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

doi:10.3233/FAIA200175

Algorithms for Dynamic Argumentation Frameworks: An Incremental SAT-Based Approach

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

Abstract. Motivated by the fact that argumentation is intrinsically a dynamic process, the study of representational and computational aspects of dynamics in argumentation is starting to gain more traction. This is also witnessed by the most recent 2019 edition of the International Competition on Computational Models of Argumentation (ICCMA 2019), which introduced a new track focusing on dynamic argumentation frameworks. In this paper, we present an efficient Boolean satisfiability (SAT) based approach to reasoning over dynamic argumentation frameworks. In particular, based on employing incremental SAT solving, we detail algorithms covering all of the reasoning tasks—credulous and skeptical acceptance, as well as the computation of a single and all extensions—and semantics—complete, preferred, stable, and grounded—constituting the ICCMA 2019 dynamic track. Furthermore, we demonstrate empirically that an implementation of the approach is highly competitive.

1 INTRODUCTION

Computational aspects of argumentation is a vibrant area of artificial intelligence research. Abstract argumentation frameworks (AFs) [23] constitute a central formalism for knowledge representation and reasoning in AI argumentation, allowing for modelling disputes between agents in a simple yet flexible manner. Syntactically, AFs take the form of directed graphs, where nodes represent abstract arguments (of agents), edges between nodes represent attacks between arguments and counterarguments, and semantics identify subsets of jointly acceptable arguments [7].

In various application scenarios, such as disputes between agents in online social networks [30], the attack relation is subject to temporal changes, as e.g. disputes may change (be retracted or added) due to new available knowledge. This underlines the dynamic nature of argumentation [12, 9]. To account for different types of dynamics in AFs, going beyond studying different aspects of static AFs, various new types of problem settings have been recently considered and investigated [22] (as further outlined as related work in Section 7).

The focus of this work is on a specific recently formalized type of AF dynamics, where the attack structure of a given initial AF is subject to iterative changes in the form of removing and adding individual attacks, thereby modelling the temporal nature of AFs. In particular, the computational tasks involving these dynamic argumentation frameworks deal with iteratively deciding the acceptance of a given argument, or computing a single or all extensions, under a given semantics after each iterative change to the AF. Naturally, the computational hardness of deciding acceptance and enumerating extensions for a given static AF under different AF semantics [25, 31] directly implies that the computational tasks involving dynamic AFs are likewise computationally challenging. Addressing partially this challenge, first algorithms for specific variants of this problem setting have been very recently proposed [28, 29, 1, 2]. As the most recent development, the newest 2019 edition of the International Competition on Computational Models of Argumentation (ICCMA 2019) introduced a new track focusing on dynamic argumentation frameworks, covering both skeptical and credulous acceptance and computation of a single and all extensions as well as four central argumentation semantics (grounded, complete, preferred, and stable) [15].

In this paper, we describe incremental Boolean satisfiability (SAT) based algorithms for computational tasks involving dynamic argumentation frameworks. In particular, the algorithms cover all of the reasoning tasks—credulous and skeptical acceptance, as well as the computation of a single and all extensions—and semantics—complete, preferred, stable, and grounded—of the dynamic track of ICCMA 2019. Building on the success of SAT-based approaches for reasoning over static AFs [26, 33, 19, 37], we describe SAT encodings for the dynamic case as a basis of the incremental approach; and present in detail SAT-based algorithms that make use of the assumptions interface of a SAT solver—as well as further optimizations based on knowledge obtained from previous calls—to avoid making redundant SAT solver calls during the iterative computation. A preliminary implementation of the approach was shown to be competitive with other proposed solutions in the ICCMA 2019 competition (`https://www.iccma2019.dmi.unipg.it/results/results-dyn.html`). Here we present results from a more extensive empirical evaluation of the approach extending the ICCMA 2019 benchmarks to a higher number of changes to the underlying AFs, showing that the approach scales noticeably better than currently available implementations of alternative approaches [32].

The rest of the paper is organized as follows. We first give necessary background on argumentation frameworks (Section 2) and in particular dynamic argumentation frameworks and the associated key computational problems focused on in this work (Section 3). We will then describe in detail our SAT-based approach to reasoning about dynamic argumentation frameworks: the SAT encodings used within the approach (Section 4) and the incremental SAT-based algorithms for dynamic acceptance and extension computation problems (Section 5). Before conclusions, we present results from an empirical evaluation of the approach (Section 6).

2 ARGUMENTATION FRAMEWORKS

We start by recalling argumentation frameworks [23], the argumentation semantics [6, 7] and the static versions of computational

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

tasks [25] considered 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* $R \subseteq A \times A$ *is the attack relation. The pair* $(a, b) \in R$ *indicates that* a *attacks* 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 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* $R = \{(a, b), (b, a), (b, c), (c, d), (d, e), (e, c)\}$*. The AF* F *is represented as a directed graph in Figure 1. Here* a *defends itself against the attack from* b*, and, on the other hand* d *is defended by the set* $\{b, d\}$ *since* b *attacks* c*, the only attacker of* d*.*

Semantics for AFs are defined via functions σ which assign to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions corresponding to jointly acceptable subsets of arguments. We consider for σ the functions cf, adm, com, grd, prf, and stb, which stand for conflict-free, admissible, complete, grounded, preferred, and stable, respectively. To formally define the semantics, we recall two standard auxiliary concepts.

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 considered in this work are now 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* $S = \mathcal{F}_F(S)$*,*
- $S \in grd(F)$ *iff* $S \in com(F)$ *and* $\nexists S' \in com(F)$ *with* $S' \subset S$*,*
- $S \in prf(F)$ *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, i.e., an extension under the semantics* σ*.*

Following from Definition 3, the subset-relations

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

hold for any AF F.

Two central reasoning tasks on AFs are deciding whether a given argument is credulously or skeptically accepted under a prescribed semantics.

Definition 4. *Let* $F = (A, R)$ *be an AF and* $\sigma \in \{cf, adm, com, grd, prf, stb\}$ *an AF semantics. We say that an argument* $a \in A$ *is* credulously accepted under σ *if* $a \in \bigcup \sigma(F)$*, and* skeptically accepted under σ *if* $a \in \bigcap \sigma(F)$*.*

That is, an argument is credulously accepted if it is contained in at least one σ-extension, and skeptically accepted if it is contained in all σ-extensions of the given AF. We denote the task of credulous acceptance under semantics σ by DC-σ and the task of skeptical acceptance by DS-σ.

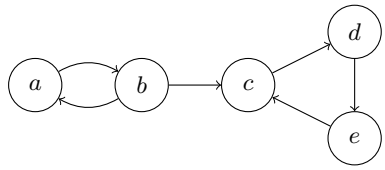

Figure 1. Argumentation framework F from Example 1.

Example 2. *Consider the AF* F *in Example 1 and Figure 1. The* σ*-extensions of* F *are* $com(F) = \{\emptyset, \{a\}, \{b, d\}\}$*,* $prf(F) = \{\{a\}, \{b, d\}\}$*,* $stb(F) = \{\{b, d\}\}$*, and* $grd(F) = \{\emptyset\}$*. The argument* a *is credulously, but not skeptically, accepted under preferred semantics. On the other hand,* a *is neither credulously nor skeptically accepted under stable.*

The computational complexity of deciding the credulous acceptance of a query argument under complete, preferred, and stable semantics is NP-complete [21] (note that the problems coincide under admissible, complete, and preferred semantics). On the other hand, deciding the skeptical acceptance of an argument is coNP-complete under stable semantics, and Π_2^{P}-complete under preferred semantics [24]. Skeptical acceptance under complete coincides with skeptical acceptance under grounded, which in turn coincides with credulous acceptance under grounded. This follows from the fact that the grounded extension is the unique subset-minimal complete extension: each of these problems is decidable in polynomial time [20] since the grounded extension of any AF can be computed in polynomial time by simulating the iterative application of the characteristic function. For an overview of complexity in abstract argumentation, we refer the reader to [25].

3 DYNAMIC ARGUMENTATION FRAMEWORKS

In this work we use the term *dynamic argumentation frameworks* to refer to standard AFs augmented with an update, i.e., a change to the attack structure of the AF, following [28, 29, 1, 2].[2] The extensions of the updated AF can clearly change due to the update (consider e.g. adding a self-attack to an argument that is contained in an extension). Hence, also the acceptance status of a query argument is subject to change. In particular, we focus on the computational tasks of the ICCMA 2019 track over dynamic AFs [15] where in addition to an AF, a sequence of changes (rather than a single one) to the attack structure of the AF is provided as input, and the task is to answer the query (credulous or skeptical acceptance, or the computation of any or all extensions) for each of the AFs defined by the sequence—with the goal of answering the queries for updates (potentially more efficiently than computing from scratch).

Example 3. *We continue from Example 1. Consider now the AF* F_1 *obtained by removing the attack* (b, c) *from the AF* F*. This AF is illustrated in Figure 2. Now, the* σ*-extensions of* F_1 *are* $com(F_1) = \{\emptyset, \{a\}, \{b\}\}$*,* $prf(F_1) = \{\{a\}, \{b\}\}$*,* $stb(F_1) = \emptyset$*, and* $grd(F_1) = \{\emptyset\}$*. The complete and preferred extension* $\{b, d\}$ *of* F *has now lost the argument* d*, and* F_1 *has no stable extension. The status of* a *remains the same under preferred semantics. However, under stable semantics,* a *is now skeptically accepted, since no stable extension exists.*

[2] We note that other uses of the term dynamic argumentation frameworks do appear in the literature [36].

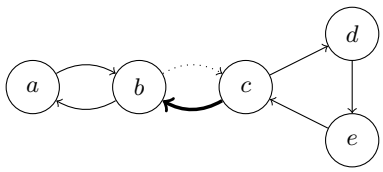

Figure 2. Argumentation frameworks F_1 and F_2 from Example 2, with the removed attack as a dotted edge and the added attack as a thick edge.

If, on the other hand, we add the attack (c, b) to F_1, as illustrated again in Figure 2, we obtain another AF F_2. The extensions of F_2 are $com(F_2) = \{\emptyset, \{a\}\}$, $prf(F_2) = \{\{a\}\}$, $stb(F_2) = \emptyset$, and $grd(F_2) = \{\emptyset\}$. Now the complete and preferred extension $\{b\}$ of F_1 has disappeared in F_2. However, the acceptance status of a under these semantics remains the same. Again, every argument is skeptically accepted under stable, since no stable extension exists.

We will consider a standard AF augmented with a sequence of changes, i.e., additions and removals, to the attack structure. The sequence naturally defines a sequence of attack structures, using which we may distinguish between *static* and *dynamic* attacks of the dynamic AF.

Definition 5. *Consider an input AF $F = (A, R)$, along with a sequence of changes $\chi = (\chi_1, \ldots, \chi_n)$ where each χ_i is either $+(a, b)$ (addition of an attack) or $-(a, b)$ (deletion of an attack) for some $a, b \in A$. The sequence χ then generates a sequence of attack structures $(R_0, R_1, \ldots, R_n)$ recursively, where $R_0 = R$ (i.e., the attack structure given as input), and*

$$R_i = \begin{cases} R_{i-1} \cup (a, b) & \text{if } \chi_i = +(a, b) \text{ and } a, b \in A, \\ R_{i-1} \setminus (a, b) & \text{if } \chi_i = -(a, b) \text{ and } a, b \in A. \end{cases}$$

We define static and dynamic attacks R^s *and* R^d *via*

$$R^s = \bigcap_{i=0}^{n} R_i,$$
$$R^d = \left(\bigcup_{i=0}^{n} R_i \right) \setminus R^s,$$

where $(R_i)_{i=0}^n$ is the sequence of attack structures generated by χ. The corresponding dynamic AF *is $F^\chi = (A, R, \chi)$.*

That is, R^s consists of *static* attacks that are part of every attack structure of the sequence $(R_i)_{i=0}^n$, and R^d contains *dynamic* attacks which are included in some structure of the sequence, but not all.

Example 4. *Consider the input AF $F = (A, R)$ from Example 1 on page 2, and the changes $\chi = (-(b, c), +(c, b))$ introduced in Example 2, forming the dynamic AF $F^\chi = (A, R, \chi)$. The static and dynamic attacks are*

$$R^s = \{(a, b), (b, a), (c, d), (d, e), (e, c)\},$$
$$R^d = \{(b, c), (c, b)\}.$$

Note that in Examples 1 and 2, the singleton $\{a\}$ is a preferred extension in every AF of the sequence. That is, we could answer the question "is b skeptically accepted" more efficiently in F_1 and F_2 just by checking whether the counterexample extension $\{a\}$ from F_0 is still preferred. The algorithms developed in this work cover cases like this.

4 ENCODINGS

We continue by providing encodings in Boolean satisfiability (SAT) for the semantics considered in this paper, extending the standard encodings for argumentation semantics [13] to the dynamic case by essentially conditioning relevant parts of the formulas with the existence of an attack. The encodings will be utilized as main ingredients of the algorithms for the dynamic versions of the acceptance, search, and enumeration problems in the extension-based setting. In particular, our goal is to instantiate a SAT solver with an input formula ϕ_σ, from which the σ-extensions of all AFs in the sequence can be extracted.

Consider now a dynamic AF $F^\chi = (A, R, \chi)$, with $\chi = (\chi_1, \ldots, \chi_n)$, and the corresponding static and dynamic attacks R^s and R^d generated by χ. We declare Boolean variables x_a for all $a \in A$ and $r_{a,b}$ for all $(a, b) \in R^d$ with the following interpretations:

- $r_{a,b} = \top$ iff $(a, b) \in R_i$,
- $x_a = \top$ iff $a \in E$ for some $E \in \sigma(F_i)$,

where $F_i = (A, R_i)$ is the AF to which changes $\chi_1, \ldots, \chi_i$ have been applied (recall Definition 5). The $r_{a,b}$ variables introduced by the dynamic attacks in particular will be central to incremental SAT computations. In addition, we introduce auxiliary variables $s_{a,b}$ for each $(a, b) \in R^d$, y_a for each $a \in A$, and $z_{a,b}$ for each $(a, b) \in R^d$, binding their interpretations with equivalences to the x_a and $r_{a,b}$ variables as follows:

$$s_{a,b} \leftrightarrow (r_{a,b} \wedge x_a),$$
$$y_a \leftrightarrow \left(\bigvee_{(b,a) \in R^s} x_b \vee \bigvee_{(b,a) \in R^d} s_{b,a} \right),$$
$$z_{a,b} \leftrightarrow (r_{a,b} \rightarrow y_a).$$

Intuitively, $s_{a,b}$ is true iff the attack (a, b) exists and the source a is accepted, y_a is true iff the argument a is attacked by an accepted argument (i.e. defeated), and $z_{a,b}$ is true iff the existence of the attack (a, b) implies that the source is defeated.

Now, the Boolean formula

$$\phi_{cf}(F^\chi) = \bigwedge_{(a,b) \in R^s} (\neg x_a \vee \neg x_b) \wedge \bigwedge_{(a,b) \in R^d} (r_{a,b} \rightarrow (\neg x_a \vee \neg x_b))$$

encodes conflict-free sets of the dynamic AF F^χ, since for all static attacks we require that we cannot accept both the source and the target of the attack, and for dynamic attacks we require this only in the case that the corresponding attack exists.

Likewise the formula $\phi_{adm}(F^\chi)$ defined as

$$\phi_{cf}(F^\chi) \wedge \bigwedge_{a \in A} \left(x_a \rightarrow \left(\bigwedge_{(b,a) \in R^s} y_b \wedge \bigwedge_{(b,a) \in R^d} z_{b,a} \right) \right)$$

represents admissible sets, requiring them to be conflict-free and, additionally, for each argument a that if we accept a, we need to defend a against all static and dynamic attacks—that is, for each possible attack, if it exists, we need to accept an argument that counterattacks the source of the attack.

The encoding for complete semantics is given by $\varphi_{com}(F^\chi)$ defined as

$$\phi_{adm}(F^\chi) \wedge \bigwedge_{a \in A} \left(\left(\bigwedge_{(b,a) \in R^s} y_b \wedge \bigwedge_{(b,a) \in R^d} z_{b,a} \right) \rightarrow x_a \right)$$

stating additionally the converse; if we defend an argument, we need to include it in the extension. This, together with admissibility, corresponds exactly to complete extensions.

Finally, our encoding for stable semantics is

$$\phi_{stb}(F^\chi) = \phi_{cf}(F^\chi) \wedge \bigwedge_{a \in A} (x_a \vee y_a).$$

stating that stable extensions are conflict-free sets which attack every argument outside the set, i.e., either we accept an argument or accept some attacker of the argument. Note that the formulas $\phi_{cf}(F^\chi), \phi_{adm}(F^\chi), \phi_{com}(F^\chi)$ and $\phi_{stb}(F^\chi)$ are essentially in conjunctive normal form, the input format required for SAT solvers.

The following proposition summarizes the key properties of the encodings. The statement follows from the fact that via performing unit propagation on $r_{a,b}$ and $\neg r_{a,b}$ literals the encodings reduce to the standard SAT encodings of argumentation semantics [13].

Proposition 1. *Given a dynamic AF $F^\chi = (A, R, \chi)$, where $\chi = (\chi_1, \ldots, \chi_n)$, let $F_i = (A, R_i)$, $0 \leq i \leq n$, where R_i is defined according to Definition 5, and*

$$\text{ATT}(F_i) = \{r_{a,b} \mid (a,b) \in R_i\} \cup \{\neg r_{a,b} \mid (a,b) \notin R_i\}.$$

In the AF F_i,

- *$\phi_\sigma(F^\chi) \wedge x_a \wedge \text{ATT}(F_i)$ is satisfiable if and only if the argument $a \in A$ is credulously accepted under $\sigma \in \{com, stb\}$, and*
- *$\phi_\sigma(F^\chi) \wedge \neg x_a \wedge \text{ATT}(F_i)$ is satisfiable if and only if the argument $a \in A$ is not skeptically accepted under $\sigma \in \{com, stb\}$.*

5 ALGORITHMS

Making use of the just-described encodings, we detail an incremental SAT-based approach to credulous and skeptical reasoning and computing a single extension and all extensions for each attack structure of a given dynamic AF $F^\chi = (A, R, \chi)$. In particular, we make use of the so-called assumption interface of a SAT solver to enable incrementality, meaning that a SAT solver needs only be instantiated once for the whole sequence of changes, and information learned by the SAT solver from the previous iterations can be used for speeding up computations in the forthcoming iterations [27]. Here the "attack variables" $r_{a,b}$ play a crucial role as *assumptions* at iteration i in the form of $\text{ATT}(F_i)$ (recall Proposition 1) which are separately communicated to the SAT solver at each iteration for achieving incrementality. Furthermore, we outline checks on the persistence of the previously found solution both in the positive (e.g. the query argument was accepted during the previous iteration) and the negative (e.g. the query argument was not accepted previously) case to further speed up computation.

5.1 Acceptance

Following Proposition 1, our SAT-based approach to credulous acceptance and skeptical acceptance under $\sigma \in \{com, stb\}$ is described in pseudocode as Algorithm 1. On line 1, we initialize two empty sets E (extension) and C (core) for storing information from the previous iteration, and s for storing intermediate results. Here, depending on the reasoning mode $M \in \{\text{DC}, \text{DS}\}$ (DC for credulous, DS for skeptical), the algorithm incrementally queries a SAT solver at each iteration i with the input formula $\phi_\sigma(F^\chi) \wedge q$ under the assumptions

$$\text{ATT}(F_i) = \{r_{a,b} \mid (a,b) \in R_i\} \cup \{\neg r_{a,b} \mid (a,b) \notin R_i\}$$

Algorithm 1 Algorithm for dynamic credulous and dynamic skeptical acceptance under $\sigma \in \{com, stb\}$ on input $F^\chi = (A, R, \chi)$ and query $q \in A$.

```
1:  E ← ∅, C ← ∅, s ← empty Boolean array of size n + 1
2:  if M = DC  q ← x_q else q ← ¬x_q
3:  for i = 0, 1, ..., n
4:      if i > 0 and s[i − 1] = 1
5:          α ← ATT(F_i) ∪ {x_a | a ∈ E} ∪ {¬x_a | a ∉ E}
6:          ρ ← SAT.SOLVE(φ_σ(F^χ) ∧ q, α)
7:          if ρ = sat
8:              s[i] ← 1
9:              continue
10:     else if i > 0 and s[i − 1] = 0
11:         (a, b) ← |χ_i|
12:         ∘ ← sign(χ_i)
13:         if (∘ = + and r_{a,b} ∉ C) or (∘ = − and ¬r_{a,b} ∉ C)
14:             s[i] ← 0
15:             continue
16:     ρ ← SAT.SOLVE(φ_σ(F^χ) ∧ q, ATT(F_i))
17:     if ρ = sat
18:         τ ← SAT.GETMODEL()
19:         E ← {a ∈ A | τ(x_a) = ⊤}
20:         s[i] ← 1
21:     else
22:         C ← SAT.GETUNSATCORE()
23:         s[i] ← 0
24: for i = 0, 1, ..., n
25:     if (M = DC and s[i] = 1) or (M = DS and s[i] = 0)
26:         print "YES"
27:     else print "NO"
```

(SAT.SOLVE($\phi_\sigma(F^\chi) \wedge q$, ATT($F_i$)) on line 16), where q is x_a in the credulous case and $\neg x_a$ in the skeptical case (line 2). This accounts to deciding whether a given query argument $a \in A$ is credulously accepted and, respectively, is not skeptically accepted, in the AF $F_i = (A, R_i)$. Since com and prf coincide for credulous acceptance, this algorithm also covers credulous acceptance under preferred semantics. The same algorithm is also applicable to the polynomial-time computable case of the grounded semantics. This is guaranteed by running the polynomial-time main propagation mechanism of unit propagation in the SAT solver (instead of making a full-blown SAT solver call) on the SAT encoding of the complete semantics, which is enough for determining acceptance under grounded semantics in F_i by checking whether unit propagation assigns x_a to true [33].

Algorithm 1 includes two optimizations: one for the case where a was determined as being credulously accepted (or not being skeptically accepted) in the previous AF $F_{i-1} = (A, R_{i-1})$ (the "positive" case), and one for the case where a was determined as not being credulously accepted (or being skeptically accepted) in F_{i-1} (the "negative" case).

In the positive case, we have obtained a witnessing extension E through iteration $i - 1$ of the algorithm (SAT.GETMODEL() on line 18). Now, instead of immediately calling the SAT solver at iteration i, we can first check whether the witness extension found in the previous iteration is also an extension of F_i, in which case we can directly proceed to iteration $i + 1$. In practice, however, this check can also be implemented as a linear-time (in practice very fast) check by calling the SAT solver under further assumptions enforcing the witnessing extension E (on lines 4–9). By doing so, satisfiability is

established through mere deterministic polynomial-time propagation within the SAT solver.

In the negative case, the SAT solver can provide an unsatisfiable core C, i.e., an explanation of which $r_{a,b}$ and $\neg r_{a,b}$ assumed at the previous iteration $i-1$ the solver needed to contruct an unsatisfiability proof (SAT.GETUNSATCORE() on line 22). Before calling the SAT solver at iteration i, we check whether the call is necessary (lines 10–15 in Algorithm 1); if the change $\chi_i = \pm(a,b)$ in the attack structure from iteration $i-1$ to i does not involve an $(\neg)r_{a,b}$ in the unsatisfiable core C, the core also witnesses the fact that the SAT solver call at i will necessarily report unsatisfiability, and hence at iteration i no SAT solver call is needed.[3]

Skeptical acceptance under preferred. Recall that skeptical acceptance for $\sigma = prf$ is in general a Π_2^{P}-complete problem. Hence a single SAT solver call as performed at each iteration of Algorithm 1 is not enough in this case. To cover this Π_2^{P}-complete problem, we generalize Algorithm 1 with an incremental adaptation of the counterexample-guided abstraction refinement (CEGAR) procedure of [26] originally proposed for static acceptance to dynamic argumentation frameworks. A particular non-trivial part of this generalization is that care needs to be taken in order to make sure that the so-called blocking clauses, which exclude subsets of the current extension, added (as in [26] for the static case) at a particular iteration, will not interfere with correctness of results for the forthcoming iterations.

The resulting algorithm is outlined in pseudocode as Algorithm 2. On line 1 we initialize an empty set E for storing the potential counterexample extension found during the previous iteration, set B for storing additional assumptions needed for deactivating blocking clauses from previous iterations, and array s for storing intermediate results. We initialize ϕ to $\phi_{com}(F^\chi)$ corresponding to the encoding for complete semantics on line 2—throughout the algorithm, we will keep updating ϕ via adding clauses, and calling the SAT solver with ϕ and corresponding assumptions α. Each iteration i on lines from 15 to 37 follows [26]. We include the literal $\neg x_a$ to the assumptions $\textsc{Att}(F_i)$ corresponding to the attack structure (line 15), with the goal of finding a counterexample to skeptical acceptance under preferred, that is, a preferred extension not containing the query argument. Then, we iteratively ask a SAT solver for a complete extension not containing the query argument (line 17). If one is not found, we accept the argument, since then it is contained in all preferred extensions (lines 18–21). Otherwise, we iteratively subset-maximize the complete extension (extracted from the truth assignment on lines 22–23) using the same SAT solver (lines 24–31), resulting in a counterexample candidate. Then, if further including the query argument (via including x_a as an assumption on line 32) into the counterexample candidate results in the subset not being a complete extension of F_i, the counterexample is valid, i.e., it is a preferred extension not containing the query argument, and hence we reject (lines 33–37). Otherwise, we continue by again asking for a complete extension not containing the query argument, this time ruling out all subsets of the previously found complete extension using the blocking clauses added to the solver during subset-maximization (line 26).

Algorithm 2 also includes an optimization for the case where a was not skeptically accepted during the previous iteration, i.e., a counterexample extension E was found during iteration $i-1$. At iteration i, we first check whether the counterexample extension is a complete extension of F_i (lines 5–6), and if it is, whether there is a superset containing the counterexample extension (lines 7–14). In this case, we may directly proceed to iteration $i+1$ as the counterexample is still valid, and we have potentially saved resources via a single polynomial-time propagation check (line 6) and a single SAT call (line 10), as opposed to making multiple calls iteratively.

To make the approach fully incremental for dynamic argumentation frameworks, we need to ensure that we do not consider blocking clauses added during the previous iterations; otherwise correctness will be lost due to the earlier blocking clauses ruling out counterexample candidates wrt. the current AF F_i. We ensure this critical detail via declaring additional Boolean variables b_i, $i = 0, \ldots, n$, with the interpretation $b_i = \top$ iff we are at iteration i. At each iteration, we use the literals $\neg b_0, \neg b_1, \ldots, \neg b_{i-1}$ (lines 13, 20, and 36) and b_i (lines 8 and 15) as assumptions and condition each blocking clause with b_i (lines 9 and 26). This ensures that all blocking clauses added

Algorithm 2 Dynamic skeptical acceptance under preferred semantics on input $F^\chi = (A, R, \chi)$ and query $q \in A$.

```
1: E ← ∅, B ← ∅, s ← empty Boolean array of size n + 1
2: φ ← φ_com(F^χ)
3: for i = 0, 1, ..., n
4:     if i > 0 and s[i − 1] = 0
5:         α ← Att(F_i) ∪ {x_a | a ∈ E} ∪ {¬x_a | a ∉ E} ∪ B
6:         ρ ← SAT.solve(φ, α)
7:         if ρ = sat
8:             α ← Att(F_i) ∪ {x_a | a ∈ E} ∪ B ∪ {b_i}
9:             φ ← φ ∧ (b_i → ⋁_{a∈A\E} x_a)
10:            ρ ← SAT.solve(φ, α)
11:            if ρ = unsat
12:                s[i] ← 0
13:                B ← B ∪ {¬b_i}
14:                continue
15:    α ← Att(F_i) ∪ {¬x_q} ∪ B ∪ {b_i}
16:    while true
17:        ρ ← SAT.solve(φ, α)
18:        if ρ = unsat
19:            s[i] ← 1
20:            B ← B ∪ {¬b_i}
21:            break
22:        τ ← SAT.getModel()
23:        E ← {a ∈ A | τ(x_a) = ⊤}
24:        while true
25:            α ← α ∪ {x_a | a ∈ E}
26:            φ ← φ ∧ (b_i → ⋁_{a∈A\E} x_a)
27:            ρ ← SAT.solve(φ, α)
28:            if ρ = sat
29:                τ ← SAT.getModel()
30:                E ← {a ∈ A | τ(x_a) = ⊤}
31:            else break
32:        α ← (α \ {¬x_q}) ∪ {x_q}
33:        ρ ← SAT.solve(φ, α)
34:        if ρ = unsat
35:            s[i] ← 0
36:            B ← B ∪ {¬b_i}
37:            break
38: for i = 0, 1, ..., n
39:     if s[i] = 1 print "YES" else print "NO"
```

[3] In practice, the unsatisfiable core does not need to be checked explicitly, since it in general holds that if the core is valid for iteration i, then the SAT solver call at iteration i will consist of unit propagation in linear time deriving unsatisfiability.

Table 1. Cumulative running times (s) with timeouts included as 1800 s, number of timeouts (out of 326 instances) in brackets.

		Number of changes											
Task	**Solver**	8		16		32		64		128		256	
DC-CO-D	μ-TOKSIA	**115.03**	(0)	**140.78**	(0)	**192.45**	(0)	**366.50**	(0)	**950.08**	(0)	**2630.54**	(**0**)
	COQUIAAS	271.02	(0)	344.36	(0)	494.77	(0)	890.87	(0)	2353.99	(0)	7262.56	(1)
DC-ST-D	μ-TOKSIA	**81.44**	(0)	**93.18**	(0)	**113.64**	(0)	**185.22**	(0)	**426.70**	(0)	**1189.93**	(0)
	COQUIAAS	128.95	(0)	164.82	(0)	227.86	(0)	382.99	(0)	891.98	(0)	3812.97	(0)
DS-ST-D	μ-TOKSIA	**180.94**	(0)	**199.55**	(**0**)	**321.39**	(**0**)	**642.84**	(**0**)	**1729.48**	(**0**)	**5037.50**	(**0**)
	COQUIAAS	2172.89	(0)	2546.26	(1)	3499.76	(1)	5735.87	(1)	10178.92	(2)	23198.09	(6)
DS-PR-D	μ-TOKSIA	**356.54**	(0)	**483.88**	(0)	**1171.59**	(**0**)	**3006.67**	(**0**)	**6097.49**	(**1**)	**17856.47**	(**2**)
	COQUIAAS	4798.30	(0)	7808.13	(0)	14856.26	(4)	23279.47	(6)	39405.65	(14)	59949.88	(26)

at previous iterations are trivially satisfied, and that blocking clauses added during the current iteration are active.

5.2 Extension Enumeration

The approaches outlined in the previous section for deciding acceptance over dynamic argumentation frameworks can be readily adjusted also for computing a single or all extensions of a given dynamic framework $F^\chi = (A, R, \chi)$ without a query argument. In particular, for computing a single extension at each iteration, we can disregard the query argument a in Algorithm 1. In this case, the negative check (using an unsatisfiable core) is only applicable in the case of stable semantics, since the other semantics are guaranteed to yield at least one extension. For preferred semantics, we use the same procedure as for complete semantics, but in addition iteratively subset-maximize the extension E that was found using blocking clauses of the form

$$\bigwedge_{a \in E} (b_i \rightarrow x_a) \wedge \left(b_i \rightarrow \bigvee_{a \in A \setminus E} x_a \right),$$

where b_i is again a fresh Boolean variable corresponding to iteration i. Each solver call at iteration i is then performed under assumptions $\neg b_0, \neg b_1, \ldots, b_i$, which again ensure that the blocking clauses added during the previous iterations are trivially satisfied.

Generalizing to extension enumeration, for $\sigma \in \{com, stb\}$ at iteration i after each found extension E we add the blocking clause

$$b_i \rightarrow \bigvee_{a \in E} x_a \vee \bigvee_{a \in A \setminus E} x_a$$

and continue to query the SAT solver for a further extension until reaching unsatisfiability. At each iteration i, each SAT solver call call is performed under the assumptions $\neg b_0, \neg b_1, \ldots, b_i$ to avoid issues with blocking clauses at previous iterations. For extension enumeration under preferred semantics, at each iteration we further subset-maximize a complete extension similarly as in the procedure for skeptical acceptance under preferred semantics (Algorithm 2), adding blocking clauses

$$b_i \rightarrow \bigvee_{a \in A \setminus E} x_a$$

for each preferred extension E.

6 EXPERIMENTS

We present empirical results on the performance of an implementation of the incremental SAT-based approach to reasoning over dynamic argumentation frameworks described in the previous section. The implementation, named μ-TOKSIA[4], uses Glucose (version 4.1) [5] as the SAT solver, making use of its incremental mode [4]. The system is available online in open source at `https://bitbucket.org/andreasniskanen/mu-toksia`.

6.1 Benchmarks and Setup

As a basis of the evaluation, we used the 326 AFs and the query arguments and attack structure changes from the dynamic track of ICCMA 2019 (`https://www.iccma2019.dmi.unipg.it`). In the competition, 8 attack structure changes (modifications) were considered for each AF. We extend our evaluation to up to 256 changes for obtaining a more general view of scalability by appending uniformly at random selected additions and deletions of attacks to the ICCMA 2019 change lists. Due to space constraints, here we focus on the arguably central NP-hard acceptence problems of the dynamic track, i.e., the problems DC-CO-D (credulous acceptance under complete), DC-ST-D (credulous stable), DS-ST-D (skeptical stable) and DS-PR-D (skeptical preferred).

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

We compare the performance of μ-TOKSIA to that of the currently available system COQUIAAS supporting the computational tasks of the dynamic track of ICCMA 2019. While—in addition to μ-TOKSIA and COQUIAAS—the PYGLAF solver [3] also participated in the dynamic track of ICCMA 2019, we observed that it reported—inline with the ICCMA 2019 results—wrong results on 13 (DC-CO-D), 13 (DC-ST-D), 24 (DS-ST-D), and 150 (DS-PR-D) out of the total of 326 benchmark instances with 8 changes, and therefore are unable to present a meaning comparison to PYGLAF at this time. Furthermore, we note that the prototype system presented in [2] appears to implement the DS-PR-D task of ICCMA 2019; however, despite our efforts we were unable to obtain an implementation of the system from the authors for evaluation.

6.2 Results

A comparison between μ-TOKSIA and COQUIAAS [32] is summarized in Table 1 and Figure 3. Overall, the results show a similar tendency for each of the problem variants, μ-TOKSIA significantly outperforming COQUIAAS. As seen from Table 1, μ-TOKSIA times out only on 3 instances with 128 and 256 changes on skeptical acceptance under preferred, while COQUIAAS has several timeouts over several tasks. We observe that skeptical acceptance under preferred results in highest running times for both of the systems, which

[4] *Muutoksia* means "changes" in Finnish.

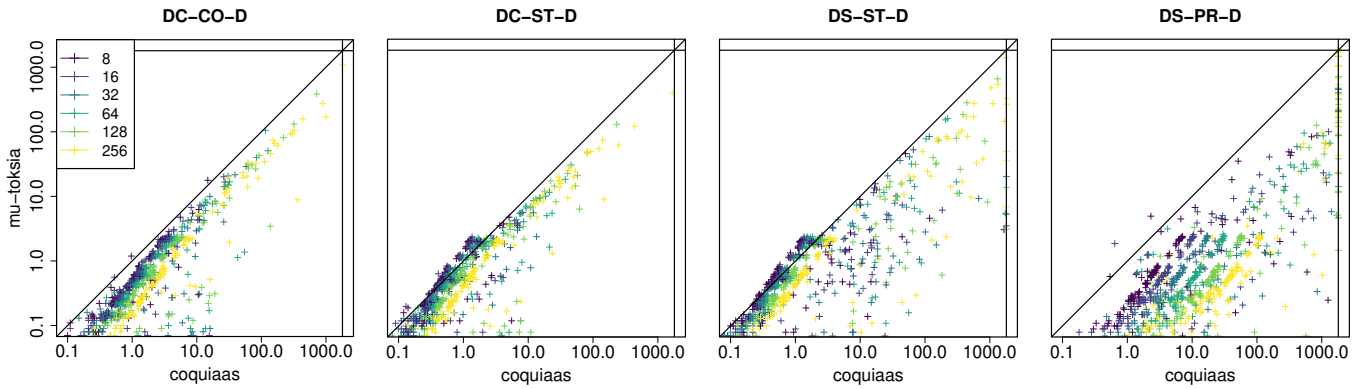

Figure 3. μ-TOKSIA vs. COQUIAAS on DC-CO-D, DC-ST-D, DS-ST-D, and DS-PR-D.

is in-line with the increased complexity of this problem variant compared to the other variants. However, as seen in Figure 3 (right), the running times of μ-TOKSIA are not considerably influenced by the number of changes, while running times of COQUIAAS increase as more changes are considered. We expect this behavior of μ-TOKSIA to be a result of the fully incremental SAT-based approach, which pays off in particular for the computationally hardest problem variant. Furthermore, as highlighted in Figure 4, the "positive" check of the counterexample extension obtained at the previous iteration being a valid counterexample for the current iteration (recall Section 5) provides noticeable runtime improvements. This can be seen in Figure 4, providing a comparison between the performance of μ-TOKSIA with (y-axis) and without (x-axis) the positive check. At times, the improvements due to the positive check are drastic; dropping from a runtime of 1000 seconds to just over 10 seconds for the same instance.

7 RELATED WORK

Dynamics in abstract argumentation has raised increasing attention during the recent years [22]. Algorithms for determining a single extension [1] and the skeptical acceptance of an argument under preferred semantics [2] in dynamic argumentation frameworks have been recently developed, building on previous work on incremental computation of the grounded and ideal extension in dynamic AFs [28, 29]. In contrast to our work, the authors develop solver-independent techniques by considering arguments that have potentially been affected by the update and using any solver for the computation over a smaller AF. Our approach is purely SAT-based, relying on the power of incremental solving instead of external calls, and can support essentially all major semantics. Likewise, the division-based method of [34], further refined in [8], incrementally computes extensions of an updated AF, using a notion of the affected part of the arguments and reusing the extensions of the original AF; however, no implementation or experimental evaluation is provided. Changes to AFs in the form of adding and removing attacks or arguments have also been considered from purely representational pespectives, both in the form of atomic changes [18, 14, 16, 17, 35] and changes involving sets of arguments and incident attacks [12, 10, 11]. Finally, we mention that the approach to dynamic argumentation frameworks presented in this work builds on the earlier success of SAT-based approaches to reasoning over static AFs [26, 33, 19].

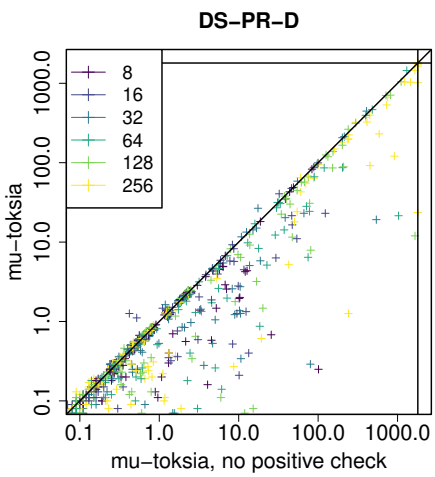

Figure 4. μ-TOKSIA on task DS-PR-D: with (y-axis) vs without (x-axis) positive check.

8 CONCLUSIONS

Temporal changes to argumentation frameworks are inherent in various practical settings, such as disputes among agents in social networks. Motivated by this, we detailed a fully incremental Boolean satisfiability based approach to credulous and skeptical acceptance, computing a single extension, and extension enumeration under a sequence of changes to the attack structure of a given argumentation framework. Our SAT-based approach covers all of the three semantics (complete, preferred, stable) for which a static acceptance problem is NP-hard, as well as the polynomial-time computable case of grounded semantics through guaranteed polynomial-time propagation on the level of propositional encodings, hence covering all of the computational tasks featured in the special dynamic track that was organized as part of the 3rd International Competition on Computational Models of Argumentation (ICCMA 2019). We described encodings for the dynamic case and algorithms which make full use of the assumptions interface of a SAT solver as well as optimizations to avoid making redundant SAT solver calls based on knowledge obtained from previous calls. We showed through an extensive empirical evaluation, extending the ICCMA 2019 dynamic track benchmarks to higher numbers of changes, that the approach is highly competitive in practice.

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] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi, 'Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach', in *Proc. IJCAI*, ed., Carles Sierra, pp. 49–55. ijcai.org, (2017).

[2] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi, 'An efficient algorithm for skeptical preferred acceptance in dynamic argumentation frameworks', in *Proc. IJCAI*, ed., Sarit Kraus, pp. 18–24. ijcai.org, (2019).

[3] Mario Alviano. The pyglaf argumentation reasoner. `https://www.iccma2019.dmi.unipg.it/papers/ICCMA19_paper_6.pdf`.

[4] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon, 'Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction', in *Proc. SAT*, eds., Matti Järvisalo and Allen Van Gelder, volume 7962 of *Lecture Notes in Computer Science*, pp. 309–317. Springer, (2013).

[5] Gilles Audemard and Laurent Simon, 'On the Glucose SAT solver', *Int. J. Artif. Intell. Tools*, **27**(1), 1–25, (2018).

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

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

[8] Pietro Baroni, Massimiliano Giacomin, and Beishui Liao, 'On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: A division-based method', *Artif. Intell.*, **212**, 104–115, (2014).

[9] Ringo Baumann, 'Splitting an argumentation framework', in *Proc. LPNMR*, eds., James P. Delgrande and Wolfgang Faber, volume 6645 of *Lecture Notes in Computer Science*, pp. 40–53. Springer, (2011).

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

[11] Ringo Baumann, 'Context-free and context-sensitive kernels: Update and deletion equivalence in abstract argumentation', in *Proc. ECAI*, eds., Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 63–68. IOS Press, (2014).

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

[13] Philippe Besnard and Sylvie Doutre, 'Checking the acceptability of a set of arguments', in *Proc. NMR*, eds., James P. Delgrande and Torsten Schaub, pp. 59–64, (2004).

[14] Pierre Bisquert, Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex, 'Change in argumentation systems: Exploring the interest of removing an argument', in *Proc. SUM*, eds., Salem Benferhat and John Grant, volume 6929 of *Lecture Notes in Computer Science*, pp. 275–288. Springer, (2011).

[15] Stefano Bistarelli, Lars Kotthoff, Francesco Santini, and Carlo Taticchi, 'Containerisation and dynamic frameworks in ICCMA'19', in *Proc. COMMA*, eds., Matthias Thimm, Federico Cerutti, and Mauro Vallati, volume 2171 of *CEUR Workshop Proceedings*, pp. 4–9. CEUR-WS.org, (2018).

[16] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre, 'Dynamics in argumentation with single extensions: Abstraction principles and the grounded extension', in *Proc. ECSQARU*, eds., Claudio Sossai and Gaetano Chemello, volume 5590 of *Lecture Notes in Computer Science*, pp. 107–118. Springer, (2009).

[17] Guido Boella, Souhila Kaci, and Leendert W. N. van der Torre, 'Dynamics in argumentation with single extensions: Attack refinement and the grounded extension (extended version)', in *Proc. ArgMAS*, eds., Peter McBurney, Iyad Rahwan, Simon Parsons, and Nicolas Maudet, volume 6057 of *Lecture Notes in Computer Science*, pp. 150–159. Springer, (2009).

[18] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex, 'Change in abstract argumentation frameworks: Adding an argument', *J. Artif. Intell. Res.*, **38**, 49–84, (2010).

[19] Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati, 'How we designed winning algorithms for abstract argumentation and which insight we attained', *Artif. Intell.*, **276**, 1–40, (2019).

[20] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis, 'Symmetric argumentation frameworks', in *Proc. ECSQARU*, ed., Lluis Godo, volume 3571 of *Lecture Notes in Computer Science*, pp. 317–328. Springer, (2005).

[21] Yannis Dimopoulos and Alberto Torres, 'Graph theoretical structures in logic programs and default theories', *Theor. Comput. Sci.*, **170**(1-2), 209–244, (1996).

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

[23] 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).

[24] Paul E. Dunne and Trevor J. M. Bench-Capon, 'Coherence in finite argument systems', *Artif. Intell.*, **141**(1/2), 187–203, (2002).

[25] 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).

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

[27] Niklas Eén and Niklas Sörensson, 'Temporal induction by incremental SAT solving', *Electr. Notes Theor. Comput. Sci.*, **89**(4), 543–560, (2003).

[28] Sergio Greco and Francesco Parisi, 'Efficient computation of deterministic extensions for dynamic abstract argumentation frameworks', in *Proc. ECAI*, eds., Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 1668–1669. IOS Press, (2016).

[29] Sergio Greco and Francesco Parisi, 'Incremental computation of deterministic extensions for dynamic argumentation frameworks', in *Proc. JELIA*, eds., Loizos Michael and Antonis C. Kakas, volume 10021 of *Lecture Notes in Computer Science*, pp. 288–304. Springer, (2016).

[30] Nadin Kökciyan, Nefise Yaglikci, and Pinar Yolum, 'An argumentation approach for resolving privacy disputes in online social networks', *ACM Trans. Internet Techn.*, **17**(3), 27:1–27:22, (2017).

[31] Markus Kröll, Reinhard Pichler, and Stefan Woltran, 'On the complexity of enumerating the extensions of abstract argumentation frameworks', in *Proc. IJCAI*, ed., Carles Sierra, pp. 1145–1152. ijcai.org, (2017).

[32] Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly. CoQuiAAS v3.0 ICCMA 2019 solver description. `https://www.iccma2019.dmi.unipg.it/papers/ICCMA19_paper_5.pdf`.

[33] Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly, 'CoQuiAAS: A constraint-based quick abstract argumentation solver', in *Proc. ICTAI*, pp. 928–935. IEEE Computer Society, (2015).

[34] Bei Shui Liao, Li Jin, and Robert C. Koons, 'Dynamics of argumentation systems: A division-based method', *Artif. Intell.*, **175**(11), 1790–1814, (2011).

[35] Tjitze Rienstra, Chiaki Sakama, and Leendert W. N. van der Torre, 'Persistence and monotony properties of argumentation semantics', in *Proc. TAFA*, eds., Elizabeth Black, Sanjay Modgil, and Nir Oren, volume 9524 of *Lecture Notes in Computer Science*, pp. 211–225. Springer, (2015).

[36] Nicolás D. Rotstein, Martín O. Moguillansky, Alejandro Javier García, and Guillermo Ricardo Simari, 'A dynamic argumentation framework', in *Proc. COMMA*, eds., Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pp. 427–438. IOS Press, (2010).

[37] Matthias Thimm and Serena Villata, 'The first international competition on computational models of argumentation: Results and analysis', *Artif. Intell.*, **252**, 267–294, (2017).