μ–toksia: An Efficient Abstract Argumentation Reasoner

Andreas Niskanen and **Matti Järvisalo**
HIIT, Department of Computer Science, University of Helsinki, Finland

Abstract

We describe the μ–toksia argumentation reasoning system. The system supports a range of different reasoning tasks over both standard and dynamic abstract argumentation frameworks under essentially all central argumentation semantics, covering all tracks and reasoning tasks considered in the most recent International Competition on Computational Models of Argumentation (ICCMA 2019). μ–toksia ranked first in all reasoning tasks in the main track of ICCMA 2019, and has been shown to scale noticeably better on the dynamic track tasks than its current competitors. In this paper, we provide an overview of μ–toksia and its algorithmic and implementation-level details, and provide further empirical evidence beyond ICCMA 2019 on the efficiency of μ–toksia compared to related systems.

1 Introduction

Argumentation is a central area of knowledge representation and reasoning research. Abstract argumentation (Dung 1995), with Dung's argumentation frameworks (AFs) as the key formalism, is one of the main directions in argumentation research. Despite the simplistic syntactic form of AFs, central reasoning tasks in AFs such as credulous and skeptical acceptance of arguments, and extension enumeration, are computationally difficult under various argumentation semantics (Dvorák and Dunne 2018). Various practical algorithmic approaches for reasoning in AFs have been developed (Charwat et al. 2015; Cerutti et al. 2018). The biennial series of International Competitions on Computational Models of Argumentation (ICCMA) (Thimm and Villata 2017; Gaggl et al. 2020), with its 3rd instantiation in 2019 (Bistarelli et al. 2018), provides further incentives for developing efficient argumentation reasoning systems. ICCMA 2019 expanded the range of reasoning tasks by introducing a so-called dynamic track, generalizing the standard acceptance and enumeration tasks to sequences of related tasks resulting from iteratively changing the attack structure of the AF on which reasoning is performed (Liao, Jin, and Koons 2011; Baroni, Giacomin, and Liao 2014; Alfano, Greco, and Parisi 2017; Alfano, Greco, and Parisi 2019).

This paper describes the μ–toksia argumentation reasoning system. The system supports a range of different reasoning tasks over both standard and dynamic abstract argumentation frameworks under essentially all central argumentation semantics, covering all tracks and reasoning tasks considered in ICCMA 2019. Motivated by various successes in employing declarative solving for reasoning in AFs and their generalizations (Dvorák et al. 2014; Cerutti, Giacomin, and Vallati 2019; Alviano 2019; Lagniez, Lonca, and Mailly 2015; Egly, Gaggl, and Woltran 2010; Gaggl et al. 2015; Bistarelli, Rossi, and Santini 2017) as a fundamental choice of attack μ–toksia makes heavy use of state-of-the-art Boolean satisfiability (SAT) solvers in optimized ways. In terms of empirical performance, μ–toksia arguably represents the current state of the art in abstract argumentation reasoners: it ranked first in all reasoning tasks in the main track of ICCMA 2019. Furthermore, in an extended evaluation of the system on the tasks of the ICCMA 2019 dynamic track, μ–toksia has been recently shown to scale noticeably better than its current competitors (Niskanen and Järvisalo 2020).

In this paper, we provide an overview of μ–toksia and its various algorithmic and implementation-level details. While the algorithmic approach implemented in μ–toksia for the dynamic track tasks is described in a separate work (Niskanen and Järvisalo 2020), here we describe details on how μ–toksia implements the standard "static" tasks, and provide extensive empirical results beyond ICCMA 2019 benchmarks on the efficiency of μ–toksia compared to competing systems (including ones not participating in ICCMA 2019).

2 Abstract Argumentation

An *argumentation framework (AF)* is a pair $F = (A, R)$, where A is a (finite) set of arguments and $R \subseteq A \times A$ is a set of attacks. An argument $a \in A$ is defended by a set $S \subseteq A$ if for all $(b, a) \in R$ there is $c \in S$ with $(c, b) \in R$. For $S \subseteq A$, define $\mathcal{D}_F(S) = \{a \in A \mid a \text{ is defended by } S\}$. The range of $S \subseteq A$ is $\mathcal{R}_F(S) = S \cup \{a \in A \mid (b, a) \in R, b \in S\}$.

Argumentation semantics (Baroni, Caminada, and Giacomin 2018) define sets of jointly acceptable arguments called extensions. Given an AF $F = (A, R)$, a set $S \subseteq A$ is *conflict-free* if there are no $x, y \in S$ with $(x, y) \in R$. We denote the collection of conflict-free sets of F as $cf(F)$. For a conflict-free set $S \in cf(F)$, we denote

- $S \in \mathit{adm}(F)$ if $S \subseteq \mathcal{D}_F(S)$, $S \in \mathit{com}(F)$ if $S = \mathcal{D}_F(S)$,
- $S \in \mathit{prf}(F)$ if $S \in \mathit{com}(F)$ and there is no $S' \in \mathit{com}(F)$ with $S' \supset S$,
- $S \in \mathit{stb}(F)$ if $\mathcal{R}_F(S) = A$,
- $S \in \mathit{sem}(F)$ if $S \in \mathit{com}(F)$ and there is no $S' \in \mathit{com}(F)$ with $\mathcal{R}_F(S') \supset \mathcal{R}_F(S)$,
- $S \in \mathit{stg}(F)$ if $S \in \mathit{cf}(F)$ and there is no $S' \in \mathit{cf}(F)$ with $\mathcal{R}_F(S') \supset \mathcal{R}_F(S)$,
- $S \in \mathit{grd}(F)$ if $S = \bigcap \mathit{com}(F)$,
- $S \in \mathit{id}(F)$ if $S \in \mathit{adm}(F)$, $S \subseteq \bigcap \mathit{prf}(F)$ and there is no $S' \subseteq \bigcap \mathit{prf}(F)$ with $S' \supset S$.

The functions $\mathit{adm}, \mathit{com}, \mathit{prf}, \mathit{stb}, \mathit{sem}, \mathit{stg}, \mathit{grd}, \mathit{id}$ stand for admissible, complete, preferred, stable, semi-stable, stage, grounded, and ideal semantics, respectively. For a semantics σ, a set $E \in \sigma(F)$ is an extension under σ, or a σ-extension. An argument $a \in A$ is credulously (skeptically) accepted under σ if it is in some (all) σ-extension(s).

Apart from deciding credulous/skeptical acceptance and finding a single or all extensions of a given AF, μ–toksia supports the problem setting of the ICCMA 2019 dynamic track (Bistarelli et al. 2018), where in addition to an AF, a sequence of changes to the attack structure of the AF is provided (Alfano, Greco, and Parisi 2017; Alfano, Greco, and Parisi 2019), and the task is to output the answer to the task at hand for all AFs defined by the sequence.

Let $F = (A, R)$ be an AF, and $\chi = (\chi_1, \ldots, \chi_n)$ a sequence 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

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

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

3 μ–toksia

3.1 Overview

The μ–toksia system supports credulous (DC) and skeptical (DS) acceptance of an argument, outputting a single extension (SE), and extension enumeration (EE) under semantics *com*, *prf*, *stb*, *sem*, *stg*, *grd*, and *id* with respect to a given AF (and a query argument), as well as DC, DS, SE, and EE for dynamic AFs under semantics *com*, *prf*, *stb*, and *grd* (as in ICCMA 2019). The system is based heavily on incremental use of a SAT solver (Eén and Sörensson 2003): a SAT solver is instantiated only once during a single execution μ–toksia, and iterative computations are done keeping the state of the SAT solver through its API.

$$\bigwedge_{a \in A} \left(y_a \leftrightarrow \left(\bigvee_{(b,a) \in R^s} x_b \vee \bigvee_{(b,a) \in R^d} s_{b,a} \right) \right)$$

$$\bigwedge_{(a,b) \in R^d} (s_{a,b} \leftrightarrow (r_{a,b} \wedge x_a)) \qquad \bigwedge_{(a,b) \in R^d} (z_{a,b} \leftrightarrow (r_{a,b} \rightarrow y_a))$$

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

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

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

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

Figure 1: SAT encodings used in μ–toksia.

3.2 Algorithmic Details and Optimization

The exact SAT encodings used by μ–toksia are shown in Figure 1, based on standard encodings of argumentation semantics (Besnard and Doutre 2004). Variables x_a indicate that argument $a \in A$ is included in a σ-extension, and y_a that $a \in A$ is attacked by it. Variables $r_{a,b}$ are used to represent dynamic attacks $(a, b) \in R^d$ to cover dynamic tasks, enabling incremental computation, along with $s_{a,b}$ and $z_{a,b}$ variables (Niskanen and Järvisalo 2020). For tasks over fixed AFs ("static" tasks), $R^d = \emptyset$ and the encodings reduce to the standard encodings. We first give semantics-specific details for static tasks and then shortly outline the generalization to dynamic AFs.

Grounded semantics Unit propagation (standard in SAT solvers) on $\phi_{\mathit{com}}(F)$ gives the polytime computable grounded extension (Lagniez, Lonca, and Mailly 2015). Acceptance is decided by checking if x_q for the query argument $q \in A$ is assigned to true after propagation.

Complete semantics NP-complete credulous acceptance of $q \in A$ is decided by invoking the SAT solver on $\phi_{\mathit{adm}}(F) \wedge x_q$. Skeptical acceptance is decided via grounded, and the grounded extension gives a single extension. For extension enumeration, all models of $\phi_{\mathit{com}}(F)$ are enumerated, blocking each found extension E by adding the clause $\bigvee_{a \in E} \neg x_a \vee \bigvee_{a \in A \setminus E} x_a$ until the SAT solver reports unsatisfiability.

Stable semantics We first compute the grounded extension E_{grd}, as it is a subset of every stable extension. Credulous acceptance of $q \in A$ is decided on $\phi_{\mathit{stb}}(F) \wedge \bigwedge_{a \in E_{\mathit{grd}}} x_a \wedge x_q$, skeptical acceptance by unsatisfiability of $\phi_{\mathit{stb}}(F) \wedge \bigwedge_{a \in E_{\mathit{grd}}} x_a \wedge \neg x_q$. Single extension and extension enumeration is performed on $\phi_{\mathit{stb}}(F) \wedge \bigwedge_{a \in E_{\mathit{grd}}} x_a$.

Preferred semantics NP-complete credulous acceptance is decided by credulous acceptance under complete. For Π_2^p-complete skeptical acceptance, we implement the iterative SAT-based CEGAR approach of (Dvorák et al. 2014) (without so-called shortcuts in contrast to Cegartix).

Semi-stable and stage semantics We define Boolean variables $\bigwedge_{a \in A}(r_a \leftrightarrow (x_a \vee y_a))$ representing the range of the extension determined by the x_a variables. For credulous and skeptical acceptance, we implement the CEGAR algorithms from (Dvorák et al. 2014), omitting all shortcuts. For stage semantics, we first check if a stable extension exists using assumptions $\bigwedge_{a \in A}\ r_a$, and in the positive case instead invoke the algorithm for stable semantics. For a single semi-stable extension, we subset-maximize a complete extension (resp. conflict-free set for stage) with respect to the range. For enumeration, at each iteration we assume the range after subset-maximization, iteratively enumerate all complete (resp. conflict-free) extensions with this range, and iterate after adding a clause blocking all subsets of this range.

Ideal semantics The union of admissible extensions is obtained on $\phi_{com}(F)$, adding clauses $\bigvee_{a \in A \setminus E} x_a$ for each extension found. If the query is not in the union, we reject. Then, we compute the arguments in the union not attacked by the union. If the query is not in this set, we reject. Then, we assume that all arguments outside this set are not in an extension, and iteratively subset-maximize a complete extension within this set. This corresponds exactly to the ideal extension (Dunne, Dvorák, and Woltran 2013; Wallner, Weissenbacher, and Woltran 2013).

Generalization to dynamic AFs The $r_{a,b}$ variables condition standard SAT encodings for each dynamic attack $(a, b) \in R^d$, and are used as assumptions through the SAT solver API to switch such attacks on-off (present-absent) during iterative computations on dynamic AFs. This allows for using the same techniques as just-described for static AFs also on dynamic AFs. The algorithmic details for the dynamic tasks are fully detailed in (Niskanen and Järvisalo 2020).

3.3 Implementation, Availability and Usage

The system, implemented in C++ and using standard STL data structures (apart from employing a hash function from the Boost library), is available under the open-source MIT licence at https://bitbucket.org/andreasniskanen/mu-toksia. Its design aims at avoiding applications of specialized algorithms to cover different types of special cases, and makes extensive use of SAT solver APIs. The system includes interfaces to the Glucose (Audemard and Simon 2018) and CryptoMiniSAT (Soos, Nohl, and Castelluccia 2009) SAT solvers. A generic SAT solver interface `SATSolver.h` is available for integrating any SAT solver with an assumptions interface.

Usage μ–toksia is invoked via command line by

```
./mu-toksia -p <task> -f <file> -fo <format>
            [-a <query>] [-m <modfile>]
```

where `<task>` is the task (e.g., `DS-PR` for skeptical acceptance under *prf*, or `DC-CO-D` for credulous acceptance under *com* on a dynamic AF), `<file>` the input AF filename, `<format>` either `apx` or `tgf`, `<query>` the query argument (for acceptance), and `<modfile>` the file with changes to the attack structure (for dynamic tasks).

Table 1: Number of solved instances, number of times contributed to VBS, cumulative time over solved instances for all solvers.

task	solver	# solved	VBS	time
DC-CO	argmat-sat	308	104	13390.86
	ArgSemSAT	305	94	7366.16
	Aspartix	317	10	16588.00
	Cegartix	309	104	13353.49
	CoQuiAAS	307	65	7450.00
	pyglaf	313	0	18313.79
	μ–toksia	**325**	**222**	15395.26
DC-ST	argmat-sat	315	**183**	13753.50
	ArgSemSAT	302	83	17039.21
	Aspartix	**326**	65	13898.51
	Cegartix	308	86	16105.28
	CoQuiAAS	316	114	11079.49
	pyglaf	318	1	15165.57
	μ–toksia	**326**	133	14697.66
DS-ST	ArgSemSAT	287	85	44126.20
	Aspartix	**316**	72	19148.97
	Cegartix	292	71	32495.35
	CoQuiAAS	291	**171**	11816.92
	pyglaf	300	2	25002.93
	μ–toksia	312	141	21892.89
DS-PR	argmat-sat	300	116	20704.71
	ArgSemSAT	318	93	14411.45
	Aspartix	299	2	10160.48
	Cegartix	302	25	43282.10
	CoQuiAAS	277	70	17303.92
	pyglaf	287	44	2064.21
	μ–toksia	**326**	**194**	22819.06
DC-SST	argmat-sat	301	133	17587.45
	ArgSemSAT	312	88	20558.02
	Aspartix	256	0	13949.41
	Cegartix	330	64	25046.95
	CoQuiAAS	246	31	25147.22
	pyglaf	276	2	28572.59
	μ–toksia	**336**	**235**	7464.34
DS-SST	argmat-sat	304	123	17782.17
	ArgSemSAT	313	93	22485.71
	Aspartix	254	2	11638.68
	Cegartix	301	18	42691.19
	CoQuiAAS	270	54	18407.72
	pyglaf	278	53	3363.16
	μ–toksia	**327**	**216**	23720.55
DC-STG	argmat-sat	273	127	25299.99
	Aspartix	232	8	6159.26
	Cegartix	284	17	48685.60
	CoQuiAAS	205	28	23186.74
	pyglaf	230	16	6648.47
	μ–toksia	**317**	**250**	19518.67
DS-STG	argmat-sat	311	152	15427.73
	Aspartix	231	2	4689.58
	Cegartix	278	2	42670.30
	CoQuiAAS	252	30	23817.32
	pyglaf	247	42	1921.85
	μ–toksia	**314**	**207**	17605.09
DC-ID	argmat-sat	239	26	32990.52
	Aspartix	319	6	36913.05
	Cegartix	300	35	26760.89
	CoQuiAAS	283	59	18201.06
	pyglaf	298	23	28364.07
	μ–toksia	**327**	**254**	23388.07

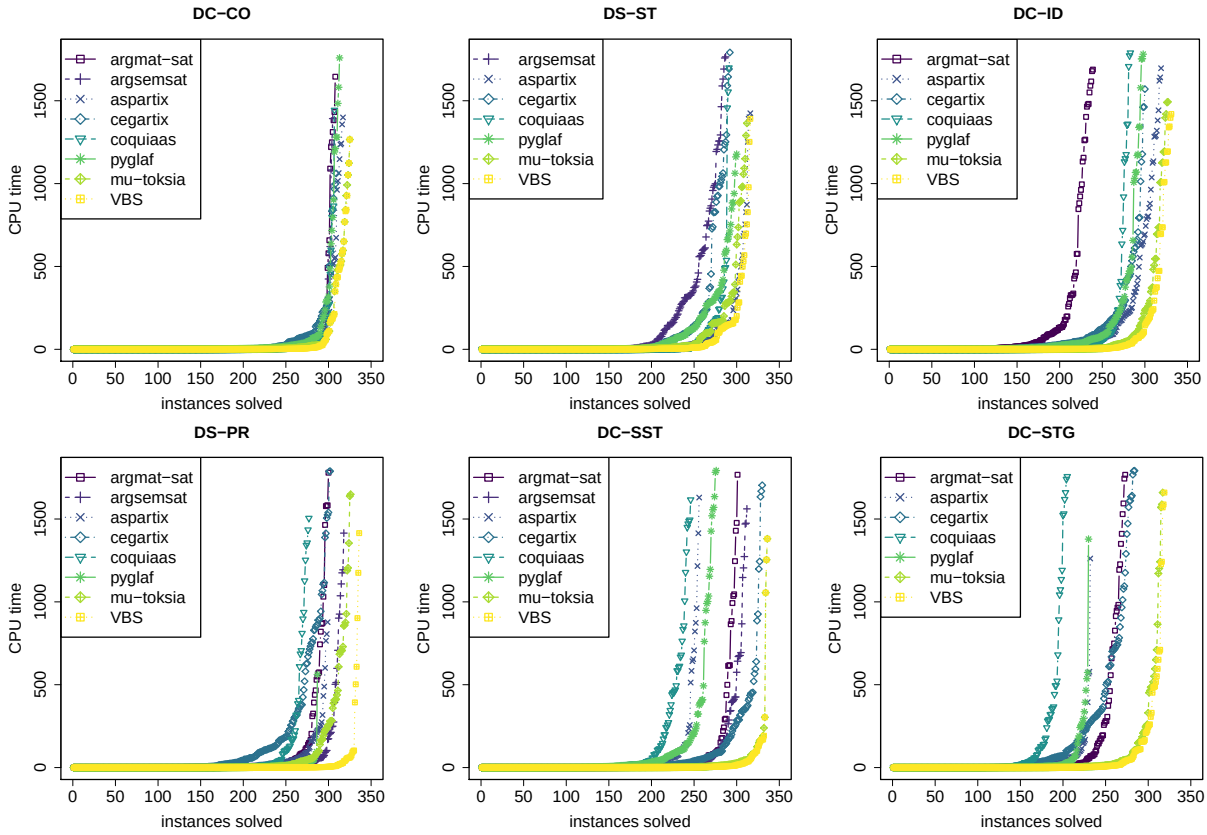

Figure 2: Performance comparison on selected tasks.

Input formats μ–toksia accepts AF files as input in the standard APX and TGF formats. APX uses predicates `arg` and `att` to represent arguments and attacks. In TGF each argument is represented by a string separated by whitespace, then `#` as delimiter, followed by the attacks separated by whitespace. Dynamic changes to the attack structure are represented with `+att(a,b).` and `-att(a,b).` in APX, and `+a b` and `-a b` in TGF.

4 Empirical Evaluation

In ICCMA 2019, μ–toksia consistently outperformed all other solvers. However, the ICCMA 2019 instances are very easy for declarative approaches with almost no timeouts in the main track. For a further evaluation of μ–toksia, we report results on acceptance problems on ICCMA 2017 instances which are empirically more challenging. We compare μ–toksia to the ICCMA 2017 top-performing solvers of CO, ST, PR, SST, STG, and ID tasks and the top-3 solvers on these tasks in ICCMA 2019:[1] argmat-sat (ICCMA 2017 version) (Pu, Ya, and Luo 2017), ArgSemSAT (ICCMA 2017 version) (Cerutti, Giacomin, and Vallati 2019), Cegartix (ICCMA 2017 version) (Dvorák et al. 2014), pyglaf (ICCMA 2017 version) (Alviano 2019), Aspartix (version V19-4) (Dvorák et al. 2020), and CoQuiAAS (ICCMA 2019 version) (Lagniez, Lonca, and Mailly 2015). For this evaluation, we use CryptoMiniSAT version 5.6.8 in μ–toksia. CryptoMiniSAT is used with default parameters, with the exception of deciding variables to their positive polarities in the CEGAR algorithms, which yields modest improvements w.r.t. the default. The experiments were run on nodes with 8-core Intel Xeon E5-2670 2.6-GHz CPUs and 64-GB RAM. We set a per-instance 1800-second time (three times longer than the 10-minute ICCMA time limit) and 16-GB memory limit.

The results are summarized in Table 1 (all static tasks) and Figure 2 (selected tasks). Here μ–toksia ranks first on tasks DC-CO, DS-PR, DC-SST, DS-SST, DC-STG, DS-STG, and DC-ID, and ties with Aspartix on DC-ST with respect to the number of solved instances; i.e., the only exception is that Aspartix solves more instances than μ–toksia on DS-ST (316 vs 312). We hypothesize it to be due to the close relationship of stable semantics and answer set (stable model) semantics (Dung 1995). All solvers are essentially on par on DC-CO. Interestingly, on DC-SST, DC-STG, and DC-ID, μ–toksia is in particular very close to the performance of the virtual best solver (VBS, comprised of taking for each instance the best runtime over all solvers). Furthermore, μ–toksia makes the largest contribution to the VBS on all tasks except for *-ST.

[1] On DS-ST argmat-sat reported incorrectly "NO" on 128 instances that have no extensions; we exclude it from the evaluation on DS-ST. pyglaf 2019 is also excluded due to noticeable numbers of incorrect answers across the tasks. Further, Aspartix was incorrect on DS-SST on 2 instances.

5 Conclusion

We described the SAT-based μ–toksia argumentation reasoning system and provided further empirical results (beyond ICCMA 2019 where μ–toksia dominated the competition) on the efficiency of μ–toksia compared to other argumentation reasoners.

Acknowledgments

This work has been financially supported by Academy of Finland (grants 322869, 328718) and University of Helsinki Doctoral Programme in Computer Science (DoCS). Computational resources were provided by Finnish Grid and Cloud Infrastructure (urn:nbn:fi:research-infras-2016072533).

References

Alfano, G.; Greco, S.; and Parisi, F. 2017. Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach. In *Proc. IJCAI*, 49–55. ijcai.org.

Alfano, G.; Greco, S.; and Parisi, F. 2019. An efficient algorithm for skeptical preferred acceptance in dynamic argumentation frameworks. In *Proc. IJCAI*, 18–24. ijcai.org.

Alviano, M. 2019. Argumentation reasoning via circumscription with Pyglaf. *Fundam. Inform.* 167(1-2):1–30.

Audemard, G., and Simon, L. 2018. On the Glucose SAT solver. *Int. J. Artif. Intell. T.* 27(1):1840001:1–1840001:25.

Baroni, P.; Caminada, M.; and Giacomin, M. 2018. Abstract argumentation frameworks and their semantics. In *Handbook of Formal Argumentation*. College Publications. chapter 4, 159–236.

Baroni, P.; Giacomin, M.; and Liao, B. 2014. 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.

Besnard, P., and Doutre, S. 2004. Checking the acceptability of a set of arguments. In *Proc. NMR*, 59–64.

Bistarelli, S.; Kotthoff, L.; Santini, F.; and Taticchi, C. 2018. Containerisation and dynamic frameworks in ICCMA'19. In *Proc. SAFA*, volume 2171 of *CEUR Workshop Proceedings*, 4–9. CEUR-WS.org.

Bistarelli, S.; Rossi, F.; and Santini, F. 2017. A ConArg-based library for abstract argumentation. In *Proc. ICTAI*, 374–381. IEEE Computer Society.

Cerutti, F.; Gaggl, S. A.; Thimm, M.; and Wallner, J. P. 2018. Foundations of implementations for formal argumentation. In *Handbook of Formal Argumentation*. College Publications. chapter 14, 689–767.

Cerutti, F.; Giacomin, M.; and Vallati, M. 2019. How we designed winning algorithms for abstract argumentation and which insight we attained. *Artif. Intell.* 276:1–40.

Charwat, G.; Dvorák, W.; Gaggl, S. A.; Wallner, J. P.; and Woltran, S. 2015. Methods for solving reasoning problems in abstract argumentation - A survey. *Artif. Intell.* 220:28–63.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–358.

Dunne, P. E.; Dvorák, W.; and Woltran, S. 2013. Parametric properties of ideal semantics. *Artif. Intell.* 202:1–28.

Dvorák, W., and Dunne, P. E. 2018. Computational problems in formal argumentation and their complexity. In *Handbook of Formal Argumentation*. College Publications. chapter 13, 631–687.

Dvorák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2014. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.* 206:53–78.

Dvorák, W.; Rapberger, A.; Wallner, J. P.; and Woltran, S. 2020. ASPARTIX-V19 - an answer-set programming based system for abstract argumentation. In *Proc. FoIKS*, volume 12012 of *LNCS*, 79–89. Springer.

Eén, N., and Sörensson, N. 2003. Temporal induction by incremental SAT solving. *Electron. Notes Theor. Comput. Sci.* 89(4):543–560.

Egly, U.; Gaggl, S. A.; and Woltran, S. 2010. Answer-set programming encodings for argumentation frameworks. *Argument & Computation* 1(2):147–177.

Gaggl, S. A.; Manthey, N.; Ronca, A.; Wallner, J. P.; and Woltran, S. 2015. Improved answer-set programming encodings for abstract argumentation. *Theor. Pract. Log. Prog.* 15(4-5):434–448.

Gaggl, S. A.; Linsbichler, T.; Maratea, M.; and Woltran, S. 2020. Design and results of the second international competition on computational models of argumentation. *Artif. Intell.* 279.

Lagniez, J.; Lonca, E.; and Mailly, J. 2015. CoQuiAAS: A constraint-based quick abstract argumentation solver. In *Proc. ICTAI*, 928–935. IEEE Computer Society.

Liao, B. S.; Jin, L.; and Koons, R. C. 2011. Dynamics of argumentation systems: A division-based method. *Artif. Intell.* 175(11):1790–1814.

Niskanen, A., and Järvisalo, M. 2020. Algorithms for dynamic argumentation frameworks: An incremental SAT-based approach. In *Proc. ECAI*, FAIA. IOS Press. To appear.

Pu, F.; Ya, H.; and Luo, G. 2017. argmat-sat: Applying SAT solvers for argumentation problems based on Boolean matrix algebra. http://argumentationcompetition.org/2017/argmat-sat.pdf.

Soos, M.; Nohl, K.; and Castelluccia, C. 2009. Extending SAT solvers to cryptographic problems. In *Proc. SAT*, volume 5584 of *LNCS*, 244–257. Springer.

Thimm, M., and Villata, S. 2017. The first international competition on computational models of argumentation: Results and analysis. *Artif. Intell.* 252:267–294.

Wallner, J. P.; Weissenbacher, G.; and Woltran, S. 2013. Advanced SAT techniques for abstract argumentation. In *Proc. CLIMA*, volume 8143 of *LNCS*, 138–154. Springer.