Advanced SAT Techniques for Abstract Argumentation

Johannes P. Wallner, Georg Weissenbacher, and Stefan Woltran

Institute of Information Systems, Vienna University of Technology,
Favoritenstraße 9-11, A-1040 Vienna, Austria

Abstract. In the area of propositional satisfiability (SAT), tremendous progress has been made in the last decade. Today's SAT technology covers not only the standard SAT problem, but also extensions thereof, such as computing a backbone (the literals which are true in all satisfying assignments) or minimal corrections sets (minimal subsets of clauses which if dropped leave an originally unsatisfiable formula satisfiable). In this work, we show how these methods can be applied to solve important problems from the area of abstract argumentation. In particular, we present new systems for semi-stable, ideal, and eager semantics. Our experimental results demonstrate the feasibility of this approach.

Keywords: Abstract Argumentation, Propositional Satisfiability, Argumentation Systems

1 Introduction

Argumentation is an interdisciplinary subfield of Artificial Intelligence [4] with links to psychology, linguistics, philosophy and legal theory. Formal methods of argumentation are nowadays embedded in decision support systems [1], E-Democracy tools [9], multi-agent systems [34], and many more. Dung's abstract model of argumentation [13] (and variants thereof) plays a central role in many of these applications providing a common core for diverse aspects of argumentation formalisms. This clearly calls for efficient systems and significant progress and variety in implementing Dung's argumentation semantics has been achieved over the last years (for an overview, see [11]).

One central method is to reduce the argumentation problem at hand to a formula in propositional logic. Reductions of this kind make highly sophisticated SAT solvers amenable for the field of argumentation. Using classical propositional logic to evaluate Dung-style argumentation frameworks was first advocated by Besnard and Doutre in [5] and later extended to quantified propositional logic [22, 2] in order to efficiently reduce abstract argumentation problems with complexity beyond NP. However, these methods have not been implemented yet.

The goal of this paper is to demonstrate how modern SAT technology can be used for solving such hard problems in the area of argumentation. In particular, we consider two extensions of the SAT problem, namely *minimal correction sets* (MCSes) [28, 31] and *backbones* [32]. A minimal correction set is a minimal subset

of the clauses of an unsatisfiable SAT instance which, if dropped, results in a satisfiable formula. The backbone of a propositional formula ϕ is the set of all literals that evaluate to true in all interpretations that satisfy ϕ.

We demonstrate that these methods suit particular argumentation problems surprisingly well, simplifying the design of the actual procedures. The work which is closest to the methods we propose here is the CEGARTIX system [19], which relies on iterative calls of standard SAT-solvers. Our modular approach results in reduced engineering effort, allowing for rapid prototyping of abstract argumentation systems that immediately benefit from future improvements of SAT technology. Moreover, our results indicate that MCSes and backbones can be more broadly applied to reasoning problems in the AI domain since they directly treat typical features of such problems making the design of the reductions easier compared to reductions to standard (quantified) propositional logic.

Moreover, our experimental results are very promising and show that the proposed methods are competitive to the CEGARTIX system. We recall that experimental results in [19] show that CEGARTIX outperforms other reduction approaches like ASPARTIX [21], although the number of calls to the SAT engine is exponential (with respect to the instance size) in the worst case due to the high complexity of the problems. One reason for the good performance of CEGARTIX is that it performs certain semantic-specific optimizations between the SAT-calls while in monolithic reductions like the ASPARTIX approach, where the entire problem is reduced at once and given to a "black-box" solver, the domain specific short-cuts have to be identified by the underlying systems.

The structure of the paper and its main contribution are as follows: After reviewing abstract argumentation, we present SAT-based techniques to compute backbones and MCSes (Section 2.2). Section 3 contains our main results: we provide new proof procedures for semi-stable [8] and eager semantics [7] based on MCSes and backbones; and show how the ideal semantics [14] can be realized via a backbone. In Section 4 we present our experimental evaluation showing that for the ideal semantics we achieve a significant performance gain over existing systems, and that for semi-stable reasoning we outperform the CEGARTIX system.

Our new systems and test instances are freely available under the link `www.dbai.tuwien.ac.at/research/project/argumentation/sat-based`.

2 Background

2.1 Abstract Argumentation

In this section we introduce (abstract) argumentation frameworks [13] and recall the semantics we study in this paper.

Definition 1. *An* argumentation framework (AF) *is a pair $F = (A, R)$ where A is a set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that a attacks b.*

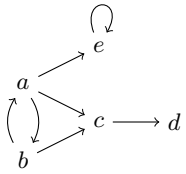

Fig. 1. Example argumentation framework

An argumentation framework can be represented as a directed graph, as shown in the following example.

Example 1. Let $F = (A, R)$ be an AF with $A = \{a, b, c, d, e\}$ and $R = \{(a, b), (b, a), (a, c), (b, c), (c, d), (e, e)\}$. The corresponding graph representation is depicted in Fig. 1.

A semantics for argumentation frameworks is given via a function σ which assigns to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. In this paper we focus on the semi-stable [8], eager [7] and ideal [14] semantics. These are based on the stable and preferred semantics [13] and a fundamental notion underlying all of these is the concept of an admissible set. Hence we consider for σ the functions *adm*, *prf*, *stb*, *sem*, *ideal*, and *eager* which stand for admissible, preferred, stable, semi-stable, ideal, and eager extensions, respectively. We will introduce these concepts in the following.

The basic concept for all the semantics considered in this paper is the admissible set. Admissibility has two requirements, namely conflict-freeness and defense of all arguments in the set.

Definition 2. *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is conflict-free in F, if there are no $a, b \in S$, such that $(a, b) \in R$. We say that an argument $a \in A$ is defended by a set $S \subseteq A$ in F if, for each $b \in A$ such that $(b, a) \in R$, there exists a $c \in S$ such that $(c, b) \in R$.*

Admissible sets are then conflict-free sets of arguments, where each argument in the set is defended by the set.

Definition 3. *Let $F = (A, R)$ be an AF. A set $S \subseteq A$ is admissible in F, if S is conflict-free in F; and each $a \in S$ is defended by S in F.*

Maximal admissible sets, w.r.t. subset-inclusion are called preferred extensions and accept as many arguments as possible, without violating admissibility.

Definition 4. *Let $F = (A, R)$ be an AF. An admissible set $S \subseteq A$ is a preferred extension in F, if there is no admissible set $S' \subseteq A$ such that $S \subsetneq S'$.*

A basic property of the preferred semantics is that admissible sets and hence preferred extensions always exist for any given framework. A popular semantics

for which this is not the case is the stable semantics. For the definition of the stable semantics and the closely related semi-stable semantics we make use of the concept of the range of a given set S of arguments, which is simply the set itself and everything it attacks, i.e. given an AF $F = (A, R)$ and $S \subseteq A$, then the range of S, denoted by S_R^+ is given by $S_R^+ \stackrel{\text{def}}{=} S \cup \{a \mid (b, a) \in R, b \in S\}$.

Definition 5. *Let $F = (A, R)$ be an AF. A conflict-free set $S \subseteq A$ in F is a stable extension in F, if $S_R^+ = A$. An admissible set E in F is a semi-stable extension in F if there does not exist a set T admissible in F, with $E_R^+ \subset T_R^+$.*

A basic property of these two semantics is that if an AF has stable extensions, then the semi-stable and stable semantics coincide [8]. The intuition is that semi-stable extensions should be "close" to stable extensions, in case no stable extensions exist.

Example 2. Consider the AF from Example 1. Then we have the following admissible sets, respectively extensions: $adm(F) = \{\emptyset, \{a\}, \{b\}, \{a, d\}, \{b, d\}\}$; $stb(F) = \{\{a, d\}\}$; $prf(F) = \{\{a, d\}, \{b, d\}\}$; and $sem(F) = \{\{a, d\}\}$. Note that if we would add a single isolated self-attacking argument to F, i.e. $F' = (A', R')$ with $A' = A \cup \{f\}$ and $R' = R \cup \{(f, f)\}$, then $stb(F') = \emptyset$, but the set of semi-stable extensions would remain the same, i.e. $sem(F') = sem(F)$.

Notice that all the semantics introduced until now in this paper may have multiple extensions. Reasoning tasks on AFs w.r.t. a semantics σ, apart from simple enumeration of all extensions, include the credulous and skeptical acceptance of arguments. An argument is credulously (skeptically) accepted for a semantics and an AF, if it is present in at least one extension (in all extensions) of the semantics.

Definition 6. *Given an AF $F = (A, R)$, a semantics σ and an argument $a \in A$ then we define the following reasoning tasks. The decision problem $\mathsf{Cred}_\sigma(a, F)$ answers* yes *if $a \in \bigcup \sigma(F)$ and* no *otherwise. The decision problem $\mathsf{Skept}_\sigma(a, F)$ answers* yes *if $a \in \bigcap \sigma(F)$ and* no *otherwise. Let $\mathsf{AllCred}_\sigma(F) \stackrel{\text{def}}{=} \bigcup \sigma(F)$ and $\mathsf{AllSkept}_\sigma(F) \stackrel{\text{def}}{=} \bigcap \sigma(F)$.*

Example 3. Applying the reasoning tasks to the AF in Example 1, we have for the preferred semantics the following credulously and skeptically accepted arguments: $\mathsf{AllCred}_{prf}(F) = \{a, b, d\}$ and $\mathsf{AllSkept}_{prf}(F) = \{d\}$.

The remaining two semantics we study in this paper are the ideal and eager semantics, which take a particular skeptical stance and are among the so-called unique-status semantics, i.e. always have a unique extension for any AF.

Definition 7. *Let $F = (A, R)$ be an AF. For an admissible set $S \in adm(F)$, it holds that*

- *$S \in ideal(F)$, if $S \subseteq \mathsf{AllSkept}_{prf}(F)$ and there is no $T \in adm(F)$ with $S \subset T \subseteq \mathsf{AllSkept}_{prf}(F)$;*

Table 1. Computational complexity of reasoning in AFs.

σ	*stb*	*adm*	*prf*	*sem*	*ideal*	*eager*
Cred_σ	NP-c	NP-c	NP-c	Σ_2^P-c	in Θ_2^P	Π_2^P-c
Skept_σ	coNP-c	trivial	Π_2^P-c	Π_2^P-c	in Θ_2^P	Π_2^P-c

– $S \in \mathit{eager}(F)$, *if* $S \subseteq \mathsf{AllSkept}_{sem}(F)$ *and there is no* $T \in \mathit{adm}(F)$ *with* $S \subset T \subseteq \mathsf{AllSkept}_{sem}(F)$.

That is, the ideal and eager extensions are the maximal-admissible sets w.r.t. subset-inclusion, composed only of arguments skeptically accepted under preferred, respectively semi-stable semantics.

Example 4. Continuing the Example 2, based on the AF in Example 1, then $\mathit{ideal}(F) = \{\emptyset\}$; and $\mathit{eager}(F) = \{\{a, d\}\}$. Note that although $\mathsf{AllSkept}_{prf}(F) = \{d\}$, the set $\{d\}$ is not admissible in F.

Given the set of skeptically accepted arguments w.r.t. preferred or semi-stable semantics to compute the unique subset-maximal admissible set composed only of the arguments skeptically accepted, we can make use of the following function, which we call restricted characteristic function [15].

Definition 8. *Let* $F = (A, R)$ *be an AF. Then* $\hat{\mathcal{F}}_F : 2^A \to 2^A$ *is the restricted characteristic function of* F *and is defined by* $\hat{\mathcal{F}}_F(S) \stackrel{\text{def}}{=} \{a \in S \mid a \text{ is defended by } S\}$.

This function iteratively removes arguments from S, which are not defended by S in F. Applying the function at most $|A|$ times for an AF $F = (A, R)$ yields the maximal admissible set $U \subseteq S$, w.r.t. subset-inclusion. Note that this function is not to be confused with the characteristic function, which one can use for defining semantics of AFs.

The computational complexity of all the semantics considered in this paper is high and in many cases "beyond" NP. The complexity of semi-stable has been investigated in [20], eager in [16] and ideal in [15]. See Table 1 for details. We briefly recall the complexity classes here. The class Σ_2^P contains decision problems that can be decided in polynomial time using a nondeterministic Turing machine with access to an NP-oracle, i.e. it can solve a problem in NP in one step. The class Π_2^P is defined as the complementary class of Σ_2^P. The class Θ_2^P contains decision problems that can be solved by a deterministic polynomial time algorithm which is allowed to make $O(n)$ non-adaptive calls to the NP-oracle.

2.2 Boolean Satisfiability

This section provides an overview of the propositional SAT problem, satisfiability solvers, and extensions of the SAT problem – in particular *minimal correction sets* [28, 31] and *backbones* [32] – and iterative SAT-based algorithms for these problems. For an introduction we refer the reader to the tutorial paper [29].

Algorithm 1 Iterative Probing (computes the backbone of ϕ)

Require: ϕ is satisfiable
Ensure: returns $\{\ell \,|\, \ell \in \{a, \neg a \,|\, a \in A\} \wedge \forall I \,.\, I \models \phi \wedge \ell \vee I \models \neg\phi\}$

```
1: S = ∅
2: let I : A → 𝔹 be such that I ⊨ φ                    ▷ I may be partial
3: for all ℓ ∈ {a, ¬a | a ∈ A} with I ⊨ ℓ do
4:     if φ ∧ ¬ℓ is unsatisfiable then
5:         S = S ∪ {ℓ}; φ = φ ∪ {ℓ}
6:     else let J be such that J ⊨ φ ∧ ¬ℓ in
7:         I = {a ↦ v | a ∈ A, v ∈ 𝔹, I(a) = v ∧ J(a) = v}
8:     end if
9: end for
10: return S
```

Propositional Logic. We work in the standard setting of propositional logic over a set $A \stackrel{\text{def}}{=} \{a, b, c, \ldots\}$ of propositional atoms, and the standard logical connectives $\wedge$, $\vee$, and $\neg$ (denoting conjunction, disjunction, and negation, respectively). A literal ℓ is an atom $a \in A$ or its negation $\neg a$. A clause C is a set of literals representing the disjunction $\bigvee_{\ell \in C} \ell$. A propositional formula in Conjunctive Normal Form (CNF) is a conjunction of clauses, also represented as a set of clauses. An interpretation $I : A \to \mathbb{B}$ maps atoms to boolean values $\mathsf{T}, \mathsf{F} \in \mathbb{B}$. An interpretation I satisfies a formula ϕ (denoted by $I \models \phi$) if ϕ evaluates to T under the (potentially partial) assignment determined by I. A formula ϕ is satisfiable if there exists an interpretation I such that $I \models \phi$, and unsatisfiable otherwise.

SAT Solvers. A satisfiability solver is a decision procedure which determines whether a given formula ϕ (in CNF) is satisfiable or not. Contemporary SAT solvers are capable of solving instances with hundreds of thousands of literals and clauses. SAT solvers largely owe their success to efficient search heuristics (e.g., [30]) and conflict-driven back-tracking [33]. The latter technique avoids the repeated exploration of similar portions of the search space by augmenting the original instance ϕ with conflict clauses C derived from ϕ (i.e., $\models \neg\phi \vee C$).

Modern SAT solvers operate in an iterative manner: conflict clauses derived from a previous instance ϕ can be retained in a subsequent run of the solver on a formula ψ if $\phi \subseteq \psi$. In addition, the back-tracking capabilities of SAT solvers make it possible to fix a tentative assignment (or *assumption*, respectively) for a subset S of A in form of a conjunction of literals over S. Assumptions can be discarded in subsequent calls. This capability to perform iterative calls is crucial to the performance of the SAT-based algorithms presented below.

Backbones. The backbone of a satisfiable propositional formula ϕ comprises the literals over A that are true in every interpretation I satisfying ϕ. To compute the backbone of a formula ϕ (with $I \models \phi$), the currently most efficient algorithms (according to [37, 32]) iteratively "probe" each atom $a \in A$ by subsequently checking the satisfiability of $\phi \wedge \ell$ (with $\ell \stackrel{\text{def}}{=} \neg a$ if $I(a) = \mathsf{T}$, and

Algorithm 2 Minimal Correction Sets

Require: $\phi \stackrel{\text{def}}{=} \bigcup_i \{C_i\}$ is unsatisfiable
Ensure: returns set $\mathcal{M}$ of all minimal correction sets for ϕ
1: $\psi = \bigcup_i \{(a_i \vee C_i)\}$, with $L \stackrel{\text{def}}{=} \bigcup_i \{a_i\}$ a set of fresh atoms
2: $k = 1$
3: $\mathcal{M} = \emptyset$
4: **while** ψ is satisfiable **do**
5: $\quad \psi_k = \psi \wedge \mathit{AtMost}(k, L)$
6: $\quad$ **while** ψ_k is satisfiable **do**
7: $\quad\quad$ **let** I be such that $I \models \psi_k$
8: $\quad\quad \mathcal{M} = \mathcal{M} \cup \{\{C_i \mid a_i \in L \wedge I(a_i) = \mathsf{T}\}\}$
9: $\quad\quad$ **let** D be $\{\neg a_i \mid a_i \in L \wedge I(a_i) = \mathsf{T}\}$
10: $\quad\quad \psi_k = \psi_k \wedge D$
11: $\quad\quad \psi = \psi \wedge D$
12: $\quad$ **end while**
13: $\quad k = k + 1$
14: **end while**
15: *return* $\mathcal{M}$

$\ell \stackrel{\text{def}}{=} a$ otherwise). Algorithm 1 illustrates the basic structure of such an implementation. Practical implementations incorporate techniques such as excluding variables with opposing values in subsequent satisfying assignments (line 7 of Algorithm 1), clause reuse, and variable filtering [37, 32].

Minimal Correction Sets. Given an unsatisfiable formula ϕ, a *minimal correction set* is a minimal subset $\psi \subseteq \phi$ such that $\phi \setminus \psi$ is satisfiable. The constraints $\chi \subseteq \phi$ are *hard* if we require that $\psi \cap \chi = \emptyset$ (conversely, the clauses $\phi \setminus \chi$ are *soft*).

Numerous techniques to compute MCSes exist (e.g., [28, 36, 24, 35]), and the field is still advancing: the algorithm presented in the upcoming publication [31], for instance, partitions ϕ into one satisfied and r unsatisfied subsets ($\mathcal{S}$ and $\mathcal{U}_1, \ldots, \mathcal{U}_r$) and computes MCSes by heuristically moving clauses from $\mathcal{U}_i$ to $\mathcal{S}$.

Our implementation for semi-stable and eager semantics (see Sections 3 and 4) does not inherently depend on the implementation details of the MCS algorithm. Algorithm 2 shows a simplified version of the algorithm in [28] that underlies our implementation. Each (soft) clause C is augmented up front with *relaxation literal* a that does not occur anywhere else in ϕ (line 1). (A common optimization is to instrument only clauses contained in an unsatisfiable subset of ϕ.) The effect of dropping C can now be simulated by choosing an interpretation which maps a to T. Given a set $L \subset A$ of relaxation literals, a cardinality constraint $\mathit{AtMost}(k, L) \stackrel{\text{def}}{=} |\{a \in L \mid I(a) = \mathsf{T}\}| \leq k$ (encoded as a propositional formula [12, 3]) limits the number of clauses that can be dropped. Algorithm 2 derives *all* MCSes by *systematically* enumerating assignments of relaxation literals for increasingly larger values of k (cf. the outer loop starting in line 4). The inner loop (line 6) enumerates all MCSes of size k by incrementally *blocking* MCSes represented by a conjunction of relaxation literals $\neg D$.

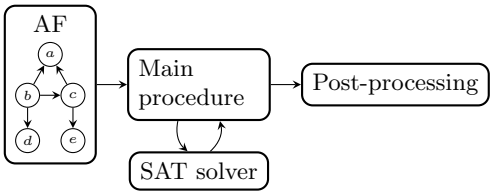

Fig. 2. Basic workflow for the algorithms based on iterative SAT procedures.

3 Algorithms

In this section we will present algorithms to solve reasoning problems associated with three kinds of complex semantics on AFs, namely the semi-stable, eager and ideal semantics. The basic idea is to utilize state-of-the-art SAT solvers. Due to the high complexity it is unlikely that we can in general compactly answer the reasoning tasks within one propositional encoding and one invocation of a SAT solver. To tackle this problem we look at multiple calls to the SAT solver, in particular *iterative* calls.

The basic workflow of our algorithms is depicted in Fig. 2. We first translate the given AF to boolean constraints, i.e. into sets of boolean clauses. The main procedure now formulates queries to the SAT solver and iteratively adapts the calls depending on already computed calls. After the main procedure is finished we apply post-processing if needed.

On a more abstract level, we apply the MCS algorithm to solve reasoning tasks under the semi-stable semantics, in particular $\mathsf{AllSkept}_{sem}$, and the backbone algorithm to solve $\mathsf{AllCred}_{adm}$. Both approaches are based on iterative calls to a SAT solver.

The eager semantics is based on the semi-stable semantics and, given the skeptically accepted arguments under the semi-stable semantics from the MCS algorithm, one can compute the unique eager extension in polynomial time by means of a post-processing step. The algorithm behind the ideal semantics is more complicated and is taken from [16]. The difficult part of this algorithm from a computational point of view is to compute $\mathsf{AllCred}_{adm}$; the remainder can be done by a similar post-processing technique as for eager.

In the following we will show how this works in detail. In Section 3.1 we show how to use MCSes (Algorithm 2) to compute the semi-stable and eager extensions, and in Section 3.2 we show how to utilize backbones (Algorithm 1) to compute the ideal extensions of a given framework. In both cases we build on existing reductions to SAT [5] for the admissible and stable semantics.

We will first recall the propositional formula representing admissible sets of a given AF from [5], in form of sets of disjunctions of atoms, i.e. in CNF. The basic idea is that every atom represents an argument. By slightly abusing our notation we use the set of arguments for a given AF and the set of propositional

atoms of the constructed formula interchangeably.

$$adm_{A,R} \stackrel{\text{def}}{=} \bigcup_{(a,b)\in R} \{(\neg a \vee \neg b)\} \cup \bigcup_{(b,c)\in R} \{(\neg c \vee \bigvee_{(a,b)\in R} a)\} \tag{1}$$

The first part of the formula (1) encodes the conflict-free property and the second part the defense of arguments. Now using the result from [5] we have for any AF $F = (A, R)$ that $adm(F) = \{S \mid I \models adm_{A,R}, S = \{a \mid I(a) = \mathsf{T}\}\}$, i.e. the interpretations satisfying $adm_{A,R}$, projected to the atoms mapped to true, directly correspond to the admissible sets of F.

3.1 MCS Algorithm for Semi-stable and Eager Semantics

Computing semi-stable extensions inherently requires to compute admissible sets, which are subset-maximal w.r.t. the range. The MCS algorithm computes subset-minimal sets of clauses of a formula in CNF, which if removed result in a satisfiable formula. The idea to exploit the MCS algorithm for the semi-stable semantics is to encode the range as satisfied clauses of a propositional formula for a given interpretation and additionally requiring that the result is admissible.

For this to work we slightly adapt the formulas from [5] for the stable semantics. Given an AF $F = (A, R)$ we define the following formulas.

$$in\_range_{a,R} \stackrel{\text{def}}{=} (a \vee \bigvee_{(b,a)\in R} b) \tag{2}$$

$$all\_in\_range_{A,R} \stackrel{\text{def}}{=} \bigcup_{a\in A} \{in\_range_{a,R}\} \tag{3}$$

The formula $in\_range_{a,R}$ indicates whether the argument a is in the range w.r.t. the atoms set to true in an interpretation. In other words, for an AF $F = (A, R)$ and $a \in A$ we have, $I \models in\_range_{a,R}$ iff $a \in S_R^+$ for $S = \{b \mid I(b) = \mathsf{T}\}$. The formula $all\_in\_range_{A,R}$ is satisfied if all arguments are in the range. Taking the formulas $adm_{A,R}$ and $all\_in\_range_{A,R}$ together conjunctively, denoted by $stb_{A,R}$, results in a formula equivalent to the stable formula in [5].

$$stb_{A,R} \stackrel{\text{def}}{=} adm_{A,R} \cup all\_in\_range_{A,R} \tag{4}$$

An interpretation I which satisfies $adm_{A,R}$ for a given AF $F = (A, R)$ and a *subset-maximal* set of clauses of $all\_in\_range_{A,R}$ corresponds to a semi-stable extension of F. Consequently, we can derive semi-stable extensions from the correction sets computed with the MCS algorithm, as long as no clause from $adm_{A,R}$ is dropped. That is, we consider the clauses of the formula $adm_{A,R}$ as *hard* constraints and the clauses in $all\_in\_range_{A,R}$ as *soft* constraints. Note also, since any AF $F = (A, R)$ has at least one admissible set, we know that $adm_{A,R}$ is always satisfiable. If $stb_{A,R}$ is satisfiable, meaning that F has stable extensions, then immediately this computation yields the stable extensions, which are equal to the semi-stable extensions.

The following proposition shows this result more formally. For a given propositional formula ϕ in CNF and an interpretation I, we define ϕ^I to be the set of clauses in ϕ, which are satisfied by I, i.e. $\phi^I \stackrel{\text{def}}{=} \{C \in \phi \mid I \models C\}$.

Proposition 1. *Let $F = (A, R)$ be an AF and $\mathcal{I}_{sem} = \{S \mid I \models adm_{A,R}, S = \{a \mid I(a) = \mathsf{T}\}, \nexists I' : I' \models adm_{A,R}$ s.t. $all\_in\_range^I_{A,R} \subset all\_in\_range^{I'}_{A,R}\}$. Then $sem(F) = \mathcal{I}_{sem}$.*

Proof. Let $F = (A, R)$ be an AF. Assume $E \in sem(F)$, then define the following interpretation I with $I(a) = \mathsf{T}$ iff $a \in E$. Then $I \models adm_{A,R}$, since E is admissible by definition and due to [5] we know that I satisfies $adm_{A,R}$. Suppose now there exists an interpretation I' such that $I' \models adm_{A,R}$ and $all\_in\_range^I_{A,R} \subset all\_in\_range^{I'}_{A,R}$. But then E would not be maximal w.r.t. the range and hence no semi-stable extension of F.

Assume $E \in \mathcal{I}_{sem}$, which implies $E \in adm(F)$ and as above let I be an interpretation with $I(a) = \mathsf{T}$ iff $a \in E$. Suppose there exists a set $S \in adm(F)$ with $E^+_R \subset S^+_R$. Then $all\_in\_range^I_{A,R} \subset all\_in\_range^{I'}_{A,R}$ for an interpretation I' defined as $I'(a) = \mathsf{T}$ iff $a \in S$, which is a contradiction.

The MCS algorithm can now be straightforwardly applied for the reasoning tasks for the semi-stable semantics we study in this paper, that is the algorithm can be easily adapted to yield an enumeration of all semi-stable extensions, answer credulous or skeptical queries or enumerate all arguments skeptically accepted. Since we need the set of skeptically accepted arguments for computation of the eager extension, we will present this variant in Algorithm 3.

Algorithm 3 MCS-AllSkept$_{sem}$

Require: AF $F \stackrel{\text{def}}{=} (A, R)$
Ensure: returns AllSkept$_{sem}(F)$

$\phi = \{a_i \vee C_i \mid C_i \in all\_in\_range_{A,R}\}$ with $L \stackrel{\text{def}}{=} \bigcup_i \{a_i\}$ a set of fresh atoms
$\psi = adm_{A,R} \cup \phi$
$k = 0$
$X = A$
while ψ is satisfiable and $k \leq |A|$ **do**
    $\psi_k = \psi \cup AtMost(k, L)$
    $X = X \cap Probing(\psi_k)$
    while ψ_k is satisfiable **do**
        let I be such that $I \models \psi_k$
        let D be $\{\neg a_i \mid a_i \in L \wedge I(a_i) = \mathsf{T}\}$
        $\psi_k = \psi_k \wedge D$
        $\psi = \psi \wedge D$
    end while
    $k = k + 1$
end while
return X

Algorithm 3 computes the set $\mathsf{AllSkept}_{sem}(F)$ for a given AF $F = (A, R)$. The formula ψ consists of the clauses for admissibility and the instrumented clauses of $all\_in\_range_{A,R}$, i.e. these clauses may be dropped during the running time. The idea is that if $I \models \psi_k$, then $E = \{a \mid I(a) = \mathsf{T}\}$ is an admissible set in F and $|E_R^+| = |A| - k$, since we allow to drop k clauses of $all\_in\_range_{A,R}$ and block previously computed MCSes. This means that E is a semi-stable extension of F, since there is no assignment I' which satisfies $adm_{A,R}$ and a superset of $all\_in\_range^I_{A,R}$. We need to slightly modify Algorithm 2 to incorporate our reasoning task. We utilize the backbone algorithm in line 7 to compute in X the set of skeptically accepted arguments. Since all satisfying interpretations of ψ_k are semi-stable extensions we compute the set of atoms set to true in all such interpretations by applying Algorithm 1. There exists alternatives and optimizations to compute MCSes and Algorithm 3 can be adapted to work with these as long as all satisfying assignments can be computed w.r.t. the formula reduced by each of its MCSes separately.

Using the Algorithm 3 for solving the $\mathsf{AllSkept}_{sem}$ problem, we can use its output to calculate the unique eager extension, since we just have to compute the subset-maximal admissible set within $\mathsf{AllSkept}_{sem}(F)$ for an AF F. For this we apply the restricted characteristic function a number of times bounded by the number of arguments in the framework, i.e. $\hat{\mathcal{F}}_F^{|A|}(\mathsf{AllSkept}_{sem}(F))$ results in the eager extension of F.

3.2 Backbone Algorithm for Ideal Semantics

For the ideal semantics we make use of a method proposed in [16], which we recall in Algorithm 4. The important point for our instantiation of this algorithm is that we essentially need to compute $\mathsf{AllCred}_{adm}$ and afterwards again, as before for the eager semantics, a post-processing with the function $\hat{\mathcal{F}}_F$. We define for an AF $F = (A, R)$ the auxiliary notion of adjacent arguments of an argument: $adj(a) \stackrel{\text{def}}{=} \{x \mid (x, a) \in R \text{ or } (a, x) \in R\}$. Additionally we define the restriction of an attack relation for a set S by $R_{|S} \stackrel{\text{def}}{=} \{(a, b) \in R \mid a \in S \text{ and } b \in S\}$.

Briefly put, Algorithm 4 computes first the credulously accepted arguments w.r.t. admissible sets and then a set X, which consists of all of the credulously accepted arguments, except those, which have an adjacent argument also credulously accepted. This set acts as a kind of approximation of the skeptically

Algorithm 4 Ideal-Extension [16]

Require: AF $F \stackrel{\text{def}}{=} (A, R)$
Ensure: returns $ideal(F)$
1: $Cred = \mathsf{AllCred}_{\mathsf{adm}}(F)$
2: $Out = A \setminus Cred$
3: $X = \{x \in Cred \mid adj(x) \subseteq Out\}$
4: $F' = (X \cup Out, R_{|(X \cup Out)})$
5: *return* $\hat{\mathcal{F}}_{F'}^{|A|}(X)$

accepted arguments w.r.t. the preferred semantics. Constructing then the new framework F' and computing the restricted characteristic function at most $|A|$ times in this new framework for X, suffices for computing the ideal extension.

Now it is straightforward to instantiate this with the help of a backbone algorithm. Given an AF $F = (A, R)$, we first simply compute the backbone of $adm_{A,R}$. Let S be the output of Algorithm 1 on this formula, then $O = \{a \mid \neg a \in S\}$ be the set of variables set to false in every satisfying interpretation of $adm_{A,R}$. Since we know that this formula is satisfiable, this means that $(A \setminus O) = \mathsf{AllCred}_{adm}(F)$. The rest of Algorithm 4 can be achieved with post-processing.

4 Experimental Evaluation

In this section we will present our concrete implementation of the presented algorithms and an experimental evaluation.

In our implementations the overall workflow from Fig. 2 is handled by Unix shell scripts and for the main procedures we utilize already implemented MCS and backbone solvers. For all our implementations we adopt the input language from the ASPARTIX system [21], a system capable of solving many problems on AFs, based on the answer-set programming (ASP) paradigm. The first step of the workflow, the translation of this language to a boolean formula, is handled by a parser implemented in C++.

Our instantiation of the presented algorithm for semi-stable semantics covers the reasoning tasks of enumerating all extensions, credulous and skeptical reasoning, as well as computing the set of all skeptically accepted arguments for a given AF. For the main procedure we utilize the CAMUS solver [28] in version 1.0.5, which we slightly modified to handle our reasoning tasks. The distinction between hard and soft constraints is implemented in CAMUS by the possibility to supply additional clauses for the relaxation literals. Based on this we implemented the necessary post-processing for computing the eager extension with an ASP call using the clingo ASP solver [26], version 3.0.4. Note that the post-processing step is inherently computable in polynomial time, the ASP solver for this step was used for its declarative and easy-to-use nature. The implementation for the ideal semantics to compute its unique extensions is based on the backbone solver JediSAT [37], version 0.2 beta. The post-processing step is again handled by an ASP call.

To show the feasibility of our approach, we conducted preliminary experiments for checking the performance of the presented algorithms. All tests were executed under OpenSUSE with Intel Xeon processors (2.33 GHz) and 49 GB memory. We note that this high amount of memory is not actually used by our algorithms, we set a hard limit of 4 GB memory usage on all runs, which was never reached.

Regarding test instances for our experiments, we note that, as identified in [17], there is still need for benchmark libraries for AFs. Without such standard libraries artificially generated AFs are the main source of test instances. Therefore, we follow the line of [19] for benchmarking and used randomly gen-

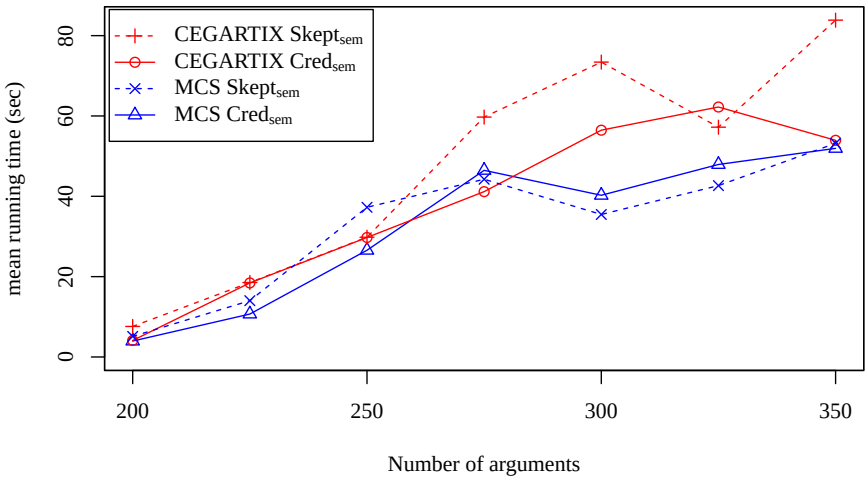

Fig. 3. Mean running time for CEGARTIX and the MCS-based algorithm.

erated AFs for testing. For our random creation of AFs we fix a number of arguments and insert for any pair of arguments (a, b) with $a \neq b$ the attack from a to b with a given probability $p \in \{0.1, 0.2, 0.3, 0.4\}$. For each parameter we created ten random AFs. We considered AFs (A, R) of size $|A| \in \{100, 150, 200, 225, 250, 275, 300, 325, 350\}$, which totaled in 360 AFs. We have chosen to use larger random AFs than in [19], since the SAT-based procedures appear to be able to handle small-sized AFs very well.

For all runs we enforced a timeout of five minutes and measure the whole time for the workflow from Fig. 2, i.e. combining parsing, solving and post-processing time. We tested the following reasoning tasks.

- Credulous and skeptical reasoning for semi-stable semantics
- Enumeration of all semi-stable extensions
- Computing the ideal extension
- Computing the eager extension

We compare credulous and skeptical reasoning for semi-stable semantics with CEGARTIX [19], a SAT-based system for reasoning tasks in abstract argumen-

Table 2. Number of solved instances for CEGARTIX and the MCS-based algorithm.

reasoning task \ $\lvert A\rvert$	200	225	250	275	300	325	350	% solved overall
CEGARTIX Cred$_{sem}$	120	120	112	91	71	64	50	74.8%
CEGARTIX Skept$_{sem}$	120	120	104	84	69	60	48	72%
MCS Cred$_{sem}$	117	120	117	111	85	77	76	83.7%
MCS Skept$_{sem}$	117	120	116	102	79	73	73	81%

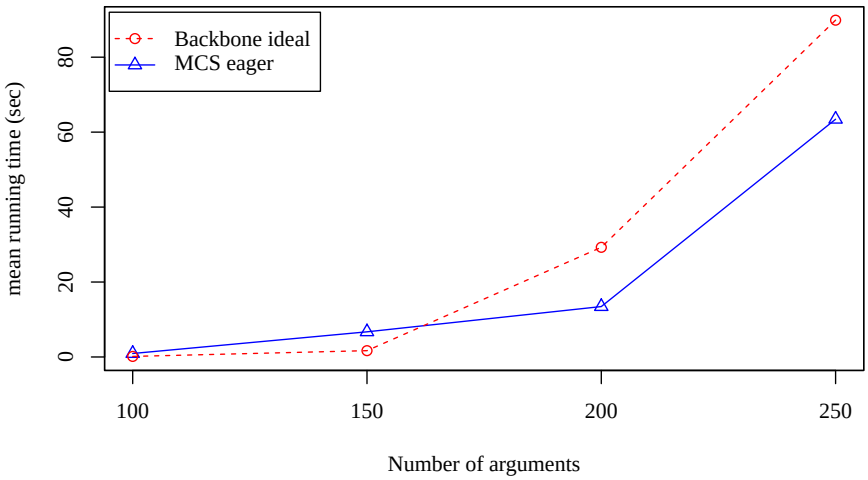

Fig. 4. Mean running time for computing the ideal respectively eager extension.

tation, which was shown to be a competitive solver. We chose version 0.1a of CEGARTIX for our tests, since in this version CEGARTIX is able to utilize incremental SAT-solving techniques and further versions of CEGARTIX mainly feature capabilities to use different SAT solvers. We let both CEGARTIX and the MCS-based approach compute the queries for three pre-specified arguments for AFs with at least 200 arguments, i.e. credulous and skeptical acceptance with three different arguments. This gives us 120 queries per AF size and in total 840 queries. The results are summarized in Fig. 3, where we show the mean running time in seconds for both approaches, *excluding* timed out runs. We grouped together queries on AFs with the same number of arguments. We see that the MCS-based approach is competitive and in cases even somewhat outperforming CEGARTIX. Note that by excluding the timeouts, which are shown in Table 2, the figures slightly favor CEGARTIX for large AFs.

It is interesting to note that the expected edge density, which we set between 0.1 and 0.4 appears to play an important role for the performance of the SAT-based approaches. Out of the total 212 timeouts encountered for credulous reasoning under semi-stable semantics for the solver CEGARTIX for all considered queries, 113 were on AFs with 0.1, 75 on AFs with 0.2 and 24 on AFs with 0.3 expected edge density. Showing a similar picture, the MCS-approach had 137 total timeouts and 105 of them with 0.1 and 32 with 0.2 expected edge density. For skeptical reasoning the results are similar.

For comparing our MCS-approach w.r.t. the enumeration of all semi-stable extensions we use an ASP approach [18] utilizing metasp for our performance test. For this ASP approach we used gringo 3.0.5 and claspD 1.1.4 [26]. We tested both approaches on the same AFs as for the credulous and skeptical

reasoning under semi-stable semantics and out of the 280 AFs we tested, the MCS-approach solved (i.e. enumerated all semi-stable extensions) 172 instances while ASP with metasp solved only seven instances within the time limit of five minutes.

For ideal and eager semantics, we report the mean computation time for AFs of size $|A| \in \{100, 150, 200, 250\}$ in Fig. 4 to compute the unique extension. Hence we compute the ideal respectively eager extension for each AF separately, which gives us 40 computations per number of arguments and 160 such calls in total per semantics. We encountered one timeout for eager reasoning on AFs with size 200 and ten with AFs of size 250. For ideal reasoning we encountered 17 timeouts with AFs of size 250. Other systems capable of solving these tasks are e.g. ASPARTIX, but which could only solve instances with a low number of arguments, i.e. AFs with less than 30 arguments, which is the reason we excluded this system in a comparison with our implementations. For ideal reasoning ASPARTIX uses a complex ASP encoding technique [23] for the DLV solver [27] (we used build BEN/Dec 16 2012 of DLV). The system ConArg [6], which is based on constraint satisfaction solvers, appears to be more competitive. ConArg is a visual tool, so more detailed performance comparisons are subject of future work. We tested some randomly generated AFs with 100 and 150 arguments and let ConArg compute the ideal extension, which it solved within ten seconds for the AFs with 100 arguments and took more than a minute for AFs with 150 arguments, but one has to factor in that a graphical representation of large graphs may consume a part of the resources needed for solving the problem.

5 Conclusion

In this paper, we presented new algorithms utilizing extensions of the SAT problem for hard tasks in abstract argumentation. In particular we showed how to solve reasoning tasks under the semi-stable and eager semantics using an MCS solver and based an algorithm for the ideal semantics on the computation of a backbone of a boolean formula. Reduction-based approaches for semantics in abstract argumentation include transformations to equational systems [25], propositional logic [5] and quantified boolean formulas [2, 22]. Our approach differs from these in that we do not use a single encoding for the whole problem, but rather solving partial problems iteratively using solvers for extensions to the SAT problem. Preliminary experiments using our approaches are very promising, showing a good performance without much engineering effort. The benefit of applying SAT-solvers for abstract argumentation is also witnessed by a very recent related approach [10] for enumeration of preferred extensions. Our approach for semi-stable semantics can be adapted for preferred semantics and a performance comparison with the systems [10, 19] is an interesting subject for future work. Further interesting directions are on one side incorporating optimizations developed in the SAT community for our approaches and on the other side applying the proposed methods to further hard problems in abstract argumentation and extensions thereof. Not in the least, this indicates that modern

SAT technology might be well applicable to other hard problems in the areas of knowledge representation and AI.

Acknowledgements

This research has been supported by the Austrian Science Fund (FWF) through projects I1102 and P25518-N23, and the Austrian National Research Network S11403-N23 (RiSE), as well as the Vienna Science and Technology Fund (WWTF) through project VRG11-005.

References

1. Amgoud, L., Prade, H.: Using arguments for making and explaining decisions. Artif. Intell. 173(3-4), 413–436 (2009)
2. Arieli, O., Caminada, M.W.A.: A QBF-based formalization of abstract argumentation semantics. J. Applied Logic 11(2), 229–252 (2013)
3. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality networks: a theoretical and empirical study. Constraints 16(2), 195–221 (2011)
4. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in Artificial Intelligence. Artif. Intell. 171(10-15), 619–641 (2007)
5. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: NMR'04. pp. 59–64 (2004)
6. Bistarelli, S., Santini, F.: Conarg: A constraint-based computational framework for argumentation systems. In: ICTAI'11. pp. 605–612 (2011)
7. Caminada, M.W.A.: Comparing two unique extension semantics for formal argumentation: Ideal and eager. In: BNAIC'07. pp. 81–87 (2007)
8. Caminada, M.W.A., Carnielli, W.A., Dunne, P.E.: Semi-stable Semantics. J. Log. Comput. 22(5), 1207–1254 (2012)
9. Cartwright, D., Atkinson, K.: Using computational argumentation to support e-participation. IEEE Intelligent Systems 24(5), 42–52 (2009)
10. Cerutti, F., Dunne, P.E., Giacomin, M., Vallati, M.: A SAT-based Approach for Computing Extensions on Abstract Argumentation. In: TAFA'13 (2013)
11. Charwat, G., Dvořák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Implementing Abstract Argumentation – A Survey. Technical Report DBAI-TR-2013-82, Vienna University of Technology (2013)
12. Codish, M., Zazon-Ivry, M.: Pairwise cardinality networks. In: LPAR'10. LNCS, vol. 6355, pp. 154–172. Springer (2010)
13. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77(2), 321–358 (1995)
14. Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artif. Intell. 171(10-15), 642–674 (2007)
15. Dunne, P.E.: The computational complexity of ideal semantics. Artif. Intell. 173(18), 1559–1591 (2009)
16. Dvořák, W., Dunne, P.E., Woltran, S.: Parametric properties of ideal semantics. In: IJCAI'11. pp. 851–856 (2011)
17. Dvořák, W., Gaggl, S.A., Szeider, S., Woltran, S.: Benchmark libraries for argumentation. In: Agreement Technologies, LGTS, vol. 8, chap. The Added Value of Argumentation, pp. 389–393. Springer (2013)

18. Dvořák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Making use of advances in answer-set programming for abstract argumentation systems. In: INAP'11. pp. 117–130 (2011)
19. Dvořák, W., Järvisalo, M., Wallner, J.P., Woltran, S.: Complexity-sensitive decision procedures for abstract argumentation. In: KR'12. pp. 54–64 (2012)
20. Dvořák, W., Woltran, S.: Complexity of semi-stable and stage semantics in argumentation frameworks. Inf. Process. Lett. 110(11), 425–430 (2010)
21. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument and Computation 1(2), 147–177 (2010)
22. Egly, U., Woltran, S.: Reasoning in argumentation frameworks using quantified boolean formulas. In: COMMA'06. FAIA, vol. 144, pp. 133–144 (2006)
23. Faber, W., Woltran, S.: Manifold answer-set programs and their applications. In: LPNMR'11, LNCS, vol. 6565, pp. 44–63. Springer (2011)
24. Felfernig, A., Schubert, M., Zehentner, C.: An efficient diagnosis algorithm for inconsistent constraint sets. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 26(1), 53–62 (2012)
25. Gabbay, D.M.: An equational approach to argumentation networks. Argument & Computation 3(2-3), 87–142 (2012)
26. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam Answer Set Solving Collection. AI Communications 24(2), 105–124 (2011)
27. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Trans. Comput. Log. 7(3), 499–562 (2006)
28. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reasoning 40(1), 1–33 (2008)
29. Malik, S., Weissenbacher, G.: Boolean satisfiability solvers: techniques and extensions. In: Software Safety and Security - Tools for Analysis and Verification. NATO Science for Peace and Security Series, IOS Press (2012)
30. Malik, S., Zhao, Y., Madigan, C.F., Zhang, L., Moskewicz, M.W.: Chaff: Engineering an efficient SAT solver. DAC'01 pp. 530–535 (2001)
31. Marques-Silva, J., Heras, F., Janota, M., Previti, A., Belov, A.: On computing minimal correction subsets. In: IJCAI'13 (2013)
32. Marques-Silva, J., Janota, M., Lynce, I.: On computing backbones of propositional theories. In: ECAI'10. FAIA, vol. 215, pp. 15–20. IOS Press (2010)
33. Marques-Silva, J., Sakallah, K.A.: GRASP – a new search algorithm for satisfiability. In: ICCAD'96. pp. 220–227 (1996)
34. McBurney, P., Parsons, S., Rahwan, I. (eds.): Argumentation in Multi-Agent Systems - 8th International Workshop, ArgMAS 2011, Revised Selected Papers, LNCS, vol. 7543. Springer (2012)
35. Nöhrer, A., Biere, A., Egyed, A.: Managing SAT inconsistencies with HUMUS. In: Workshop on Variability Modelling of Software-Intensive Systems. pp. 83–91. ACM (2012)
36. Rosa, E.D., Giunchiglia, E., Maratea, M.: Solving satisfiability problems with preferences. Constraints 15(4), 485–515 (2010)
37. Zhu, C.S., Weissenbacher, G., Sethi, D., Malik, S.: SAT-based techniques for determining backbones for post-silicon fault localisation. In: HLDVT. pp. 84–91 (2011)