Enforcement in Abstract Argumentation via Boolean Optimization

Andreas Niskanen

MSc thesis
University of Helsinki
Department of Computer Science

Helsinki, November 30, 2016

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Andreas Niskanen

Työn nimi — Arbetets titel — Title

Enforcement in Abstract Argumentation via Boolean Optimization

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
MSc thesis	November 30, 2016	58

Tiivistelmä — Referat — Abstract

Computational aspects of argumentation are a central research topic of modern artificial intelligence. A core formal model for argumentation, where the inner structure of arguments is abstracted away, was provided by Dung in the form of abstract argumentation frameworks (AFs). AFs are syntactically directed graphs with the nodes representing arguments and edges representing attacks between them. Given the AF, sets of jointly acceptable arguments or extensions are defined via different semantics. The computational complexity and algorithmic solutions to so-called static problems, such as the enumeration of extensions, is a well-studied topic.

Since argumentation is a dynamic process, understanding the dynamic aspects of AFs is also important. However, computational aspects of dynamic problems have not been studied thoroughly. This work concentrates on different forms of enforcement, which is a core dynamic problem in the area of abstract argumentation. In this case, given an AF, one wants to modify it by adding and removing attacks in a way that a given set of arguments becomes an extension (extension enforcement) or that given arguments are credulously or skeptically accepted (status enforcement).

In this thesis, the enforcement problem is viewed as a constrained optimization task where the change to the attack structure is minimized. The computational complexity of the extension and status enforcement problems is analyzed, showing that they are in the general case NP-hard optimization problems. Motivated by this, algorithms are presented based on the Boolean optimization paradigm of maximum satisfiability (MaxSAT) for the NP-complete variants, and counterexample-guided abstraction refinement (CEGAR) procedures, where an interplay between MaxSAT and Boolean satisfiability (SAT) solvers is utilized, for problems beyond NP. The algorithms are implemented in the open source software system Pakota, which is empirically evaluated on randomly generated enforcement instances.

Avainsanat — Nyckelord — Keywords
abstract argumentation, argumentation dynamics, computational complexity, maximum satisfiability

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

Argumentation is intrinsically present in many aspects of human interaction and communication. For instance, political debates and legal discussions contain arguments presented by agents with different values and beliefs. Thus it is essential to understand the mechanisms of argumentation in an exact manner. Within recent years, the study of representational and computational aspects of argumentation has become a core topic of artificial intelligence research. There is also evidence of various applications of describing, analyzing, and implementing argumentation systems, including decision support tools [3], legal reasoning [20] and multi-agent systems [72].

Several formal models of argumentation have been developed. These can in general be divided into structured and abstract models. Structured argumentation studies the construction of arguments, which are often regarded as a pair of premises and a conclusion [24]. In this way conflicts, or attacks between arguments can be defined precisely, and different kinds of attacks can be distinguished, e.g., undercutting (attacking a premise) and rebutting (attacking a conclusion) [24].

In this work, we focus on abstract models. Abstract argumentation does not take into account the internal structure of arguments, but rather regards them as atomic, abstract entities denoted by symbols. The main focus is hence on the relation between the arguments, and the goal is to determine from the relation which arguments can be jointly accepted. The central formal model for abstract argumentation was provided in [49] in the form of abstract argumentation frameworks (AFs). Syntactically, AFs are directed graphs, with nodes representing the arguments and edges representing attacks between arguments. Given an AF, one wants to define which arguments can be accepted in a single point of view. This is achieved via different kinds of AF semantics [12], which define extensions, i.e., jointly acceptable subsets of arguments, of the given AF. Acceptance of an argument can then be defined by the argument being contained in some (credulous acceptance) or all (skeptical acceptance) extensions.

Computational problems in the field of abstract argumentation, such as checking the acceptance of an argument or verifying whether a given set is indeed an extension, are often computationally very hard, namely NP-complete or even surpassing NP [53]. There are several algorithmic solutions to computational problems over AFs, with multiple system implementations available [34], and even a biannual competition for evaluating the implementations [87]. While some of the approaches are specialized algorithms for a specific computational problem [19, 77, 78], most handle the computational complexity via a declarative approach. This means that the original problem instance is encoded using a constraint modeling language, such as Boolean satisfiability (SAT) [32, 33, 56], answer set programming (ASP) [54, 59, 61], or finite-domain constraint programming (CP) [29]. The problem instance

is then solved as an instance of the corresponding constraint satisfaction problem, and the solution is decoded back into a solution of the original problem. These approaches tend to be today the most efficient in terms of the empirical runtime [87], which is largely due to the combination of improvements in solver technology and advances in modeling the computational problems in efficient ways. Some approaches to problems beyond the complexity class NP utilize iterative approaches, where e.g. SAT solvers are used as practical NP-oracles by calling them several times, refining the solution each time [32, 33, 56].

The computational problems where we perform acceptance queries on a given AF are in this work regarded as static, or non-dynamic, since we assume that the AF does not change in time. However, this is not generally a realistic assumption, since argumentation is a dynamic process—new arguments or even new agents can enter a dialogue at any point, and original arguments and attacks may lose their validity. It is therefore natural to study dynamic aspects of argumentation frameworks, which has been an increasingly active area of research in formal argumentation [15, 18, 28, 41, 42, 45, 46, 83]. On the other hand, there has been little research on computational aspects of dynamic problems in abstract argumentation. The aim of this work is to bridge this gap by analyzing and providing algorithms for different forms of the enforcement problem [16, 17, 28, 43, 75, 90], which is a fundamental problem in dynamics of abstract argumentation.

In this context enforcing means in general changing the original argumentation framework in light of new information, in such a way that given properties are satisfied in the new framework. Often one seeks to also minimize the amount of change, in which case enforcement can be seen as a discrete optimization problem. We focus on two forms of enforcement, namely, argument-fixed extension enforcement and status enforcement. In extension enforcement, the given AF is modified by adding and removing attacks in a way that the given set of arguments becomes an extension or a part of it. In status enforcement, on the other hand, the property to satisfy is the credulous or skeptical acceptance of a given set of arguments, also by adding and removing attacks.

The main contributions of this thesis are the following.

- We establish the computational complexity of extension and status enforcement under several central AF semantics, such as the admissible, complete, stable and preferred semantics, showing that there are polynomial-time solvable, NP-complete and second-level Σ_2^P-complete variants of the problems.

- We provide algorithms for the NP-hard variants of the problem based on harnessing Boolean satisfiability and maximum satisfiability (MaxSAT) solvers. For the NP-complete variants, we describe partial maximum satisfiability encodings of the problem. For the Σ_2^P-complete ones, we

provide a procedure based on a counterexample-guided abstraction refinement (CEGAR) [37, 38] scheme instead of encoding the problem in MaxSAT, since the direct encoding would presumably be exponential in size.

- We implement the MaxSAT encodings and the CEGAR algorithms, resulting in the software system Pakota. Pakota is the first system in its generality for solving enforcement problems. To the best of our knowledge, the single other solver for extension enforcement was recently proposed in [43], but supports only the stable semantics. We describe the Pakota system in detail, and evaluate it empirically on randomly generated enforcement benchmark instances, using different MaxSAT solvers, providing an overview to the scalability of the system and the impact of the choice of the MaxSAT solver. In addition, we provide benchmarks and benchmark generators for enforcement instances.

This thesis is organized as follows. In Section 2 we define abstract argumentation frameworks, several kinds of AF semantics considered in this work, and the notions of credulous and skeptical acceptance of an argument. In Section 3 extension and status enforcement are introduced and precisely defined as optimization problems. The computational complexity of the corresponding decision problems is analyzed in Section 4 under different AF semantics and other problem parameters, providing complexity proofs for a large part of the NP-complete variants. Section 5 provides an overview of Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT)—the generalization of SAT to an optimization problem. In Section 6 we propose encodings of the NP-complete enforcement problems in MaxSAT, using it as a declarative language. In addition, an approach based on counterexample-guided abstraction refinement is deployed in order to solve the problems beyond the complexity class NP, detailed in Section 7. The system implementing these encodings and procedures—Pakota—is introduced in Section 8, with details on the system architecture, features, algorithms, input format, usage, and options. Pakota is empirically evaluated and the results are presented in Section 9, also providing results on the impact of the choice of the MaxSAT solver. Finally, we conclude the thesis in Section 10 by stating the main contributions and future directions of this work.

Some of the results of this thesis have been published in international conferences. In the 30th AAAI Conference on Artificial Intelligence (AAAI-16) [90], the complexity of extension enforcement was analyzed and algorithms for solving the problem provided. In the 25th International Joint Conference on Artificial Intelligence (IJCAI-16) [75] we focused on the status enforcement problem, also providing complexity analysis and algorithms. In the 15th European Conference on Logics in Artificial Intelligence (JELIA-16) [74], a detailed description of the Pakota system was provided.

2 Abstract Argumentation

In this section we overview some of the central concepts of abstract argumentation. We start by defining argumentation frameworks, the central formal model for abstract argumentation as provided by [49], and the notion of defense in an argumentation framework.

Definition 1. An *argumentation framework (AF)* is a pair $F = (A, R)$ where A is the (finite) set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that argument a attacks argument b. An argument $a \in A$ is *defended* (in F) by a set $S \subseteq A$ if, for each $b \in A$ such that $(b, a) \in R$, there exists a $c \in S$ such that $(c, b) \in R$.

Argumentation frameworks can be regarded quite intuitively as directed graphs, where nodes represent the arguments and edges represent attacks between individual arguments, as is shown by the following example.

Example 1. *Figure 1 illustrates two argumentation frameworks $F = (A, R)$ and $F' = (A, R')$, where $A = \{a, b, c, d\}$, $R = \{(a, b), (b, c), (c, d), (d, c)\}$ and $R' = \{(a, b), (b, c), (c, c), (c, d)\}$. In the AF F, the argument c is defended by the set $\{a, c\}$, since for both attacks on c there exists a counterattack.*

Figure 1: Two simple argumentation frameworks F (left) and F' (right).

Given an AF, semantics define sets of jointly acceptable arguments, i.e., extensions, in different ways. Formally, a semantics is a function σ which maps each AF $F = (A, R)$ to a set $\sigma(F) \subseteq 2^A$ of extensions. An extension can be viewed as a particular justified point of view one can assume in a discussion.

In this work semantics $\sigma \in \{adm, com, grd, prf, stb, sem, stg\}$ are considered, which stand for the admissible, complete, grounded, preferred, stable, semi-stable and stage semantics, respectively. The admissible, complete, grounded, stable, and preferred semantics [49] are considered as the classical AF semantics, also called Dung's semantics (although [49] did not consider admissible as a semantics). Later on, the semi-stable semantics was introduced in [31] and the stage semantics in [89]. Other meaningful AF semantics exist as well, e.g., the naive [30, 40], ideal [50], and cf2 [13, 14], but are not considered in this work. The notion of the characteristic function and the range allows us to define the considered semantics in a concise manner.

Definition 2. Given an AF $F = (A, R)$, the *characteristic function* $\mathcal{F}_F : 2^A \to 2^A$ of F is $\mathcal{F}_F(S) = \{x \in A \mid x \text{ is defended by } S\}$. Moreover, for a set $S \subseteq A$, the *range* of S is $S^+_R = S \cup \{x \mid (y, x) \in R, y \in S\}$.

In other words, in an AF $F = (A, R)$, for a subset of arguments $S \subseteq A$, the image of S under the characteristic function $\mathcal{F}_F(S)$ yields exactly the set of those arguments for which, if attacked, there exists a counterattack from S. The range of S is simply the union of S and all arguments attacked by S.

Example 2. *Consider the AF F in Figure 1 on page 4. Letting $S = \{a\}$, the image of S under the characteristic function is $\mathcal{F}_F(\{a\}) = \{a, c\}$. In addition, the range of S is $S_R^+ = \{a, b\}$.*

Definition 3. 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 denote the collection of conflict-free sets of F by $cf(F)$. For a conflict-free set $S \in cf(F)$, it holds that

- $S \in adm(F)$ iff $S \subseteq \mathcal{F}_F(S)$;
- $S \in com(F)$ iff $S = \mathcal{F}_F(S)$;
- $S \in grd(F)$ iff S is the least fixed-point of $\mathcal{F}_F$;
- $S \in prf(F)$ iff $S \in adm(F)$ and there is no $T \in adm(F)$ with $S \subset T$;
- $S \in stb(F)$ iff $S_R^+ = A$;
- $S \in sem(F)$ iff $S \in adm(F)$ and there is no $T \in adm(F)$ with $S_R^+ \subset T_R^+$; and
- $S \in stg(F)$ iff there is no $T \in cf(F)$ with $S_R^+ \subset T_R^+$.

Next we provide some intuition regarding the semantics considered, following [12]. Since extensions are subsets of arguments that can be jointly accepted, or describe a single coherent point of view on an issue, it is natural to consider only such subsets which are conflict-free. The admissibility criterion extends this notion by requiring that the extension must defend itself against all attacks—that is, the inclusion of an argument in the extension can be justified since all attacks on the argument are defended against from within. Complete extensions again extend the notion of admissibility—if an argument is defended by the extension, it must be included, and vice versa. The grounded extension is guaranteed to be unique, and as the subset-minimal complete extension, includes as few arguments as possible and describes the most skeptical point of view one can take under the complete semantics. Preferred extensions, on the other hand, are subset-maximal admissible ones, that is, every argument that can be included in the extension (without violating admissibility) must be included. Stable semantics provide a black-and-white view—if an argument is not in the extension, it must be attacked by some other argument in the extension. This is captured theoretically by the notion of range. Since stable extensions

Table 1: Extensions of the two AFs from Figure 1.

σ	$\sigma(F)$	$\sigma(F')$
adm	$\{\emptyset, \{a\}, \{a, c\}, \{a, d\}, \{d\}\}$	$\{\emptyset, \{a\}\}$
com	$\{\{a\}, \{a, c\}, \{a, d\}\}$	$\{\{a\}\}$
grd	$\{\{a\}\}$	$\{\{a\}\}$
prf	$\{\{a, c\}, \{a, d\}\}$	$\{\{a\}\}$
stb	$\{\{a, c\}, \{a, d\}\}$	$\emptyset$
sem	$\{\{a, c\}, \{a, d\}\}$	$\{\{a\}\}$
stg	$\{\{a, c\}, \{a, d\}\}$	$\{\{a, d\}, \{b, d\}\}$

do not exist in every argumentation framework (i.e., there exists an AF F such that $stb(F) = \emptyset$), and are the only main semantics that have this property, one is interested in an approximation of the stable extensions. Such an approximation is provided via the semi-stable (and stage) semantics, which maximize the range without breaking admissibility (conflict-freeness). Indeed, semi-stable and stage extensions exist in all (finite) AFs, and if a stable extension exists, stable, semi-stable and stage extensions coincide.

Example 3. *Consider again the AFs F and F' in Figure 1 on page 4. Table 1 captures all extensions under the semantics defined previously. Notice that for the AF F, the preferred, stable, semi-stable and stage extensions coincide. In addition, the AF F' has no stable extensions.*

The following states a well-known theorem on relationships between semantics. We provide a proof for completeness.

Theorem 4. *For an AF F, the following inclusions hold.*

$$cf(F) \supseteq adm(F) \supseteq com(F) \supseteq prf(F) \supseteq sem(F) \supseteq stb(F).$$

Proof. We show that the inclusions hold from left to right. The first inclusion is trivial by definition.

For the second, let $S \in com(F)$. Now $S = \mathcal{F}_F(S)$, so $S \subseteq \mathcal{F}_F(S)$ and therefore $S \in adm(F)$.

Now let $S \in prf(F)$, that is, $S \in adm(F)$ and there is no $T \in adm(F)$ with $S \subset T$. To show that $S \in com(F)$, suppose on the contrary that this does not hold. Then we have $S \neq \mathcal{F}_F(S)$, but since $S \in adm(F)$, $S \subset \mathcal{F}_F(S)$. This yields a contradiction, since $\mathcal{F}_F(S) \in adm(F)$ because $S \in adm(F)$, so the third inclusion is proven.

Let $S \in sem(F)$. Now $S \in adm(F)$ and there is no $T \in adm(F)$ such that $S_R^+ \subset T_R^+$. Suppose again on the contrary that $S \notin prf(F)$. Therefore there exists $T \in adm(F)$ such that $S \subset T$. We yield a contradiction by showing that in this case $S_R^+ \subset T_R^+$. Suppose that this does not hold, which implies $S_R^+ = T_R^+$. Since now $T \setminus S \neq \emptyset$, take $x \in T \setminus S$, so $x \in T_R^+ = S_R^+$.

But $x \notin S$, and therefore $x \in S_R^+ \setminus S$, so there must exist $y \in S$ such that $(y, x) \in R$. This yields a contradiction, since $x \in T$ and $y \in S \subset T$, which cannot hold, since $T \in adm(F) \subseteq cf(F)$. Therefore $S_R^+ \subset T_R^+$, which is the desired contradiction for the proof of the fourth inclusion.

For the last inclusion, note that if $S \in stb(F)$, then $S_R^+ = A$, so $S \in adm(F)$ and there cannot be a $T \in adm(F)$ such that $S_R^+ \subset T_R^+$. □

The inclusions of Theorem 4 can also be strict. As shown in Table 1, the AFs in Figure 1 on page 4 provide an example of such strict inclusions.

Semantics also provide a way to define the acceptance of a given argument. Two ways are considered—an argument is credulously accepted if it is in some extension, and skeptically accepted if it is in all extensions. These acceptance criteria correspond to an argument being accepted in some possible worlds, or all of them. Therefore credulous acceptance provides a considerably more relaxed notion of acceptance than skeptical acceptance.

Definition 5. Let $F = (A, R)$ be an AF, and σ a semantics. Under the semantics σ, an argument $a \in A$ is

- *credulously accepted* iff $a \in \bigcup \sigma(F)$, and
- *skeptically accepted* iff $a \in \bigcap \sigma(F)$.

Example 4. *Consider again the AF F in Figure 1 on page 4. Since*

$$com(F) = \{\{a\}, \{a, c\}, \{a, d\}\},$$

we know that argument a is skeptically accepted under the complete semantics (and the only such argument), since the intersection of the complete extensions is the singleton of a, i.e., a is contained in every extension. In addition, arguments a, c, and d are credulously accepted, but argument b is not, since the union of the extensions is $\{a, c, d\}$.

Finally, we note that in addition to the extension-based view on semantics considered in this work, some work instead focuses on the labeling-based view in abstract argumentation [12]. The main difference is that instead of the binary set-inclusion-based classification (an argument is either in an extension or out), three labels—*in*, *out*, and *undecided*—are used. All semantics considered in this work can also be defined using the labeling idea in an equivalent way, where all sets of arguments that can be labeled *in* form the set of extensions under a semantics. Furthermore, the central contributions of this thesis could be similarly rephrased in terms of labelings.

3 Enforcement in Abstract Argumentation

In general, enforcement means adjusting a given AF in light of new information in a way that certain properties hold for the modified AF, for

instance, a set being an extension or arguments being accepted. Enforcing can be viewed as an optimization problem, where the task is to minimize the amount of change to the framework. In this section, we describe two variants of the problem—extension and status enforcement—regarded as discrete optimization problems.

3.1 Extension Enforcement

In the extension enforcement problem, the aim is to modify the given argumentation framework or change the semantics in light of new, more reliable information in such a way that a given set becomes an extension [17]. The extension enforcement problem in abstract argumentation was proposed in [17], where the (im)possibilities of enforcing a set of arguments was studied under different conditions, namely, addition of new arguments and attacks from and to them (normal, strong and weak expansions). In [16], the problem was generalized to enforcing a set of arguments under *minimal change*[1], that is, the question is not only whether the set *can* be enforced (a decision problem), but *how many changes* to the original framework are required at minimum in order to achieve the desired enforcement (an optimization problem) under a certain distance measure (a pseudometric in the space of all AFs). The removal of arguments and attacks associated with them was studied in [28], also generalizing the extension enforcement problem to arbitrary goals expressible in propositional logic. All of these approaches have supposed that the initial attacks of the AF are fixed, and instead new arguments can arrive or old ones can leave. However, the dual where arguments are fixed and the attack relation is dynamic also makes sense, proposed in [43] as *argument-fixed extension enforcement*, proving that a solution always exists and providing an algorithmic solution to the problem by encoding it as integer programming. In this work, we extend the approach by providing a nearly full complexity analysis and an implementation for multiple AF semantics utilizing maximum satisfiability as the optimization engine.

Extension enforcement as defined in [16, 17] is based on *expansions* of AFs, which are additions of new arguments and attacks to the original AF. This approach makes sense in a dialogue-based setting, where the question is how to make (enforce) a certain point of view (extension) acceptable by drawing new information into the discussion. Three expansions are considered:

- normal expansion, where new arguments are added and such new attacks that either the attacker or the attacked is a new argument;
- strong expansion, where new arguments are added and such new attacks that the new arguments are not attacked by the original ones; and

[1]Note that in [16], the word 'minimal' is used as a synonym for 'minimum'.

- weak expansion, where new arguments are added and such new attacks that the original arguments are not attacked by the new ones.

However, as noted in [17, 43], enforcement under these expansions is impossible in the general case. Motivated by this, the argument-fixed extension enforcement problem was proposed in [43], where instead of adding new arguments and attacks, the arguments are fixed, and the original attack structure may be subject to any change, viewed as an optimization problem by minimizing the number of changes necessary. This approach is viable e.g. in a setting where an agent has observed that the given set is actually an extension, and the agent's AF needs to be adjusted in light of this information to be consistent. The authors also define in [43] the enforcement problem with an additional parameter, (non-)strictness. It has been shown that argument-fixed extension enforcement is always possible [43].

Formally, the task of extension enforcement is to modify the attack structure R of an AF $F = (A, R)$ in a way that a given set T becomes (a subset of) an extension under a given semantics σ. *Strict* enforcement requires that the given set of arguments has to be a σ-extension, while in *non-strict* enforcement it is required to be a subset of a σ-extension. We denote strict by s and non-strict by ns.

Denote by

$$enf(F, T, s, \sigma) = \{R' \mid F' = (A, R'),\ T \in \sigma(F')\},$$

the set of attack structures that strictly enforce T under σ for an AF F, and by

$$enf(F, T, ns, \sigma) = \{R' \mid F' = (A, R'),\ \exists T' \in \sigma(F')\ :\ T' \supseteq T\}$$

for non-strict enforcement.

Define the Hamming distance between two attack structures by

$$|R \Delta R'| = |R \setminus R'| + |R' \setminus R|,$$

which is the cardinality of the symmetric difference, or the number of changes (additions or removals of attacks) of an enforcement. Extension enforcement is considered as an optimization problem, where the number of changes is minimized. Formally, the problem can be stated as follows.

Optimal Extension Enforcement ($M \in \{s, ns\}$)
Input: AF $F = (A, R)$, $T \subseteq A$, and semantics σ.
Task: Find an AF $F^* = (A, R^*)$ with

$$R^* \in \operatorname*{arg\,min}_{R' \in enf(F,T,M,\sigma)} |R \Delta R'|.$$

Example 5. *Let F' be the AF in Figure 1 on page 4. Consider enforcing $T = \{a\}$ non-strictly under the stable semantics. First we note that the AF F' does not have any stable extensions, and therefore we need to modify the attack structure in some way, that is, at least one attack has to be added or removed. One way of doing this optimally is adding the attack (a, c) to the AF, so that $\{a, d\}$ is a stable extension in the solution AF. Another way would be to remove the existing attack (c, c) from the attack structure, in which case $\{a, c\}$ becomes a stable extension. The resulting AFs F^* and F^{**} are shown in Figure 2.*

Figure 2: The argumentation frameworks F^* and F^{**} of Example 5.

For non-strict extension enforcement, the problems of enforcing an extension under the admissible, complete and preferred semantics coincide.

Theorem 6. *Let $F = (A, R)$ be an AF, $T \subseteq A$. Now*

$$enf(F, T, ns, adm) = enf(F, T, ns, com) = enf(F, T, ns, prf).$$

Proof. First, suppose $R' \in enf(F, T, ns, prf)$, and let $F' = (A, R')$. Since now there exists a superset $T' \supseteq T$ such that $T' \in prf(F')$, and in addition $prf(F') \subseteq com(F')$, it also holds that $T' \in com(F')$. Therefore $R' \in enf(F, T, ns, com)$, which shows that $enf(F, T, ns, prf) \subseteq enf(F, T, ns, com)$ holds. The inclusion $enf(F, T, ns, com) \subseteq enf(F, T, ns, adm)$ is handled similarly by noting that complete extensions of the new AF are a subset of the admissible extensions.

Now, suppose $R' \in enf(F, T, ns, adm)$, and let again $F' = (A, R')$. From the definition we know that there exists $T' \supseteq T$ such that $T' \in adm(F')$. Let now $T'' \in adm(F')$ be a maximal element with respect to set inclusion such that $T' \subseteq T''$. It follows that $T'' \in prf(F')$, and since $T \subseteq T''$, we know that $R' \in enf(F, T, ns, prf)$. Therefore the final inclusion $enf(F, T, ns, adm) \subseteq enf(F, T, ns, prf)$ also holds, establishing equality between the three sets. □

The fact that the optimal solutions are also the same for non-strict extension enforcement under the admissible, complete, and preferred semantics follows trivially from the previous theorem.

3.2 Status Enforcement

Status enforcement is a variant of extension enforcement, where instead of enforcing a single point of view, the goal is to enforce given arguments' statuses either *positively* —as to accept them—or *negatively*—to reject (i.e.,

not accept) them, either following credulous or skeptical acceptance [75]. In credulous status enforcement, the solution AF will have positively enforced arguments justified by some point of view, and negatively enforced arguments not justified by any means. In the dual, skeptical status enforcement, positively enforced arguments will be justified without any conflicting viewpoints, and negatively enforced arguments have some conflicting point of view.

Formally, the goal is to modify the attack structure R of a given AF $F = (A, R)$ such that for given disjoint sets of arguments P and N, $P \cap N = \emptyset$, the following holds: all arguments in P are credulously/skeptically accepted and all arguments in N are not credulously/skeptically accepted. It is said that the set P of arguments is then *positively* enforced, and the set N *negatively* enforced.

For the credulous status enforcement problem, denote by $\mathit{cred}(F, P, N, \sigma)$ the set

$$\{R' \mid F' = (A, R'), P \subseteq \bigcup \sigma(F'), N \cap \bigcup \sigma(F') = \emptyset\},$$

that is, the set of attack structures where all arguments in P are credulously accepted, i.e., each argument in P is contained in *some* σ-extension, and each argument in N is not credulously accepted (not contained in *any* σ-extension).

For skeptical status enforcement, denote by $\mathit{skept}(F, P, N, \sigma)$ the set

$$\{R' \mid F' = (A, R'), P \subseteq \bigcap \sigma(F'), N \cap \bigcap \sigma(F') = \emptyset\},$$

that is, the set of attack structures where all arguments in P are skeptically accepted, i.e. each argument in P is contained in *all* σ-extensions, and each argument in N is not skeptically accepted (not contained in *some* σ-extension).

Note that by definition $A \subseteq \bigcap \sigma(F')$ if $\sigma(F') = \emptyset$. From the semantics considered in this work, only the stable semantics may admit no extensions for a given AF. This implies that if $N = \emptyset$, under the stable semantics any set of arguments $P \subseteq A$ can be skeptically enforced by an AF that has no stable extensions. Due to this, for skeptical enforcement under the stable semantics we additionally require that the solution AF has at least one stable extension, i.e., $\sigma(F') \neq \emptyset$, where F' is the solution AF.

Here status enforcement is also considered as an optimization problem, where the number of changes to the attack structure is minimized. Formally, we distinguish between credulous and skeptical status enforcement as follows.

Optimal Credulous Status Enforcement
Input: AF $F = (A, R)$, $P, N \subseteq A$, and semantics σ.
Task: Find an AF $F^* = (A, R^*)$ with

$$R^* \in \underset{R' \in \mathit{cred}(F,P,N,\sigma)}{\arg\min} |R \Delta R'|.$$

Optimal Skeptical Status Enforcement
Input: AF $F = (A, R)$, $P, N \subseteq A$, and semantics σ.
Task: Find an AF $F^* = (A, R^*)$ with

$$R^* \in \underset{R' \in skept(F,P,N,\sigma)}{\arg\min} |R \Delta R'|.$$

Example 6. *Let F be the AF in Figure 1 on page 4. Consider enforcing $P = \{c\}$ skeptically under the stable semantics. By adding the attack (a, d), we make the current stable extension $\{a, d\}$ conflicting, in which case $\{a, c\}$ is the only stable extension in the new AF, so c is skeptically accepted. The resulting AF F^* is shown in Figure 3.*

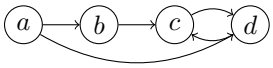


Figure 3: The argumentation framework F^* of Example 6.

The problems of credulous status enforcement under the admissible, complete, and preferred semantics coincide. Therefore the optimal solutions to the problems are also the same.

Theorem 7. *Let $F = (A, R)$ be an AF and $P, N \subseteq A$ two disjoint subsets of arguments. Now*

$$cred(F, P, N, adm) = cred(F, P, N, com) = cred(F, P, N, prf).$$

Proof. From the definitions of the three sets, it suffices to show that

$$\bigcup adm(F') = \bigcup com(F') = \bigcup prf(F')$$

for any AF F'. Since $prf(F') \subseteq com(F') \subseteq adm(F')$, the inclusions

$$\bigcup adm(F') \supseteq \bigcup com(F') \supseteq \bigcup prf(F')$$

follow straightforwardly. Let now $a \in \bigcup adm(F')$, that is, there exists $E \in adm(F')$ such that $a \in E$. Let $E' \in adm(F')$ be a maximal element with respect to the set inclusion. It follows that $E' \in prf(F')$, and since $a \in E'$, we know that $a \in \bigcup prf(F')$, with shows that the inclusion

$$\bigcup adm(F') \subseteq \bigcup prf(F')$$

also holds, establishing equality between the three sets. □

Table 2: Complexity of extension enforcement.

σ	strict	non-strict
Conflict-free	in P	in P
Admissible	in P	NP-c
Stable	in P	NP-c
Complete	NP-c	NP-c
Grounded	NP-c	NP-c
Preferred	Σ_2^P-c	NP-c
Semi-stable	Σ_2^P-c	Σ_2^P-c
Stage	coNP-hard and in Σ_2^P	Σ_2^P-c

Table 3: Complexity of status enforcement.

	$N = \emptyset$		general case	
σ	credulous	skeptical	credulous	skeptical
Conflict-free	in P	trivial	in P	trivial
Admissible	NP-c	trivial	Σ_2^P-c	trivial
Stable	NP-c	Σ_2^P-c	Σ_2^P-c	Σ_2^P-c
Complete	NP-c	NP-c	Σ_2^P-c	NP-c
Grounded	NP-c	NP-c	NP-c	NP-c
Preferred	NP-c	in Σ_3^P	Σ_2^P-c	in Σ_3^P

4 Computational Complexity of Enforcement

We recall the definitions of computational complexity classes [79] to the extent relevant to this work. A decision problem is a computational problem where the output is either YES or NO. The fundamental class P consists of all decision problems that can be decided by a deterministic Turing machine in polynomial time, that is, for which there exists a deterministic polynomial-time algorithm. The complexity class NP, on the other hand, contains all decision problems that can be decided by a nondeterministic Turing machine in polynomial time, that is, verifying whether a (nondeterministic) guess is indeed a solution can be computed in polynomial time. The complementary class of NP is denoted by coNP, and contains those decision problems for which the YES instances are exactly the NO instances of a corresponding problem in NP.

For a complexity class $\mathcal{C}$, a $\mathcal{C}$-oracle is a procedure that solves each problem in $\mathcal{C}$ using a constant amount of time. Using this definition, we can define the complexity class $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$ as all decision problems solvable in nondeterministic polynomial time using an NP-oracle, and the complexity class Σ_3^P as all decision problems solvable in nondeterministic polynomial time using a Σ_2^P-oracle. The complexity classes Σ_2^P and Σ_3^P are second-level and third-level classes in the polynomial hierarchy [86], respectively. A problem P is $\mathcal{C}$-hard if every problem P' in $\mathcal{C}$ is polynomial-time reducible to P, i.e., there is a polynomial-time algorithm that transforms an instance

I' of P' to an instance I of P such that I' is a YES instance of P' if and only if I is a YES instance of P. A problem P is complete for a complexity class $\mathcal{C}$, denoted by $\mathcal{C}$-c, if P is in $\mathcal{C}$ and P is $\mathcal{C}$-hard.

The overview of the computational complexity of enforcement is given in Table 2 and Table 3, for extension and status enforcement, respectively. The decision variant of each optimization problem is considered, i.e., in addition we are given an integer $k \geq 0$ and asked whether the optimization problem has a solution with cost (the number of changes to the attack structure required) less than or equal to k. In this section we overview some of the main complexity proofs. For the complexity of extension enforcement, the rest of the proofs can be found in [90], and for the complexity of status enforcement, in [75]. We start off by showing that strict extension enforcement is solvable in polynomial time under the conflict-free, admissible and stable semantics.

Theorem 8. *Strict extension enforcement for the conflict-free, admissible, and stable semantics is in* P*.*

Proof. Let $F = (A, R)$ be an AF, $T \subseteq A$ the set to be enforced, and $\sigma \in \{cf, adm, stb\}$. Suppose $T \neq \emptyset$, since otherwise the problem is trivial. Let $t \in T$ be an arbitrary argument in the set.

1. $\sigma = cf$: Let $R^* = R \setminus (T \times T)$, that is, R^* is formed by deleting all attacks where both arguments are in T.

2. $\sigma = adm$: Let

$$R^* = (R \setminus (T \times T)) \\ \cup \{(t, a) \mid a \in A \setminus T, \exists (a, b) \in R \colon b \in T, \nexists (c, a) \in R \colon c \in T\},$$

   that is, remove all attacks inside T and add attacks to ensure admissibility by defending from all attacks outside T.

3. $\sigma = stb$: Let

$$R^* = (R \setminus (T \times T)) \\ \cup \{(t, a) \mid a \in A \setminus T, \nexists b \in T \colon (b, a) \in R\},$$

   removing all attacks inside T and extending the range to arguments not attacked by T.

Now for all semantics σ, $F^* = (A, R^*)$ is a solution AF where clearly $T \in \sigma(F^*)$. These solutions are also optimal, since in each case no less modifications can be made to the original attack structure. □

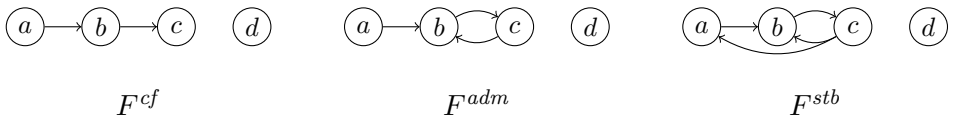
The following example illustrates how to construct the solution in polynomial time.

Example 7. *Consider the AF $F = (A, R)$ in Figure 1 from page 4. Suppose we want to enforce $T = \{c, d\}$ strictly.*

1. *Under $\sigma = cf$, it suffices to remove the attacks (c, d) and (d, c), yielding the AF $F^{cf} = (A, R^{cf})$ illustrated in Figure 4.*

2. *Under $\sigma = adm$, in addition to removing the attacks inside T, we need a counterattack on the argument b, since it attacks T. This is accomplished via adding the attack (c, b), forming the AF F^{adm} illustrated in Figure 4.*

3. *Under $\sigma = stb$, T must be conflict-free and the range of T must be the whole set of arguments A. Therefore we must delete attacks (c, d) and (d, c), and add attacks from T to $A \setminus T = \{a, b\}$, e.g., (c, a) and (c, b). The resulting AF F^{stb} is shown in Figure 4.*



F^{cf} F^{adm} F^{stb}

Figure 4: Optimal solutions to strict extension enforcement in Example 7.

For non-strict extension enforcement, on the other hand, only instances of conflict-free semantics are polynomial time solvable.

Theorem 9. *Non-strict extension enforcement for the conflict-free semantics is in* P.

Proof. Similarly to Theorem 8, let $F = (A, R)$ be an AF and $T \subseteq A$ the set to be non-strictly enforced. Now letting $R^* = R \setminus (T \times T)$ we have an optimal solution AF $F^* = (A, R^*)$, since clearly $T \in cf(F^*)$ and all attacks inside T have to be removed for it to be a conflict-free set. □

The NP-hardness proofs for non-strict extension enforcement are based on a reduction from the credulous acceptance problem under the same semantics. Recall that an argument is credulously accepted under semantics σ if and only if it is contained in some σ-extension of the given AF. The problem was first proven to be NP-complete in the context of graph theory [47], where semikernels correspond to admissible extensions and kernels to stable extensions. We provide a full proof directly on argumentation frameworks, utilizing a part of the same structure as in the original proof. The structure also provides insight to the kinds of reductions typically used in analyzing the complexity of computational problems in abstract argumentation.

Theorem 10. *Credulous acceptance of an argument under the admissible and stable semantics is* NP*-complete.*

Proof. Let $F = (A, R)$ be an AF, $a \in A$ an argument, and $\sigma \in \{adm, stb\}$. Nondeterministically guess a subset $E \subseteq A$. It is now polynomial-time verifiable whether $E \in \sigma(F)$ and $a \in E$, which implies that $a \in \bigcup \sigma(F)$, that is, a is credulously accepted under σ. Therefore credulous acceptance under these semantics is in NP.

For NP-hardness, we reduce CNF-SAT (see Section 5.1) to the problem of credulous acceptance. Let $\varphi = C_1 \wedge \cdots \wedge C_n$ be a formula in CNF over the Boolean variables X. Denote by $\overline{X}$ the set of negative literals $\{\neg x \mid x \in X\}$, and by C the set of symbols corresponding to clauses $\{c_i \mid i = 1, \ldots, n\}$. Construct an AF $F = (A, R)$ with the arguments

$$A = X \cup \overline{X} \cup C \cup \{a\}$$

and the attack structure

$$\begin{aligned} R = {} & \{(x, \neg x) \mid x \in X\} \cup \{(\neg x, x) \mid x \in X\} \\ & \cup \{(x, c_i) \mid x \in C_i\} \cup \{(c_i, a) \mid i = 1, \ldots, n\}, \end{aligned}$$

i.e., attacks exist between each pair of a positive and negative literal, from each literal to a clause containing it, and from each clause to the additional argument a. An illustration of the AF F is shown in Figure 5. Note that the reduction AF F can be constructed in polynomial time. The claim is that φ is satisfiable if and only if a is credulously accepted under $\sigma \in \{adm, stb\}$.

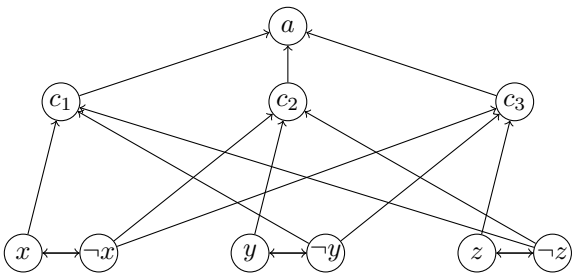


Figure 5: Illustration of the reduction used in the NP-hardness proof, given the CNF formula $\varphi = (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z)$.

For the direction from left to right, suppose φ is satisfiable. Then there exists a truth assignment $\tau \colon X \to \{0, 1\}$ such that for each clause there exists a literal l for which $\tau(l) = 1$. Let now

$$E = \{a\} \cup \{x \in X \mid \tau(x) = 1\} \cup \{\neg x \in \overline{X} \mid \tau(x) = 0\} \subseteq A,$$

i.e., the union of the singleton of a and each literal that evaluates to true under τ. First note that $E \in cf(F)$, since there are no attacks between $\{a\}$ and any of the literals, and since τ is a function, it cannot map a variable x to both 0 and 1, hence no positive and negative literal of the same variable is included in E.

1. $\sigma = adm$: We claim that the set E is an admissible extension. To see this, note that positive literals in E attack negative ones, and negative literals attack positive ones. Each clause argument c_i attacks a, but for each clause, there exists a literal which is evaluated to true, and that literal is included in E. This shows that E defends each of its members, and therefore $E \in adm(F)$.

2. $\sigma = stb$: We claim that the set E is a stable extension. To see this, note that the only arguments in $A \setminus E$ are the literals that evaluate to false under τ, and each clause argument c_i. But for each literal that evaluates to false, there exists an attacker that evaluates to true, and for each clause argument, there exists a literal which is evaluated to true, and these literals are included in E. Therefore the range of E is A, that is, $E \in stb(F)$.

In both cases $E \in \sigma(F)$, which implies that a is credulously accepted under σ, since $a \in E$.

For the converse, suppose a is credulously accepted. Then there exists an extension $E \in \sigma(F)$ such that $a \in E$. Construct the following truth assignment $\tau \colon X \to \{0, 1\}$,

$$\tau(x) = \begin{cases} 1 & \text{if } x \in E, \\ 0 & \text{otherwise.} \end{cases}$$

We claim that τ is a satisfying truth assignment for φ. First note that τ is a well-defined function, since $E \in cf(F)$, so no positive and negative literal for the same variable can occur in E.

1. $\sigma = adm$: Now for each attack $(c_i, a) \in R$ there must be an attack to c_i from E. By construction the only argument attacking c_i is a literal contained in the corresponding clause.

2. $\sigma = stb$: Since E is conflict-free and contains a, no clause argument c_i can be can be contained in E. Since now the range of E is A, for each c_i, there must be an attack on c_i from E. Again, by construction the only argument attacking c_i is a literal contained in the corresponding clause.

Now for each clause there must be a literal included in E, which is then evaluated true under τ. This shows that φ is satisfiable, since each clause of the formula is satisfied.

We have now shown that the credulous acceptance problem under σ is in NP and NP-hard. Thus credulous acceptance under the admissible and stable semantics is NP-complete. □

We move on to NP-completeness proofs for non-strict extension enforcement. Under the admissible, complete, stable, and preferred semantics the problem is NP-complete.

Theorem 11. *Non-strict enforcement for the admissible, complete, stable, and preferred semantics is* NP*-complete.*

Proof. Recall that non-strict extension enforcement under the admissible, complete and preferred semantics coincides (Theorem 6). Therefore it is enough to consider one of these, for instance the admissible semantics. Let $F = (A, R)$ be an AF, $T \subseteq A$ the set to be enforced, $\sigma \in \{adm, stb\}$, and $k \geq 0$ an integer. Let R' be a nondeterministic guess for the attack structure of the proposed solution AF $F' = (A, R')$, and T' the guess for the superset of T that is a σ-extension. Since the necessary checks $|R \Delta R'| \leq k$, $T \subseteq T'$ and $T' \in \sigma(F')$ can be computed in polynomial time, it is clear that non-strict extension enforcement under these semantics is in NP.

NP-hardness follows from a reduction from the NP-complete credulous acceptance problem under semantics σ. This problem is reduced to non-strict extension enforcement as follows. Let $F = (A, R)$ be an AF, and $a \in A$ an argument whose credulous acceptance is to be checked. Define a non-strict extension enforcement instance with the same AF F and the singleton set $T = \{a\}$ to be enforced. The reduction can clearly be computed in polynomial time. Suppose T can be enforced with 0 changes to the attack structure. This implies that a is credulously accepted, since there exists $T' \supseteq T$ such that $T' \in \sigma(F)$. On the other hand, if a is credulously accepted, there exists some $T' \in \sigma(F)$ such that $T \subseteq T'$, that is, T is non-strictly enforced with 0 changes. □

The computational complexity of the *union* of non-strict and strict extension enforcement under the stable semantics was established to be NP-hard in [43]. Therefore Theorems 8 and 11 provide a more fine-grained analysis of the computational complexity, showing that under the stable semantics, strict extension enforcement is in fact in P, and non-strict extension enforcement is NP-complete.

For the Σ_2^P-completeness proof for non-strict extension enforcement under the semi-stable and stage semantics, we show Σ_2^P-hardness via a reduction from the Σ_2^P-complete credulous acceptance problem under the same semantics [51, 55]. Membership is shown also via a similar guess-and-check, but this time verifying whether a subset of arguments is a semi-stable or stage extension is in fact in coNP [31, 47].

Theorem 12. *Non-strict enforcement for the semi-stable and stage semantics is Σ_2^P-complete.*

Proof. Let $F = (A, R)$ be an AF, $T \subseteq A$ the set to be enforced, $\sigma \in \{sem, stg\}$, and $k \geq 0$ an integer. Let R' be a nondeterministic guess for the attack structure of the proposed solution AF $F' = (A, R')$, and T' the guess for the superset of T that is a σ-extension. Since the checks $|R \Delta R'| \leq k$, $T \subseteq T'$ can be computed in polynomial time, and $T' \in \sigma(F')$ via a single call to

an NP-oracle, it is clear that non-strict extension enforcement under these semantics is in Σ_2^P.

Σ_2^P-hardness follows from a reduction from the Σ_2^P-complete credulous acceptance problem under semantics σ. Let $F = (A, R)$ be an AF, and $a \in A$ an argument whose credulous acceptance is to be checked. Define a non-strict extension enforcement instance with the same AF F and the set $T = \{a\}$ to be enforced. The reduction can clearly be computed in polynomial time. Now T can be enforced with 0 changes to the attack structure if and only if a is credulously accepted. □

Moving on to results for status enforcement, skeptical status enforcement under the conflict-free and admissible semantics is trivial, since the empty set is always a conflict-free and an admissible set, and therefore the intersection of all conflict-free and admissible sets is always the empty set, i.e., $\bigcap \sigma(F) = \emptyset$ since $\emptyset \in \sigma(F)$. In addition, under the conflict-free semantics credulous status enforcement is polynomial-time solvable, which is stated in the following theorem.

Theorem 13. *Credulous status enforcement under the conflict-free semantics is in* P.

Proof. Let $F = (A, R)$ be an AF, and $P, N \subseteq A$, $P \cap N = \emptyset$ two disjoint subsets of arguments. An optimal solution is given by $F^* = (A, R^*)$, where

$$R^* = (R \setminus \{(a, a) \mid a \in P\}) \cup \{(a, a) \mid a \in N\},$$

i.e., removing all self-attacks on arguments in P and adding self-attacks on arguments in N. Since for all $p \in P$ the singleton $\{p\}$ must be a conflict-free set, and for all $n \in N$ the singleton $\{n\}$ must not be conflict-free, the solution is optimal, since not removing or adding any self-attack would not satisfy these properties. □

On the other hand, credulous status enforcement for the admissible, complete, stable and preferred semantics is NP-complete when $N = \emptyset$.

Theorem 14. *Even when $N = \emptyset$, credulous status enforcement for the admissible, complete, stable, and preferred semantics is* NP*-complete.*

Proof. Recall that credulous status enforcement under the admissible, complete, and preferred semantics coincides. We consider the admissible semantics. Let $F = (A, R)$ be an AF, and $P \subseteq A$ the set of arguments to be positively enforced under $\sigma \in \{adm, stb\}$, and $k \geq 0$ an integer. Nondeterministically guess the attack structure R' of the solution AF $F' = (A, R')$, and for each $p \in P$, a subset of arguments E_p such that $p \in E_p$. Now the cardinality check $|R \Delta R'| \leq k$ and the extension check $E_p \in \sigma(F')$ can be computed in polynomial time for the semantics σ considered. Therefore credulous status enforcement under σ is in NP.

NP-hardness follows by a reduction from the NP-complete credulous acceptance problem. We reduce it to credulous status enforcement as follows. Let $F = (A, R)$ be an AF, and $a \in A$ an argument whose credulous acceptance is to be checked. Construct an instance of credulous status enforcement with the AF F, and the singleton $P = \{a\}$ the set to be positively enforced. The reduction can be clearly computed in polynomial time. It is now clear that a is credulously accepted if and only if P is credulously enforced with $k = 0$ changes to the attack structure. □

5 Maximum Satisfiability

Recall that our goal is to develop algorithms for the extension and status enforcement problems, the decision variants of which are NP-hard. In abstract argumentation, many static problems have been solved using declarative approaches, where the instance of the problem is encoded into an instance of a constraint language, such as answer set programming (ASP) [54, 59, 61] or Boolean satisfiability (SAT) [32, 33, 56]. In this case, the problems can be solved by calling the corresponding solver, and decoded back into the solution of the original problem. Since these approaches have been the most efficient ones in practice [87], we intend to provide algorithmic solutions to the dynamic problems of enforcement in a similar manner—using maximum satisfiability (MaxSAT) as the optimization engine for tackling the NP-hard optimization problems.

5.1 SAT

We recall basic concepts related to Boolean satisfiability (SAT) [26] before moving on to maximum satisfiability. The satisfiability problem was the first problem shown to be NP-complete [39]. An instance of the satisfiability problem consists of a propositional formula over a finite set of Boolean variables. The question is whether there exists a truth assignment that satisfies the formula, i.e., the formula is true under the assignment. If such a satisfying truth assignment exists, we say that the formula is satisfiable.

It is a well-known fact that every propositional formula can be converted to conjunctive normal form (CNF) [88, 91], and CNF provides a standard way of representing SAT instances. The conversion of every propositional formula to CNF can be done using a distinct new variable for each subformula, yielding a formula in CNF that is linear in the size of the original formula, with a linear number of new variables introduced. This procedure is called the Tseitin encoding [88, 91], and is a standard approach in the field of satisfiability solving.

In CNF, a propositional formula is given as a conjunction of clauses, or disjunctions of literals. Literals are Boolean variables or their negations.

Definition 15. A literal l is a Boolean variable x or its negation $\neg x$, which are called positive and negative literals, respectively. A clause C is a disjunction of literals $\bigvee_{i=1}^{m} l_i$. A CNF-formula φ is a conjunction of clauses $\bigwedge_{i=1}^{n} C_i$.

The semantics of CNF formulas is defined as follows.

Definition 16. Given a SAT instance in CNF φ, a truth assignment τ is a function from the variables X of φ to $\{0, 1\}$. A positive literal x is true if $\tau(x) = 1$, and a negative literal $\neg x$ is true if $\tau(x) = 0$. If a literal l is true, then we denote $\tau(l) = 1$. A clause $C = \bigvee_{i=1}^{m} l_i$ is satisfiable, if for some $i = 1, \ldots, m$ we have $\tau(l_i) = 1$. In this case we denote $\tau(C) = 1$. A formula $\varphi = \bigwedge_{i=1}^{n} C_i$ is satisfied by the truth assignment τ, if for every $i = 1, \ldots, n$ it holds that $\tau(C_i) = 1$. In this case we call τ a satisfying truth assignment, and say that the formula is satisfiable (SAT). Otherwise it is unsatisfiable (UNSAT).

We illustrate these concepts with the following example.

Example 8. *Consider the following instance of satisfiability.*

$$\varphi = \neg x_1 \wedge (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$$

The variables of this instance are x_1, x_2, x_3, and the clauses are $\neg x_1$, $x_1 \vee x_2$, $\neg x_2 \vee x_3$ and $x_1 \vee x_2 \vee \neg x_3$. To determine whether the formula is satisfiable, first observe that $\neg x_1$ has to be true under such an assignment τ, and therefore $\tau(x_1) = 0$. From this we can derive that in the clause $x_1 \vee x_2$ the literal x_1 is false, and in order to satisfy the clause, we must set $\tau(x_2) = 1$, in which case the last clause is also satisfied. Using the same inference, from the clause $\neg x_2 \vee x_3$ we derive $\tau(x_3) = 1$, and we have found a satisfying truth assignment. Therefore the formula is satisfiable. If, however, the last clause would be of the form $x_1 \vee \neg x_2 \vee \neg x_3$, i.e., the literal x_2 is flipped from positive to negative, we would notice that the formula is unsatisfiable.

A routine for solving instances of the satisfiability problem is called a SAT solver, which have shown to be a success story in many aspects. Starting from a maximum of a few hundred variables and clauses [84], in a couple of decades SAT solvers have advanced to the point where instances with up to millions of variables and clauses can be solved [68]. This is mostly due to efficient procedures developed, such as conflict-driven clause learning [70], which is the base algorithm of the most well-performing solvers [64, 68]. SAT solvers have been used in many practical applications, such as bounded model checking [25], formal verification of hardware [27] and software [66], and automated planning [69, 81].

5.2 MaxSAT

Since not all propositional formulas are satisfiable, it is natural to consider the problem where the task is to satisfy as many clauses as possible. This problem is called maximum satisfiability (MaxSAT), and it is the optimization variant of the satisfiability problem. MaxSAT can be generalized by considering that a subset of the clauses called hard clauses must be satisfied, the remaining part forming the soft clauses, where the task is to satisfy as many as possible. This variant is called partial MaxSAT. In addition, each soft clause can have a non-negative weight associated to it, in which case the goal is to maximize the sum of the satisfied soft clauses. Many real-world optimization problems can be expressed as instances of MaxSAT, including probabilistic inference [80], hardware design debugging [35, 36], reasoning over biological networks [62], correlation clustering [21], learning Bayesian networks [22], static reasoning in AFs [60], and AF synthesis [76].

In maximum satisfiability, the task is to find a truth assignment to a given propositional formula φ in CNF such that the maximum number of clauses is satisfied. We define the cost of a truth assignment τ by

$$\textsc{cost}(\varphi, \tau) = \sum_{C \in \varphi} (1 - \tau(C)),$$

i.e., the number of unsatisfied clauses. A truth assignment τ is optimal if

$$\textsc{cost}(\varphi, \tau) \leq \textsc{cost}(\varphi, \tau')$$

for all possible truth assignments τ'. An optimal truth assignment is a solution to the maximum satisfiability problem.

In partial maximum satisfiability, the instance is a pair $\varphi = (\varphi_h, \varphi_s)$, where φ_h and φ_s are CNF-formulas consisting of hard clauses and soft clauses, respectively. The task is now to find a truth assignment such that all hard clauses are satisfied and the maximum number of soft clauses are satisfied. In this case, we define the cost of a truth assignment τ by

$$\textsc{cost}(\varphi, \tau) = \sum_{C \in \varphi_s} (1 - \tau(C)),$$

i.e., the number of unsatisfied soft clauses. Optimality is defined in a similar way. An optimal truth assignment that satisfies all hard clauses is a solution to the partial MaxSAT problem.

In weighted partial maximum satisfiability, the instance is a triple $\varphi = (\varphi_h, \varphi_s, w)$, where $w \colon \varphi_s \to \mathbb{Z}_+$ is a function assigning to each soft clause a non-negative weight. In this case, the cost of a truth assignment τ is defined as

$$\textsc{cost}(\varphi, \tau) = \sum_{C \in \varphi_s} w(C) \cdot (1 - \tau(C)),$$

i.e., the sum of unsatisfied soft clauses. Again, a solution is optimal if no other solution has a lower cost.

Example 9. *Consider an instance of partial MaxSAT $\varphi = (\varphi_h, \varphi_s)$, where the hard clauses are*

$$\varphi_h = (\neg x_1 \vee \neg x_2) \wedge (\neg x_2 \vee \neg x_3),$$

and the soft clauses are

$$\varphi_s = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3).$$

The hard clauses state that either x_1 or x_2 is false, and either x_2 or x_3 is false, and are therefore satisfiable by simply setting $\tau(x_2) = 0$. In this case also all of the soft clauses can be satisfied by $\tau(x_1) = 1 = \tau(x_3)$, which implies that the solution is optimal with $\textsc{cost}(\varphi, \tau) = 0$.

Maximum satisfiability can be used for efficiently encoding various discrete optimization problems. In the following we give a simple example illustrating how problems can be encoded as MaxSAT.

Example 10. *We recall the classical maximum independent set (MIS) problem. Given a directed graph $G = (V, E)$, the task is to find a subset of the nodes $V' \subseteq V$ such that for all $u, v \in V'$, there is no $(u, v) \in E$, and there exists no other subset $V'' \subseteq V$ such that $V' \subset V''$—that is, V' is subset-maximal. The problem can be encoded as partial maximum satisfiability as follows.*

For each node $v \in V$, define Boolean variables x_v with the interpretation that x_v is assigned to true ($\tau(x_v) = 1$) if and only if x_v is in an independent set. The hard clauses encode the fact that for each edge $(u, v) \in E$, it cannot be that both endpoints x_u and x_v are assigned to true. This is expressed compactly as

$$\varphi_h = \bigwedge_{(u,v) \in E} (\neg x_u \vee \neg x_v).$$

The soft clauses, on the other hand, take care of the optimization part. Since we want to find an independent set of maximum size, we want to satisfy as many variables as possible, expressed simply as

$$\varphi_s = \bigwedge_{v \in V} x_v.$$

Now the optimal solution to the partial MaxSAT problem corresponds to a maximum independent set $V' = \{v \in V \mid \tau(x_v) = 1\}$.

Example 11. *Let $G = (V, E)$ be a graph with $V = \{a, b, c, d\}$ and $E = \{(a, b), (b, c), (c, d), (d, b)\}$, illustrated in Figure 6. The corresponding hard clauses for this instance are*

$$\varphi_h = (\neg x_a \vee \neg x_b) \wedge (\neg x_b \vee \neg x_c) \wedge (\neg x_c \vee \neg x_d) \wedge (\neg x_d \vee \neg x_b),$$

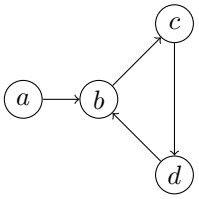


Figure 6: A directed graph $G = (V, E)$.

and the soft clauses

$$\varphi_s = x_a \wedge x_b \wedge x_c \wedge x_d.$$

An optimal solution is given by setting $\tau(x_a) = 1 = \tau(x_d)$ *and* $\tau(x_b) = 0 = \tau(x_c)$*, which corresponds to a maximum independent set* $\{a, d\}$ *in the graph.*

5.3 Algorithms for Solving MaxSAT

Here we give a brief overview of the main classes of algorithms for solving maximum satisfiability. Many solvers that solve the MaxSAT problem exactly have been developed over the past few years [2, 5, 44, 71, 73, 82]. In addition, the biannual MaxSAT evaluation [7] compares the efficiency of these solvers in practice.

One approach for MaxSAT solving is based on the general branch-and-bound scheme [1]. The idea is based on a search tree, the root of which consists of all possible solutions, i.e., the whole search space. The algorithm then keeps assigning variables, branching and dividing the search space at each assignment. The formula may be simplified after the assignment, after which the lower and upper bounds for the cost of any solution of the simplified formula are computed and stored in the corresponding node. If the lower bound is equal to the upper bound, no better solutions may be found in the branch, and the algorithm backtracks in the tree, assigning the next variable. These algorithms have shown to be very effective in random instances [7].

Another viable option is to solve MaxSAT via translating it to integer programming (IP) [6]. To each soft clause C_i, we add a relaxation variable b_i, forming the relaxed soft clause $C_i \vee b_i$. Now, if b_i is true, the relaxed clause is satisfied, and if b_i is false, the original soft clause must be satisfied. Now the optimization part may be expressed as minimizing the function $\sum_i w_i b_i$, where w_i is the weight of the corresponding soft clause. Each hard clause C_i may be encoded as a linear inequality of the form

$$\sum_{j \in C_i^+} x_j + \sum_{j \in C_i^-} (1 - x_j) \geq 1,$$

where C_i^+ contains the indices of positive literals and C_i^- the indices of negative ones. The constants may then be moved to the right hand side, in

which case the right hand side is the sum of negative literals incremented by one, and the left hand side is a sum of positive and negative variables. Finally, each integer variable is restricted to the interval $[0, 1]$.

A large family of modern MaxSAT algorithms are based on using a SAT solver as a subroutine [4]. The algorithm begins by relaxing the soft clauses by adding a new variable b_i to each soft clause $C_i \in \varphi$. In this case k soft clauses may be disabled by setting the corresponding relaxation variables to true. The iterative approach then proceeds by initializing k to zero and checking via a SAT solver whether $\varphi \wedge (\sum_i b_i \leq k)$ is satisfiable, where the cardinality constraint is encoded in CNF (many efficient approaches exist to encode this general constraint—see e.g. [11, 58, 85]). If it is, we know that k of the soft clauses can not be simultaneously satisfied, and we return the truth assignment given by the SAT solver. If not, we increment k by one, and continue by SAT checking, until a solution is found or until we reach the point where k is equal to the number of soft clauses. This linear search can be replaced straightforwardly by a binary search, improving the effectiveness of the algorithm. Another variant first finds a satisfying assignment to the hard clauses, and then checks whether the whole formula is satisfiable with k equal to the number of soft clauses falsified decremented by one. If it is, we have found a better assignment and can iterate by setting k equal to the number of satisfied soft clauses minus one, and if not, the current assignment is optimal.

As witnessed by results of recent MaxSAT evaluations [7], some of the most successful MaxSAT solvers on real world instance classes are the core-guided [2, 5, 71, 73] and SAT/IP hybrid approaches [44, 82]. Both of these rely on the idea of using a SAT solver as a core extractor. An unsatisfiable core of a MaxSAT instance is such a subset of the soft clauses that is unsatisfiable together with the hard clauses. Many modern SAT solvers return a core when reporting unsatisfiability. These can be used to guide the search more effectively than by disabling k arbitrary soft clauses. The core-guided algorithms iterate and relax each soft clause in the current unsatisfiable core, adding a cardinality constraint over the relaxation variables to the instance. This way the cardinality constraints are expressed only over those relaxation variables the corresponding soft clauses of which are found in some core, and not over the whole set of soft clauses. Additionally, disjoint cores can be used to improve the efficiency. The algorithm may then search either via linear or binary search, as in iterative approaches.

Finally, SAT/IP hybrid approaches, also called implicit hitting set algorithms, are used for solving MaxSAT [44, 82]. The idea is based on the insight that for any optimal hitting set H over the set of cores of the instance φ, $\varphi \setminus H$ is satisfiable, and any such satisfying assignment is an optimal solution. Now, using a SAT solver, we iteratively accumulate unsatisfiable cores of the MaxSAT instance, and using the IP solver, we find an optimal implicit hitting set of the unsatisfiable cores, implicit meaning that we have

not necessarily found all cores of the instance. Clauses in the hitting set are removed for the next SAT solver call, and the procedure is repeated until the SAT solver reports satisfiability, in which case the hitting set corresponds to an optimal solution.

6 Enforcement via MaxSAT

Motivated by the NP-hardness of some variants of the extension and status enforcement problems (recall Tables 2 and 3), we use a declarative approach, harnessing the power of maximum satisfiability solvers, to provide algorithmic solutions to the problem variants. In this section, we start by providing (partial) MaxSAT encodings that can be used to directly solve each NP-complete variant of the enforcement problem. These problems are strict extension enforcement under the complete semantics, non-strict extension enforcement under the admissible, complete, preferred and stable semantics, and credulous status enforcement with $N = \emptyset$ under the admissible and stable semantics. Here generally the hard clauses encode the structure of the problem, and soft clauses contain the optimality criterion. We return to the computationally harder problem variants in the next section.

6.1 Soft Clauses for Optimization

In MaxSAT, soft clauses are used to encode the objective function, i.e., the function over which we are optimizing. Let $F = (A, R)$ be an AF of an enforcement instance. For each $a, b \in A$, define Boolean variables $r_{a,b}$ with the interpretation $\tau(r_{a,b}) = 1$ iff the attack (a, b) is included in the attack structure R' of the solution AF $F' = (A, R')$. In both extension and status enforcement, the objective function to be minimized is the number of changes to the original attack structure. This can be expressed by the soft clauses

$$\varphi_s(F) = \bigwedge_{a,b \in A} r'_{a,b},$$

where

$$r'_{a,b} = \begin{cases} r_{a,b} & \text{if } (a, b) \in R, \\ \neg r_{a,b} & \text{if } (a, b) \notin R. \end{cases}$$

The above soft clauses are satisfiable if and only if there are no changes to the attack structure. By maximizing the number of clauses satisfied, we are minimizing the Hamming distance between the original and solution AFs.

6.2 Hard Clauses for Extension Enforcement

In addition to the objective function, we need to encode the underlying properties of the problem into the MaxSAT instance. This is accomplished

via hard clauses. Let $F = (A, R)$ be an AF and $T \subseteq A$ the set to be enforced under semantics σ. In addition to the attack variables, for each $a \in A$, define Boolean variables x_a with the intended meaning of $\tau(x_a) = 1$ iff a is included in the extension of the new AF F'.

For non-strict extension enforcement, define

$$\textsc{ext}(ns, F, T) = \bigwedge_{a \in T} x_a,$$

which states that T is a subset of a σ-extension.

For conflict-free sets, if an attack is present between two arguments, they cannot occur in the same extension. This is stated as

$$\textsc{ext}(ns, F, T, cf) = \textsc{ext}(ns, F, T) \wedge \bigwedge_{a,b \in A} (r_{a,b} \rightarrow (\neg x_a \vee \neg x_b)) .$$

For admissible semantics, if there is an attack on some argument in an extension, there must be another argument in the same extension that attacks the attacker, i.e., the attacked argument is defended by the extension, encoded as

$$\textsc{ext}(ns, F, T, adm) = \textsc{ext}(ns, F, T, cf) \wedge \bigwedge_{a,b \in A} \left((x_a \wedge r_{b,a}) \rightarrow \bigvee_{c \in A} (x_c \wedge r_{c,b}) \right) .$$

For stable semantics, the range of the extension is the whole set of arguments. Therefore, if an argument is not included in the extension, there must be an attack on it from some argument in the extension. This fact is expressed by

$$\textsc{ext}(ns, F, T, stb) = \textsc{ext}(ns, F, T, cf) \wedge \bigwedge_{a \in A} \left(\neg x_a \rightarrow \bigvee_{b \in A} (x_b \wedge r_{b,a}) \right) .$$

We move on to strict enforcement. Since here the values of the x_a variables are fixed, there is no need to use them in the encoding at all. Now conflict-free sets can be encoded as

$$\textsc{ext}(s, F, T, cf) = \bigwedge_{a,b \in T} \neg r_{a,b},$$

i.e., all attacks inside the set T are simply removed.

Admissible semantics are encoded by

$$\textsc{ext}(s, F, T, adm) = \textsc{ext}(s, F, T, cf) \wedge \bigwedge_{a \in T} \bigwedge_{b \in A \setminus T} \left(r_{b,a} \rightarrow \bigvee_{c \in T} r_{c,b} \right) .$$

For complete semantics, in addition all defended arguments must be included in the extension. Therefore, there must be an attack on all arguments outside T that is not defended against by T, expressed by the formula

$$\textsc{ext}(s, F, T, com) = \textsc{ext}(s, F, T, adm) \wedge \bigwedge_{a \in A \setminus T} \bigvee_{b \in A} \left(r_{b,a} \wedge \bigwedge_{c \in T} \neg r_{c,b} \right) .$$

6.3 Hard Clauses for Status Enforcement

We continue by providing hard clauses for status enforcement. Let $F = (A, R)$ be an AF and $P, N \subseteq A$ disjoint sets of arguments whose status is to be enforced under a semantics σ. We use the same attack variables $r_{a,b}$ for each $a, b \in A$ to express the attack structure of the solution AF. Furthermore, for each $a \in A$ and $p \in P$, define variables x_a^p, with the interpretation that $\tau(x_a^p) = 1$ corresponds to $a \in E_p$, where E_p is any σ-extension containing the enforced argument p.

For credulous status enforcement, we must make sure that each positively enforced argument $p \in P$ is included in some extension E_p, and all arguments $n \in N$ are not found in any extension E_p. This is achieved for credulous status enforcement via

$$\textsc{status}(\mathit{cred}, F, P, N, \sigma) = \bigwedge_{p \in P} \left(\varphi_\sigma^p(F) \wedge x_p^p \wedge \bigwedge_{n \in N} \neg x_n^p \right),$$

where $\varphi_\sigma^p(F)$ encodes the semantics σ such that the x_a^p variables correspond to an extension $E_p \in \sigma(F')$ with $F' = (A, R')$, where R' is defined by the attack variables $r_{a,b}$. Like in non-strict extension enforcement, conflict-free sets are now encoded as

$$\varphi_{cf}^p(F) = \bigwedge_{a,b \in A} \left(r_{a,b} \rightarrow (\neg x_a^p \vee \neg x_b^p) \right),$$

stating that if there is an attack, no arguments can be included in the extension. We encode the admissible semantics via

$$\varphi_{adm}^p(F) = \varphi_{cf}^p(F) \wedge \bigwedge_{a,b \in A} \left((x_a^p \wedge r_{b,a}) \rightarrow \bigvee_{c \in A} (x_c^p \wedge r_{c,b}) \right),$$

that is, the admissible extension is conflict-free, and each argument contained in the extension is defended by the extension. For stable semantics, our encoding is

$$\varphi_{stb}^p(F) = \varphi_{cf}^p(F) \wedge \bigwedge_{a \in A} \left(\neg x_a^p \rightarrow \bigvee_{b \in A} (x_b^p \wedge r_{b,a}) \right),$$

which states that the stable extension is conflict-free, and the range of the extension is the whole set of arguments.

The credulous status enforcement problem for $N \neq \emptyset$ under the admissible and stable semantics, and the skeptical status enforcement problem under the stable semantics are Σ_2^P-complete. We describe our approach to tackle the computational complexity in the next section.

6.4 Capturing Further Variants

Extension enforcement and status enforcement as presented in this work allow arbitrary changes to the attack structure, and do not allow additional arguments. However, we note that by adjusting the MaxSAT encoding of the problem in an appropriate way, further variants of these problems can be solved via the same approach. First of all, the encoding allows for introducing any bounded number of additional arguments A' by considering $A \cup A'$ instead of the set A in the clauses of the encoding. In addition, fixing a part of the attack structure of the solution AF is accomplished by making the corresponding soft clauses hard. In other words, if we want the attacks $R^+ \subseteq A \times A$ to be present, and the attacks $R^- \subseteq A \times A$ not to be present in the solution AF, we can introduce the hard clauses

$$\bigwedge_{(a,b)\in R^+} r_{a,b} \wedge \bigwedge_{(a,b)\in R^-} \neg r_{a,b}$$

to the encoding, and modify the soft clauses by enumerating through the set $(A \times A) \setminus (R^+ \cup R^-)$ instead of $A \times A$. These changes allow for expressing also the normal, strong, and weak expansions. However, as noted in [17, 43], in this case the solution is not guaranteed to exist.

Weighted argument systems, as introduced in [52], are an extension of Dung's argumentation frameworks. In addition to the arguments and attacks, a non-zero weight is assigned to each attack, corresponding to the relative strength of the attack. Similarly to this approach, the enforcement problem can be modified by introducing a non-zero weight to each $(a, b) \in A \times A$ via the weight function $w\colon A \times A \to \mathbb{Z}_+$. Intuitively, if $(a, b) \in R$, the weight $w(a, b)$ corresponds to the strength of the existing attack, i.e., how reluctant we are to remove it, and if $(a, b) \notin R$, the weight $w(a, b)$ expresses how reluctant we are to add the attack. Now, using weighted partial MaxSAT instead of partial MaxSAT, this problem can be solved via the same approach by assigning to each soft clause $r'_{a,b}$ the weight $w(a, b)$.

7 Counterexample-guided Abstraction Refinement

In this section, we present algorithms for solving enforcement problems beyond NP, i.e., the Σ_2^P-complete problems of strict extension enforcement under the preferred, semi-stable, and stage semantics; non-strict extension enforcement under the semi-stable and (presumably) stage semantics; credulous status enforcement with $N \neq \emptyset$ under the admissible and stable semantics; and skeptical status enforcement under the stable semantics. Motivated by completeness of these problem variants for the second level of the polynomial hierarchy, we employ a general approach called counterexample-guided abstraction refinement [37, 38] (CEGAR). The basic idea is to start with an NP-abstraction, which is a problem in NP and an overapproximation of

the original problem, to solve it, leading to a candidate solution. Then we use an NP-oracle (such as a SAT solver) iteratively to check whether the candidate is indeed a solution to the original, harder problem by asking it for a counterexample. If there are none, we are essentially done and can exit the loop, and if there is, we refine the abstraction by adding constraints which rule out the counterexample given by the oracle, and continue by solving the abstraction. We note that there are no guarantees on finding the actual solution before an exponential blow-up of the encoding of the refinement, but with efficient refinements this approach has been shown to be effective in practice in different domains, see for examples [48, 67].

7.1 CEGAR Algorithm for Extension Enforcement

We start by providing CEGAR algorithms solving the strict extension enforcement problem under the preferred, semi-stable, and stage semantics, and the non-strict extension enforcement problem under the semi-stable and stage semantics, which have all either been shown or conjectured to be Σ_2^P-complete [90]. In this case, the NP-abstractions used are the admissible or complete semantics for the preferred and semi-stable semantics, and conflict-free sets for stage, since these overapproximate the problem at hand (recall Definition 3 and Theorem 4). The abstraction is solved using a MaxSAT solver, leading to a candidate solution. The validity of the candidate solution is checked using a SAT solver as an 'NP-oracle', by constructing a formula which is unsatisfiable if and only if T has been enforced in the new AF. If the candidate is not a solution to the original problem, we obtain a counterexample from the SAT solver, and add refinement clauses to the MaxSAT solver in order to constrain the abstraction further. This procedure is repeated until we have no counterexample and reach a solution to the original problem.

Let $F = (A, R)$ be an AF and $T \subseteq A$ the set to be enforced under semantics $\sigma \in \{prf, sem, stg\}$, and let $M \in \{ns, s\}$ be the type of enforcement (non-strict or strict). The CEGAR algorithm for solving these problems is

Algorithm 1 CEGAR-based extension enforcement for $\sigma \in \{prf, sem, stg\}$ with $M = s$, and for $\sigma \in \{sem, stg\}$ with $M = ns$.

1: **if** $\sigma \in \{prf, sem\}$ **then** $\chi \leftarrow com$ **else** $\chi \leftarrow cf$
2: $\varphi_h \leftarrow \text{EXT}(M, F, T, \sigma)$
3: **if** $M = ns$ **then** $\varphi_h \leftarrow \varphi_h \wedge \text{RANGE}(A)$
4: **while** true **do**
5: $\quad (c, \tau) \leftarrow \text{MAXSAT}(\varphi_h, \varphi_s(F))$
6: $\quad result \leftarrow \text{SAT}(\text{CHECK}(M, A, \tau, S, \sigma))$
7: $\quad$ **if** $result = unsatisfiable$ **then** return (c, τ)
8: $\quad$ **else** $\varphi_h \leftarrow \varphi_h \wedge \text{REFINE}(\tau, M)$

presented as Algorithm 1, the details of which are as follows. The first step is to choose the base semantics, denoted by χ, that acts as an overapproximation of the original problem. We choose conflict-free sets for stage and the admissible or complete semantics for semi-stable and preferred, since these are solvable via a direct MaxSAT call (recall Section 6.2). Next, we construct the MaxSAT clauses for solving the corresponding extension enforcement problem under the semantics χ, and enter the main loop of the algorithm. In the loop, we solve the problem via a MaxSAT solver call, extracting a truth assignment τ. Then we use it to construct the candidate solution AF $F' = (A, R')$, where

$$R' = \{(a, b) \mid a, b \in A, \tau(r_{a,b}) = 1\}.$$

Now we can use a SAT solver to check whether F' is also a solution to the enforcement problem under the semantics σ. Recall that for strict enforcement, we need to check if $T \in \sigma(F')$, and for non-strict enforcement, we check if $T' \in \sigma(F')$, where

$$T' = \{a \in A \mid \tau(x_a) = 1\}.$$

For this SAT solver call, we encode the base semantics as in [23], defining variables x_a for each $a \in A$ with the interpretation $a \in E \in \sigma(F')$. Now, conflict-free sets are encoded as

$$\textsc{extension}(F', cf) = \bigwedge_{(a,b)\in R'} (\neg x_a \vee \neg x_b),$$

stating that for all attacks in R', either the attacker or the attacked argument is not contained in the conflict-free set. Admissible semantics can be expressed by the clauses

$$\textsc{extension}(F', adm) = \textsc{extension}(F', cf) \wedge \bigwedge_{(b,a)\in R'} \left(x_a \rightarrow \left(\bigvee_{(c,b)\in R} x_c \right) \right),$$

that is, an admissible extension is conflict-free, and if there is an attack on an argument in the extension, there must be a defending argument in the extension. Finally, we encode complete semantics by

$$\begin{aligned}\textsc{extension}(F', com) = \textsc{extension}(F', adm) \\ \wedge \bigwedge_{a\in A} \left(\left(\bigwedge_{(b,a)\in R'} \left(\bigvee_{(c,b)\in R'} x_c \right) \right) \rightarrow x_a \right),\end{aligned}$$

i.e., complete extensions are admissible, and each argument that is defended is also included in the extension. For preferred semantics, we check whether there is a strict superset of T, which we express via

$$\textsc{superset}(F', T, prf) = \bigwedge_{a\in T} x_a \wedge \bigvee_{a\in A\setminus T} x_a.$$

In the case of semi-stable and stage semantics, we need to likewise search for a superset with respect to the range of T (for strict enforcement) or T' (for non-strict enforcement), which we denote by S. To accomplish this, we define for each $a \in A$ new Boolean variables x_a^+ which encode the range of each argument as a conjunction of equivalences

$$\textsc{range}(F') = \bigwedge_{a \in A} \left(x_a^+ \leftrightarrow \left(x_a \vee \bigvee_{(b,a) \in R'} x_b \right) \right).$$

Now, under $\sigma \in \{sem, stg\}$, checking whether there is a superset of S with respect to the range is encoded as

$$\textsc{superset}(F', S, \sigma) = \bigwedge_{a \in S_{R'}^+} x_a^+ \wedge \bigvee_{a \in A \setminus S_{R'}^+} x_a^+ \wedge \textsc{range}(F').$$

If such a superset that is a χ-extension exists, the formula

$$\textsc{check}(M, A, \tau, S, \sigma) = \textsc{extension}(F', \sigma) \wedge \textsc{superset}(F', S, \sigma)$$

is satisfiable, providing a counterexample that F' is not an actual solution, that is, T or T' is not a σ-extension in F'.

For the refinement step, we define the shorthand that encodes the attack structure R' of F' via

$$\textsc{attack}(\tau) = \bigwedge_{(a,b) \in R'} r_{a,b} \wedge \bigwedge_{\{a,b\} \subseteq A, (a,b) \notin R'} \neg r_{a,b}.$$

The current abstraction ψ is in the strict case refined by

$$\textsc{refine}(\tau, s) = \neg \textsc{attack}(\tau),$$

which rules out the attack structure of the candidate solution. In the non-strict case, however, it may be that the attack structure of our candidate solution is indeed correct, but the superset T' of T is not. In other words, there might be a χ-extension T'' that is also a σ-extension in F'. If this is the case, then the range of the set T'' is not a subset of the range of T'. This is ruled out by

$$\textsc{refine}(\tau, ns) = \neg \textsc{attack}(\tau) \vee \bigvee_{a \in A \setminus T_{R'}'^+} x_a^+,$$

where we have defined range variables (Line 3) via

$$\textsc{range}(A) = \bigwedge_{a \in A} \left(x_a^+ \leftrightarrow \left(x_a \vee \bigvee_{b \in A} (r_{b,a} \wedge x_b) \right) \right).$$

Algorithm 1 solves strict enforcement for $\{prf, sem, stg\}$ and non-strict enforcement for $\{sem, stg\}$ optimally, as at each iteration the current abstraction is solved optimally.

Example 12. *Consider the AF F in Figure 1 on page 4, and enforcing $T = \{b\}$ strictly under the preferred semantics. We go through a hypothetical run of the CEGAR algorithm, proceeding as follows. Start by defining Boolean variables*

$$r_{a,a}, r_{a,b}, r_{a,c}, r_{a,d}, r_{b,a}, r_{b,b}, r_{b,c}, r_{b,d}, r_{c,a}, r_{c,b}, r_{c,c}, r_{c,d}, r_{d,a}, r_{d,b}, r_{d,c}, r_{d,d}.$$

Using complete as the semantics for the abstraction, we form the hard clauses $\varphi_h = \text{EXT}(s, F, T, com)$ along with the soft clauses $\varphi_s(F)$, and enter the CEGAR loop of Algorithm 1. Suppose the MaxSAT solver returns a truth assignment that corresponds to an AF $F' = (A, R')$, where

$$R' = \{(a, b), (b, a), (c, d), (d, c)\},$$

i.e., the attack (b, a) has been added and the attack (b, c) removed, illustrated in Figure 7.

Now, we form an instance of SAT with the variables x_a, x_b, x_c and x_d, and the clauses

$$\begin{aligned}\text{CHECK}(s, A, \tau, T, prf) &= \text{EXTENSION}(F', adm) \wedge \text{SUPERSET}(F', T, prf)\\ &= (\neg x_a \vee \neg x_b) \wedge (\neg x_c \vee \neg x_d)\\ &\quad \wedge (x_b \rightarrow x_b) \wedge (x_a \rightarrow x_a)\\ &\quad \wedge (x_d \rightarrow x_d) \wedge (x_c \rightarrow x_c)\\ &\quad \wedge x_b \wedge (x_a \vee x_c \vee x_d).\end{aligned}$$

This instance is satisfiable via the truth assignment $\tau(x_a) = 0$, $\tau(x_b) = 1$, $\tau(x_c) = 1$, $\tau(x_d) = 0$, corresponding to the admissible extension $\{b, c\}$. Therefore we get a counterexample, and we refine the solution via adding the hard clause

$$\text{REFINE}(\tau, s) = \neg\text{ATTACK}(\tau)$$

and continue in the CEGAR loop. Now suppose the MaxSAT solver suggests the attack structure

$$R'' = \{(a, b), (b, a), (b, c), (b, d), (c, d), (d, c)\},$$

i.e., the attacks (b, a) and (b, d) have been added into the framework, illustrated in Figure 7. Again, we form an instance of SAT with the variables x_a, x_b, x_c and x_d, and the clauses

$$\begin{aligned}\text{CHECK}(s, A, \tau, T, prf) &= \text{EXTENSION}(F', adm) \wedge \text{SUPERSET}(F', T, prf)\\ &= (\neg x_a \vee \neg x_b) \wedge (\neg x_b \vee \neg x_c)\\ &\quad \wedge (\neg x_b \vee \neg x_d) \wedge (\neg x_c \vee \neg x_d)\\ &\quad \wedge (x_b \rightarrow x_b) \wedge (x_a \rightarrow x_a)\\ &\quad \wedge (x_c \rightarrow x_d) \wedge (x_d \rightarrow (x_b \vee x_c))\\ &\quad \wedge (x_c \rightarrow (x_b \vee x_c))\\ &\quad \wedge x_b \wedge (x_a \vee x_c \vee x_d)\end{aligned}$$

Since we must set $\tau(x_b) = 1$ due to the unit clause x_b, we know that we must set $\tau(x_a) = 0$, $\tau(x_c) = 0$ and $\tau(x_d) = 0$ due to the clauses encoding conflict-freeness. Therefore the instance is unsatisfiable, since the clause $x_a \vee x_c \vee x_d$ is not satisfied. Now we know that the AF $F' = (A, R')$ is an optimal solution to the enforcement problem.

Figure 7: The argumentation frameworks F' (left) and F'' (right) in Example 12.

7.2 CEGAR Algorithm for Status Enforcement

We continue by describing the CEGAR procedures for solving the credulous status enforcement problem with negatively enforced arguments under the admissible and stable semantics, and the skeptical status enforcement problem under the stable semantics, which are complete for the second level of polynomial hierarchy (recall Section 4). We start with credulous status enforcement. Since for $N = \emptyset$ the corresponding problems are solvable via a direct MaxSAT solver call, we can use those as the NP-abstractions of the general credulous status enforcement problem to generate a candidate solution. Next we check with a SAT solver whether there exists a counterexample, i.e., whether there exists an extension in the candidate AF containing some negatively enforced argument. Then we repeat the procedure until no counterexample is found, in which case the candidate solution is a solution to the original problem.

Let $F = (A, R)$ be an AF, and $P, N \subseteq A$, $P \cap N = \emptyset$, sets of arguments whose status is to be enforced positively and negatively, respectively, under semantics $\sigma \in \{adm, stb\}$. The details of the CEGAR algorithm (Algorithm 2) are as follows. First, we construct the MaxSAT clauses for the credulous status enforcement problem via $\textsc{status}(cred, F, P, N, \sigma)$, and enter the CEGAR loop. The problem is solved by a MaxSAT solver, generating

Algorithm 2 CEGAR-based status enforcement for AF $F = (A, R)$, $P, N \subseteq A$, $\sigma \in \{adm, stb\}$, $M \in \{cred, skept\}$

```
1: φ ← STATUS(M, F, P, N, σ)
2: while true do
3:     (c, τ) ← MaxSAT(φ_h, φ_s(F))
4:     result ← SAT(CHECK(M, A, τ, P, N, σ))
5:     if result = unsatisfiable then return (c, τ)
6:     else φ_h ← φ_h ∧ REFINE(τ)
```

a candidate solution (Line 3) $F' = (A, R')$, where each $p \in P$ is enforced positively and each $n \in N$ is not contained in the witness extensions via

$$R' = \{(a, b) \mid a, b \in A, \tau(r_{a,b}) = 1\}.$$

Then we still need to check that each $n \in N$ is not contained in any σ-extension of the AF F', which is achieved via a SAT solver call on Line 4. For the SAT check, we encode the base semantics as in [23]. The encodings were defined in Section 7.1, and we denote them by $\textsc{extension}(F', \sigma)$ for $\sigma \in \{adm, stb\}$. Now the clauses of the SAT check are

$$\textsc{check}(cred, A, \tau, P, N, \sigma) = \textsc{extension}(F', \sigma) \wedge \bigvee_{n \in N} x_n,$$

which states that some argument in N is credulously accepted, or in some σ-extension. If it is satisfiable, we have a counterexample, and proceed to the refinement step, where we exclude the attack structure R' of the candidate solution by adding the clauses

$$\textsc{refine}(\tau) = \neg\textsc{attack}(\tau)$$

to the initial MaxSAT encoding. Algorithm 2 describes this procedure, which solves the optimal status enforcement problem, since in each iteration step the current abstraction is solved optimally.

We proceed to skeptical status enforcement under stable semantics, which is in the restricted ($N = \emptyset$) and general case Σ_2^P-complete. The CEGAR procedure now acts in a dual way by first computing a candidate solution where all negatively enforced arguments are not skeptically accepted under stable, i.e., not contained in some witness extension. This is encoded by

$$\textsc{status}(skept, F, P, N, \sigma) = \bigwedge_{n \in N} \left(\varphi_\sigma^n(F) \wedge \neg x_n^n \wedge \bigwedge_{p \in P} x_p^n\right).$$

Again, we extract an AF $F' = (A, R')$ using the truth assignment extracted from the MaxSAT solver call, but we still need to check whether all positively enforced arguments are contained in every stable extension. The clauses of the SAT check are

$$\textsc{check}(skept, A, \tau, P, N, stb) = \textsc{extension}(F', \sigma) \wedge \bigvee_{p \in P} \neg x_p,$$

and if this formula is satisfiable, there is a stable extension not containing a positively enforced argument, i.e., a positively enforced argument is not skeptically accepted. In this case we have a counterexample, so we can rule out the current attack structure via the same refinement clause. In the restricted case $N = \emptyset$, we initially enforce a stable extension containing P via STATUS, and proceed similarly.

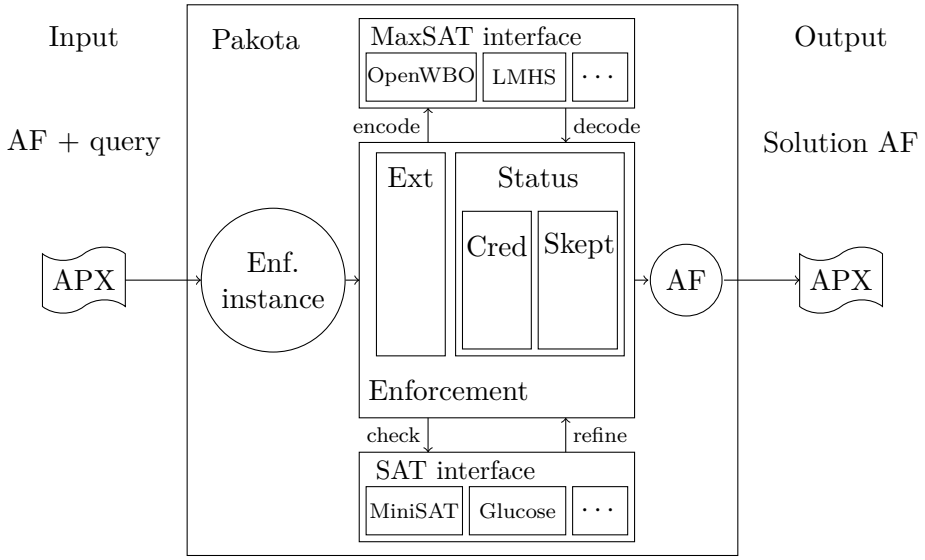


Figure 8: System architecture of Pakota.

8 Implementation

We have implemented the encodings and algorithms described in Sections 6 and 7, resulting in the software system dubbed Pakota.[2] The Pakota system is implemented in the C++ programming language. The source code is available at

`http://www.cs.helsinki.fi/group/coreo/pakota/`

under the MIT license. In what follows, we describe the main components and system architecture of the system (Section 8.1) and main features of Pakota (Section 8.2), detail the implemented algorithms (Section 8.3), input and output specifications (Section 8.4), and usage (Section 8.5).

8.1 System Architecture

The system architecture of Pakota is shown in Figure 8. Pakota accepts input for the extension enforcement problem and for the credulous and skeptical status enforcement problem in the so-called APX format (see Section 8.4), which is parsed into an enforcement instance. The algorithms implemented in Pakota that solve the given enforcement instance form the main component of the system and are described in Section 8.3, employing a MaxSAT solver, or, for problem variants beyond NP, interacting MaxSAT and SAT solvers. Pakota offers a generic MaxSAT interface for plugging in the MaxSAT solver

[2] 'Pakota' is the imperative form of 'to enforce' in Finnish language.

of choice and already includes MaxSAT solvers Open-WBO [71] (version 1.3.1) and LMHS [82] (version 2015.11), and the SAT solvers MiniSAT [57] (version 2.2.0, included with LMHS) and Glucose [8, 9, 10] (version 3.0, included with Open-WBO). We detail usage of the MaxSAT interface in Section 8.2.

The implemented algorithms for the enforcement problems can be classified according to whether they solve an NP problem or a second-level problem. For the former, the enforcement instance is encoded in a MaxSAT instance and the solution given by a MaxSAT solver is decoded to construct a solution AF to the enforcement problem, again in the APX format. In the case that the given task is a second-level problem, the algorithms implement a counterexample-guided abstraction refinement procedure, thereby iteratively querying the MaxSAT solver to construct candidate solutions and checking whether the candidate is indeed a solution to the enforcement problem via a SAT solver. In case the candidate is a solution, the decoded AF is returned in the APX format. Otherwise, i.e., in case the candidate is a non-solution, the current MaxSAT encoding is iteratively refined until an actual optimal solution is found.

8.2 Features

We overview the general features of the Pakota system, including supported AF semantics, problem variants, and solver interfaces.

8.2.1 Supported Semantics and Reasoning Modes

Pakota currently supports optimally enforcing an extension both strictly and non-strictly under the admissible, complete, preferred, stable, semi-stable, and stage semantics. Further, Pakota implements optimal credulous status enforcement under the admissible, complete, preferred, and stable semantics, and optimal skeptical status enforcement under the stable semantics. An overview of the supported semantics and reasoning modes is given in Table 4 with further implementation details for certain parameter choices discussed in more detail in Section 8.3.

8.2.2 MaxSAT and SAT Solver Interfaces

Essentially any MaxSAT solver whose source code is available can be plugged into the system. This is enabled in Pakota by offering two interfaces, `MaxSATSolver.h` and `SATSolver.h`. By creating new classes that implement these interfaces and defining the pure virtual functions declared in them, one can compile and link these to the Pakota system, which will then use the corresponding MaxSAT and SAT solvers for solving the enforcement problems. As an implementation-level detail, note that, if the MaxSAT solver uses a SAT solver internally, which is usually the case, an easy solution to

Table 4: Extension and status enforcement problems supported by Pakota.

Enforcement	parameters	semantics	Encoding/Algorithm
extension	ns	adm, com, prf	EXT(ns, F, T, adm)
extension	ns	stb	EXT(ns, F, T, stb)
extension	ns	sem,stg	Algorithm 1
extension	s	com	EXT(s, F, T, com)
extension	s	prf,sem,stg	Algorithm 1
status	$cred$, $N = \emptyset$	adm, com, prf	STATUS$(cred, A, P, \emptyset, adm)$
status	$cred$, $N = \emptyset$	stb	STATUS$(cred, A, P, \emptyset, stb)$
status	$cred$	adm, com, prf	Algorithm 2
status	$cred$	stb	Algorithm 2
status	$skept$	stb	Algorithm 2

potential naming conflicts is to use the same SAT solver as the SAT solver in CEGAR procedures within Pakota. The source code of Pakota already includes implementations of these interfaces for two different MaxSAT solvers, Open-WBO [71] and LMHS [82], allowing the use of these solvers simply by editing the `MAXSAT_SOLVER` parameter in the included Makefile before compiling.

8.2.3 MaxSAT and IP Encodings

In addition to directly solving extension and status enforcement instances, Pakota can for the NP variants of the problems output the internal MaxSAT encodings both in the standard WCNF MaxSAT input format as well as integer programs (IPs) in the standard LP format (applying the standard textbook encoding of MaxSAT as IP [6]). The latter option allows for calling state-of-the-art IP solvers, such as CPLEX [65] or Gurobi [63], on the encodings.

8.3 Algorithms

Depending on the inherent complexity of the problems, Pakota solves the extension or status enforcement problem at hand by either encoding the problem in MaxSAT (NP-complete problems), or within a counterexample-guided abstraction refinement (CEGAR) scheme utilizing a MaxSAT solver in an iterative or incremental fashion (problems complete for the second level of polynomial hierarchy). Table 4 provides details, depending on the chosen parameters and semantics, for each problem variant, whether it is solved via direct encoding to MaxSAT or via a MaxSAT-based CEGAR algorithm (detailed as Algorithms 1 and 2).

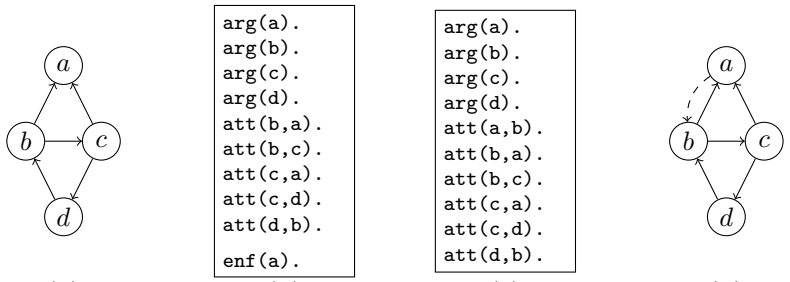
8.4 Input Format

For extension enforcement, the input AF and enforcement request are specified using the following predicates, extending the APX format for AFs.

`arg(X).`	`X` is an argument
`att(X,Y).`	there is an attack from `X` to `Y`
`enf(X).`	enforce argument `X`

As in extension enforcement, for status enforcement the AF is represented using the `arg` and `att` predicates. The arguments to be positively and negatively enforced are represented via the `pos` and `neg` predicates, respectively. For example, `pos(a).` enforces argument a positively. The reasoning mode between credulous and skeptical is chosen from the command line.

Example 13. *The enforcement of argument a for the AF in Fig. 9(a) is specified in the Pakota input format as shown in Fig. 9(b). On this input, Pakota may return the output shown in Fig. 9(c), i.e., the AF in Fig. 9(d).*



(a) (b) (c) (d)

Figure 9: Pakota input and output formats by example.

8.5 Usage and Options

After compilation, the Pakota system is used from the command line as follows.

```
./pakota <file> <mode> <sem> [options]
```

The command line arguments enabling the choice of AF semantics and reasoning mode are the following.

```
<file>    : Input filename for enforcement instance in apx format.
<mode>    : Enforcement variant: mode={strict|non-strict|cred|skept}
    strict     : strict extension enforcement
    non-strict : non-strict extension enforcement
```

```
    cred       : credulous status enforcement
    skept      : skeptical status enforcement
<sem>     : Argumentation semantics. sem={adm|com|stb|prf|sem|stg}
    adm        : admissible
    com        : complete
    stb        : stable
    prf        : preferred
    sem        : semi-stable (only for extension enforcement)
    stg        : stage (only for extension enforcement)
```

Furthermore, command line options `-h` (for help message), `-v` (for version number), `-o` (for specifying output to file) and `-t` (for outputting NP-encodings in WCNF and LP formats) are available.

8.6 Benchmarks and Generators

The Pakota webpage also offers sets of benchmarks for both extension enforcement and status enforcement in the Pakota input format. Furthermore, we provide via the webpage our benchmark generator software, AfGen and EnfGen, which we used to generate the benchmark sets. The AF generator AfGen forms argumentation frameworks in APX format implementing the Erdős-Rényi random digraph model. The generator is called as

```
./afgen <args> <prob>
```

where parameters `<args>` and `<prob>` specify the number of arguments and the probability of an attack in the output AF. The generator forms an argumentation framework with arguments $1, \ldots,$ `<args>`, including an attack between each pair of arguments independently with probability `<prob>`.

The enforcement instance generator EnfGen takes as input an AF in APX format, and produces an enforcement instance. It is called as

```
./enfgen <file> <mode> <enfs>
```

where `<file>` is the input AF and `<mode>` is either `ext` or `status`, corresponding to extension and status enforcement, respectively. In case of extension enforcement, `<enfs>` is an integer stating the number of arguments to be enforced, and for status enforcement, `<enfs>` is a pair of integers, corresponding to the number of positively and negatively enforced arguments. The generator reads the arguments from the AF and samples the enforced arguments uniformly at random, without replacement.

9 Empirical Evaluation

We present results from an empirical evaluation of the Pakota system by providing scalability experiments under different problem variants for extension and status enforcement, both for the NP- and Σ_2^P-complete problems. In addition we evaluate the impact of the choice of the underlying MaxSAT solver. For the NP-complete problem variants, we used the state-of-the-art MaxSAT solvers MaxHS [44], Maxino [2], MSCG [73], Open-WBO [71], and WPM [5], using the MaxSAT Evaluation 2015 versions, as well as the commercial IBM CPLEX integer programming solver (version 12.6), by directly solving the CNF encoding as produced by Pakota. For CEGAR, we compare the performance of Open-WBO and LMHS [82] as the underlying MaxSAT solvers, as supported by Pakota. Out of the solvers considered, Maxino, MSCG, Open-WBO, and WPM are core-guided MaxSAT solvers, and MaxHS and LMHS are SAT-IP hybrids.

For extension enforcement, we also compare the performance of Pakota and the only other solver for extension enforcement we are aware of, described in [43], and obtained from the author and developer of the solver Jean-Guy Mailly. We refer to it as the IJCAI'15 solver. We note that the obtained version of the IJCAI'15 solver does not support any other AF semantics than stable, and therefore no results for other semantics on this solver can be presented.

The experiments were run on 2.83-GHz Intel Xeon E5440 quad-core machines with 32-GB RAM and Debian GNU/Linux 8 using a timeout of 900 seconds per instance. We generated the benchmarks using our AfGen and EnfGen generators. The generated instances are available at

`http://www.cs.helsinki.fi/group/coreo/pakota/`

9.1 Results for Extension Enforcement

To obtain benchmarks for extension enforcement, for each number of arguments $|A| \in \{25, 50, \dots\}$ and each edge probability $p \in \{0.05, 0.1, 0.2, 0.3\}$, we generated five AFs. For each AF, we generated five enforcement instances with $|T|$ enforced arguments, for each $|T|/|A| \in \{0.05, 0.1, 0.2, 0.3\}$. We thus obtained 400 instances for each $|A|$.

For the NP-complete extension enforcement problems, we consider the tasks of enforcing an extension non-strictly under the admissible and stable semantics, and strictly under the complete semantics. In Figure 10 (left) we show the median runtimes for Open-WBO and CPLEX on non-strict admissible, strict complete, and non-strict stable extension enforcement, along with the IJCAI'15 solver [43] on non-strict stable, with respect to the number of arguments in the problem instance. The median runtimes for Open-WBO are clearly lower than for the rest of the solvers, which indicates

that using core-guided MaxSAT solvers is a promising approach for solving the extension enforcement problem, with most of the instances even with $|A| = 350$ solved under one second. The performance of the IJCAI'15 solver is almost the same as that of CPLEX, which comes as no surprise, since it also encodes the problem instance as an instance of IP, using CPLEX as the underlying solver.

Turning to the Σ_2^P-complete problem of enforcing an extension strictly under the preferred semantics, we show the median runtimes for the CEGAR algorithm in Figure 10 (right) w.r.t. the number of arguments in the instance. The runtime for each instance is included as a point in the plot, visualizing the distribution of the runtimes. We used Open-WBO as the MaxSAT solver, and the complete semantics as the base abstraction. Even with 175 arguments, most of the instances are solved within one second, which demonstrates the practical applicability of the CEGAR approach on this particular problem.

Similarly, for the Σ_2^P-complete problem of extension enforcement under the semi-stable semantics, Figure 11 includes plots for the strict (left) and non-strict (right) variants of the problem, again using Open-WBO as the MaxSAT solver and the complete semantics as the base abstraction. Comparing to the plot in Figure 10 (right), we can see similar scalability as in the case of preferred semantics. For the strict version (left), we can see unusually high median runtimes at around 50 arguments, the cause of which is unclear. This could be related to how the algorithms of the MaxSAT and SAT solvers behave on the encodings used in this problem.

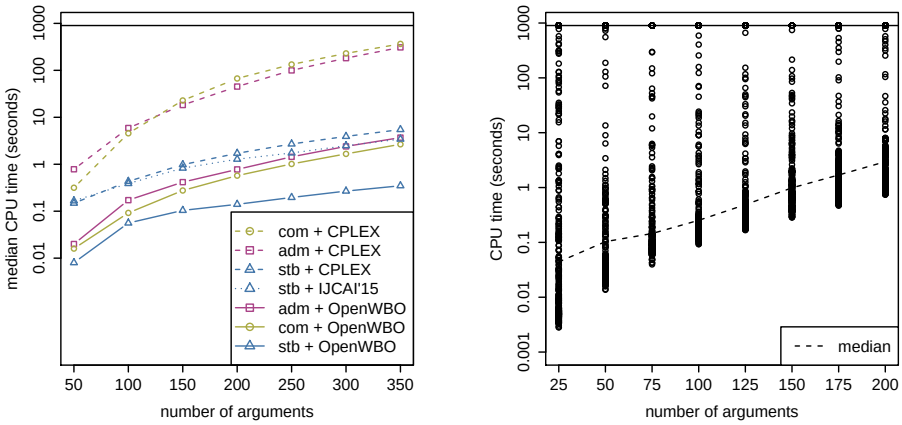


Figure 10: Left: NP-complete extension enforcement (non-strict under admissible and stable, strict under complete); right: Σ_2^P-complete strict extension enforcement under the preferred semantics.

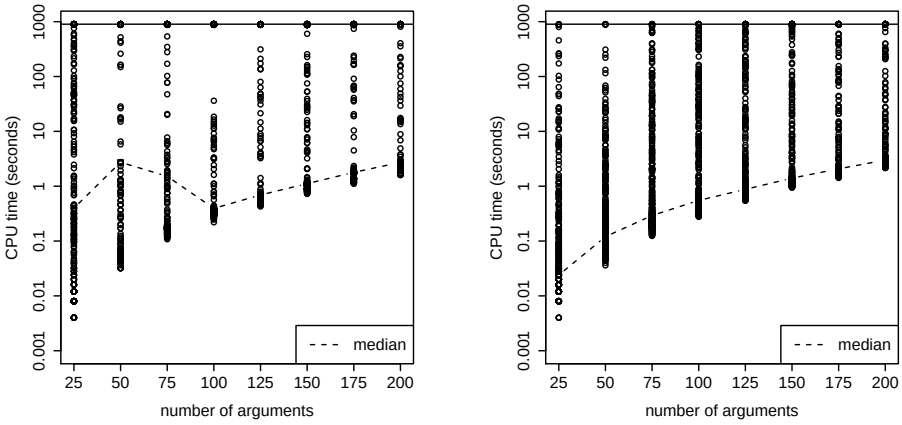


Figure 11: Σ_2^P-complete extension enforcement under the semi-stable semantics. Left: strict extension enforcement; right: non-strict extension enforcement.

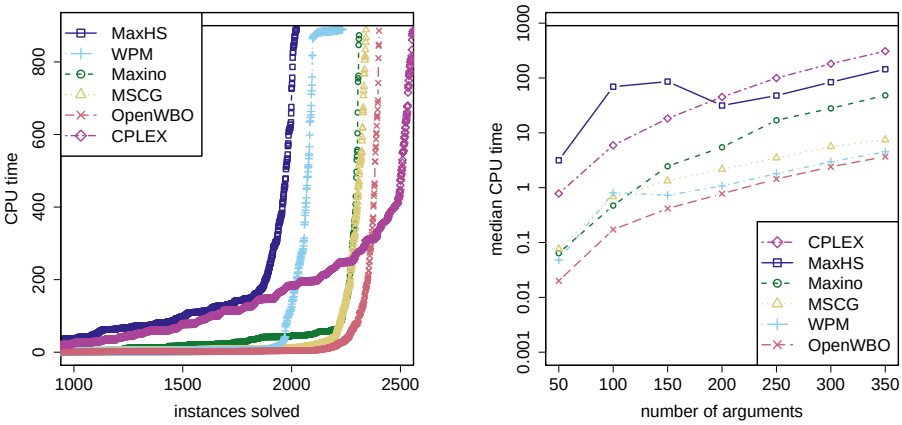


Figure 12: MaxSAT solver comparison for NP-complete non-strict extension enforcement under the admissible semantics.

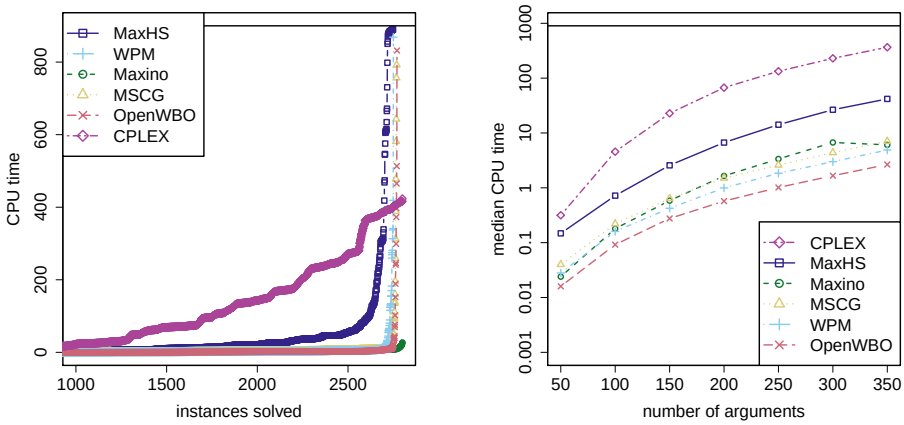


Figure 13: MaxSAT solver comparison for NP-complete strict extension enforcement under the complete semantics.

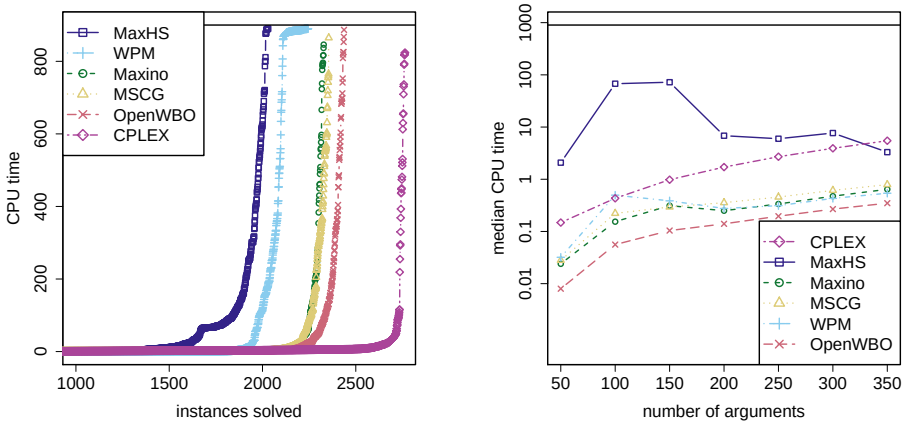


Figure 14: MaxSAT solver comparison for NP-complete non-strict extension enforcement under the stable semantics.

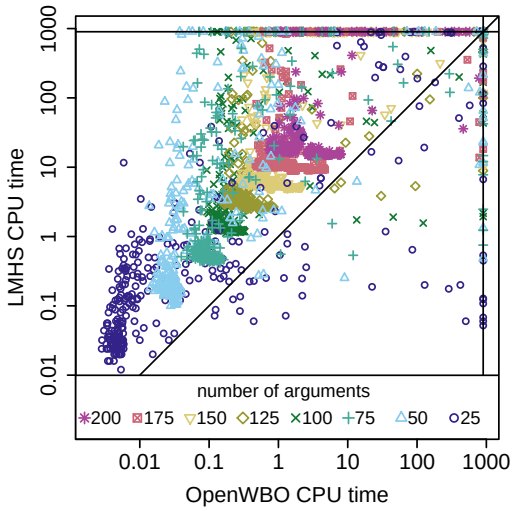


Figure 15: MaxSAT solver comparison for Σ_2^P-complete strict extension enforcement under preferred semantics.

In Figures 12 (right), 13 (right) and 14 (right) we demonstrate the median runtimes for Open-WBO, WPM, Maxino, MSCG, MaxHS, and CPLEX, with respect to the number of arguments. These plots also indicate that Open-WBO performs better than the rest of the solvers in terms of median runtimes—all in all, the core-guided MaxSAT solvers Open-WBO, WPM, Maxino and MSCG perform the best. On non-strict extension enforcement under admissible (Figure 12, right) and stable (Figure 14), the median runtimes for MaxHS are surprisingly high between 100 and 150 arguments, which is not the case for strict enforcement under complete (Figure 13). Since MaxHS is the only SAT-IP hybrid MaxSAT solver in this comparison, a possible cause is that the algorithm of the solver on the encoding of the problem behaves differently on smaller instances. However, the reason for this is unclear, and would need further investigation beyond this work. Another interesting fact is that the cactus plots in Figures 12 (left), 13 (left) and 14 (left) reveal that CPLEX performs considerably better in terms of solving more instances than the rest of the solvers, although for non-strict admissible (Figure 12 (left)) and non-strict stable (Figure 14 (left)), it runs slower than most of the other solvers. In addition, as seen from Figure 13, on strict complete CPLEX and Maxino are the only solvers which solve all of the instances generated within the timeout limit. We can also see from the plots that for strict enforcement under complete semantics, all of the MaxSAT solvers solve more instances compared to non-strict admissible and non-strict stable, indicating that strict complete is empirically easier on these instances.

In Figure 15 we illustrate the performance of the MaxSAT solvers Open-WBO and LMHS on the CEGAR approach for strict enforcement under the preferred semantics, by including the runtime of each instance as a point in

the plot. We can see that most of the points in the plot are on the upper-left side, indicating that the core-guided solver Open-WBO solves most of the generated instances faster than the SAT-IP hybrid LMHS, and therefore provides the better option for the MaxSAT solver in the CEGAR algorithm.

The results of the empirical evaluation for stage semantics are not shown here, since most of the instances timeout even at 25 arguments. This is due to using the conflict-free semantics as the base abstraction, which basically just tells the MaxSAT solver to remove all attacks inside the enforced set. This results in the solver guessing truth assignments, i.e., the attack structure, without any further constraints. Therefore in practice the algorithm has too many iterations in the CEGAR loop, resulting in high runtimes. To improve the approach, we hypothesize that a more restricting base abstraction, or better refinement clauses, are needed. This would be an interesting idea for future work beyond this thesis. It would also be interesting to investigate whether the encodings for other semantics could be optimized further.

In summary, we have shown that using MaxSAT solvers and a CEGAR-based approach for extension enforcement is practically viable, solving instances with 200 arguments and beyond. In general, core-guided MaxSAT solvers provide the best alternative in terms of the median runtimes.

9.2 Results for Status Enforcement

To generate benchmark instances for status enforcement, for each $|A| \in \{20, 40, \ldots, 200\}$ and edge probability $p \in \{0.05, 0.1, \ldots, 0.35\}$, we generated 10 AFs. For each AF, we generated an enforcement instance containing

$$(|P|, |N|) \in \{(1,0), (2,0) \ldots, (5,0), (5,1), (2,2), (1,5)\}$$

positively and negatively enforced arguments. This gave a total of 560 status enforcement instances for each $|A|$.

The mean runtimes for the NP-complete problem of credulous status enforcement under the admissible semantics for $N = \emptyset$ (left), and for the Σ_2^P-complete problem of credulous and skeptical status enforcement under the stable semantics (right) are shown in Figure 16, using Open-WBO as the MaxSAT solver. Timeouts are included when calculating the mean as 900 seconds. The number of positively enforced arguments significantly affects the empirical hardness of the problem, as seen from the left plot. After 100 arguments in the instance, for $|P| > 2$, the mean runtime starts approaching the timeout limit, indicating that most of the instances are not solved. From the right plot, we can clearly see that under the stable semantics, credulous status enforcement is easier than skeptical status enforcement. In the credulous case, again the number of positive arguments seems to be the most significant factor for the hardness of the problem. In the skeptical case, however, the numbers of positively and negatively enforced arguments do not seem to significantly affect the runtimes.

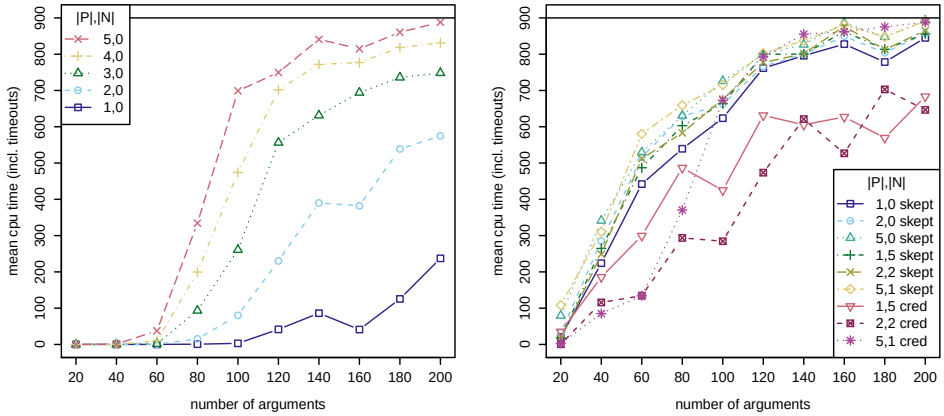


Figure 16: Left: NP-complete credulous status enforcement under the admissible semantics ($N = \emptyset$); right: Σ_2^P-complete status enforcement under the stable semantics.

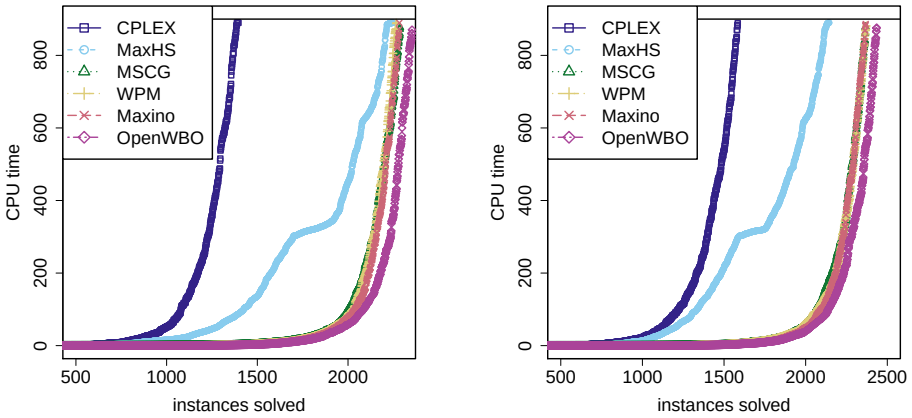


Figure 17: MaxSAT solver comparison for NP-complete credulous status enforcement ($N = \emptyset$) under the admissible (left) and stable (right) semantics.

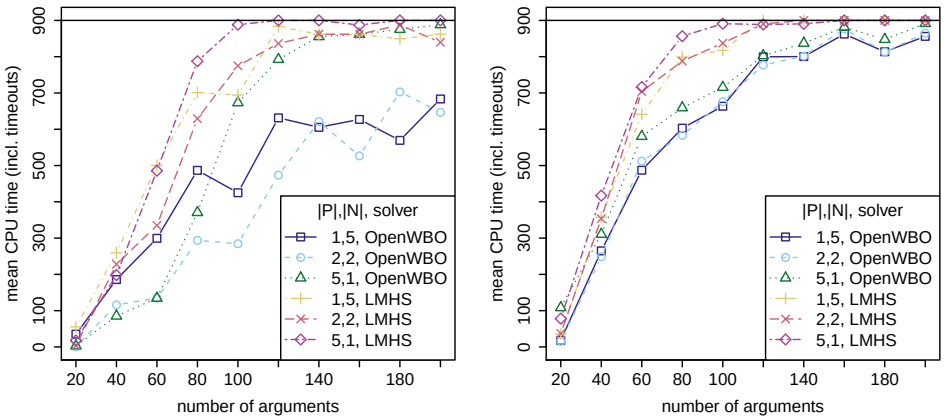


Figure 18: MaxSAT solver comparison for Σ_2^P-complete credulous (left) and skeptical (right) status enforcement under the stable semantics with $N \neq \emptyset$.

The MaxSAT solver comparison for NP-complete credulous status enforcement with $N = \emptyset$ under the admissible (left) and stable (right) semantics is shown in Figure 17. Again, out of all the solvers considered, core-guided MaxSAT solvers Open-WBO, Maxino, MSCG and WPM perform better than other solvers. Open-WBO is clearly the best option, solving more instances than the rest of the solvers, and doing it faster than the rest of the solvers. It is interesting, however, that for the credulous status enforcement problem CPLEX solves less instances than any other solver, but in extension enforcement it solved more instances than any other solver. Figure 18 provides the mean runtimes for the Σ_2^P-complete tasks of credulous and skeptical status enforcement under the stable semantics with $N \neq \emptyset$, using Open-WBO and LMHS as the MaxSAT solvers in the CEGAR algorithm. The core-guided solver Open-WBO clearly outperforms the SAT-IP hybrid LMHS in terms of the mean runtimes.

Summarizing the results for status enforcement, similarly to extension enforcement, core-guided MaxSAT solvers provide the best alternative out of the solvers considered, both for directly encoding the problem, and for the CEGAR-based approach. However, status enforcement is clearly much harder to solve in practice. For the NP-complete variants, this can be seen from Figure 17 by noting that under 2500 instances are solved within the timeout limit by the best solver, out of the total 5600 instances. Perhaps better encodings could be developed in order to solve the problem efficiently. These would also be useful in the CEGAR algorithm for status enforcement. The CEGAR algorithm could be improved further by using more restricting refinement clauses, as in the current approach only the current attack structure is forbidden.

Finally, beyond this work, an interesting question to investigate is why core-guided solvers seem to be the best alternative for both extension and status enforcement. This problem could be tackled via theoretical analysis of the MaxSAT algorithms and how the encodings of the problems affect their performance.

10 Conclusions

The dynamic aspects of argumentation frameworks is a recent and currently active research direction. The computational aspects of dynamics, however, have not been studied on a large scale. This thesis contributes by providing complexity results and algorithms for two fundamental dynamic computational problems in abstract argumentation—extension and status enforcement. We defined the problems as discrete optimization problems. We analyzed the computational complexity of the problem under different problem parameters, proving that the general problem is NP-hard, with the complexity ranging from polynomial-time solvable to the completeness on the

second level of the polynomial hierarchy. We described algorithmic solutions to the problem, utilizing a declarative approach where maximum satisfiability is used as the constraint optimization language. For the NP-complete variants of the problem, we provided direct encodings in MaxSAT, and for the Σ_2^P-complete variants, an iterative CEGAR-based procedure where in addition SAT solvers are used as practical NP-oracles. We implemented these algorithms, resulting in the software system Pakota, which we described in high detail and evaluated on random benchmarks with different parameters.

There are various potential ways to extend this work in the future. The computational complexity map is not yet full—there are no completeness results for strict extension enforcement under the stage semantics and skeptical status enforcement under the preferred semantics. Filling these blanks is a natural way of continuing the research on these problems. In addition, no algorithms have been developed for extension enforcement under the grounded semantics and skeptical status enforcement under the complete, grounded and preferred semantics. These procedures could be described and implemented, extending the existing system. The current implementation could also be optimized further by using more efficient encodings, or abstractions and refinements during the CEGAR procedure. Finally, the enforcement problem could be studied in other argumentation formalisms, and the suitability of similar approaches to computation to the ones developed in this thesis could be studied.

Acknowledgements

This work has been funded by Academy of Finland under grants 251170 COIN, 276412, and 284591; and Research Funds of the University of Helsinki. I thank my instructors Matti Järvisalo and Johannes P. Wallner for guidance, help, and support, for the supervision of this thesis, and for their collaboration on [74, 75, 90].

Bibliography

[1] Alison G. Doig Ailsa H. Land: *An automatic method of solving discrete programming problems.* Econometrica, 28(3):497–520, 1960.

[2] Mario Alviano, Carmine Dodaro, and Francesco Ricca: *A MaxSAT algorithm using cardinality constraints of bounded size.* In Qiang Yang and Michael Wooldridge (editors): *Proc. IJCAI*, pages 2677–2683. AAAI Press / IJCAI, 2015.

[3] Leila Amgoud and Henri Prade: *Using arguments for making and explaining decisions.* Artificial Intelligence, 173(3-4):413–436, 2009.

[4] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy: *SAT-based MaxSAT algorithms*. Artificial Intelligence, 196:77–105, 2013.

[5] Carlos Ansótegui, Frédéric Didier, and Joel Gabàs: *Exploiting the structure of unsatisfiable cores in MaxSAT*. In Qiang Yang and Michael Wooldridge (editors): *Proc. IJCAI*, pages 283–289. AAAI Press, 2015.

[6] Carlos Ansótegui and Joel Gabàs: *Solving (weighted) partial MaxSAT with ILP*. In Carla P. Gomes and Meinolf Sellmann (editors): *Proc. CPAIOR*, volume 7874 of *Lecture Notes in Computer Science*, pages 403–409. Springer, 2013.

[7] Josep Argelich, Chu Min Li, Felip Manyá, and Jordi Planes: *MaxSAT evaluation 2016*. `http://maxsat.ia.udl.cat/introduction/`. [Online; accessed 18-10-2016].

[8] Gilles Audemard and Laurent Simon: *Predicting learnt clauses quality in modern SAT solvers*. In Craig Boutilier (editor): *Proc. IJCAI*, pages 399–404. AAAI Press / IJCAI, 2009.

[9] Gilles Audemard and Laurent Simon: *GLUCOSE 2.1: Aggressive—but reactive—clause database management, dynamic restarts*. In *Pragmatics of SAT (Workshop of SAT'12)*, 2012.

[10] Gilles Audemard and Laurent Simon: *Refining restarts strategies for SAT and UNSAT*. In Michela Milano (editor): *Proc. CP*, volume 7514 of *Lecture Notes in Computer Science*, pages 118–126. Springer, 2012.

[11] Olivier Bailleux and Yacine Boufkhad: *Efficient CNF encoding of Boolean cardinality constraints*. In Francesca Rossi (editor): *Proc. CP*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2003.

[12] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin: *An introduction to argumentation semantics*. Knowledge Engineering Review, 26(4):365–410, 2011.

[13] Pietro Baroni and Massimiliano Giacomin: *Solving semantic problems with odd-length cycles in argumentation*. In Thomas D. Nielsen and Nevin Lianwen Zhang (editors): *Proc. ECSQARU*, volume 2711 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2003.

[14] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida: *SCC-recursiveness: a general schema for argumentation semantics*. Artificial Intelligence, 168(1-2):162–210, 2005.

[15] Ringo Baumann: *Normal and strong expansion equivalence for argumentation frameworks*. Artificial Intelligence, 193:18–44, 2012.

[16] Ringo Baumann: *What does it take to enforce an argument? Minimal change in abstract argumentation.* In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas (editors): *Proc. ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 127–132. IOS Press, 2012.

[17] Ringo Baumann and Gerhard Brewka: *Expanding argumentation frameworks: Enforcing and monotonicity results.* In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari (editors): *Proc. COMMA*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 75–86. IOS Press, 2010.

[18] Ringo Baumann and Gerhard Brewka: *AGM meets abstract argumentation: Expansion and revision for Dung frameworks.* In Qiang Yang and Michael Wooldridge (editors): *Proc. IJCAI*, pages 2734–2740. AAAI Press, 2015.

[19] Ringo Baumann, Gerhard Brewka, and Renata Wong: *Splitting argumentation frameworks: An empirical evaluation.* In Sanjay Modgil, Nir Oren, and Francesca Toni (editors): *TAFA Workshop 2011, Revised Selected Papers*, volume 7132 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2012.

[20] Trevor J. M. Bench-Capon, Henry Prakken, and Giovanni Sartor: *Argumentation in legal reasoning.* In Guillermo Simari and Iyad Rahwan (editors): *Argumentation in Artificial Intelligence*, pages 363–382. Springer, 2009.

[21] Jeremias Berg and Matti Järvisalo: *Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability.* Artificial Intelligence, in press.

[22] Jeremias Berg, Matti Järvisalo, and Brandon Malone: *Learning optimal bounded treewidth Bayesian networks via maximum satisfiability.* In Samuel Kaski and Jukka Corander (editors): *Proc. AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 86–95. JMLR.org, 2014.

[23] Philippe Besnard and Sylvie Doutre: *Checking the acceptability of a set of arguments.* In James P. Delgrande and Torsten Schaub (editors): *Proc. NMR*, pages 59–64, 2004.

[24] Philippe Besnard and Anthony Hunter: *Elements of Argumentation.* MIT Press, 2008.

[25] Armin Biere: *Bounded model checking.* In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (editors): *Handbook of Satisfiabil-*

ity, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 14, pages 457–481. IOS Press, 2009.

[26] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (editors): *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[27] Armin Biere and Wolfgang Kunz: *SAT and ATPG: Boolean engines for formal hardware verification.* In Lawrence T. Pileggi and Andreas Kuehlmann (editors): *Proc. ICCAD*, pages 782–785. ACM / IEEE Computer Society, 2002.

[28] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex: *Enforcement in argumentation is a kind of update.* In Weiru Liu, V. S. Subrahmanian, and Jef Wijsen (editors): *Proc. SUM*, volume 8078 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2013.

[29] Stefano Bistarelli and Francesco Santini: *ConArg: A constraint-based computational framework for argumentation systems.* In Taghi M. Khoshgoftaar and Xingquan (Hill) Zhu (editors): *Proc. ICTAI*, pages 605–612. IEEE Computer Society, 2011.

[30] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni: *An abstract, argumentation-theoretic approach to default reasoning.* Artificial Intelligence, 93:63–101, 1997.

[31] Martin W. A. Caminada, Walter Alexandre Carnielli, and Paul E. Dunne: *Semi-stable semantics.* Journal of Logic and Computation, 22(5):1207–1254, 2012.

[32] Federico Cerutti, Paul E. Dunne, Massimiliano Giacomin, and Mauro Vallati: *Computing preferred extensions in abstract argumentation: A SAT-based approach.* In Elizabeth Black, Sanjay Modgil, and Nir Oren (editors): *TAFA Workshop 2013, Revised Selected Papers*, volume 8306 of *Lecture Notes in Computer Science*, pages 176–193. Springer, 2014.

[33] Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati: *ArgSemSAT: Solving argumentation problems using SAT.* In Simon Parsons, Nir Oren, Chris Reed, and Federico Cerutti (editors): *Proc. COMMA*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 455–456. IOS Press, 2014.

[34] Günther Charwat, Wolfgang Dvořák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran: *Methods for solving reasoning problems in abstract argumentation - A survey.* Artificial Intelligence, 220:28–63, 2015.

[35] Yibin Chen, Sean Safarpour, João Marques-Silva, and Andreas G. Veneris: *Automated design debugging with maximum satisfiability.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 29(11):1804–1817, 2010.

[36] Yibin Chen, Sean Safarpour, Andreas G. Veneris, and João P. Marques-Silva: *Spatial and temporal design debug using partial MaxSAT.* In Fabrizio Lombardi, Sanjukta Bhanja, Yehia Massoud, and R. Iris Bahar (editors): *Proc. ACM Great Lakes Symposium on VLSI*, pages 345–350. ACM, 2009.

[37] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith: *Counterexample-guided abstraction refinement for symbolic model checking.* Journal of the ACM, 50(5):752–794, 2003.

[38] Edmund M. Clarke, Anubhav Gupta, and Ofer Strichman: *SAT-based counterexample-guided abstraction refinement.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 23(7):1113–1123, 2004.

[39] Stephen A. Cook: *The complexity of theorem-proving procedures.* In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman (editors): *Proc. STOC*, pages 151–158. ACM, 1971.

[40] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis: *Symmetric argumentation frameworks.* In Lluis Godo (editor): *Proc. ECSQARU*, volume 3571 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2005.

[41] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis: *On the revision of argumentation systems: Minimal change of arguments statuses.* In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter (editors): *Proc. KR*, pages 52–61. AAAI Press, 2014.

[42] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis: *A translation-based approach for revision of argumentation frameworks.* In Eduardo Fermé and João Leite (editors): *Proc. JELIA*, volume 8761 of *Lecture Notes in Computer Science*, pages 397–411. Springer, 2014.

[43] Sylvie Coste-Marquis, Sébastien Konieczny, Jean Guy Mailly, and Pierre Marquis: *Extension enforcement in abstract argumentation as an optimization problem.* In Qiang Yang and Michael Wooldridge (editors): *Proc. IJCAI*, pages 2876–2882. AAAI Press, 2015.

[44] Jessica Davies and Fahiem Bacchus: *Exploiting the power of MIP solvers in MAXSAT.* In Matti Järvisalo and Allen Van Gelder (ed-

itors): *Proc. SAT*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2013.

[45] Jérôme Delobelle, Sébastien Konieczny, and Srdjan Vesic: *On the aggregation of argumentation frameworks.* In Qiang Yang and Michael Wooldridge (editors): *Proc. IJCAI*, pages 2911–2917. AAAI Press, 2015.

[46] Martin Diller, Adrian Haret, Thomas Linsbichler, Stefan Rümmele, and Stefan Woltran: *An extension-based approach to belief revision in abstract argumentation.* In Qiang Yang and Michael Wooldridge (editors): *Proc. IJCAI*, pages 2926–2932. AAAI Press, 2015.

[47] Yannis Dimopoulos and Alberto Torres: *Graph theoretical structures in logic programs and default theories.* Theoretical Computer Science, 170(1-2):209–244, 1996.

[48] Alastair F. Donaldson, Alexander Kaiser, Daniel Kroening, Michael Tautschnig, and Thomas Wahl: *Counterexample-guided abstraction refinement for symmetric concurrent programs.* Formal Methods in System Design, 41(1):25–44, 2012.

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

[50] Phan Minh Dung, Paolo Mancarella, and Francesca Toni: *Computing ideal sceptical argumentation.* Artificial Intelligence, 171(10-15):642–674, 2007.

[51] Paul E. Dunne and Martin Caminada: *Computational complexity of semi-stable semantics in abstract argumentation frameworks.* In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing (editors): *Proc. JELIA*, volume 5293 of *Lecture Notes in Computer Science*, pages 153–165. Springer, 2008.

[52] Paul E. Dunne, Anthony Hunter, Peter McBurney, Simon Parsons, and Michael Wooldridge: *Weighted argument systems: Basic definitions, algorithms, and complexity results.* Artificial Intelligence, 175(2):457–486, 2011.

[53] Paul E. Dunne and Michael Wooldridge: *Complexity of abstract argumentation.* In Guillermo Ricardo Simari and Iyad Rahwan (editors): *Argumentation in Artificial Intelligence*, pages 85–104. Springer, 2009.

[54] Wolfgang Dvořák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran: *Making use of advances in answer-set programming for abstract argumentation systems.* In Hans Tompits, Salvador Abreu,

Johannes Oetsch, Jörg Pührer, Dietmar Seipel, Masanobu Umeda, and Armin Wolf (editors): *Proc. INAP 2011 and Proc. WLP 2011, Revised Selected Papers*, volume 7773 of *Lecture Notes in Computer Science*, pages 114–133. Springer, 2013.

[55] Wolfgang Dvořák and Stefan Woltran: *Complexity of semi-stable and stage semantics in argumentation frameworks.* Information Processing Letters, 110(11):425–430, 2010.

[56] Wolfgang Dvořák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran: *Complexity-sensitive decision procedures for abstract argumentation.* Artificial Intelligence, 206:53–78, 2014.

[57] Niklas Eén and Niklas Sörensson: *An extensible SAT-solver.* In Enrico Giunchiglia and Armando Tacchella (editors): *Proc. SAT 2003*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.

[58] Niklas Eén and Niklas Sörensson: *Translating pseudo-Boolean constraints into SAT.* Journal on Satisfiability, Boolean Modeling and Computation, 2(1-4):1–26, 2006.

[59] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran: *Answer-set programming encodings for argumentation frameworks.* Argument & Computation, 1(2):147–177, 2010.

[60] Wolfgang Faber, Mauro Vallati, Federico Cerutti, and Massimiliano Giacomin: *Solving set optimization problems by cardinality optimization with an application to argumentation.* In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen (editors): *Proc. ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 966–973. IOS Press, 2016.

[61] Sarah Alice Gaggl, Norbert Manthey, Alessandro Ronca, Johannes Peter Wallner, and Stefan Woltran: *Improved answer-set programming encodings for abstract argumentation.* Theory and Practice of Logic Programming, 15(4-5):434–448, 2015.

[62] João Guerra and Inês Lynce: *Reasoning over biological networks using maximum satisfiability.* In Michela Milano (editor): *Proc. CP*, volume 7514 of *Lecture Notes in Computer Science*, pages 941–956. Springer, 2012.

[63] Gurobi: *Gurobi Optimizer 7.0.* `http://www.gurobi.com/`. [Online; accessed 18-10-2016].

[64] Marijn Heule, Matti Järvisalo, and Tomas Balyo: *SAT competition 2016*. `http://baldur.iti.kit.edu/sat-competition-2016/`. [Online; accessed 18-10-2016].

[65] IBM: *ILOG CPLEX Optimizer*. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`. [Online; accessed 18-10-2016].

[66] Franjo Ivancic, Zijiang Yang, Malay K. Ganai, Aarti Gupta, and Pranav Ashar: *Efficient SAT-based bounded model checking for software verification*. Theoretical Computer Science, 404(3):256–274, 2008.

[67] Mikolás Janota, Radu Grigore, and João Marques-Silva: *Counterexample guided abstraction refinement algorithm for propositional circumscription*. In Tomi Janhunen and Ilkka Niemelä (editors): *Proc. JELIA*, volume 6341 of *Lecture Notes in Computer Science*, pages 195–207. Springer, 2010.

[68] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon: *The international SAT solver competitions*. AI Magazine, 33(1), 2012.

[69] Henry A. Kautz and Bart Selman: *Planning as satisfiability*. In Bernd Neumann (editor): *Proc. ECAI*, pages 359–363. Wiley, 1992.

[70] João P. Marques-Silva, Inês Lynce, and Sharad Malik: *Conflict-driven clause learning SAT solvers*. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (editors): *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 4, pages 131–153. IOS Press, 2009.

[71] Ruben Martins, Vasco M. Manquinho, and Inês Lynce: *Open-WBO: A modular MaxSAT solver*. In Carsten Sinz and Uwe Egly (editors): *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014.

[72] Peter McBurney, Simon Parsons, and Iyad Rahwan (editors): *ArgMAS 2011 Revised Selected Papers*, volume 7543 of *Lecture Notes in Computer Science*. Springer, 2012.

[73] Antonio Morgado, Alexey Ignatiev, and João Marques-Silva: *MSCG: Robust core-guided MaxSAT solving*. Journal on Satisfiability, Boolean Modeling and Computation, 9:129–134, 2015.

[74] Andreas Niskanen, Johannes Peter Wallner, and Matti Järvisalo: *Pakota: A system for enforcement in abstract argumentation*. In Loizos Michael and Antonis C. Kakas (editors): *Proc. JELIA*, volume 10021 of *Lecture Notes in Computer Science*, pages 385–400. Springer.

[75] Andreas Niskanen, Johannes Peter Wallner, and Matti Järvisalo: *Optimal status enforcement in abstract argumentation.* In Subbarao Kambhampati (editor): *Proc. IJCAI*, pages 1216–1222. IJCAI/AAAI Press, 2016.

[76] Andreas Niskanen, Johannes Peter Wallner, and Matti Järvisalo: *Synthesizing argumentation frameworks from examples.* In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen (editors): *Proc. ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 551–559. IOS Press, 2016.

[77] Samer Nofal, Katie Atkinson, and Paul E. Dunne: *Algorithms for decision problems in argument systems under preferred semantics.* Artificial Intelligence, 207:23–51, 2014.

[78] Samer Nofal, Paul E. Dunne, and Katie Atkinson: *On preferred extension enumeration in abstract argumentation.* In Bart Verheij, Stefan Szeider, and Stefan Woltran (editors): *Proc. COMMA*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, pages 205–216. IOS Press, 2012.

[79] Christos H. Papadimitriou: *Computational complexity.* Addison-Wesley, 1994.

[80] James D. Park: *Using weighted MAX-SAT engines to solve MPE.* In Rina Dechter and Richard S. Sutton (editors): *Proc. AAAI*, pages 682–687. AAAI Press / The MIT Press, 2002.

[81] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä: *Planning as satisfiability: parallel plans and algorithms for plan search.* Artificial Intelligence, 170(12-13):1031–1080, 2006.

[82] Paul Saikko, Jeremias Berg, and Matti Järvisalo: *LMHS: A SAT-IP hybrid MaxSAT solver.* In Nadia Creignou and Daniel Le Berre (editors): *Proc. SAT*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer, 2016.

[83] Florence Dupin de Saint-Cyr, Pierre Bisquert, Claudette Cayrol, and Marie-Christine Lagasquie-Schiex: *Argumentation update in YALLA (Yet Another Logic Language for Argumentation).* International Journal of Approximate Reasoning, 75:57–92, 2016.

[84] Bart Selman, Hector J. Levesque, and David G. Mitchell: *A new method for solving hard satisfiability problems.* In William R. Swartout (editor): *Proc. AAAI*, pages 440–446. AAAI Press / The MIT Press, 1992.

[85] Carsten Sinz: *Towards an optimal CNF encoding of Boolean cardinality constraints.* In Peter van Beek (editor): *Proc. CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005.

[86] Larry J. Stockmeyer: *The polynomial-time hierarchy.* Theoretical Computer Science, 3(1):1–22, 1976.

[87] Matthias Thimm, Serena Villata, Federico Cerutti, Nir Oren, Hannes Strass, and Mauro Vallati: *Summary report of the First International Competition on Computational Models of Argumentation.* AI Magazine, 37(1):102–104, 2016.

[88] Gregory S. Tseitin: *On the complexity of derivation in propositional calculus.* In *Automation of Reasoning 2: Classical Papers on Computational Logic 1967–1970*, pages 466–483. Springer, 1983. Originally in Russian: [91].

[89] Bart Verheij: *Two approaches to dialectical argumentation: Admissible sets and argumentation stages.* In John Jules Meyer and Linda van der Gaag (editors): *Proc. NAIC*, pages 357–368. Utrecht University, 1996.

[90] Johannes Peter Wallner, Andreas Niskanen, and Matti Järvisalo: *Complexity results and algorithms for extension enforcement in abstract argumentation.* In Dale Schuurmans and Michael Wellman (editors): *Proc. AAAI*, pages 1088–1094. AAAI Press, 2016.

[91] Григорий С. Цейтин: О сложности вывода в исчислении высказываний. В Исследования по конструктивной математике и математической логике. II, том 8 из Записки научных семинаров ЛОМИ, страницы 234–259. 1968.