ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200174

Strong Refinements for Hard Problems in Argumentation Dynamics

Andreas Niskanen and **Matti Järvisalo** [1]

Abstract. Going beyond the more classically studied reasoning problems over argumentation frameworks (AFs), the study of dynamics in argumentation gives rise to new types of computational challenges. This work studies ways of extending the scalability of computational approaches to reasoning about dynamics of abstract argumentation frameworks. In particular, we focus on three recently proposed optimization problems underlying AF dynamics—two variants of enforcement in abstract argumentation and the synthesis of argumentation frameworks from examples—for semantics under which the problems are (presumably) complete for the second level of the polynomial hierarchy. As the main contributions, we show that by bridging recent theoretical results on the persistence of extensions under changes to the structure of AFs with Boolean satisfiability (SAT) counterexample-guided abstraction refinement algorithms for the considered problems, the scalability of state-of-the-art practical algorithms for each of the three problems can be significantly improved.

1 INTRODUCTION

The study of representational and computational aspects of argumentation has become a core topic in artificial intelligence research [1]. Dung's argumentation frameworks (AFs) [30], taking the form of directed graphs in which nodes represent abstract arguments and edges attacks between arguments, provide a central formal model for argumentation in AI via extension-based semantics [30, 4], expressed as sets of jointly acceptable arguments.

The development of computational approaches to various fundamental reasoning tasks underlying aspects of argumentation is a major research direction with AI argumentation. A main focus in this line of research has until recently been on developing practical systems for efficiently solving various reasoning tasks over a given fixed argumentation framework [13]—in particular, skeptical and credulous acceptance of prescribed arguments under different AF semantics. However, argumentation is intrinsically a dynamic process. Motivated by this, computational approaches to reasoning about dynamic aspects of AFs have recently received attention from various different viewpoints, such as expansions [7, 5, 8], revision [12, 18, 19, 43, 8, 25], enforcement [7, 6, 10, 20, 46, 28], update [10, 26, 21, 27], aggregation [17, 32, 23, 22, 24, 14] (see [11] for a survey on aggregation issues), synthesis [42], among other approaches [29].

Similarly as in the case of static reasoning tasks such as acceptance of arguments in a fixed AF [33], the various types of reasoning problems underlying argumentation dynamics are often computationally challenging, surpassing in complexity the first level of the polynomial hierarchy for specific semantics such as the central preferred semantics [46, 40, 42]. As dynamic problems naturally give rise to optimization problems—in contrast to "mere" decision problems as in the case of deciding acceptance of arguments—the problem variants with beyond-NP complexity pose great challenges from the perspective of developing scalable practical algorithms.

In this work we focus on improving the scalability of current state-of-the-art declarative approaches to several types of problems arising from the study of AF dynamics. In terms of the computational problems, we focus of two different forms of enforcement problems—namely, extension enforcement [7, 20, 46, 41] and status enforcement [40]—and AF synthesis [42]. In enforcement problems, the goal is to structurally adjust a given AF in a smallest possible way so that a given subset of its arguments becomes an extension (extension enforcement) or so that the AF will support given positive and negative acceptance statuses of its arguments (status enforcement). On the other hand, generalizing the notion of realizability [31], AF synthesis deals with finding (or synthesizing) an AF that is semantically closest to a given set of potentially conflicting positive and negative examples of extensions and non-extensions.

For each of the problems, the current state-of-the-art practical algorithms are based on harnessing Boolean satisfiability (SAT) solvers and their optimization-extensions, i.e., maximum satisfiability (MaxSAT) solvers, for finding optimal solutions to the task at hand [46, 40, 42]. Common to each of the problems is that, under specific central AF semantics and reasoning modes, the problems are complete for the second level of the polynomial hierarchy, which rules out direct compact encodings of the problems in terms of (Max)SAT. Indeed, the current state-of-the-art approaches to the second-level variants of these problems are based on the general principle of SAT-based counterexample-guided abstraction refinement (CEGAR) [15, 16]. The CEGAR approaches work iteratively by optimally solving an over-abstraction of the problem via considering a different, easier (in particular NP-complete) variant of the problem, and using a SAT solver to check for possible counterexamples for the found solution candidates, until a provably optimal solution to the original second-level complete problem variant is found.

Central to scaling up SAT-based CEGAR for such beyond-NP AF dynamics problems is the ability to strongly refine the NP-abstraction during the iterations of the algorithms. In this respect, the current state-of-the-art CEGAR approaches to extension enforcement, status enforcement, and AF synthesis fall somewhat short.

As the main contributions of this work, we show that recent results on the persistence of labelings under adding and removing attacks in AFs [44] yield a uniform way of noticeably improving the scalability of CEGAR algorithms to extension enforcement, status

[1] HIIT, Department of Computer Science, University of Helsinki, Finland, email: first.last@helsinki.fi

enforcement, and AF synthesis. In particular, bringing the theoretical results on the persistence of extensions under updates to AFs into the practical algorithmic realm by applying the theoretical results to obtain noticeably stronger refinements in the CEGAR approaches, we show that significantly larger sets of solution candidates can be ruled out at each iteration of the CEGAR approaches based on a single counterexample. This results in obtaining significant improvements in terms of running times and scalability to larger instance sizes over the current state of the art in practical algorithms to second-level complete variants of extension enforcement, status enforcement, and AF synthesis.

The rest of the paper is organized as follows. After necessary background on argumentation frameworks (Section 2) and the extension enforcement, status enforcement, and AF synthesis problems (Section 3), we describe in a uniform way the SAT-based CEGAR approaches for the three problems (Section 4). We then explain how the more naive refinements applied in the current CEGAR implementations can be generalized to obtain noticeably stronger refinement steps (Section 5) and show that the strong refinements yield significant scalability improvements (Section 6).

2 ABSTRACT ARGUMENTATION

We start by recalling argumentation frameworks [30] and the argumentation semantics [3, 4]) used in this work.

Definition 1. *An* argumentation framework (AF) *is a pair $F = (A, R)$, where A is a finite, non-empty set of arguments and $R \subseteq A \times A$ an attack relation. An arguments a attacks b if $(a, b) \in R$. 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 is a $c \in S$ such that $(c, b) \in R$.*

Example 1. *Let $F = (A, R)$ be an AF with the set of arguments $A = \{a, b, c, d, e\}$ and the attack relation $R = \{(a, b), (b, a), (c, b), (c, d), (d, e), (e, c)\}$. The AF F is represented as a directed graph in Figure 1.*

Semantics for AFs are functions σ which map each AF $F = (A, R)$ to a set $\sigma(F) \subseteq 2^A$ of extensions. We consider $\sigma \in \{cf, adm, com, prf, stb\}$, which stand for conflict-free, admissible, complete, preferred, and stable, respectively.

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 $S^+_R = S \cup \{x \in A \mid (y, x) \in R, y \in S\}$.*

The semantics we consider can now be defined as follows.

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 $S = \mathcal{F}_F(S)$,*
- *$S \in prf(F)$ iff $S \in adm(F)$ and $\nexists S' \in adm(F)$ with $S \subset S'$,*
- *$S \in stb(F)$ iff $S^+_R = A$.*

If $S \in \sigma(F)$, then S is called a σ-extension.

For any AF F, the semantics satisfy the subset relations $cf(F) \supseteq adm(F) \supseteq com(F) \supseteq prf(F) \supseteq stb(F)$.

Example 2. *For the AF F in Example 1, it holds that $adm(F) = com(F) = \{\emptyset, \{a\}\}$, $prf(F) = \{\{a\}\}$, and $stb(F) = \emptyset$.*

3 DYNAMIC PROBLEMS

We focus on the optimization problems underlying (argument-fixed) extension enforcement [20, 46], status enforcement [40], and argumentation framework synthesis [42], and in particular, variants of these problems that are (presumably) hard for the second level of the polynomial hierarchy. Briefly put, extension and status enforcement aim to update a given input AF in order to reach a goal while minimizing the number of changes, and the goal of AF synthesis is to construct an AF that optimally represents given sets of potentially conflicting examples as extensions.

3.1 Extension Enforcement

Strict extension enforcement aims to modify the attack structure of an AF in order to make a subset of arguments provided as input a σ-extension, while the non-strict version requires only that the input set is a subset of a σ-extension [20]. Whereas the non-strict version is NP-complete for several AF semantics, the strict version under preferred semantics is complete for the second level of the polynomial hierarchy, in particular, Σ^{P}_2-complete [46]. We focus on the latter, computationally more challenging problem variant.

Given an AF $F = (A, R)$ and a subset $T \subseteq A$ of arguments, the task is to modify the attack structure R such that T becomes a preferred extension. Denote by

$$strict(F, T, prf) = \{R' \mid F' = (A, R'),\ T \in prf(F')\}$$

the solution set of attack structures. Further, denote the number of changes between two attack structures R and R' as $|R \Delta R'| = |R \setminus R'| + |R' \setminus R|$. Strict extension enforcement is then the optimization problem of minimizing the number of changes to the input attack structure:

Strict Extension Enforcement under Preferred Semantics
INPUT: AF $F = (A, R)$, $T \subseteq A$.
TASK: Find an AF $F^* = (A, R^*)$ with

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

Example 3. *Consider again the AF in Example 1. Suppose we want to enforce $T = \{d\}$ strictly under preferred semantics. Clearly some changes to the attack structure are needed, as the unique preferred extension is $\{a\}$. Removing the attack (c, d) results in an AF in which T is admissible. However, the resulting AF also has an admissible superset $\{a, c, d\}$ since $\{a\}$ is still admissible, d now defends c against the attack from e, and there are no conflicts between a, c, d, so this does not suffice. Further removing the attack (a, b) and adding a self-attack (c, c) results in the AF $F' = (A, R')$ with $R' = \{(b, a), (c, b), (c, c), (d, e), (e, c)\}$ in which $\{d\}$ is the unique non-empty admissible extension and hence the unique preferred extension. One can verify that this is indeed an optimal solution, i.e., one needs at least three changes in order to enforce T under preferred.*

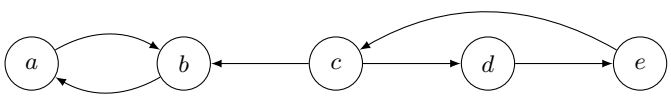

Figure 1. Argumentation framework from Example 1.

3.2 Status Enforcement

Status enforcement problem aims to modify an input AF with the goal that certain input arguments are—either credulously or skeptically–accepted, i.e., enforced *positively*, and others are not, that is, enforced *negatively* [40]. While credulous status enforcement is NP-complete for several AF semantics when there are no arguments that are enforced negatively, the credulous status enforcement problem (without restrictions) is complete for the second level, and so is skeptical status enforcement under stable semantics (even without negatively enforced arguments) as well [40]. Again, we will focus on these computationally more challenging problem variants.

Formally, in the credulous (resp. skeptical) status enforcement problem we are given an initial AF $F = (A, R)$, with two disjoint subsets of arguments $P, N \subseteq A$, $P \cap N = \emptyset$. The task is now to modify the attack structure R in such a way that all arguments in P are credulously (resp. skeptically) accepted and all arguments in N are not (resp. skeptically) accepted under σ. For *credulous status enforcement*, denote the set of solution attack structures

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

as $cred(F, P, N, \sigma)$, and for *skeptical status enforcement* by $skept(F, P, N, \sigma)$, defined as

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

We set the additional constraint for skeptical status enforcement that a solution AF F' has at least one extension, as otherwise by definition all arguments are skeptically accepted. Again, status enforcement is considered as an optimization problem, where the goal is to minimize the number of changes to the original attack structure.

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

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

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

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

More formally, skeptical status enforcement under stable is Σ_2^{P}-complete even if $N = \emptyset$. Furthermore, for an arbitrary N, credulous status enforcement under admissible and stable is also Σ_2^{P}-complete [40].

Example 4. *Consider again the AF in Example 1, with a unique non-empty admissible extension $\{a\}$. Suppose we want to enforce $P = \{d\}$ and $N = \{a\}$ credulously under admissible, i.e., we want d to be included in at least one admissible extension, and a not to appear in any of them. Removing the attack (c, d) results in an AF with $\{d\}$ as an admissible extension. However, the resulting AF also has $\{a\}$ as an admissible extension, so this is not a valid solution. Adding or removing any other attacks does not result in valid solutions, either. However, if we additionally add the self-attack (a, a), resulting in the AF $F = (A, R')$ with $R' = \{(a, a), (a, b), (b, a), (c, b), (d, e), (e, c)\}$, a is no longer in any admissible set and $\{d\}$ is still admissible. One can verify that this solution is optimal.*

3.3 AF Synthesis

Realizability of AFs is the problem of deciding whether there exists an AF the extensions of which is exactly the collection of subsets of arguments given as input [31]. AF synthesis generalizes realizability by relaxing the requirement of exact knowledge of extensions and non-extensions, aiming to construct an AF which is semantically closest to the knowledge provided in the form of weighted examples [46]. Again, under various central AF semantics, such as complete and stable, AF synthesis is in NP. However, AF synthesis under preferred semantics is conjectured to be Σ_2^{P}-complete and is also noticeably challenging to solve in practice [46].

Formally, in the AF synthesis problem, we are given a non-empty set of arguments A, and two sets of *positive* and *negative* examples E^+ and E^- over A. An *example* (over A) is a pair (S, w) with $S \subseteq A$ and $w > 0$ an integer representing the weight of the example. An AF synthesis instance is of the form $P = (A, E^+, E^-, \sigma)$ with σ an AF semantics. For a fixed example $e = (S, w)$, denote $S_e = S$ and $w_e = w$. Given an AF $F = (A, R)$, a positive (resp. negative) example e is *satisfied* if $S_e \in \sigma(F)$ (resp. $S_e \notin \sigma(F)$). The *cost* of an AF F is

$$cost(P, F) = \sum_{e \in E^+} w_e \cdot I(S_e \notin \sigma(F)) + \sum_{e \in E^-} w_e \cdot I(S_e \in \sigma(F)),$$

i.e., the sum of the weights of the examples that are not satisfied by F. The goal of the AF synthesis problem is to find an AF of minimum cost, i.e., that optimally represents the examples given as input.

AF Synthesis
INPUT: $P = (A, E^+, E^-, \sigma)$
TASK: Find an AF F^* with

$$F^* \in \operatorname*{arg\,min}_{F=(A,R^*)} (cost(P, F)).$$

Example 5. *Consider the AF synthesis instance $P = (A, E^+, E^-, prf)$ with $A = \{a, b, c, d, e\}$, $E^+ = \{(\{a, b\}, 1), (\{b, c\}, 1)\}$, and $E^- = \{(\{d, e\}, 1)\}$. In this case, it is possible to construct an optimal solution AF with zero cost, namely, via $R = \{(a, c), (c, a), (d, d), (e, e)\}$. The preferred extensions of the AF $F = (A, R)$ are exactly the sets of the positive examples $\{a, b\}$ and $\{b, c\}$.*

4 COUNTEREXAMPLE-GUIDED ABSTRACTION REFINEMENT

Counterexample-guided abstraction refinement (CEGAR), originally proposed in the context of model checking [15, 16], is a general framework suitable for iteratively solving hard decision and optimization problems. The idea is to start with an abstraction, i.e., an overapproximation of the problem, of lesser computational complexity. At each iteration, a solution to the currrent abstraction is obtained, and checked whether the solution to the abstraction is an actual solution to the original problem. If it is not, a counterexample witnessing this fact is obtained, and based on the counterexample the abstraction is refined by excluding at least the solution candidate from further consideration. Finally, a solution to the abstraction which has no counterexamples will correspond to a solution of the original problem, or, in case the abstraction is determined to have no more solutions, it follows that there are no solutions to the actual instance being solved. A critical aspect in constructing practically efficient CEGAR algorithms is the development of *strong refinements*, i.e., ideas for

ruling out not only the latest solution candidate, but also many other ones from further consideration, based on a single counterexample to the latest solution candidate in order to limit the number of iterations and avoid bloating up the size of the abstraction unnecessarily. This is also our focus in the context of the three AF dynamics problems.

Employing CEGAR has been shown to be effective in various domains, including nonmonotonic reasoning [37], QBF solving [38], classical planning [45], and, more importantly, for solving static [34] and dynamic [46, 42] reasoning problems in abstract argumentation. Here we focus on MaxSAT-based CEGAR algorithms for problems hard for the second level, where the abstraction is solved using a MaxSAT solver, and the search for a counterexample employs a SAT solver. This means that our abstractions must be problems in NP in order to obtain succinct encodings.

For background on MaxSAT, recall that for a Boolean variable x, there are two literals, x and $\neg x$. A clause is a disjunction ($\vee$) of literals. A truth assignment τ is a function from variables to true (1) and false (0). Satisfaction is defined as usual. A weighted partial MaxSAT (or simply MaxSAT) instance consists of hard clauses φ_h, soft clauses φ_s, and a weight function w associating to each soft clause $C \in \varphi_s$ a positive weight $w(C)$. An assignment τ is a solution to a MaxSAT instance $(\varphi_h, \varphi_s, w)$ if τ satisfies φ_h. The cost of τ, $c(\tau)$, is the sum of weights of the soft clauses not satisfied by τ. A solution τ to MaxSAT instance φ is optimal if $c(\tau) \leq c(\tau')$ for any solution τ' to φ.

4.1 CEGAR for Enforcement

We start by describing the CEGAR approaches to extension enforcement [46] and status enforcement [40] in a uniform way. The CEGAR algorithm proceeds as follows (outlined in pseudocode as Algorithm 1). Suppose $M \in \{\mathit{strict}, \mathit{cred}, \mathit{skept}\}$, an AF $F = (A, R)$, and a semantics σ are given as input, along with a set $S = T$ in the case of strict extension enforcement ($M = \mathit{strict}$) or a pair of sets $S = (P, N)$ in the case of status enforcement ($M = \mathit{cred}, \mathit{skept}$).

For extension and status enforcement, changes to the attack structure are encoded using variables $r_{a,b}$ for each $a, b \in A$, with the interpretation that $(a, b) \in R^*$ for an optimal attack structure R^* iff $\tau(r_{a,b}) = 1$ for an optimal truth assignment τ. Now, using soft clauses (line 2)

$$\textsc{NoChange}(F) = \bigwedge_{(a,b)\in R} r_{a,b} \wedge \bigwedge_{(a,b)\in (A\times A)\setminus R} \neg r_{a,b}$$

with unit weights, the number of changes to the original attack structure is minimized.

The NP-abstractions $\textsc{Abstraction}(M, F, S, \sigma)$ are encoded using hard clauses (line 1). For strict extension enforcement under preferred, a suitable abstraction is to instead enforce a complete extension [46]. For credulous status enforcement under $\sigma \in \{adm, stb\}$, the complexity drops to NP for $N = \emptyset$, and hence as an abstraction only the arguments in P are enforced, and for skeptical status enforcement under $\sigma = stb$, likewise only the arguments in N are enforced [40].

A MaxSAT solver is called iteratively (line 4) in order to obtain a candidate attack structure from the optimal truth assignment τ via $R_\tau = \{(a, b) \in A \times A \mid \tau(r_{a,b}) = 1\}$ (line 5). The validity of this solution is checked by asking a SAT solver for a counterexample (line 6), encoded as a Boolean formula $\textsc{counterexample}(F_\tau, S, \sigma)$ utilizing the standard SAT encodings of argumentation semantics [9]. For strict extension enforcement under preferred, a counterexample is an admissible extension of F_τ that is a superset of the set T to be enforced [46]. On the other hand, for credulous status enforcement, a counterexample is a σ-extension that contains an argument in N, and for skeptical status enforcement, a σ-extension that does not contain an argument in P [40].

If a counterexample is found, according to [46, 40] the abstraction is refined by adding the clause

$$\textsc{refine}(F_\tau) = \bigvee_{(a,b)\in (A\times A)\setminus R_\tau} r_{a,b} \vee \bigvee_{(a,b)\in R_\tau} \neg r_{a,b} \qquad (1)$$

which simply excludes the current attack structure (line 8). In this work, we refer to this clause as the *trivial refinement*.

Algorithm 1 CEGAR for extension and status enforcement. Input: $M \in \{\mathit{strict}, \mathit{cred}, \mathit{skept}\}$, AF $F = (A, R)$, $S = T$ if $M = \mathit{strict}$ and $S = (P, N)$ otherwise, semantics σ.

1: $\varphi_h \leftarrow \textsc{Abstraction}(M, F, S, \sigma)$
2: $\varphi_s \leftarrow \textsc{NoChange}(F)$
3: **while** true **do**
4: $\quad(c, \tau) \leftarrow \textsc{MaxSAT}(\varphi_h, \varphi_s)$
5: $\quad F_\tau \leftarrow (A, R_\tau)$
6: $\quad(r, \tau') \leftarrow \textsc{SAT}(\textsc{counterexample}(F_\tau, S, \sigma))$
7: $\quad$**if** $r = \mathit{unsat}$ **then** return F_τ
8: $\quad$**else** $\varphi_h \leftarrow \varphi_h \wedge \textsc{refine}(F_\tau)$

4.2 CEGAR for AF Synthesis

In the CEGAR approach to AF synthesis under preferred semantics [42], the output attack structure is likewise encoded using variables $r_{a,b}$ for each $a, b \in A$, additionally making use of variables Ext_e for each example $e \in E^+ \cup E^-$, with the interpretation that $S_e \in \sigma(F)$ for an optimal AF F iff $\tau(\text{Ext}_e) = 1$ for an optimal truth assignment τ. The presumed second-level hardness of AF synthesis under preferred is due to positive examples, since negative examples can be encoded succintly in SAT, as described in [42]. In terms of positive examples, the abstraction is formed using the complete semantics via hard clauses of the form $\text{Ext}_e \rightarrow \varphi_{com}(S_e)$ for each $e \in E^+$, where $\varphi_{com}(S_e)$ encodes using the $r_{a,b}$ variables that S_e is a complete extension, and a soft clause (Ext_e) is introduced for each $e \in E^+$ with weights corresponding to the weights of the examples. In terms of negative examples, the encoding follows similar lines of reasoning but is somewhat more complicated, and detailed in [42].

In the CEGAR loop, a MaxSAT solver is called iteratively, obtaining a candidate solution AF $F_\tau = (A, R_\tau)$. Then, the algorithm iterates over all satisfied positive examples e, and for each of these, checks whether there exists a counterexample, i.e., an admissible extension that is a superset of S_e. If such a counterexample is found, the algorithm proceeds by adding the clause

$$\text{Ext}_e \rightarrow \left(\bigvee_{(a,b)\in (A\times A)\setminus R_\tau} r_{a,b} \vee \bigvee_{(a,b)\in R_\tau} \neg r_{a,b} \right)$$

to the abstraction, stating that if one wants to satisfy e, one needs a different attack structure. Note that the consequent of the implication is exactly the *trivial refinement* in the CEGAR algorithm for enforcement (Equation 1).

5 STRONG REFINEMENTS

For each of the three problems we consider, the counterexample provided by the SAT solver in the CEGAR algorithm comes in terms

of an admissible or a stable extension of the candidate solution AF. In the original CEGAR approaches, the refinement used straightforwardly rules out exactly the attack structure of the candidate solution. This can be a considered as a weakest possible—or "trivial"—refinement, since it is guaranteed that no other attack structures are ever ruled out by a refinement step. This means that in the worst case the number of CEGAR iterations will equal the number of solutions to the original abstraction, the number of which can be very large. Thus the quest for stronger refinements is essential towards improving the scalability of the CEGAR algorithm for each of the three problems.

In particular, we may interpret the trivial refinement clause as stating "remove an existing attack or add a non-existing attack". Thus a natural way to seek for stronger refinements is to aim to exclude from the refinements literals that correspond to removing or adding attacks which do not change the counterexample extension.

So-called $\{\text{IN}, \text{OUT}, \text{UNDEC}\}$-labelings provide an alternative way of defining the standard AF semantics [4]. The persistence of a labeling, i.e., whether a labeling is still a labeling in an AF that has been updated via adding or removing attacks, was recently studied from a representational point of view [44] under the complete, preferred, grounded, stable, and semi-stable semantics. Under these semantics, labelings are in one-to-one correspondence with extensions [4]. We put the results for stable semantics into action computationally by taking into account the labels of the endpoints of each attack in the counterexample AF.

Most recently, the results of [44] for the persistence of a stable labeling, i.e., a stable extension, was extended to admissible semantics in the context of incomplete argumentation frameworks [39]. In the context of plain AFs, the results are summarized in the following propositions. For an extension $E \in \sigma(F)$ for an AF F, denote $\text{IN}(E) = E$, $\text{OUT}(E) = \{a \in A \mid (b, a) \in R, b \in E\}$, and $\text{UNDEC}(E) = A \setminus (\text{IN}(E) \cup \text{OUT}(E))$. Consider first adding an attack to an AF.

Proposition 1. *Let $F = (A, R)$ be an AF, $E \in \sigma(F)$, $\sigma \in \{adm, stb\}$, and $(a, b) \in (A \times A) \setminus R$. If $a \in \text{OUT}(E)$ or $b \notin \text{IN}(E)$, then $E \in \sigma(F^+)$ for $F^+ = (A, R \cup \{(a, b)\})$.*

That is, adding an attack with the source already attacked by E, or the target outside E, has no effect on E being an extension.

Now consider removing an attack from an AF.

Proposition 2. *Let $F = (A, R)$ be an AF, $E \in \sigma(F)$, $\sigma \in \{adm, stb\}$, and $(a, b) \in R$. If $a \notin \text{IN}(E)$ or $b \notin \text{OUT}(E)$, then $E \in \sigma(F^-)$ for $F^- = (A, R \setminus \{(a, b)\})$.*

In words, removing an attack from an argument that is not in E or removing an attack to an argument that is not attacked by E has no effect on E being an extension.

Based on these propositions, we define in the context of the CEGAR algorithms a *strong refinement* clause $\text{REFINE}(F_\tau, E)$ for the enforcement and synthesis problems, where F_τ is the candidate AF provided by the MaxSAT solver and E is the counterexample extension provided by the SAT solver, defined as

$$\bigvee_{\substack{(a,b) \in ((A \times A) \setminus R_\tau) \cap \\ ((\text{IN}(E) \cup \text{UNDEC}(E)) \times \text{IN}(E))}} r_{a,b} \vee \bigvee_{\substack{(a,b) \in R_\tau \cap \\ (\text{IN}(E) \times \text{OUT}(E))}} \neg r_{a,b}. \qquad (2)$$

Due to Propositions 1 and 2, this refinement clause is also valid in the sense that it does not exclude any solutions of the problem at hand. For extension and status enforcement, this can be formalized as follows.

Theorem 3. *Consider the algorithm resulting from replacing $\text{REFINE}(F_\tau)$ (line 8 in Algorithm 1) with $\text{REFINE}(F_\tau, E)$ as defined in Equation 2, where $E = \{a \in A \mid \tau'(a) = 1\}$ is the counterexample extension extracted from the satisfying truth assignment τ' of the SAT solver call on line 6. This algorithm is correct for both the extension and status enforcement problem.*

Proof. (sketch) Let $\chi = adm$ for strict extension enforcement under preferred and credulous status enforcement under admissible, and $\chi = stb$ otherwise (status enforcement under stable). At any iteration, suppose that both the MaxSAT and SAT calls were satisfiable and that we have extracted both a counterexample AF $F_\tau = (A, R_\tau)$ and a counterexample extension $E \in \chi(F_\tau)$. If we add a literal $r_{a,b}$ with $(a, b) \in \text{OUT}(E) \times (A \setminus \text{IN}(E))$ to the refinement clause, and during a subsequent iteration the AF $F^+ = (A, R \cup \{(a, b)\})$ is an optimal solution to the current abstraction, the MaxSAT solver may provide a truth assignment corresponding to F^+, since $R \cup \{(a, b)\}$ satisfies the refinement clause by construction. By Proposition 1, $E \in \chi(F^+)$ and thus E is still a counterexample. The same holds for any literal $\neg r_{a,b}$ with $(a, b) \in (A \setminus \text{IN}(E)) \times (A \setminus \text{OUT}(E))$ and the AF $F^- = (A, R \setminus \{(a, b)\})$ by Proposition 2. That is, all literals $r_{a,b}$ with $(a, b) \in \text{OUT}(E) \times (A \setminus \text{IN}(E))$ and $\neg r_{a,b}$ with $(a, b) \in (A \setminus \text{IN}(E)) \times (A \setminus \text{OUT}(E))$ are redundant in the clause in Equation 1, and hence may be removed, resulting in the strong refinement clause (Equation 2). Since Algorithm 1 is correct [46, 40], the modified algorithm is also correct. □

The result of Theorem 3 extends to the CEGAR algorithm for AF synthesis under preferred [42].

Corollary 4. *Consider the algorithm resulting from replacing $Ext_e \rightarrow \text{REFINE}(F_\tau)$ (as described in Section 4.2) with $Ext_e \rightarrow \text{REFINE}(F_\tau, E)$, where F_τ is the candidate AF and E is an admissible superset of a positive example S_e extracted from the satisfiyng truth assignment of the SAT solver. This algorithm is correct for AF synthesis under preferred semantics.*

We illustrate the effect of the strong refinement clause in the context of status enforcement.

Example 6. *Consider Example 3 with the input AF $F = (A, R)$ from Example 1, and enforcing $P = \{d\}$ and $N = \{a\}$ credulously under admissible. Suppose the optimal truth assignment of the MaxSAT solver τ corresponds to the candidate AF $F_\tau = (A, R \cup \{(b, c)\})$, where $\{b, d\}$ is an admissible extension. However, $\{a\}$ is still in an admissible extension according to the satisfying truth assignment of the SAT solver, so we need to exclude this AF from consideration. The trivial refinement (Equation 1) gives the clause*

$$r_{a,a} \vee r_{a,c} \vee r_{a,d} \vee r_{a,e} \vee r_{b,b} \vee r_{b,d} \vee r_{b,e} \vee r_{c,a} \vee r_{c,c} \vee r_{c,e} \vee r_{d,a} \vee r_{d,a} \vee r_{d,b} \vee r_{d,c} \vee r_{d,d} \vee r_{e,a} \vee r_{e,b} \vee r_{e,d} \vee r_{e,e} \vee \neg r_{a,b} \vee \neg r_{b,a} \vee \neg r_{b,c} \vee \neg r_{c,b} \vee \neg r_{c,d} \vee \neg r_{d,e} \vee \neg r_{e,c}$$

consisting of $5^2 = 25$ literals. Clearly, e.g., adding the attack (a, c) satisfies the clause via the literal $r_{a,c}$, but has no effect on $\{a\}$ being admissible. If we instead compute $\text{IN}(E) = \{a\}$, $\text{OUT}(E) = \{b\}$ and $\text{UNDEC}(E) = \{c, d, e\}$, we obtain via Equation 2 the stronger refinement clause $r_{a,a} \vee r_{c,a} \vee r_{d,a} \vee r_{e,a} \vee \neg r_{a,b}$ with only 5 literals.

6 EXPERIMENTS

We overview results from an empirical evaluation on the impact of employing strong refinements on the practical efficiency

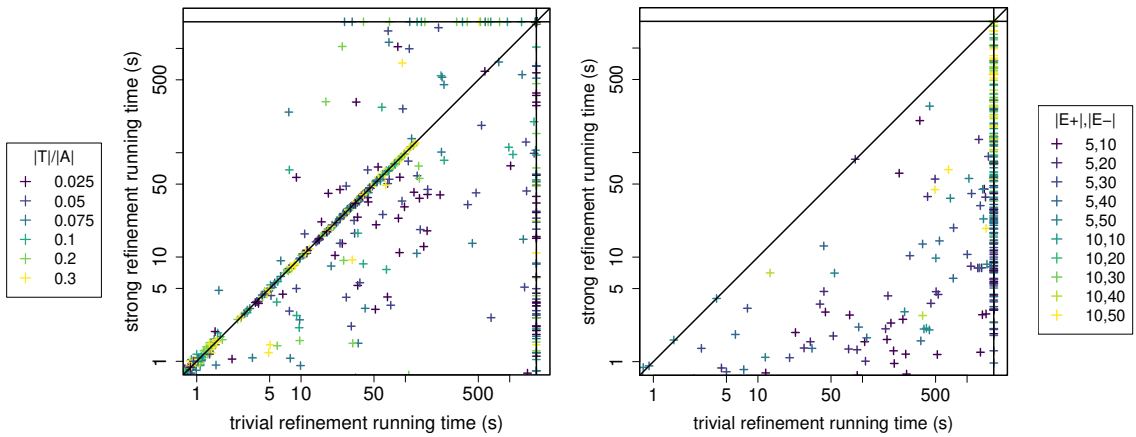

Figure 2. Trivial vs. strong refinement: strict extension enforcement (left) and AF synthesis (right) under prf.

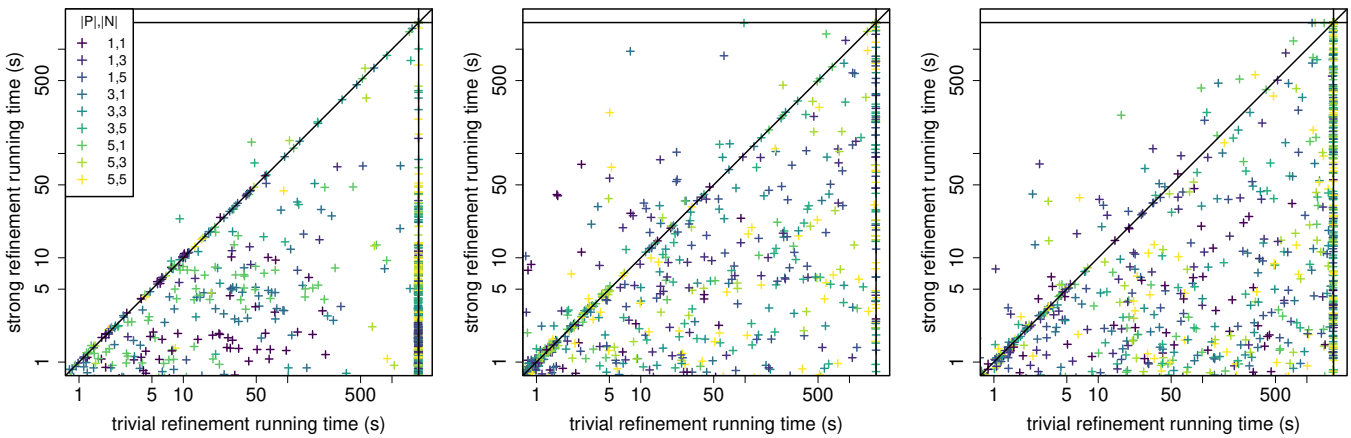

Figure 3. Trivial vs. strong refinement for status enforcement: credulous under adm (left), credulous under stb (center), skeptical under stb (right).

of the state-of-the-art CEGAR approaches to extension enforcement, status enforcement, and AF synthesis variants with second-level complexity. For the evaluation, we reimplemented the CEGAR algorithms for the variants of enforcement [46, 40] and AF synthesis [42] under the PySAT framework [35] (version 0.1.4.dev19), using RC2 [36] as the MaxSAT solver and Glucose [2] as the SAT solver. This reimplementation turned out to outperform the previous implementations (Pakota and AFSynth) of the CEGAR algorithms. The reimplementations are available via `https://www.helsinki.fi/en/researchgroups/constraint-reasoning-and-optimization/software`.

For extension and status enforcement, we used as benchmarks AFs from the 3rd International Competition on Computational Models of Argumentation (ICCMA 2019)[2] with up to 500 arguments. For each of these 221 AFs, we generated six extension enforcement instances for each $|T|/|A| = 0.025, 0.05, 0.075, 0.1, 0.2, 0.3$, following [46]. For each AF with at most 200 arguments (a total of 154), we generated nine status enforcement instances for each $|P|, |N| = 1, 3, 5$, following [40]. This resulted in a total of 1326 extension enforcement instances and 1386 status enforcement instances. For AF synthesis, we employed the exact same random instance generation method as in [42], with parameters $|A| = 20, 30, 40, 50$, $|E^+| = 5, 10$, and $|E^-| = 10, 20, 30, 40, 50$, resulting in a total of 400 instances.

All experiments were run on Intel Xeon E5-2680 v4 2.4-GHz, 256-GB machines running CentOS 7. We enforced a per-instance time limit of 1800 seconds and a memory limit of 64 GB.

[2] `https://www.iccma2019.dmi.unipg.it`

Results Overviews of the results, comparing for each of the problems the performance of the corresponding CEGAR algorithm employing the "trivial" (instance-specific running times on x-axis) and strong refinements (instance-specific running times on y-axis), are shown in Figure 2 left (for extension enforcement), Figure 2 right (for AF synthesis), and in Figure 3 (for variants of status enforcement). In summary, the strong refinements improve the overall performance of the CEGAR approach for each problem and problem variant. The most noticeable and one-sided improvements are obtained on AF synthesis. We will discuss the results in more detail individually for each of the problems.

Table 1. Mean running times (with timeouts as 1800 s) and number of timeouts (out of 221 instances): strict extension enforcement under preferred.

	Refinement type			
$\|T\|/\|A\|$	trivial		strong	
0.025	1023.32	(121)	**798.11**	**(94)**
0.05	830.51	(95)	**666.93**	**(78)**
0.075	748.53	(87)	**671.96**	**(79)**
0.1	717.16	(82)	**676.62**	**(81)**
0.2	463.21	(54)	**433.36**	**(51)**
0.3	325.47	(38)	**301.14**	**(34)**

For extension enforcement (see Figure 2 left), the number of timeouts drops by 60 when employing the strong refinement instead of the trivial one. The performance-improving effects of using the

strong refinement is especially evident on instances on which the set of arguments is relatively small, in particular for $|T|/|A| = 0.025, 0.05, 0.075$. Interestingly, there are also instances which time out for the strong refinement but not for the trivial one, but this happens on a clear minority of the instances. Why this happens is an interesting question to consider; at present, we hypothesize this to be due to non-deterministic effects in how the internal heuristics of the MaxSAT solver is influenced by the specific order and structure of the added refinement clauses. More details on the comparison are provided in Table 1 in light of the mean running times and numbers of timeouts resulting from using the different refinements. We observe that instances with a smaller $|T|/|A|$ ratio are significantly harder to solve. The strong refinement improves performance especially on these harder instances, resulting in considerably fewer timeouts, e.g., dropping from 121 to 94 for $|T|/|A| = 0.025$.

Table 2. Mean running times (with timeouts as 1800 s) and number of timeouts (out of 462 instances): status enforcement

			Refinement type			
Mode	**Semantics**	**Parameter**	**trivial**		**strong**	
cred	*adm*	$\|P\| = 1$	881.27	(214)	**22.51**	**(5)**
cred	*adm*	$\|P\| = 3$	971.08	(234)	**75.45**	**(14)**
cred	*adm*	$\|P\| = 5$	1059.88	(260)	**143.27**	**(32)**
cred	*adm*	$\|N\| = 1$	125.00	(22)	**72.88**	**(15)**
cred	*adm*	$\|N\| = 3$	1209.24	(283)	**80.17**	**(18)**
cred	*adm*	$\|N\| = 5$	1577.98	(403)	**88.18**	**(18)**
cred	*stb*	$\|P\| = 1$	218.11	(44)	**80.13**	**(13)**
cred	*stb*	$\|P\| = 3$	314.88	(67)	**156.13**	**(32)**
cred	*stb*	$\|P\| = 5$	336.72	(69)	**207.08**	**(43)**
cred	*stb*	$\|N\| = 1$	88.89	(17)	**74.52**	**(15)**
cred	*stb*	$\|N\| = 3$	235.09	(44)	**122.02**	**(24)**
cred	*stb*	$\|N\| = 5$	545.74	(119)	**246.80**	**(49)**
skept	*stb*	$\|P\| = 1$	285.31	(53)	**95.66**	**(20)**
skept	*stb*	$\|P\| = 3$	690.63	(150)	**263.35**	**(53)**
skept	*stb*	$\|P\| = 5$	854.41	(189)	**369.42**	**(72)**
skept	*stb*	$\|N\| = 1$	504.65	(103)	**212.56**	**(41)**
skept	*stb*	$\|N\| = 3$	621.36	(133)	**244.49**	**(50)**
skept	*stb*	$\|N\| = 5$	703.83	(156)	**271.11**	**(54)**

Turning to status enforcement, a comparison of the trivial and strong refinement are shown for credulous enforcement under admissible and stable in Figure 3 left and center, and skeptical enforcement under stable in Figure 3 right. The positive impact of employing the strong refinement is here even more pronounced than in the case of extension enforcement. In particular, the number of timeouts drops considerably, and the running times improve quite consistently. For example, for credulous status enforcement under admissible and instances with $|N| = 3, 5$ (Figure 3 left), a considerable number of instances on which CEGAR times out when employing the trivial refinement are solved in under 50 seconds when employing the strong refinement. The mean running times and numbers of timeouts for the variants of status enforcement are provided in Table 2. For the credulous problem variants, the positive impact of using the strong refinement increases for larger $|N|$, which also results in the most difficult-to-solve instances especially for admissible. This may be explained by the fact that negatively enforced arguments are the source of second-level hardness for the credulous status enforcement problem [40]. For the skeptical variant of status enforcement, the number of timeouts is cut by more than a half for all parameter choices by employing the strong refinement. Interestingly, the choices of $|P|$ and $|N|$ clearly affect the empirical difficulty of the instances. In fact, it appears that a contrary observation was made in [40] due the fact that with the trivial refinement the instances that reveal a dependency on runtimes on the choices of $|P|$ and $|N|$ turned out to be too difficult to solve for all choices of the parameters.

Finally, we overview results for the AF synthesis problem under preferred, which to-date is the most challenging one among the considered three problems in practice in terms of scalability wrt. the number of arguments. Here the positive impact of employing the strong refinement is the most pronounced, considerably pushing further the scalability of the CEGAR algorithm in practice. As shown in Figure 2 (right), with the trivial refinement only 6 instances out of 200 with 10 positive examples are solved in the time limit. In contrast, by employing the strong refinement a majority of the instances with 10 positive examples can now be solved. Interestingly, solving these instances gets easier when employing the strong refinement with increasing $|A|$, which is also witnessed by the number of timeouts in Table 3. We observed that this is due to the fact that the cost of optimal solutions decreases significantly with increasing $|A|$ as the probability of generating conflicting examples decreases with increasing $|A|$. This effect was not observed in the original work proposing the CEGAR algorithm for AF synthesis using the trivial refinement [42] due to its poor scalability.

Table 3. Mean running times (with timeouts as 1800s) and number of timeouts (out of 50 instances): AF synthesis under preferred

		Refinement type			
$\|A\|$	$\|E^+\|$	**trivial**		**strong**	
20	5	416.04	(10)	**53.21**	**(1)**
20	10	1676.23	(45)	**1280.45**	**(33)**
30	5	1133.46	(26)	**18.52**	**(0)**
30	10	1777.27	(49)	**1034.65**	**(25)**
40	5	1392.90	(30)	**68.01**	**(1)**
40	10	1800.00	(50)	**549.27**	**(8)**
50	5	1643.90	(40)	**90.09**	**(0)**
50	10	1800.00	(50)	**586.75**	**(8)**

7 CONCLUSIONS

The study of AF dynamics gives rise to noticeably hard optimization problems. Focusing on three recently proposed and studied aspects of dynamics—extension enforcement, status enforcement, and AF synthesis—we address the challenge of scaling up the performance of the current state of the art in practical algorithms for the problems for semantics and reasoning modes under which the problems have beyond-NP complexity. Specifically, we showed that by applying recent results on the persistence of extensions under changes to an argumentation framework, the performance of the currently best SAT-based counterexample-guided abstraction refinement algorithms for these problems can be noticeably improved. Regarding further work, we aim to study whether strong refinements can be applied to other problems over AFs that are hard for the second level of the polynomial hierarchy.

ACKNOWLEDGEMENTS

This work has been financially supported by Academy of Finland (grants 276412, 312662, 322869) and University of Helsinki Doctoral Programme in Computer Science DoCS.

REFERENCES

[1] Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Ricardo Simari, Matthias Thimm, and Serena Villata, 'Towards artificial argumentation', *AI Magazine*, **38**(3), 25–36, (2017).

[2] Gilles Audemard and Laurent Simon, 'On the Glucose SAT solver', *International Journal on Artificial Intelligence Tools*, **27**(1), 1–25, (2018).

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

[4] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin, 'Abstract argumentation frameworks and their semantics', in *Handbook of Formal Argumentation*, chapter 4, 159–236, College Publications, (2018).

[5] Ringo Baumann, 'Normal and strong expansion equivalence for argumentation frameworks', *Artif. Intell.*, **193**, 18–44, (2012).

[6] Ringo Baumann, 'What does it take to enforce an argument? Minimal change in abstract argumentation', in *Proc. ECAI*, volume 242 of *FAIA*, pp. 127–132. IOS Press, (2012).

[7] Ringo Baumann and Gerhard Brewka, 'Expanding argumentation frameworks: Enforcing and monotonicity results', in *Proc. COMMA*, volume 216 of *FAIA*, pp. 75–86. IOS Press, (2010).

[8] Ringo Baumann and Gerhard Brewka, 'AGM meets abstract argumentation: Expansion and revision for Dung frameworks', in *Proc. IJCAI*, pp. 2734–2740. AAAI Press, (2015).

[9] Philippe Besnard and Sylvie Doutre, 'Checking the acceptability of a set of arguments', in *Proc. NMR*, pp. 59–64, (2004).

[10] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex, 'Enforcement in argumentation is a kind of update', in *Proc. SUM*, volume 8078 of *LNCS*, pp. 30–43. Springer, (2013).

[11] Gustavo Adrian Bodanza, Fernando Tohmé, and Marcelo Auday, 'Collective argumentation: A survey of aggregation issues around argumentation frameworks', *Argument & Computation*, **8**(1), 1–34, (2017).

[12] Richard Booth, Souhila Kaci, Tjitze Rienstra, and Leendert W. N. van der Torre, 'A logical theory about dynamics in abstract argumentation', in *Proc. SUM*, volume 8078 of *LNCS*, pp. 148–161. Springer, (2013).

[13] Federico Cerutti, Sarah A. Gaggl, Matthias Thimm, and Johannes P. Wallner, 'Foundations of implementations for formal argumentation', in *Handbook of Formal Argumentation*, chapter 14, 689–767, College Publications, (2018).

[14] Weiwei Chen and Ulle Endriss, 'Preservation of semantic properties in collective argumentation: The case of aggregating abstract argumentation frameworks', *Artif. Intell.*, **269**, 27–48, (2019).

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

[16] Edmund M. Clarke, Anubhav Gupta, and Ofer Strichman, 'SAT-based counterexample-guided abstraction refinement', *IEEE Trans. on CAD of Integrated Circuits and Systems*, **23**(7), 1113–1123, (2004).

[17] Sylvie Coste-Marquis, Caroline Devred, Sébastien Konieczny, Marie-Christine Lagasquie-Schiex, and Pierre Marquis, 'On the merging of Dung's argumentation systems', *Artif. Intell.*, **171**(10-15), 730–753, (2007).

[18] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis, 'On the revision of argumentation systems: Minimal change of arguments statuses', in *Proc. KR*. AAAI Press, (2014).

[19] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis, 'A translation-based approach for revision of argumentation frameworks', in *Proc. JELIA*, volume 8761 of *LNCS*, pp. 397–411. Springer, (2014).

[20] Sylvie Coste-Marquis, Sébastien Konieczny, Jean-Guy Mailly, and Pierre Marquis, 'Extension enforcement in abstract argumentation as an optimization problem', in *Proc. IJCAI*, pp. 2876–2882. AAAI Press, (2015).

[21] Florence Dupin de Saint-Cyr, Pierre Bisquert, Claudette Cayrol, and Marie-Christine Lagasquie-Schiex, 'Argumentation update in YALLA (Yet Another Logic Language for Argumentation)', *Int. J. Approx. Reasoning*, **75**, 57–92, (2016).

[22] Jérôme Delobelle, Adrian Haret, Sébastien Konieczny, Jean-Guy Mailly, Julien Rossit, and Stefan Woltran, 'Merging of abstract argumentation frameworks', in *Proc. KR*, pp. 33–42. AAAI Press, (2016).

[23] Jérôme Delobelle, Sébastien Konieczny, and Srdjan Vesic, 'On the aggregation of argumentation frameworks', in *Proc. IJCAI*, pp. 2911–2917. AAAI Press, (2015).

[24] Jérôme Delobelle, Sébastien Konieczny, and Srdjan Vesic, 'On the aggregation of argumentation frameworks: operators and postulates', *J. Log. Comput.*, **28**(7), 1671–1699, (2018).

[25] Martin Diller, Adrian Haret, Thomas Linsbichler, Stefan Rümmele, and Stefan Woltran, 'An extension-based approach to belief revision in abstract argumentation', *Int. J. Approx. Reasoning*, **93**, 395–423, (2018).

[26] Sylvie Doutre, Andreas Herzig, and Laurent Perrussel, 'A dynamic logic framework for abstract argumentation', in *Proc. KR*. AAAI Press, (2014).

[27] Sylvie Doutre, Faustine Maffre, and Peter McBurney, 'A dynamic logic framework for abstract argumentation: Adding and removing arguments', in *Proc. IEA/AIE*, volume 10351 of *LNCS*, pp. 295–305. Springer, (2017).

[28] Sylvie Doutre and Jean-Guy Mailly, 'Semantic change and extension enforcement in abstract argumentation', in *Proc. SUM*, volume 10564 of *LNCS*, pp. 194–207. Springer, (2017).

[29] Sylvie Doutre and Jean-Guy Mailly, 'Constraints and changes: A survey of abstract argumentation dynamics', *Argument & Computation*, **9**(3), 223–248, (2018).

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

[31] Paul E. Dunne, Wolfgang Dvorák, Thomas Linsbichler, and Stefan Woltran, 'Characteristics of multiple viewpoints in abstract argumentation', *Artif. Intell.*, **228**, 153–178, (2015).

[32] Paul E. Dunne, Pierre Marquis, and Michael J. Wooldridge, 'Argument aggregation: Basic axioms and complexity results', in *Proc. COMMA*, eds., Bart Verheij, Stefan Szeider, and Stefan Woltran, volume 245 of *FAIA*, pp. 129–140. IOS Press, (2012).

[33] Wolfgang Dvorák and Paul E. Dunne, 'Computational problems in formal argumentation and their complexity', in *Handbook of Formal Argumentation*, eds., Pietro Baroni, Dov Gabbay, Massimiliano Giacomin, and Leendert van der Torre, chapter 13, 631–687, College Publications, (2018).

[34] Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran, 'Complexity-sensitive decision procedures for abstract argumentation', *Artif. Intell.*, **206**, 53–78, (2014).

[35] Alexey Ignatiev, António Morgado, and João Marques-Silva, 'Pysat: A Python toolkit for prototyping with SAT oracles', in *Proc. SAT*, volume 10929 of *LNCS*, pp. 428–437. Springer, (2018).

[36] Alexey Ignatiev, António Morgado, and João Marques-Silva, 'RC2: An efficient MaxSAT solver', *JSAT*, **11**, 53–64, (2019).

[37] Mikolás Janota, Radu Grigore, and João Marques-Silva, 'Counterexample guided abstraction refinement algorithm for propositional circumscription', in *Proc. JELIA*, volume 6200 of *LNCS*, pp. 195–207. Springer, (2010).

[38] Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke, 'Solving QBF with counterexample guided refinement', *Artif. Intell.*, **234**, 1–25, (2016).

[39] Andreas Niskanen, Daniel Neugebauer, Matti Järvisalo, and Jörg Rothe, 'Deciding acceptance in incomplete argumentation frameworks', in *Proc. AAAI*, (2020, in press).

[40] Andreas Niskanen, Johannes Peter Wallner, and Matti Järvisalo, 'Optimal status enforcement in abstract argumentation', in *Proc. IJCAI*, pp. 1216–1222. IJCAI/AAAI Press, (2016).

[41] Andreas Niskanen, Johannes Peter Wallner, and Matti Järvisalo, 'Extension enforcement under grounded semantics in abstract argumentation', in *Proc. KR*, pp. 178–183. AAAI Press, (2018).

[42] Andreas Niskanen, Johannes Peter Wallner, and Matti Järvisalo, 'Synthesizing argumentation frameworks from examples', *J. Artif. Intell. Res.*, **66**, 503–554, (2019).

[43] Farid Nouioua and Eric Würbel, 'Removed set-based revision of abstract argumentation frameworks', in *Proc. ICTAI*, pp. 784–791. IEEE Computer Society, (2014).

[44] Tjitze Rienstra, Chiaki Sakama, and Leendert W. N. van der Torre, 'Persistence and monotony properties of argumentation semantics', in *Proc. TAFA*, volume 9524 of *LNCS*, pp. 211–225. Springer, (2015).

[45] Jendrik Seipp and Malte Helmert, 'Counterexample-guided cartesian abstraction refinement for classical planning', *J. Artif. Intell. Res.*, **62**, 535–577, (2018).

[46] Johannes Peter Wallner, Andreas Niskanen, and Matti Järvisalo, 'Complexity results and algorithms for extension enforcement in abstract argumentation', *J. Artif. Intell. Res.*, **60**, 1–40, (2017).