap L_2 \mid L_1 \in \mathsf{NP}^{\mathsf{NP}}, L_2 \in \mathsf{coNP}^{\mathsf{NP}}\}$ where $\mathsf{NP}^{\mathsf{NP}}$ is the class of problems that can be solved by a non-deterministic Turing machine in polynomial time that has access to an NP oracle and $\mathsf{coNP}^{\mathsf{NP}}$ is the class of problems where the complement can be solved by a non-deterministic Turing machine in polynomial time that has access to an NP oracle. $\mathsf{NP}^{\mathsf{NP}}$ is also written as Σ^P_2 and $\mathsf{coNP}^{\mathsf{NP}}$ as Π^P_2. So $\mathsf{DP2}$ contains those languages that are intersections of a language in Σ^P_2 and a language in Π^P_2.

In this section, we are interested in the computational complexity of the following decision problem:

$\mathsf{ACC}^{x}_{\sigma}$ **Input**: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ and $E \subseteq \mathsf{A}$
Output: TRUE iff $E = \mathsf{Acc}^{x}_{\sigma}(\mathsf{AF})$.

for a semantics σ and $x \in \{s, c\}$.

We start with the tractable problems.

Proposition 1. $\mathsf{ACC}^s_{\mathsf{GR}}$, $\mathsf{ACC}^c_{\mathsf{GR}}$, *and* $\mathsf{ACC}^s_{\mathsf{CO}}$ *are in P.*

Many other problems are DP-complete.

Proposition 2. $\mathsf{ACC}^c_{\mathsf{CO}}$, $\mathsf{ACC}^c_{\mathsf{PR}}$, *and* $\mathsf{ACC}^c_{\mathsf{ST}}$ *are DP-complete.*

Proposition 3. $\mathsf{ACC}^s_{\mathsf{ST}}$ *is DP-complete.*

Skeptical inference with preferred semantics is (unsurprisingly) on the second level of the polynomial hierarchy.

Proposition 4. $\mathsf{ACC}^s_{\mathsf{PR}}$ *is DP2-complete.*

The results from above also allow us to easily provide an upper bound for the computational complexity of the functional problem of determining the set of acceptable arguments. For the following result, recall that $\mathsf{FNP}^{\mathsf{DP}[1]}$ is the complexity class of functional problems that can be solved by a non-deterministic Turing machine running in polynomial time that can call a DP-oracle for a constant number of times. The class $\mathsf{FNP}^{\mathsf{DP2}[1]}$ is defined analogously. Let furthermore $\mathsf{EnumACC}^{\sigma}_{x}$ denote the problem of computing $\mathsf{Acc}^{x}_{\sigma}(\mathsf{AF})$.

Corollary 1. *Let* AF *be an abstract argumentation framework.*

1. *The problems* $\mathsf{EnumACC}^{\mathsf{GR}}_s$, $\mathsf{EnumACC}^{\mathsf{GR}}_c$, $\mathsf{EnumACC}^{\mathsf{CO}}_s$ *are in FP, respectively.*
2. *The problems* $\mathsf{EnumACC}^{CO}_c$, $\mathsf{EnumACC}^{\mathsf{PR}}_c$, $\mathsf{EnumACC}^{\mathsf{ST}}_c$, $\mathsf{EnumACC}^{\mathsf{ST}}_s$ *are in* $FNP^{DP[1]}$, *respectively.*
3. *The problem* $\mathsf{EnumACC}^{\mathsf{PR}}_s$ *is in* $FNP^{DP2[1]}$.

Table 1 summarises the results of this section and also lists known complexity results for deciding credulous reasoning (Cred_σ for deciding whether on input AF and a it holds $\mathsf{AF} \models^{\mathrm{c}}_{\sigma} a$) and skeptical reasoning (Skep_σ for deciding whether on input AF and a it holds $\mathsf{AF} \models^{\mathrm{s}}_{\sigma} a$) for reference, cf. [5]. Membership proofs (see Appendix A) for the new results do derive quite naturally from proofs of those previously known results, while hardness proofs required some new and challenging techniques.

σ	Cred_σ	Skep_σ	ACC^c_σ	ACC^s_σ	$\mathsf{EnumACC}^c_\sigma$	$\mathsf{EnumACC}^s_\sigma$
GR	P	P	P	P	FP	FP
CO	NP-c	P	DP-c	P	$\mathsf{FNP}^{\mathsf{DP}[1]}$	FP
ST	NP-c	coNP-c	DP-c	DP-c	$\mathsf{FNP}^{\mathsf{DP}[1]}$	$\mathsf{FNP}^{\mathsf{DP}[1]}$
PR	NP-c	Π^{P}_2-c	DP-c	DP2-c	$\mathsf{FNP}^{\mathsf{DP}[1]}$	$\mathsf{FNP}^{\mathsf{DP}[2]}$

Table 1: Overview in existing (columns Cred_σ and Skep_σ) and new (remaining columns) complexity results; all statements are membership statements, except where a suffix "-c" indicates completeness statements.

4. Algorithms for Credulous Reasoning

We will now investigate some algorithms that compute the set $\mathsf{Acc}^c_\sigma(\mathsf{AF})$ for $\sigma \in \{\mathsf{CO}, \mathsf{ST}, \mathsf{PR}\}$. We do not consider grounded semantics here as $\mathsf{Acc}^c_{\mathsf{GR}}(\mathsf{AF})$ can be computed in polynomial time anyway, cf. see Proposition 1.

We will develop reduction-based algorithms [9, 2] and leverage SAT-solving technologies. Our encodings of acceptability problems into SAT are based on the encodings proposed initially in [10] and used in modern SAT-based argumentation solvers, see e. g. [9, 11]. We consider complete semantics first. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ be an abstract argumentation framework. For each argument $a \in \mathsf{A}$ we introduce three propositional variables $\mathtt{in}_a, \mathtt{out}_a, \mathtt{undec}_a$ which model the cases that a is in the extension, a is attacked by the extension, a is not in the extension nor attacked by it, respectively. Then define

$$\Phi^{\mathsf{CO}}_a = \left(\mathtt{out}_a \Leftrightarrow \bigvee_{b \in a^-} \mathtt{in}_b\right) \wedge \left(\mathtt{in}_a \Leftrightarrow \bigwedge_{b \in a^-} \mathtt{out}_b\right) \wedge (\mathtt{in}_a \vee \mathtt{out}_a \vee \mathtt{undec}_a)$$
$$\wedge (\neg\mathtt{in}_a \vee \neg\mathtt{out}_a) \wedge (\neg\mathtt{in}_a \vee \neg\mathtt{undec}_a) \wedge (\neg\mathtt{out}_a \vee \neg\mathtt{undec}_a)$$

and

$$\Psi^{\mathsf{CO}}_{\mathsf{AF}} = \bigwedge_{a \in \mathsf{A}} \Phi^{\mathsf{CO}}_a.$$

For any propositional formula Φ, let $\mathsf{Mod}(\Phi)$ denote its set of models. For any model ω let $E(\omega) = \{a \mid \omega(\mathtt{in}_a) = \text{TRUE}\}$. Variants of the following observations have been proven in e. g. [10].

Proposition 5. *Let* $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ *be an abstract argumentation framework.*

1. *If* $\omega \in \mathit{Mod}(\Psi^{\mathsf{CO}}_{\mathsf{AF}})$ *then* $E(\omega)$ *is a complete extension of* AF.

2. *If E is a complete extension of* AF *then there is* $\omega \in \mathit{Mod}(\Psi_{\mathsf{AF}}^{\mathit{CO}})$ *with* $E(\omega) = E$.
3. $a \in \mathsf{Acc}_{\mathsf{CO}}^{c}(\mathsf{AF})$ *if and only if* $\Psi_{\mathsf{AF}}^{\mathit{CO}} \wedge \mathtt{in}_a$ *is satisfiable.*

Due to $\mathsf{Acc}_{\mathsf{CO}}^{c}(\mathsf{AF}) = \mathsf{Acc}_{\mathsf{PR}}^{c}(\mathsf{AF})$ we set $\Psi_{\mathsf{AF}}^{\mathsf{PR}} = \Psi_{\mathsf{AF}}^{\mathsf{CO}}$ and use the encoding $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$ for credulous reasoning with preferred semantics as well.

For stable semantics, we can define a slightly simpler encoding as we do not need to encode the case in which an argument is neither in the extension nor attacked by the extension. So, for each argument $a \in \mathsf{A}$, we introduce one propositional variable $\mathtt{in}_a$, which models the case that a is in the extension. Then define

$$\Phi_a^{\mathsf{ST}} = \left(\neg \mathtt{in}_a \Leftrightarrow \bigvee_{b \in a^-} \mathtt{in}_b \right) \wedge \left(\mathtt{in}_a \Leftrightarrow \bigwedge_{b \in a^-} \neg \mathtt{in}_b \right)$$

and

$$\Psi_{\mathsf{AF}}^{\mathsf{ST}} = \bigwedge_{a \in \mathsf{A}} \Phi_a^{\mathsf{ST}}.$$

The observations from Proposition 5 apply similarly for stable semantics as well.

Proposition 6. *Let* $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ *be an abstract argumentation framework.*

1. *If* $\omega \in \mathit{Mod}(\Psi_{\mathsf{AF}}^{\mathit{ST}})$ *then* $E(\omega)$ *is a stable extension of* AF.
2. *If E is a stable extension of* AF *then there is* $\omega \in \mathit{Mod}(\Psi_{\mathsf{AF}}^{\mathit{ST}})$ *with* $E(\omega) = E$.
3. $a \in \mathsf{Acc}_{\mathsf{ST}}^{c}(\mathsf{AF})$ *if and only if* $\Psi_{\mathsf{AF}}^{\mathit{ST}} \wedge \mathtt{in}_a$ *is satisfiable.*

The above observations enable us to use SAT solving technology by encoding abstract argumentation problems into one or a series of SAT problems.[3]

Example 2. Given the abstract argumentation framework AF_1 from Example 1, $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$ is

$$(\mathtt{in}_{a_1} \vee \mathtt{out}_{a_1} \vee \mathtt{undec}_{a_1}) \wedge$$
$$(\mathtt{out}_{a_2} \Leftrightarrow \mathtt{in}_{a_1}) \wedge (\mathtt{in}_{a_2} \Leftrightarrow \mathtt{out}_{a_1}) \wedge (\mathtt{in}_{a_2} \vee \mathtt{out}_{a_2} \vee \mathtt{undec}_{a_2}) \wedge$$
$$(\mathtt{out}_{a_3} \Leftrightarrow \mathtt{in}_{a_2} \vee \mathtt{in}_{a_4}) \wedge (\mathtt{in}_{a_3} \Leftrightarrow \mathtt{out}_{a_2} \wedge \mathtt{out}_{a_4}) \wedge (\mathtt{in}_{a_3} \vee \mathtt{out}_{a_3} \vee \mathtt{undec}_{a_3}) \wedge$$
$$(\mathtt{out}_{a_4} \Leftrightarrow \mathtt{in}_{a_3}) \wedge (\mathtt{in}_{a_4} \Leftrightarrow \mathtt{out}_{a_3}) \wedge (\mathtt{in}_{a_4} \vee \mathtt{out}_{a_4} \vee \mathtt{undec}_{a_4})$$

Observe that $\mathsf{Acc}_{\mathsf{CO}}^{c}(\mathsf{AF}_1) = \{a_1, a_3, a_4\}$.

[3] Note that formulas such as $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$ can be easily turned in conjunctive normal form, the standard input format for SAT solvers, with only polynomial overhead, so we do not explicitly discuss matters related to this aspect in the following.

Algorithm 1 Algorithm IAQ^c

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{CO}, \mathsf{ST}, \mathsf{PR}\}$
Output: $\mathsf{Acc}^c_\sigma(\mathsf{AF})$
1: $S \leftarrow \emptyset$
2: **for** $a \in \mathsf{A}$ **do**
3: **if** $\textsc{Sat}(\Psi^\sigma_{\mathsf{AF}} \wedge \mathtt{in}_a)$ **then**
4: $S \leftarrow S \cup \{a\}$
5: **return** S

In the remainder of this section, we will use AF_1 as a running example to explain the behaviour of the introduced algorithms.

4.1. Iterative Acceptability Queries

A straightforward algorithm for determining $\mathsf{Acc}^c_\sigma(\mathsf{AF})$ is to exploit observations 3 of Propositions 5 and 6, respectively, and check for each $a \in \mathsf{A}$ whether $\Psi^\sigma_{\mathsf{AF}} \wedge \mathtt{in}_a$ is satisfiable using some SAT solver. We denote this algorithm IAQ^c (for *iterative acceptability queries* wrt. credulous reasoning), it is depicted as Algorithm 1. We write $\textsc{Sat}(\phi)$ for a call to an external SAT solver that evaluates to TRUE if ϕ is satisfiable.

Example 3. Assuming $\sigma = \mathsf{CO}$, the IAQ^c algorithm makes exactly one call per argument to the SAT solver, and updates S accordingly. Considering AF_1, after its initialisation $S \leftarrow \emptyset$, S is updated as follows:

1. loop on a_1: $S = \{a_1\}$
2. loop on a_2: $S = \{a_1\}$
3. loop on a_3: $S = \{a_1, a_3\}$
4. loop on a_4: $S = \{a_1, a_3, a_4\}$

The following observation regarding the correctness of the algorithm should be obvious.

Proposition 7. *Algorithm* IAQ^c *is sound and complete.*

4.2. Exhaustive extension enumeration

Another straightforward approach is to leverage the fact that SAT solvers usually do not only report on the satisfiability of a given formula but also provide a

Algorithm 2 Algorithm EEE^c

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{CO}, \mathsf{ST}, \mathsf{PR}\}$
Output: $\mathsf{Acc}^c_\sigma(\mathsf{AF})$
1: $S \leftarrow \emptyset$
2: $\Psi \leftarrow \Psi^\sigma_{\mathsf{AF}}$
3: **while** FALSE $\neq \omega = \textsc{Witness}(\Psi)$ **do**
4: $\quad S \leftarrow S \cup E(\omega)$
5: $\quad \Psi \leftarrow \Psi \wedge C(\omega)$
6: **return** S

model as a witness. For a model ω let

$$C(\omega) = \bigvee_{\omega(\alpha)=\text{TRUE}} \neg\alpha \vee \bigvee_{\omega(\alpha)=\text{FALSE}} \alpha.$$

One can then enumerate all models of formula ϕ by first retrieving any one model ω, then retrieving a model ω' of $\phi \wedge C(\omega)$, then a model ω'' if $\phi \wedge C(\omega) \wedge C(\omega')$ and so on. It is clear that all models retrieved this way are models of ϕ and that by adding $C(\omega)$, we avoid retrieving the same model on future calls again. Eventually, the formula becomes unsatisfiable, so we retrieved all the models. We can use this strategy to enumerate all complete/stable extensions of an input abstract argumentation framework (using observations 2 and 3 of Propositions 5 and 6, respectively). The union of these is then the set $\mathsf{Acc}^c_\sigma(\mathsf{AF})$. We denote this algorithm EEE^c and it is depicted as Algorithm 2. We write $\textsc{Witness}(\phi)$ for a call to an external SAT solver that evaluates to a model ω of ϕ if ϕ is satisfiable, or FALSE otherwise.

Example 4. Assuming $\sigma = \mathsf{CO}$, let us consider one by one the iterations that the EEE^c algorithm performs on AF_1.

1. $E = \{a_1, a_3\}$ is found, so

$$E(\omega) = \{a_1, a_3\} \quad S = \{a_1, a_3\} \quad C(\omega) = \{\neg a_1 \vee a_2 \vee \neg a_3 \vee a_4\}$$

2. $E = \{a_1\}$ is found, so

$$E(\omega) = \{a_1\} \quad S = \{a_1, a_3\} \quad C(\omega) = \{\neg a_1 \vee a_2 \vee a_3 \vee a_4\}$$

3. $E = \{a_1, a_4\}$ is found, so

$$E(\omega) = \{a_1, a_4\} \quad S = \{a_1, a_3, a_4\} \quad C(\omega) = \{\neg a_1 \vee a_2 \vee a_3 \vee \neg a_4\}$$

Algorithm 3 Algorithm SEE^c

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{CO}, \mathsf{ST}, \mathsf{PR}\}$
Output: $\mathsf{Acc}^c_\sigma(\mathsf{AF})$
1: $S \leftarrow \emptyset$
2: $D \leftarrow \mathsf{A}$
3: **while** FALSE $\neq \omega = \textsc{Witness}(\Psi^\sigma_{\mathsf{AF}} \wedge \bigvee_{a \in D} \mathtt{in}_a)$ **do**
4: $\quad S \leftarrow S \cup E(\omega)$
5: $\quad D \leftarrow D \setminus E(\omega)$
6: **return** S

A final iteration is then performed, but Ψ is now unsatisfiable, as no more extensions exist, and therefore $\textsc{Witness}(\Psi) = \mathit{FALSE}$, and the algorithm terminates.

The following observation regarding the correctness of the algorithm should be obvious.

Proposition 8. *Algorithm* EEE^c *is sound and complete.*

4.3. Selective extension enumeration

We now turn to our proposal of a non-trivial algorithm for computing $\mathsf{Acc}^c_\sigma(\mathsf{AF})$. A major drawback of the algorithm EEE^c is that an abstract argumentation framework may feature an exponential number of complete/stable extensions and many may overlap to a large degree. It may therefore be the case that in many iterations of the main loop in line 3 of Algorithm 2 no new arguments are added to S. To address this issue, we propose a more selective extension enumeration SEE, implemented in Algorithm 3.

Differently from Algorithm 2, the algorithm SEE^c constrains the search for further models (line 3) by requiring that at least one argument that has not already been classified as accepted, needs to be included. Indeed, at the first iteration (line 3) the SAT solver will identify a σ-extension with at least one `in` argument. The set of `in` arguments in the obtained extension will then be removed from the set D of *unvisited* arguments (line 5). From the second iteration, the SAT solver will then be forced to identify σ-extensions that intersect with the unvisited arguments. As we can see in the following example, SEE^c requires one less iteration to identify all the acceptable arguments, compared to EEE^c from the previous section.

Example 5. Assuming again $\sigma = \mathsf{CO}$, let us consider the iterations that the SEE^c algorithm performs on AF_1.

Algorithm 4 Algorithm SEEM^c

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{CO}, \mathsf{ST}, \mathsf{PR}\}$
Output: $\mathsf{Acc}^c_\sigma(\mathsf{AF})$
1: $S \leftarrow \emptyset$
2: $D \leftarrow \mathsf{A}$
3: **while** FALSE $\neq \omega = \textsc{MaxSat}(\{\mathtt{in}_a \mid a \in D\}, \Psi^\sigma_{\mathsf{AF}})$ **do**
4: $\quad S \leftarrow S \cup E(\omega)$
5: $\quad D \leftarrow D \setminus E(\omega)$
6: **return** S

1. $E = \{a_1, a_3\}$ is found, so

$$E(\omega) = \{a_1, a_3\} \qquad S = \{a_1, a_3\} \qquad D = \{a_2, a_4\}$$

2. $E = \{a_1, a_4\}$ is found, so

$$E(\omega) = \{a_1, a_4\} \qquad S = \{a_1, a_3, a_4\} \qquad D = \{a_2\}$$

Finally, since a_2 cannot be included in any complete extension, the overall formula is unsatisfiable and the execution ends. Observe, in particular, that after the algorithm found the extension $E = \{a_1, a_3\}$ in the first iteration, it will not consider the extension $E' = \{a_1\}$ in any of the following iterations as it introduces no new arguments.

Proposition 9. *Algorithm* SEE^c *is sound and complete.*

4.4. Selective extension enumeration via MaxSat

In (unweighted) MaxSat [12], formulas can be either *hard* or *soft*. The solutions of a MaxSat problem are determined among all assignments that satisfy all the hard formulas and are those that maximise the number of satisfied soft formulas. We write $\textsc{MaxSat}(S, H)$ (with a set of formulas S and a formula H) for a call to an external MaxSat solver that evaluates to a model ω that satisfies H and a maximal number of formulas in S. If H is not satisfiable, $\textsc{MaxSat}(S, H)$ evaluates to FALSE. Algorithm 4 shows our final algorithm SEEM^c for credulous reasoning.

The algorithm SEEM^c forces the MaxSat solver to maximise the set of *unvisited* arguments at each iteration. Given the simplicity and the size of the abstract argumentation framework AF_1, in this specific case, the SEEM^c algorithm would perform as SEE^c, so we do not give an additional example here.

Proposition 10. *Algorithm* SEEM^c *is sound and complete.*

Despite the fact that SEEM^c seems to be the most sophisticated algorithm so far, we will see later (in Section 6) that it is indeed outperformed in many cases by the simpler SEE^c version.

5. Algorithms for Skeptical Reasoning

We will now investigate some algorithms that compute the set $\mathsf{Acc}^s_\sigma(\mathsf{AF})$ for $\sigma \in \{\mathsf{ST}, \mathsf{PR}\}$. We do not consider grounded and complete semantics here as $\mathsf{Acc}^s_{\mathsf{GR}}(\mathsf{AF}) = \mathsf{Acc}^s_{\mathsf{CO}}(\mathsf{AF})$ can be computed in polynomial time anyway, cf. see Proposition 1.

For stable semantics, we will develop SAT-based algorithms and use the encoding $\Psi^{\mathsf{ST}}_{\mathsf{AF}}$ from the previous section. For that, we will take the following well-known fact about skeptical reasoning wrt. stable semantics and $\Psi^{\mathsf{ST}}_{\mathsf{AF}}$.

Proposition 11. *Let* $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ *be an abstract argumentation framework. Then* $a \in \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$ *if and only if* $\Psi^{ST}_{\mathsf{AF}} \wedge \neg \mathtt{in}_a$ *is unsatisfiable.*

For preferred semantics, note that the problem of deciding whether an argument a is skeptically accepted is Π^P_2-complete [5]. Thus, it cannot be solved by a single SAT-solver call (under standard complexity-theoretic assumptions). To develop similar baseline algorithms as before for skeptical reasoning under preferred semantics (in particular skeptical variants of the methods IAQ and EEE), we will make use of oracle calls of the form $\textsc{SkepPref}(\mathsf{AF}, a)$ and $\textsc{PrefExts}(\mathsf{AF})$. Here $\textsc{SkepPref}(\mathsf{AF}, a)$ returns TRUE if and only if a is skeptically accepted wrt. preferred semantics in AF (and FALSE otherwise) while $\textsc{PrefExts}(\mathsf{AF})$ returns the set of all preferred extensions of AF. These calls can be implemented by solvers capable of solving these problems, such as e. g., μ-toksia [6] or fudge [13] (which themselves use iterative calls to a SAT-solver for producing the answers).

5.1. Iterative Acceptability Queries

As in Section 4.1 for the case of credulous reasoning, a straightforward algorithm for determining $\mathsf{Acc}^s_\sigma(\mathsf{AF})$ is to check skeptical acceptance for each $a \in \mathsf{A}$ individually. We denote this algorithm IAQ^s, which is depicted as Algorithm 5. As before, we write $\textsc{Sat}(\phi)$ for a call to an external SAT solver that evaluates to TRUE if ϕ is satisfiable (and $\textsc{SkepPref}(\mathsf{AF}, a)$ to decide skeptical acceptance wrt. preferred semantics). The following observation regarding the correctness of the algorithm should be obvious.

Proposition 12. *Algorithm* IAQ^s *is sound and complete.*

Algorithm 5 Algorithm IAQ^s

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{ST}, \mathsf{PR}\}$
Output: $\mathsf{Acc}^s_\sigma(\mathsf{AF})$

$S \leftarrow \emptyset$
for $a \in \mathsf{A}$ **do**
  if $\sigma = \mathsf{ST}$ **then**
    if $\neg\textsc{Sat}(\Psi^{\mathsf{ST}}_{\mathsf{AF}} \wedge \neg \mathtt{in}_a)$ **then**
      $S \leftarrow S \cup \{a\}$
  if $\sigma = \mathsf{PR}$ **then**
    if $\textsc{SkepPref}(\mathsf{AF}, a)$ **then**
      $S \leftarrow S \cup \{a\}$
return S

5.2. Exhaustive extension enumeration

As in Section 4.2 for the case of credulous reasoning, another straightforward approach for skeptical reasoning is to exhaustively enumerate all extensions and take their intersection. For that, recall

$$C(\omega) = \bigvee_{\omega(\alpha)=\text{TRUE}} \neg\alpha \vee \bigvee_{\omega(\alpha)=\text{FALSE}} \alpha.$$

for any model ω. We denote this algorithm EEE^s and it is depicted as Algorithm 6. As before, We write $\textsc{Witness}(\phi)$ for a call to an external SAT solver that evaluates to a model ω of ϕ if ϕ is satisfiable, or FALSE otherwise (and we use $\textsc{PrefExts}(\mathsf{AF})$ for a call to a solver that enumerates all preferred extensions). The following observation regarding the correctness of the algorithm should be obvious.

Proposition 13. *Algorithm* EEE^s *is sound and complete.*

5.3. Selective extension enumeration via SAT *for skeptical reasoning wrt. stable semantics*

In this and the next section, we continue with algorithms that follow the scheme of the algorithms SEE^c and SEEM^c presented in Sections 4.3 and 4.4, respectively, but only for skeptical reasoning wrt. stable semantics. Due to the higher computational complexity of skeptical reasoning wrt. preferred semantics, those ideas cannot be applied in the same manner.

Algorithm 6 Algorithm EEE^s

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{ST}, \mathsf{PR}\}$
Output: $\mathsf{Acc}^s_\sigma(\mathsf{AF})$
1: $S \leftarrow \mathsf{A}$
2: **if** $\sigma = \mathsf{ST}$ **then**
3: $\quad \Psi \leftarrow \Psi^{\mathsf{ST}}_{\mathsf{AF}}$
4: $\quad$ **while** FALSE $\neq \omega =$ WITNESS(Ψ) **do**
5: $\quad\quad S \leftarrow S \cap E(\omega)$
6: $\quad\quad \Psi \leftarrow \Psi \wedge C(\omega)$
7: **if** $\sigma = \mathsf{PR}$ **then**
8: $\quad$ **for** $E \in$ PREFEXTS(AF) **do**
9: $\quad\quad S \leftarrow S \cap E$
10: **return** S

Algorithm 7 Algorithm SEE^s

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$
1: $S \leftarrow \mathsf{A}$
2: **while** FALSE $\neq \omega =$ WITNESS$(\Psi^{\mathsf{ST}}_{\mathsf{AF}} \wedge \bigvee_{a \in S} \neg \mathtt{in}_a)$ **do**
3: $\quad S \leftarrow S \cap E(\omega)$
4: **return** S

Again, a major drawback of the algorithm EEE^s (for stable semantics) is that an abstract argumentation framework may feature an exponential number of stable extensions and many may overlap to a large degree. So it may be the case that in many iterations of the main loop in line 4 of Algorithm 6 no new arguments are added to S. To address this, we present in Algorithm 7 the skeptical variant of selective extension enumeration SEE^s for stable semantics.

As one can see, the skeptical variant SEE^s is even a bit simpler than the credulous variant SEE^c (see Algorithm 3). We only maintain a set S of arguments that will hold the acceptable arguments once the algorithm terminates. This set is initialised with all arguments. In each iteration of the algorithm, we constrain the search for further models (line 2) by requiring that at least one argument currently assumed to be skeptically accepted must be $\neg\mathtt{in}$. Then, we take the intersection of the obtained stable extension with the set S and continue. Once no further model can be found, it is clear that all arguments in S must be contained in every stable

Algorithm 8 Algorithm SEEM^s

Input: $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$
Output: $\mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$
1: $S \leftarrow \mathsf{A}$
2: **while** FALSE $\neq \omega = \textsc{MaxSat}(\{\neg \mathtt{in}_a \mid a \in S\}, \Psi^{\mathsf{ST}}_{\mathsf{AF}})$ **do**
3: $\quad S \leftarrow S \cap E(\omega)$
4: **return** S

extension.

Proposition 14. *Algorithm* SEE^s *is sound and complete.*

5.4. Selective extension enumeration via MaxSat *for skeptical reasoning wrt. stable semantics*

We can apply the same idea when going from SEE^c to SEEM^c for skeptical reasoning wrt. stable semantics as well. Line 2 of Algorithm 7 only requires that at least one of the variables $\mathtt{in}_a$ for $a \in S$ be FALSE. Using a MaxSat solver allows us to maximise the number of arguments that can be dismissed in each iteration. Recall that $\textsc{MaxSat}(S, H)$ (with a set of formulas S and a formula H) denotes a call to an external MaxSat solver that evaluates to a model ω that satisfies H and a maximal number of formulas in S. If H is not satisfiable, $\textsc{MaxSat}(S, H)$ evaluates to FALSE. Algorithm 8 shows our final algorithm SEEM^s for skeptical reasoning.

Proposition 15. *Algorithm* SEEM^s *is sound and complete.*

As for the variant SEEM^c for credulous reasoning and despite the fact that SEEM^s seems to be the most sophisticated algorithm so far (for skeptical reasoning), we will see later (in Section 6) that it is indeed outperformed in many cases by the simpler SEE^s version.

6. Experimental Evaluation

We performed an extensive experimental evaluation to compare the runtime performance of our new algorithms. We describe the setup of this evaluation in Section 6.1 and present the results in Section 6.2. In Section 6.3 we conduct a small ablation study to show that the concrete Sat-solver has no influence on the general behaviour of our algorithms.

6.1. Experimental setup

For the experimental evaluation, we considered the following problems

EC-PR Enumerate the acceptable arguments wrt. credulous reasoning with preferred semantics

EC-ST Enumerate the acceptable arguments wrt. credulous reasoning with stable semantics

ES-PR Enumerate the acceptable arguments wrt. skeptical reasoning with preferred semantics

ES-ST Enumerate the acceptable arguments wrt. skeptical reasoning with stable semantics

Again, we do not consider complete semantics since credulous reasoning with complete semantics is equivalent to credulous reasoning with preferred semantics, and skeptical reasoning with complete semantics is equivalent to credulous/skeptical reasoning with grounded semantics, which can be solved in polynomial time and is also not considered.

For the above problems, we consider the algorithms depicted in Algorithms 1–8. More precisely, the competitors for the individual problems are:

- EC-PR: IAQ^c, EEE^c, SEE^c, SEEM^c
- EC-ST: IAQ^c, EEE^c, SEE^c, SEEM^c
- ES-PR: IAQ^s, EEE^s
- ES-ST: IAQ^s, EEE^s, SEE^s, SEEM^s

Our algorithms were implemented in C++[4]. For all calls of the form $\textsc{Sat}(\cdot)$, $\textsc{Witness}(\cdot)$ and $\textsc{MaxSat}(\cdot,\cdot)$ we used the CGSS 2.2.5 MaxSAT-solver [14, 15] based on the CaDiCaL 1.9.5 SAT-solver [16, 17]. For the problem ES-PR, all algorithms do not require $\textsc{MaxSat}(\cdot,\cdot)$ calls and thus only use CaDiCaL 1.9.5. We implemented the function $\textsc{SkepPref}(\mathsf{AF}, a)$ utilised in Algorithm 5 via a CEGAR-style approach [18], similar to that of the μ-toksia solver [6]. For Algorithm 6, the function $\textsc{PrefExts}(\mathsf{AF})$ for enumerating the preferred extensions

[4] https://github.com/aig-hagen/algorithms_for_acceptable_arguments

has been implemented via the SAT-encoding $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$ of the complete semantics and an iterative maximisation of all found models [19]. We ran the experiments on a machine running Ubuntu 20.04 with an Intel Xeon E5 3.4 GHz CPU and 192 GB of RAM. The evaluation has been conducted via the Probo2 benchmark suite [20]. We considered the benchmark datasets from the ICCMA'15 to ICCMA'23 competitions [21, 22, 23, 24]. Table 2 gives an overview on features of these datasets. There, #AFs is the number of instances in the dataset[5]; $|A|$ is the number of arguments of an instance, for which Table 2 shows the average, median and standard deviation; Avg. $|R|$ is the average number of attacks in the instances of the dataset; Avg. D is the average node degree over the whole dataset.

Dataset	#AFs	Avg. $\|A\|$	Med. $\|A\|$	Std. $\|A\|$	Avg. $\|R\|$	Avg. D
ICCMA'15	192	1980	675	2424	105396	68
ICCMA'17	1050	16638	500	151641	301409	169
ICCMA'19	326	826	196	1784	97639	239
ICCMA'21	480	87331	48200	92881	7239611	161
ICCMA'23	329	29791	796	203719	1002470	299

Table 2: Statistics for all considered benchmark datasets.

Each algorithm was given 20 minutes to compute the solution for every instance (=argumentation framework) and problem. For every algorithm and problem, we consider the number of unsolved instances and the runtime of solved instances. We also considered the PAR10 (Penalised Average Runtime) score to compare the performance of the algorithms. This score combines runtime and ability to solve, as it is calculated by considering runs that did not solve the problem as ten times the cutoff time.

6.2. Results

Tables 3–6 show the performance of the considered algorithms on all benchmarks for the respective problems EC-PR, EC-ST, ES-PR, and ES-ST. For each benchmark and problem we also include the *virtual best solver* (VBS), i. e., the solver that uses per instance the best other solver. In each table, N is the total number of instances of the benchmark set; #TO gives the number of time-outs/errors of each solver on this benchmark set; RT gives the runtime in seconds on all

[5]Note that the ICCMA'17 purposefully contains duplicate AFs. In total there are 874 unique AFs in the ICCMA'17 dataset.

ICCMA'15						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	192	0	2474.12	12.89	-
2	SEE^c	192	0	2493.56	12.99	106
3	EEE^c	192	0	2571.85	13.40	43
4	IAQ^c	192	1	2942.53	77.83	10
5	SEEM^c	192	19	11491.34	1247.35	33
ICCMA'17						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	1050	223	28450.93	2575.67	-
2	SEE^c	1050	232	29529.34	2679.55	239
3	IAQ^c	1050	240	36385.91	2777.51	133
4	SEEM^c	1050	308	31471.52	3549.97	194
5	EEE^c	1050	541	32191.84	6213.52	118
ICCMA'19						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	326	0	664.77	2.04	-
2	SEE^c	326	0	774.88	2.38	134
3	EEE^c	326	0	946.80	2.90	85
4	IAQ^c	326	0	1004.89	3.08	59
5	SEEM^c	326	4	5437.36	163.92	48
ICCMA'21						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	480	347	62511.65	8805.23	-
2	IAQ^c	480	349	68501.58	8867.71	106
3	SEE^c	480	390	40434.98	9834.24	27
4	EEE^c	480	480	0.00	12000.00	0
5	SEEM^c	480	480	0.00	12000.00	0
ICCMA'23						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	329	59	9806.91	2181.78	-
2	SEE^c	329	63	8755.90	2324.49	129
3	IAQ^c	329	68	11353.52	2514.75	53
4	SEEM^c	329	81	10092.91	2985.08	45
5	EEE^c	329	142	7603.76	5202.44	43

Table 3: Results for EC-PR on the ICCMA'15, ICCMA'17, ICCMA'19, ICCMA'21 and ICCMA'23 benchmark sets.

ICCMA'15						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	192	0	1442.36	7.51	-
2	EEE^c	192	0	1492.35	7.77	50
3	SEE^c	192	0	1520.09	7.92	95
4	IAQ^c	192	0	2336.67	12.17	25
5	SEEM^c	192	7	10497.84	492.18	22
ICCMA'17						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	1050	209	36085.22	2422.94	-
2	SEE^c	1050	219	33461.65	2534.73	260
3	IAQ^c	1050	232	35937.57	2685.65	125
4	SEEM^c	1050	279	22751.58	3210.24	133
5	EEE^c	1050	325	41943.91	3754.23	177
ICCMA'19						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	326	0	592.56	1.82	-
2	SEE^c	326	0	724.18	2.22	131
3	EEE^c	326	0	730.91	2.24	116
4	IAQ^c	326	0	790.67	2.43	43
5	SEEM^c	326	4	3008.31	156.47	36
ICCMA'21						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	480	159	99064.08	4181.38	-
2	IAQ^c	480	175	107816.26	4599.62	226
3	SEE^c	480	234	87482.91	6032.26	51
4	SEEM^c	480	368	49448.02	9303.02	27
5	EEE^c	480	401	16596.06	10059.58	17
ICCMA'23						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	329	38	14796.29	1430.99	-
2	SEE^c	329	39	16550.30	1472.80	135
3	IAQ^c	329	49	18408.16	1843.19	42
4	SEEM^c	329	57	13039.01	2118.66	37
5	EEE^c	329	81	13497.44	2995.43	77

Table 4: Results for EC-ST on the ICCMA'15, ICCMA'17, ICCMA'19, ICCMA'21 and ICCMA'23 benchmark sets.

ICCMA'15						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	192	0	934.68	4.87	-
2	EEE^s	192	0	934.68	4.87	192
3	IAQ^s	192	47	22608.66	3055.25	0
ICCMA'17						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	1050	299	42531.08	3457.65	-
2	EEE^s	1050	343	46995.16	3964.76	461
3	IAQ^s	1050	587	59977.92	6765.69	144
ICCMA'19						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	326	0	386.61	1.19	-
2	EEE^s	326	0	387.11	1.19	310
3	IAQ^s	326	9	27777.68	416.50	16
ICCMA'21						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	480	480	0	12000.00	-
2	EEE^s	480	480	0	12000.00	0
3	IAQ^s	480	480	0	12000.00	0
ICCMA'23						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	329	81	6544.26	2974.30	-
2	EEE^s	329	101	6207.97	3702.76	213
3	IAQ^s	329	136	19856.58	5020.84	35

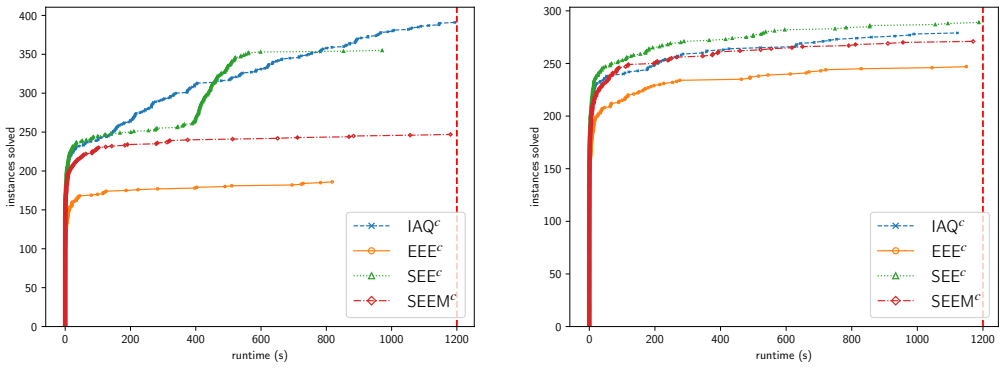
Table 5: Results for ES-PR on the ICCMA'15, ICCMA'17, ICCMA'19, ICCMA'21 and ICCMA'23 benchmark sets.

correctly solved benchmarks; PAR10 gives the average runtime where time-outs count ten times the cutoff-time, i. e., 12,000 seconds; #VBS gives the number of instances contributed to the VBS. Algorithms are ranked by the number of unsolved instances (in increasing order). In the case of ties, solvers are then ranked by runtime (in increasing order). Figures 2 and 3 visualise the performance of all approaches on the ICCMA'21 and ICCMA'23 benchmark sets (the other benchmark sets are not shown in order to keep the plots readable; their addition would not add interesting information).

The first observation when inspecting the results is that out of 15 experi-

ICCMA'15						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	192	0	1152.76	6.00	-
2	SEE^s	192	0	1203.54	6.27	30
3	EEE^s	192	0	1539.26	8.02	38
4	IAQ^s	192	0	1996.11	10.40	10
5	SEEM^s	192	2	4389.96	147.86	114
ICCMA'17						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	1050	214	34651.15	2478.72	-
2	SEE^s	1050	218	34555.19	2524.34	220
3	IAQ^s	1050	232	38677.18	2688.26	71
4	SEEM^s	1050	245	21924.75	2820.88	261
5	EEE^s	1050	318	40397.99	3672.76	139
ICCMA'19						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	326	0	461.20	1.41	-
2	SEE^s	326	0	489.73	1.50	83
3	IAQ^s	326	0	679.94	2.09	36
4	EEE^s	326	0	709.62	2.18	67
5	SEEM^s	326	1	2007.56	42.97	140
ICCMA'21						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	480	82	99532.77	2257.36	-
2	SEE^s	480	82	99801.65	2257.92	350
3	SEEM^s	480	146	110667.28	3880.56	8
4	IAQ^s	480	204	91177.72	5289.95	0
5	EEE^s	480	401	13788.57	10053.73	40
ICCMA'23						
No.	Algorithm	N	#TO	RT	PAR10	#VBS
1	VBS	329	34	15216.83	1286.37	-
2	SEE^s	329	35	15018.27	1322.24	120
3	IAQ^s	329	43	20851.61	1631.77	27
4	SEEM^s	329	67	13095.07	2483.57	100
5	EEE^s	329	79	14164.20	2924.51	48

Table 6: Results for ES-ST on the ICCMA'15, ICCMA'17, ICCMA'19, ICCMA'21 and ICCMA'23 benchmark sets.

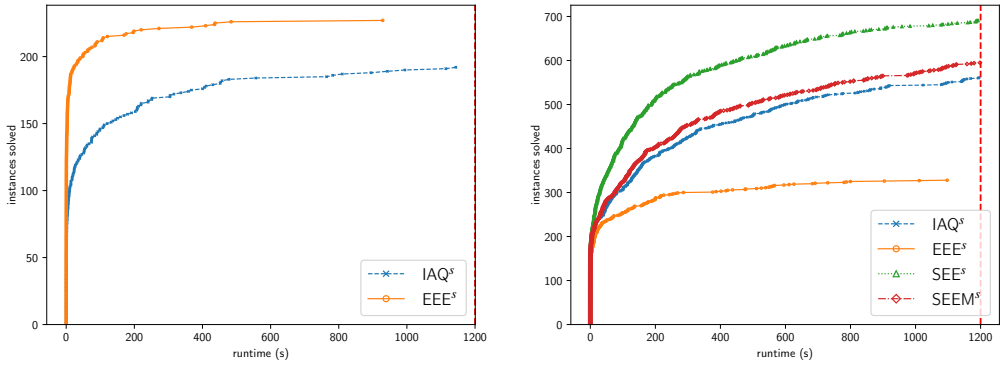


(a) EC-PR (b) EC-ST

Figure 2: Number of solved instances given the per-instance runtime by each algorithm for credulous reasoning on the ICCMA'21 and ICCMA'23 datasets.

ments, where the algorithm SEE^c/SEE^s participated (one for each pair of benchmark set and problem EC-PR, EC-ST, and ES-ST), it achieved first rank in 12 of them. In particular, SEE^c/SEE^s outperforms EEE^c/EEE^s in all but one case, namely on the ICCMA'15 data set for EC-ST. However, in that case, both algorithms have almost the same performance (no time outs and about 1500s total runtime), so there is indeed no case where EEE^c/EEE^s (significantly) outperforms SEE^c/SEE^s. This shows that exhaustive extension enumeration can generally be avoided when determining the set of acceptable arguments. SEE^c/SEE^s also outperforms IAQ^c/IAQ^s in all but two cases, namely EC-PR and EC-ST on the ICCMA'21 data set. In both cases, IAQ^c outperforms SEE^c quite significantly. The main difference between the ICCMA'21 data set and the others is that it features many large instances, which seems to benefit the IAQ^c approach. Finally, SEE^c/SEE^s also outperforms the algorithm SEEM^c/SEEM^s in all of them. The reason for this is likely that the total number of required SAT-solver calls for SEE^c/SEE^s is lower than the total number of required SAT-solver calls for SEEM^c/SEEM^s, since the latter requires a MAXSAT-solver call for finding each new extension, which results in multiple SAT-solver calls for each such extension. Although the number of found extensions by SEE^c/SEE^s is usually larger than for SEEM^c/SEEM^s, fewer SAT-solver calls are required for the former. Assuming that each SAT-solver call has (roughly) the same time consumption, SEE^c/SEE^s can outperform SEEM^c/SEEM^s.

An anomaly can be observed between 400 and 600 seconds runtime for SEE^c



(a) ES-PR (b) ES-ST

Figure 3: Number of solved instances given the per-instance runtime by each algorithm for skeptical reasoning on the ICCMA'21 and ICCMA'23 datasets.

in Figure 2a, where a couple of instances can be identified with similar runtime. Nearly all of these instances are from the ICCMA'21 dataset. In particular, these instances are those with the smallest number of arguments in the ICCMA'21 dataset, they have between $9,450$ and $12,600$ arguments and thus about 8 times less arguments than the dataset average. Comparatively, they also have about 4 fewer attacks than the dataset average. These instances are the only instances of the ICCMA'21 dataset that were solved at all by the algorithm SEE^c, all other instances resulted in a timeout. In the same instances, EEE^c and SEEM^c produce only timeouts and IAQ^c exhibits no abnormal behaviour. In general, this hints that IAQ^c scales better with an increasing number of arguments than the other algorithms.

As for the results for ES-PR (see Table 5), where only EEE^s and IAQ^s competed, it may come to a surprise that EEE^s consistently outperformed IAQ^s (with the exception of the ICCMA'21 data set where neither algorithm could solve any instance; which is again likely because the instances of ICCMA'21 are significantly larger than for the other data sets). However, one should recall that solving a single query on skeptical acceptance wrt. preferred semantics is a Π_2^P-complete problem [5] and itself involves multiple SAT-solver calls. In contrast, determining (some) preferred extension is a comparably easy task. In the best case, it can be solved by a single SAT-solver call (when the search heuristic of the solver favours labelling arguments as accepted). So even if the number of extensions is comparably large, solving a series of comparably easier tasks is beneficial to solving

fewer but harder tasks (at least in the case of skeptical acceptance wrt. preferred semantics).

Another interesting observation can be made on the results for ES-ST, see Table 6. Despite the fact that SEEM^s usually ranks at the lower end, it has a significantly large contribution to the virtual best solver. This is particularly apparent for ICCMA'15, ICCMA'17, and ICCMA'19, where SEEM^s has the majority share in the virtual best solver. The reason for this is that ICCMA'15 and ICCMA'19 (and also a large part of ICCMA'17) feature many easy instances that all solvers can solve, but where SEEM^s can solve them quite fast. Still, the overhead required for MAXSAT-solving in SEEM^s leads to more timeouts for the harder instances, which leads to the otherwise low ranking of SEEM^s.

6.3. Ablation study wrt. SAT*-solvers*

To evaluate the impact of the underlying SAT-solver that is used in each algorithm, we conducted a small ablation study, focusing only on the ICCMA'23 benchmark set and the problems EC-ST and ES-ST. In addition to the previously used SAT-solver CADICAL 1.9.5 [17], we also considered GLUCOSE 4.1 [25] and CRYPTOMINISAT 5.11.21 [26]. Tables 7 and 8 show the results for the algorithms IAQ, EEE and SEE executed with the three different SAT-solvers on the ICCMA'23 benchmark set for the problems EC-ST and ES-ST, respectively. As one can see, the choice of the concrete SAT-solver has no influence on the ranking of the three algorithmic approaches. However, one can observe that CADICAL consistently outperforms the other SAT-solvers in this domain.

7. Summary and Conclusion

In this paper, we considered the computational task of computing the set of acceptable arguments in abstract argumentation wrt. credulous and skeptical reasoning and grounded, complete, stable, and preferred semantics. Our study on computational complexity showed that the corresponding decision variants are complete for the DP family of complexity classes, mirroring results for classical problems. We presented different SAT-based algorithms for computing the set of accepted arguments wrt. the different semantics and reasoning modes, and our evaluation showed that the SEE approach turned out to be the most effective, also generally outperforming (quite surprisingly) the approaches based on maximum satisfiability solving.

For future work, both the theoretical as well as the experimental study can be extended to include further semantics such as semi-stable [27], stage [28], and

SEE^c					
No.	Algorithm	N	#TO	RT	PAR10
1	SEE^c (CaDiCaL)	329	28	10999.14	1054.71
2	SEE^c (Glucose)	329	40	15929.37	1507.38
3	SEE^c (CryptoMiniSat)	329	42	16254.35	1581.32
IAQ^c					
No.	Algorithm	N	#TO	RT	PAR10
1	IAQ^c (CaDiCaL)	329	34	11726.63	1275.76
2	IAQ^c (CryptoMiniSat)	329	49	18997.24	1844.98
3	IAQ^c (Glucose)	329	50	14301.12	1867.18
EEE^c					
No.	Algorithm	N	#TO	RT	PAR10
1	EEE^c (CaDiCaL)	329	68	10761.63	2512.95
2	EEE^c (Glucose)	329	79	15379.40	2928.20
3	EEE^c (CryptoMiniSat)	329	88	12872.30	3248.85

Table 7: Results for EC-ST on the ICCMA'23 benchmark set for the algorithms IAQ^s, EEE^s and SEE^s executed with three different SAT-solvers.

CF2 semantics [29]. Since the complexity of reasoning with CF2 semantics (NP-complete for credulous reasoning and coNP-complete reasoning for skeptical reasoning) is similar to reasoning with stable semantics, see, e. g., [5], we expect that this will also be similar to the corresponding problems analysed in this paper for stable semantics (so DP-completeness for all variants). Moreover, since skeptical reasoning with semi-stable and stage semantics is Π_2^{P}-complete, we expect that the result for $\mathsf{ACC}^s_{\mathsf{PR}}$ carries over as well (namely DP2-completeness). Since credulous reasoning with semi-stable/stage semantics is Σ_2^{P}-complete, a more challenging analysis and development of algorithmic approaches is expected for this case.

Another avenue for future work are improvements on the algorithms. In particular, algorithms IAQ, EEE, and SEE (for both credulous and skeptical reasoning) are based on iterative calls to a SAT solver. The order of these calls (in particular when using iterative SAT solving techniques) can influence overall runtime and approaches such as [30] could be used to decrease the number of required SAT solver calls. Also approaches to algorithm selection [31] or portfolio-based approaches [32] could be used to combine the advantages of the individual algorithms.

SEE^s					
No.	Algorithm	N	#TO	RT	PAR10
1	SEE^s (CaDiCaL)	329	27	11339.43	1019.27
2	SEE^s (Glucose)	329	35	11705.36	1312.17
3	SEE^s (CryptoMiniSat)	329	40	15033.60	1504.66
IAQ^s					
No.	Algorithm	N	#TO	RT	PAR10
1	IAQ^s (CaDiCaL)	329	34	11810.67	1276.02
2	IAQ^s (Glucose)	329	41	20346.03	1557.28
3	IAQ^s (CryptoMiniSat)	329	53	16891.78	1984.47
EEE^s					
No.	Algorithm	N	#TO	RT	PAR10
1	EEE^s (CaDiCaL)	329	68	11510.70	2515.23
2	EEE^s (Glucose)	329	77	13078.14	2848.26
3	EEE^s (CryptoMiniSat)	329	86	17660.49	3190.46

Table 8: Results for ES-ST on the ICCMA'23 benchmark set for the algorithms IAQ^s, EEE^s and SEE^s executed with three different SAT-solvers.

Acknowledgements The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grants 375588274, 550735820).

References

[1] P. M. Dung, On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games, Artificial Intelligence 77 (2) (1995) 321–358.

[2] F. Cerutti, S. A. Gaggl, M. Thimm, J. P. Wallner, Foundations of implementations for formal argumentation, in: Handbook of Formal Argumentation, 2018.

[3] A. Toniolo, T. J. Norman, A. Etuk, F. Cerutti, R. W. Ouyang, M. Srivastava, N. Oren, T. Dropps, J. A. Allen, P. Sullivan, Agent Support to Reasoning with Different Types of Evidence in Intelligence Analysis, in: Proceedings of AAMAS, 2015, pp. 781–789.

[4] M. Vallati, F. Cerutti, M. Giacomin, Predictive models and abstract argumentation: the case of high-complexity semantics, The Knowledge Engineering Review 34 (2019) e6.

[5] W. Dvořák, P. E. Dunne, Computational problems in formal argumentation and their complexity, in: Handbook of Formal Argumentation, 2018.

[6] A. Niskanen, M. Järvisalo, μ-toksia: An efficient abstract argumentation reasoner, in: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020), 2020.

[7] M. Thimm, F. Cerutti, M. Vallati, On computing the set of acceptable arguments in abstract argumentation, in: H. Prakken, S. Bistarelli, F. Santini, C. Taticchi (Eds.), Proceedings of the 8th International Conference on Computational Models of Argument (COMMA'20), Vol. 326 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2020, pp. 363–370.

[8] C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[9] G. Charwat, W. Dvořák, S. A. Gaggl, J. P. Wallner, S. Woltran, Methods for solving reasoning problems in abstract argumentation - A survey, Artificial Intelligence 220 (2015) 28–63.

[10] P. Besnard, S. Doutre, Checking the acceptability of a set of arguments, in: Proceedinfs of NMR, 2004.

[11] 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.

[12] C. M. Li, F. Manya, Maxsat, hard and soft constraints., in: Handbook of Satisfiability, 2009.

[13] M. Thimm, F. Cerutti, M. Vallati, Skeptical reasoning with preferred semantics in abstract argumentation without computing preferred extensions, in: Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21), 2021.

[14] H. E. Ihalainen, J. Berg, M. Järvisalo, Refined core relaxation for core-guided maxsat solving, in: 27th International Conference on Principles and Practice of Constraint Programming (CP 2021), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021, pp. 28–1.

[15] H. Ihalainen, Refined core relaxations for core-guided maximum satisfiability algorithms, Ph.D. thesis, MSc thesis, University of Helsinki, 2022, http://hdl.handle.net/10138/351207 (2022).

[16] A. Biere, K. Fazekas, M. Fleury, M. Heisinger, CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020, in: T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, M. Suda (Eds.), Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions, Vol. B-2020-1 of Department of Computer Science Report Series B, University of Helsinki, 2020, pp. 51–53.

[17] 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, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part I, Vol. 14681 of Lecture Notes in Computer Science, Springer, 2024, pp. 133–152. doi:10.1007/978-3-031-65627-9_7.

[18] W. Dvořák, M. Järvisalo, J. P. Wallner, S. Woltran, Complexity-sensitive decision procedures for abstract argumentation, Artificial Intelligence 206 (2014) 53–78.

[19] J. Klein, M. Thimm, Revisiting sat techniques for abstract argumentation, in: Computational Models of Argument, IOS Press, 2020, pp. 251–262.

[20] J. Klein, M. Thimm, probo2: A benchmark framework for argumentation solvers, in: Computational Models of Argument, IOS Press, 2022, pp. 363–364.

[21] M. Thimm, S. Villata, The first international competition on computational models of argumentation: Results and analysis, Artificial Intelligence 252 (2017) 267–294.

[22] S. A. Gaggl, T. Linsbichler, M. Maratea, S. Woltran, Design and results of the second international competition on computational models of argumentation, Artificial Intelligence 278 (2020).

[23] 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 (Preprint) (2024) 1–73.

[24] M. Järvisalo, T. Lehtonen, A. Niskanen, Solver and benchmark descriptions of iccma 2023: 5th international competition on computational models of argumentation (2023).

[25] G. Audemard, L. Simon, On the glucose SAT solver, Int. J. Artif. Intell. Tools 27 (1) (2018) 1840001:1–1840001:25. doi:10.1142/S0218213018400018.
URL https://doi.org/10.1142/S0218213018400018

[26] M. Soos, K. Nohl, C. Castelluccia, Extending SAT solvers to cryptographic problems, in: O. Kullmann (Ed.), Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings, Vol. 5584 of Lecture Notes in Computer Science, Springer, 2009, pp. 244–257. doi:10.1007/978-3-642-02777-2_24.
URL https://doi.org/10.1007/978-3-642-02777-2_24

[27] M. Caminada, Semi-stable semantics, in: The First Conference on Computational Models of Argument (COMMA'06), 2006, pp. 121–130.

[28] B. Verheij, Two approaches to dialectical argumentation: admissible sets and argumentation stages, in: Proceedings of NAIC, 1996, pp. 357–368.

[29] P. Baroni, M. Giacomin, G. Guida, SCC-recursiveness: a general schema for argumentation semantics, Artificial Intelligence 168 (1–2) (2005) 162–210.

[30] M. Janota, J. Marques-Silva, On the query complexity of selecting minimal sets for monotone predicates, Artif. Intell. 233 (2016) 73–83. doi:10.1016/J.ARTINT.2016.01.002.
URL https://doi.org/10.1016/j.artint.2016.01.002

[31] F. Cerutti, M. Giacomin, M. Vallati, Algorithm selection for preferred extensions enumeration, in: S. Parsons, N. Oren, C. Reed, F. Cerutti (Eds.), Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014, Vol. 266 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2014, pp. 221–232. doi:10.3233/978-1-61499-436-7-221.
URL https://doi.org/10.3233/978-1-61499-436-7-221

[32] M. Vallati, F. Cerutti, M. Giacomin, On the combination of argumentation solvers into parallel portfolios, in: Proceedings of the 30th Australasian Joint Conference on Artificial Intelligence (AG 2017), 2017.

[33] M. Thimm, J. P. Wallner, On the complexity of inconsistency measurement, Artificial Intelligence 275 (2019) 411–456.

Appendix A. Proofs of technical results

Proposition 1. $\mathsf{ACC}^s_{\mathsf{GR}}$, $\mathsf{ACC}^c_{\mathsf{GR}}$, *and* $\mathsf{ACC}^s_{\mathsf{CO}}$ *are in P.*

Proof. Observe first that the problems $\mathsf{ACC}^s_{\mathsf{GR}}$, $\mathsf{ACC}^c_{\mathsf{GR}}$, and $\mathsf{ACC}^s_{\mathsf{CO}}$ are actually identical as there is exactly one grounded extension and it is equal to the intersection of all complete extensions. Furthermore, as determining the grounded extension E_{gr} is in P [5], we can first compute it and then compare it to the input E in linear time. □

Proposition 2. $\mathsf{ACC}^c_{\mathsf{CO}}$, $\mathsf{ACC}^c_{\mathsf{PR}}$, *and* $\mathsf{ACC}^c_{\mathsf{ST}}$ *are DP-complete.*

Proof. Observe first that the problems $\mathsf{ACC}^c_{\mathsf{CO}}$ and $\mathsf{ACC}^c_{\mathsf{PR}}$ are identical. Recall also that verifying whether a given set E is a complete or stable extension can be done in polynomial time [5].

In order to show DP membership of ACC^c_σ (with σ being either complete or stable semantics), we first define the two languages L_1 and L_2 as follows

$$L_1 = \{(E_1, \mathsf{A} \setminus E_1) \in \mathsf{A} \times \mathsf{A} \mid E_1 \subseteq \mathsf{Acc}^x_\sigma(\mathsf{AF})\}$$
$$L_2 = \{(\mathsf{A} \setminus E_2, E_2) \in \mathsf{A} \times \mathsf{A} \mid E_2 \cap \mathsf{Acc}^x_\sigma(\mathsf{AF}) = \emptyset\}$$

Observe that $L_1 \in \mathsf{NP}$: on instance (E_1, E_2) we guess for each $a \in E_1$ a set F with $a \in F$ and verify in polynomial time whether it is indeed a complete/stable extension. Furthermore, $L_2 \in \mathsf{coNP}$: if an instance (E_1, E_2) is *not* in L_2, we can guess a set F for some $a \in E_2$ (with $a \in F$) and verify in polynomial time whether it is a complete/stable extension. Finally, $L_1 \cap L_2$ is equivalent to ACC^c_σ by projecting on the first component of the instances, showing its DP-membership.

For DP-hardness, we reduce the problem SAT-UNSAT to ACC^c_σ, which is known to be DP-complete [8]. An instance (ϕ, ψ), with two propositional formulas ϕ, ψ in conjunctive normal form with exactly three literals per clause (3-CNF), belongs to SAT-UNSAT iff ϕ is satisfiable and ψ is not satisfiable. We make use of the standard reduction of 3-CNF to abstract argumentation frameworks (see Reduction 3.6. in [5]). For a formula $\phi = c_1 \wedge \ldots \wedge c_n$ with $c_i = l_{1,i} \vee l_{2,i} \vee l_{3,i}$ over the alphabet $V_\phi = \{v_1, \ldots, v_m\}$,[6] we denote by $F_\phi = (\mathsf{A}_\phi, \mathsf{R}_\phi)$ the abstract

[6] The argument ϕ can be interpreted as "the formula ϕ is true," while the argument c_i can be read as "clause c_i is not satisfied" [5].

argumentation framework defined via

$$\begin{aligned}
\mathsf{A}_\phi &= \{a_\phi, a_{\phi,c_1}, \ldots, a_{\phi,c_n}, v_1, \ldots, v_m, \neg v_1, \ldots, \neg v_m\} \\
\mathsf{R}_\phi &= \{(a_{\phi,c_1}, a_\phi), \ldots, (a_{\phi,c_n}, a_\phi)\} \cup \\
&\qquad \{(v, a_{\phi,c_i}) \mid v \in c_i, i = 1, \ldots, n\} \cup \\
&\qquad \{(v_1, \neg v_1), (\neg v_1, v_1), \ldots, (v_m, \neg v_m), (\neg v_m, v_m)\}
\end{aligned}$$

For an SAT-UNSAT instance (ϕ, ψ) we require two additional assumptions: no clause c appearing in either ϕ or ψ is a tautology, i. e., contains both v and $\neg v$ for some atom v (the clause could be removed from the formula anyway), and ϕ and ψ have disjoint vocabularies (can be realised by renaming of atoms). Upon instance (ϕ, ψ) we construct the framework $\mathsf{AF} = (\mathsf{A}_\phi \cup \mathsf{A}_\psi, \mathsf{R}_\phi \cup \mathsf{R}_\psi)$ and ask whether

$$E = \mathsf{A}_\phi \cup \mathsf{A}_\psi \setminus \{a_\psi\}$$

is exactly the set of credulously accepted arguments wrt. complete/stable semantics. This is the case if and only if (ϕ, ψ) is a positive instance of SAT-UNSAT. To see this, assume that (ϕ, ψ) is a positive instance and observe first that stable extensions exist in AF for every instance (there is no odd loop in AF). Furthermore, every v and $\neg v$ is accepted as it defends itself against its only attacker, so there is always a complete/stable extension including it. As every clause c_i in either ϕ or ψ is not a tautology, the set $\{\overline{l}_{1,i}, \overline{l}_{2,i}, \overline{l}_{3,i}\}$ (with overlining indicating the complement literal) is conflict-free and defends c_i, therefore c_i can be credulously accepted. Furthermore, if ϕ is satisfiable there is a stable/complete extension containing a_ϕ [5]. Finally, if ψ is not satisfiable, a_ψ cannot be credulously accepted (as the only argument of AF). So if E is the set of credulously accepted arguments wrt. complete/stable semantics then (ϕ, ψ) is a positive instance of SAT-UNSAT. The reverse direction is analogous. □

Proposition 3. $\mathsf{ACC}^s_{\mathsf{ST}}$ *is DP-complete.*

Proof. The proof is similar to the proof of Proposition 2. For DP-membership we define two languages

$$\begin{aligned}
L_1 &= \{(E_1, \mathsf{A} \setminus E_1) \in \mathsf{A} \times \mathsf{A} \mid E_1 \subseteq \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})\} \\
L_2 &= \{(\mathsf{A} \setminus E_2, E_2) \in \mathsf{A} \times \mathsf{A} \mid E_2 \cap \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF}) = \emptyset\}
\end{aligned}$$

Observe that L_1 is in coNP: if an instance (E_1, E_2) is *not* in L_1 we guess a set E with $E_1 \setminus E \neq \emptyset$ and verify in polynomial time that E is stable (meaning that there

is at least one argument in E_1 that cannot be skeptically accepted). Furthermore, L_2 is in NP: for each argument $a \in E_2$ we can guess a set E with $a \notin E$ and verify in polynomial time that E is stable. Finally, $L_1 \cap L_2$ is equivalent to $\mathsf{ACC}^s_{\mathsf{ST}}$ by projecting on the first component of the instances, showing its DP-membership (the reader may also verify that the verification still works for the case of an abstract argumentation framework without stable extensions where $\mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF}) = \mathsf{A}$).

For DP-hardness, we use the same reduction as in the proof of Proposition 2, but include two new arguments x_ϕ, x_ψ, and attacks (a_ϕ, x_ϕ), (a_ψ, x_ψ). Then $E = \{x_\psi\}$ is exactly the set of skeptically accepted arguments wrt. stable semantics if and only if (ϕ, ψ) is a positive instance of SAT-UNSAT. Assume that (ϕ, ψ) is a positive instance of SAT-UNSAT, then no argument v or $\neg v$ is skeptically accepted as there is always a stable extension including its only attacker. Furthermore, no c_i is skeptically accepted as there is always a stable extension including one of its attackers (as every clause is satisfiable). Furthermore, as ϕ is satisfiable, x_ϕ is not included in the extension containing a_ϕ, which must exist [5]. As ϕ is also not tautological (this can only be the case if all its clauses are tautological), there must also be a stable extension not including a_ϕ. As ψ is not satisfiable x_ψ must be contained in every stable extension and a_ψ is in no stable extension. This shows that $E = \{x_\psi\}$ is exactly the set of skeptically accepted arguments wrt. stable semantics. The reverse direction is analogous. □

Proposition 4. $\mathsf{ACC}^s_{\mathsf{PR}}$ *is DP2-complete.*

Proof. In order to show DP2 membership of $\mathsf{ACC}^s_{\mathsf{PR}}$ we first define the two languages L_1 and L_2 as follows

$$L_1 = \{(E_1, \mathsf{A} \setminus E_1) \in \mathsf{A} \times \mathsf{A} \mid E_1 \subseteq \mathsf{Acc}^s_{\mathsf{PR}}(\mathsf{AF})\}$$
$$L_2 = \{(\mathsf{A} \setminus E_2, E_2) \in \mathsf{A} \times \mathsf{A} \mid E_2 \cap \mathsf{Acc}^s_{\mathsf{PR}}(\mathsf{AF}) = \emptyset\}$$

Observe that L_1 is in $\mathsf{coNP}^{\mathsf{NP}}$: if an instance (E_1, E_2) is *not* in L_1 we guess a set E with $E_1 \setminus E \neq \emptyset$ and verify that E is preferred (meaning that there is at least one argument in E_1 that cannot be skeptically accepted). The latter problem is in coNP [5] and an NP-oracle call is equivalent to an coNP-oracle call. Furthermore, L_2 is in $\mathsf{NP}^{\mathsf{coNP}}$: for each argument $a \in E_2$ we can guess a set E with $a \notin E$ and verify that E is preferred. Finally, $L_1 \cap L_2$ is equivalent to $\mathsf{ACC}^s_{\mathsf{PR}}$ by projecting on the first component of the instances, showing its DP2-membership.

For DP2-hardness, we reduce the DP2-complete problem $\forall\exists\mathsf{QBF}_2$ [33] to $\mathsf{ACC}^s_{\mathsf{PR}}$. Here, an instance is a pair (ϕ, ψ) of quantified Boolean formulæ of the

form

$$\phi = \forall Y \exists Z : \mu(Y,Z)$$
$$\psi = \forall Y' \exists Z' : \mu'(Y',Z')$$

where $\mu(Y,Z)$ and $\mu'(Y',Z')$ are propositional formulæ over the variables $Y \cup Z$, $Y' \cup Z'$, respectively. The pair (ϕ,ψ) is a "yes" instance of $\forall\exists\mathsf{QBF}_2$ if ϕ evaluates to TRUE and ψ evaluates to FALSE.

First, we define a generalisation of Reduction 3.7 of [5] to compile a QBF of the form

$$\phi = \forall y_1,\ldots,y_n \exists z_1,\ldots,z_m : \mu(y_1,\ldots,y_n,z_1,\ldots,z_m) \tag{A.1}$$

to abstract argumentation frameworks that works for arbitrary propositional formulæ $\mu(y_1,\ldots,y_n,\, z_1,\ldots,z_m)$ (not just CNF-formulæ). For that, we inductively define a transformation from a propositional formula μ (we now omit mentioning the variables explicitly) to an AF, i. e., $\mathsf{AF}_\mu = (\mathsf{A}_\mu, \mathsf{R}_\mu)$, via

1. If $\mu = v$ for some $v \in \{y_1,\ldots,y_n,z_1,\ldots,z_m\}$ define

$$\mathsf{A}_{y_i} = \{p_v, p_{\overline{v}}\}$$
$$\mathsf{R}_{y_i} = \{(p_v, p_{\overline{v}}), (p_{\overline{v}}, p_v)\}$$

2. If $\mu = \neg\mu'$ define

$$\mathsf{A}_{\neg\mu'} = \mathsf{A}_{\mu'} \cup \{p_{\neg\mu'}\}$$
$$\mathsf{R}_{\neg\mu'} = \mathsf{R}_{\mu'} \cup \{(p_{\mu'}, p_{\neg\mu'})\}$$

3. If $\mu = \mu' \wedge \mu''$ define

$$\mathsf{A}_{\mu'\wedge\mu''} = \mathsf{A}_{\mu'} \cup \mathsf{A}_{\mu''} \cup \{h^1_{\mu'\wedge\mu''}, h^2_{\mu'\wedge\mu''}, p_{\mu'\wedge\mu''}\}$$
$$\mathsf{R}_{\mu'\wedge\mu''} = \mathsf{R}_{\mu'} \cup \mathsf{R}_{\mu'} \cup \{(p_{\mu'}, h^1_{\mu'\wedge\mu''}), (p_{\mu''}, h^2_{\mu'\wedge\mu''}), (h^1_{\mu'\wedge\mu''}, p_{\mu'\wedge\mu''}),$$
$$(h^2_{\mu'\wedge\mu''}, p_{\mu'\wedge\mu''})\}$$

4. If $\mu = \mu' \vee \mu''$ define

$$\mathsf{A}_{\mu'\vee\mu''} = \mathsf{A}_{\mu'} \cup \mathsf{A}_{\mu''} \cup \{h_{\mu'\vee\mu''}, p_{\mu'\vee\mu''}\}$$
$$\mathsf{R}_{\mu'\vee\mu''} = \mathsf{R}_{\mu'} \cup \mathsf{R}_{\mu'} \cup \{(p_{\mu'}, h_{\mu'\vee\mu''}), (p_{\mu''}, h_{\mu'\vee\mu''}), (h_{\mu'\vee\mu''}, p_{\mu'\vee\mu''})\}$$

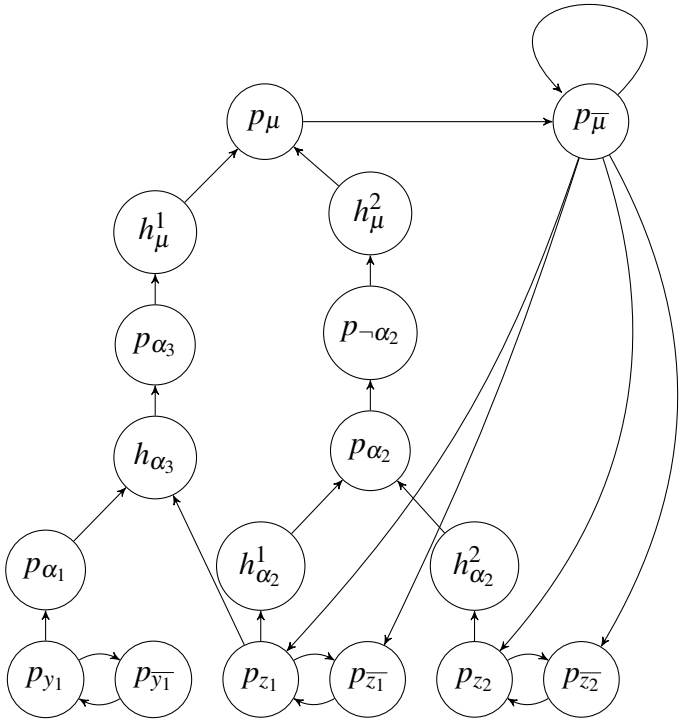


Figure A.4: Abstract argumentation framework AF_ϕ for the QBF $\phi = \forall y_1 : \exists z_1, z_2 : \mu$ with $\mu = (\neg y_1 \vee z_1) \wedge \neg(z_1 \wedge z_2)$. We abbreviate $\alpha_1 = \neg y_1$, $\alpha_2 = z_1 \wedge z_2$, and $\alpha_3 = \neg y_1 \vee z_1$

To complete the reduction, similarly to [5], we define for a QBF of the form (A.1) the AF $\mathsf{AF}_\phi = (\mathsf{A}_\phi, \mathsf{R}_\phi)$ with

$$\mathsf{A}_\phi = \mathsf{A}_\mu \cup \{p_{\overline{\mu}}\}$$
$$\mathsf{R}_\phi = \mathsf{R}_\mu \cup \{(p_\mu, p_{\overline{\mu}}), (p_{\overline{\mu}}, p_{\overline{\mu}})\} \cup \{(p_{\overline{\mu}}, z_1), \ldots, (p_{\overline{\mu}}, z_m)\}$$

Figure A.4 shows an example of the reduction. Observe that the QBF ϕ evaluates to TRUE iff p_μ is skeptically accepted in AF_ϕ wrt. preferred semantics [5]. However, note that p_μ may not be the *only* argument that is skeptically accepted (for example, in Figure A.4, p_{α_3} is skeptically accepted as well). In order for our aimed reduction to $\mathsf{ACC}^s_{\mathsf{PR}}$ to work, we need to have a clearly defined status for each argument. We address this by a process we call *cloning*. Each argument $a \in \mathsf{A}_\phi \setminus \{p_\mu, p_{\overline{\mu}}\}$ is cloned yielding an additional argument $\hat{a}$. For each attack $(a,b) \in \mathsf{R}_\phi \setminus \{(a',b') \mid a', b' \notin \{p_\mu, p_{\overline{\mu}}\}\}$ we add attacks $(a,\hat{b}), (\hat{a},b), (\hat{a},\hat{b})$.

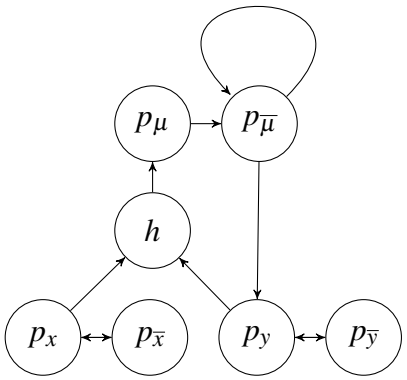


Figure A.5: Abstract argumentation framework AF_ϕ for the QBF $\phi = \forall x : \exists y : x \vee y$.

Furthermore, for each attack (a, p_μ) we add the attack $(\hat{a}, p_\mu)$ and for each attack $(p_{\overline{\mu}}, a)$ we add $(p_{\overline{\mu}}, \hat{a})$. We abbreviate the new argumentation framework by $\hat{\mathsf{AF}}_\phi = (\hat{\mathsf{A}}_\phi, \hat{\mathsf{R}}_\phi)$. Figures A.5 and A.6 show an example of the cloning process. Observe that every preferred extension E of AF_ϕ is also a preferred extension of $\hat{\mathsf{AF}}_\phi$. Furthermore, if one removes any number of arguments (except p_μ and $p_{\overline{\mu}}$) in a preferred extension of AF_ϕ and replaces them with their clones, one again obtains a preferred extension of $\hat{\mathsf{AF}}_\phi$. Finally, observe that every preferred extension of $\hat{\mathsf{AF}}_\phi$ is of that form (i. e., it cannot be the case that both an argument and its clone are not in a preferred extension, even if all their respective attackers are not in the extension; as a preferred extension is a maximal admissible set, one of them has to be included in that case). It follows, that p_μ is the *only* skeptically accepted argument in $\hat{\mathsf{AF}}_\phi$.

Now back to the reduction from $\forall\exists\mathsf{QBF}_2$ to $\mathsf{ACC}^s_{\mathsf{PR}}$. For an instance (ϕ, ψ)—we assume that ϕ and ψ are defined on disjoint vocabularies—we construct the abstract argumentation framework $\mathsf{AF}_{(\phi,\psi)} = (\mathsf{A}_{(\phi,\psi)}, \mathsf{R}_{(\phi,\psi)})$ defined via

$$\mathsf{A}_{(\phi,\psi)} = \hat{\mathsf{A}}_\phi \cup \hat{\mathsf{A}}_\psi$$
$$\mathsf{R}_{(\phi,\psi)} = \hat{\mathsf{R}}_\phi \cup \hat{\mathsf{R}}_\psi$$

Then (ϕ, ψ) is a positive instance of $\forall\exists\mathsf{QBF}_2$ if and only if $\mathsf{Acc}^s_{\mathsf{PR}}(\mathsf{AF}_{(\phi,\psi)}) = \{p_\phi\}$ by construction. □

Corollary 1. *Let* AF *be an abstract argumentation framework.*

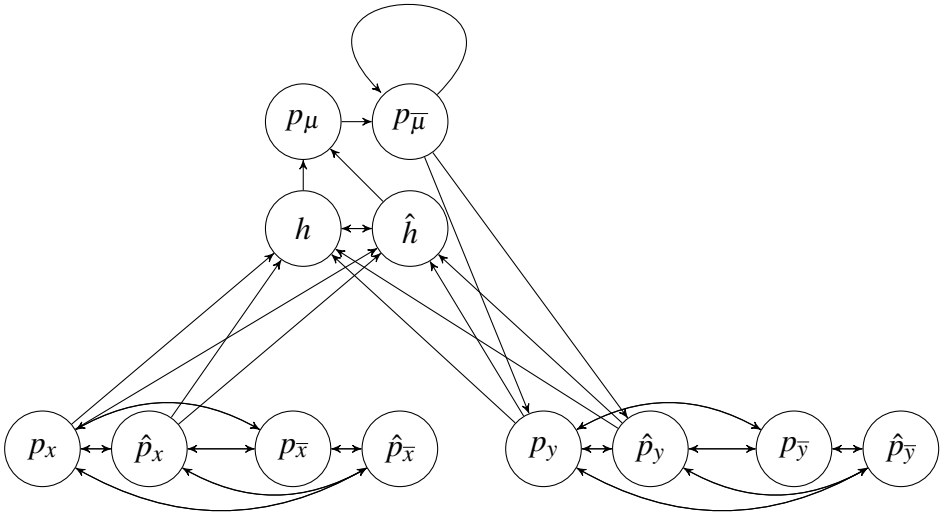


Figure A.6: Cloned abstract argumentation framework $\hat{\mathsf{AF}}_\phi$ for the QBF $\phi = \forall x : \exists y : x \vee y$.

1. *The problems* $\mathsf{EnumACC}_s^{\mathsf{GR}}$, $\mathsf{EnumACC}_c^{\mathsf{GR}}$, $\mathsf{EnumACC}_s^{\mathsf{CO}}$ *are in FP, respectively.*
2. *The problems* $\mathsf{EnumACC}_c^{CO}$, $\mathsf{EnumACC}_c^{\mathsf{PR}}$, $\mathsf{EnumACC}_c^{\mathsf{ST}}$, $\mathsf{EnumACC}_s^{\mathsf{ST}}$ *are in* $FNP^{DP[1]}$*, respectively.*
3. *The problem* $\mathsf{EnumACC}_s^{\mathsf{PR}}$ *is in* $FNP^{DP2[1]}$.

Proof. The cases in 1) follow from Proposition 1. For the other two cases, we can non-deterministically guess a set E and then verify that $E = \mathsf{Acc}_\sigma^x(\mathsf{AF})$ in DP, DP2, respectively, yielding algorithms in $\mathsf{FNP}^{\mathsf{DP}[1]}$ and $\mathsf{FNP}^{\mathsf{DP2}[1]}$, respectively. □

Proposition 5. *Let* $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ *be an abstract argumentation framework.*

1. *If* $\omega \in Mod(\Psi_{\mathsf{AF}}^{CO})$ *then* $E(\omega)$ *is a complete extension of* AF.
2. *If* E *is a complete extension of* AF *then there is* $\omega \in Mod(\Psi_{\mathsf{AF}}^{CO})$ *with* $E(\omega) = E$.
3. $a \in \mathsf{Acc}_{\mathsf{CO}}^c(\mathsf{AF})$ *if and only if* $\Psi_{\mathsf{AF}}^{CO} \wedge \mathtt{in}_a$ *is satisfiable.*

Proof. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ be an abstract argumentation framework.

1. Let $\omega \in \mathsf{Mod}(\Psi_{\mathsf{AF}}^{\mathsf{CO}})$ and define

$$E(\omega) = \{a \mid \omega(\mathtt{in}_a) = \text{TRUE}\}$$

In order to show that $E(\omega)$ is a complete extension, we have to show that $E(\omega)$ is conflict-free, admissible, and contains all all arguments it defends:

   (a) Suppose $E(\omega)$ is not conflict-free. Then there are $a, b \in E(\omega)$ such that $b \in a^-$. Due to $b \in E(\omega)$ we have $\omega(\mathtt{in}_b) = \text{TRUE}$ and due to $\omega \in \mathsf{Mod}(\Psi_{\mathsf{AF}}^{\mathsf{CO}})$ we have $\omega(\mathtt{out}_a) = \text{TRUE}$ (due to the part $(\mathtt{out}_a \Leftrightarrow \bigvee_{b \in a^-} \mathtt{in}_b)$ of $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$). Due to the part $(\neg\mathtt{in}_a \vee \neg\mathtt{out}_a)$ of $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$ we have $\omega(\mathtt{in}_a) = \text{FALSE}$, in contradiction to the assumption $a \in E(\omega)$. So $E(\omega)$ is conflict-free.

   (b) Suppose $E(\omega)$ is not admissible. Since $E(\omega)$ is conflict-free (see above), it follows that there is $a \in E(\omega)$ such that there is $b \in a^-$ and there is no $c \in b^-$ with $c \in E(\omega)$. Due to $a \in E(\omega)$, we have $\omega(\mathtt{in}_a) = \text{TRUE}$ and due to the part $(\mathtt{in}_a \Leftrightarrow \bigwedge_{b \in a^-} \mathtt{out}_b)$ of $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$ it follows $\omega(\mathtt{out}_b) = \text{TRUE}$. Then due to $(\mathtt{out}_a \Leftrightarrow \bigvee_{b \in a^-} \mathtt{in}_b)$ (with b taking the role of a) there must be $c' \in b^-$ with $\omega(\mathtt{in}_{c'}) = \text{TRUE}$ and therefore $c' \in E(\omega)$.

   (c) Due to $(\mathtt{in}_a \Leftrightarrow \bigwedge_{b \in a^-} \mathtt{out}_b) \wedge$, for every a that is attacked only by arguments b with $\omega(\mathtt{out}_b) = \text{TRUE}$, we have $\omega(\mathtt{in}_a) = \text{TRUE}$ and therefore $a \in E(\omega)$. It follows that $E(\omega)$ contains all arguments it defends.

2. Let E be a complete extension and define ω via

$$\omega(\mathtt{in}_a) = \text{TRUE} \qquad \omega(\mathtt{out}_a) = \omega(\mathtt{undec}_a) = \text{FALSE}$$

for all $a \in E$ and

$$\omega(\mathtt{out}_b) = \text{TRUE} \qquad \omega(\mathtt{in}_b) = \omega(\mathtt{undec}_b) = \text{FALSE}$$

for all $b \in E^-$ and

$$\omega(\mathtt{undec}_c) = \text{TRUE} \qquad \omega(\mathtt{in}_c) = \omega(\mathtt{out}_c) = \text{FALSE}$$

for all remaining arguments $c \in \mathsf{A} \setminus (E \cup E^-)$. It should be clear that ω is a model of $\Psi_{\mathsf{AF}}^{\mathsf{CO}}$.

3. Due to 1 and 2, $\Psi_{\mathsf{AF}}^{\mathsf{CO}} \wedge \mathtt{in}_a$ is satisfiable iff there is a complete extension E with $a \in E$, which is equivalent to $a \in \mathsf{Acc}_{\mathsf{CO}}^{c}(\mathsf{AF})$. □

Proposition 6. *Let* $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ *be an abstract argumentation framework.*

1. *If* $\omega \in \mathit{Mod}(\Psi^{ST}_{\mathsf{AF}})$ *then* $E(\omega)$ *is a stable extension of* AF.
2. *If* E *is a stable extension of* AF *then there is* $\omega \in \mathit{Mod}(\Psi^{ST}_{\mathsf{AF}})$ *with* $E(\omega) = E$.
3. $a \in \mathsf{Acc}^c_{\mathsf{ST}}(\mathsf{AF})$ *if and only if* $\Psi^{ST}_{\mathsf{AF}} \wedge \mathtt{in}_a$ *is satisfiable.*

Proof. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ be an abstract argumentation framework.

1. Let $\omega \in \mathsf{Mod}(\Psi^{\mathsf{ST}}_{\mathsf{AF}})$ and define

$$E(\omega) = \{a \mid \omega(\mathtt{in}_a) = \text{TRUE}\}$$

   In order to show that $E(\omega)$ is a stable extension, we have to show that $E(\omega)$ is conflict-free and attacks all arguments it does not contain:
   (a) Suppose $E(\omega)$ is not conflict-free. Then there are $a, b \in E(\omega)$ such that $b \in a^-$. Due to $b \in E(\omega)$ we have $\omega(\mathtt{in}_b) = \text{TRUE}$ and due to $\omega \in \mathsf{Mod}(\Psi^{\mathsf{ST}}_{\mathsf{AF}})$ we have $\omega(\mathtt{in}_a) = \text{FALSE}$ (due to the part $(\neg\mathtt{in}_a \Leftrightarrow \bigvee_{b \in a^-} \mathtt{in}_b)$ of $\Psi^{\mathsf{ST}}_{\mathsf{AF}}$). It follows $a \notin E(\omega)$, in contradiction to the assumption $a \in E(\omega)$. So $E(\omega)$ is conflict-free.
   (b) For $a \notin E(\omega)$ we have $\omega(\mathtt{in}_a) = \text{FALSE}$ and due to $(\neg\mathtt{in}_a \Leftrightarrow \bigvee_{b \in a^-} \mathtt{in}_b)$ there must be $b \in a^-$ with $\omega(\mathtt{in}_b) = \text{TRUE}$, so $b \in E(\omega)$.
2. Analogous to the proof of 2 of Proposition 5.
3. Analogous to the proof of 3 of Proposition 5. □

Proposition 7. *Algorithm* IAQ^c *is sound and complete.*

Proof. Observe that S is initialised with the empty set in line 1 and only arguments a for which $\Psi^{\sigma}_{\mathsf{AF}} \wedge \mathtt{in}_a$ is satisfiable (so arguments a for which there is an extension E with $a \in E$) are added to S in line 4. So upon termination, S contains exactly the set of credulously accepted arguments. □

Proposition 8. *Algorithm* EEE^c *is sound and complete.*

Proof. Observe that S is initialised with the empty set in line 1. In line 4, the set $E(\omega)$ is guaranteed to be an extension of AF (see Propositions 5 and 6) and added to S. So upon termination (line 6), S only contains credulously accepted arguments (showing soundness). Assume there is credulously accepted a with $a \notin S$ upon termination (towards showing completeness). Then an extension E with $a \in S$ yields a model ω_E of Ψ in line 3 (due to items 2 of Propositions 5 and 6) since all $C(\omega)$ are satisfied due to $a \notin S$ and therefore $\omega(\mathtt{in}_a) = \text{FALSE}$ for all such ω. This is in conflict with the termination criterion in line 3 and therefore $a \in S$ upon termination. □

Proposition 9. *Algorithm* SEE^c *is sound and complete.*

Proof. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{CO}, \mathsf{ST}, \mathsf{PR}\}$ and $S = \mathsf{SEE}^c(\mathsf{AF}, \sigma)$.

For soundness, let $a \in S$. Then (due to line 4) there is ω with $a \in E(\omega)$ and $\omega = \textsc{Witness}(\Psi^\sigma_{\mathsf{AF}} \wedge \bigvee_{a \in D} \mathtt{in}_a)$. Due to Proposition 5 (resp. Proposition 6), the set $E(\omega)$ is a complete (resp. stable) extension of AF, showing that a is indeed credulously acceptable wrt. complete/preferred (resp. stable) semantics.

For completeness, let $a \in \mathsf{Acc}^c_\sigma(\mathsf{AF})$ and assume $a \notin S$. Let $\hat{D}$ be the set D in the final iteration of line 3, i. e., we have $\textsc{Witness}(\Psi^\sigma_{\mathsf{AF}} \wedge \bigvee_{a \in \hat{D}} \mathtt{in}_a) = \textsc{false}$. Due to $a \notin S$ we have $a \in \hat{D}$ and since $a \in \mathsf{Acc}^c_\sigma(\mathsf{AF})$, the formula $\Psi^\sigma_{\mathsf{AF}} \wedge \bigvee_{a \in \hat{D}} \mathtt{in}_a)$ is satisfiable, contradicting $\textsc{Witness}(\Psi^\sigma_{\mathsf{AF}} \wedge \bigvee_{a \in \hat{D}} \mathtt{in}_a) = \textsc{false}$. □

Proposition 10. *Algorithm* SEEM^c *is sound and complete.*

Proof. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$, $\sigma \in \{\mathsf{CO}, \mathsf{ST}, \mathsf{PR}\}$ and $S = \mathsf{SEEM}^c(\mathsf{AF}, \sigma)$.

For soundness, let $a \in S$. Then (due to line 4) there is ω with $a \in E(\omega)$ and $\omega = \textsc{MaxSat}(\{\mathtt{in}_a \mid a \in D\}, \Psi^\sigma_{\mathsf{AF}})$. In particular, ω is a model of $\Psi^\sigma_{\mathsf{AF}}$ which assigns TRUE to $\mathtt{in}_a$. Due to Proposition 5 (resp. Proposition 6), the set $E(\omega)$ is a complete (resp. stable) extension of AF, showing that a is indeed credulously acceptable wrt. complete/preferred (resp. stable) semantics.

For completeness, let $a \in \mathsf{Acc}^c_\sigma(\mathsf{AF})$ and assume $a \notin S$. Let $\hat{D}$ be the set D in the final iteration of line 3, i. e., we have $\textsc{MaxSat}(\{\mathtt{in}_a \mid a \in \hat{D}\}, \Psi^\sigma_{\mathsf{AF}}) = \textsc{false}$. Due to $a \notin S$ we have $a \in \hat{D}$ and since $a \in \mathsf{Acc}^c_\sigma(\mathsf{AF})$, the formula $\Psi^\sigma_{\mathsf{AF}} \wedge \mathtt{in}_a$ is satisfiable, contradicting $\textsc{MaxSat}(\{\mathtt{in}_a \mid a \in \hat{D}\}, \Psi^\sigma_{\mathsf{AF}}) = \textsc{false}$. □

Proposition 11. *Let* $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ *be an abstract argumentation framework. Then* $a \in \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$ *if and only if* $\Psi^{ST}_{\mathsf{AF}} \wedge \neg \mathtt{in}_a$ *is unsatisfiable.*

Proof. $\Psi^{\mathsf{ST}}_{\mathsf{AF}} \wedge \neg \mathtt{in}_a$ is unsatisfiable if and only if there is no stable extension E of AF with $a \notin E$, which is equivalent to a being skeptically accepted, so $a \in \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$. □

Proposition 12. *Algorithm* IAQ^s *is sound and complete.*

Proof. Observe that S is initialised with the empty set in line 1 and only arguments a that are skeptically accepted are added to S in lines 5 and 8 respectively. So upon termination, S contains exactly the set of skeptically accepted arguments. □

Proposition 13. *Algorithm* EEE^s *is sound and complete.*

Proof. Observe that S is initialised with all arguments in line 1. In lines 5 and 9, respectively, the sets $E(\omega)$ (respectively E) is guaranteed to be an extension of AF and all arguments not contained in E are removed from S. So upon termination (line 10), S contains all skeptically accepted arguments (showing completeness). Assume there is an argument a that is not skeptically accepted but $a \in S$ upon termination (towards showing soundness). Then there must exist an extension E with $a \notin S$ and this extension (or another one with $a \notin S$) satisfies lines 4 or 8, respectively (or more precisely for line 4, there is a corresponding model ω for E). This is in conflict with the termination criterion in lines 4 and 8, respectively and therefore $a \notin S$ upon termination. □

Proposition 14. *Algorithm* SEE^s *is sound and complete.*

Proof. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ and $S = \mathsf{SEE}^s(\mathsf{AF})$.

For soundness, let $a \in S$. Then (due to line 3) for all ω with $\omega = \textsc{Witness}(\Psi^{\mathsf{ST}}_{\mathsf{AF}} \wedge \bigvee_{a \in S} \mathtt{out}_a)$ we have $a \in E(\omega)$ (and due to Proposition 6 these sets $E(\omega)$ are stable extensions of AF). In the final iteration of line 2 we have $\textsc{false} = \textsc{Witness}(\Psi^{\mathsf{ST}}_{\mathsf{AF}} \wedge \bigvee_{a \in S} \mathtt{out}_a)$, i. e., there is no stable extension of AF that does not include some argument of S. It follows that a is skeptically accepted wrt. stable semantics.

For completeness, let $a \in \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$ and assume $a \notin S$. First, consider the case that AF has no stable extensions. Then we have $S = \mathsf{A}$ due to line 1 and it follows $a \in S$. We, therefore, assume that AF has at least one stable extension. Since $a \in \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$, a must belong to every stable extension, in particular $a \in E(\omega)$ for every ω in line 3. It follows $a \in S$, in contradiction to the assumption. □

Proposition 15. *Algorithm* SEEM^s *is sound and complete.*

Proof. Let $\mathsf{AF} = (\mathsf{A}, \mathsf{R})$ and $S = \mathsf{SEEM}^s(\mathsf{AF})$.

For soundness, let $a \in S$. Then (due to line 3) for all ω with $\omega = \textsc{MaxSat}(\{\mathtt{out}_a \mid a \in S\}, \Psi^{\mathsf{ST}}_{\mathsf{AF}})$ we have $a \in E(\omega)$ (and due to Proposition 6 these sets $E(\omega)$ are stable extensions of AF). In the final iteration of line 2 we have $\textsc{false} = \textsc{MaxSat}(\{\mathtt{out}_a \mid a \in S\}, \Psi^{\mathsf{ST}}_{\mathsf{AF}})$, i. e., there is no stable extension of AF that does not include some argument of S. It follows that a is skeptically accepted wrt. stable semantics.

For completeness, let $a \in \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$ and assume $a \notin S$. First, consider the case that AF has no stable extensions. Then we have $S = \mathsf{A}$ due to line 1 and it follows $a \in S$. We, therefore, assume that AF has at least one stable extension. Since $a \in \mathsf{Acc}^s_{\mathsf{ST}}(\mathsf{AF})$, a must belong to every stable extension, in particular $a \in E(\omega)$ for every ω in line 3. It follows $a \in S$, in contradiction to the assumption. □