μ-toksia Participating in ICCMA 2019

Andreas Niskanen and Matti Järvisalo

HIIT, Department of Computer Science,
University of Helsinki, Finland
andreas.niskanen@helsinki.fi,matti.jarvisalo@helsinki.fi

Abstract

We shortly describe the μ-toksia system, a purely SAT-based implementation participating in the 3rd International Competition on Computational Models of Argumentation (ICCMA 2019), supporting both the classical and dynamic reasoning tasks of ICCMA 2019.

1 The μ-toksia System

The μ-toksia system is a purely SAT-based implementation supporting both the classical [5] and dynamic reasoning tasks of ICCMA 2019. By "purely a SAT-based implementation" we mean that essentially all reasoning by the system is performed by calls to a Boolean satisfiability (SAT) solver —including polynomial-time computations, such as the grounded semantics as well as incremental checks for persistence of (non-)solutions under changes in the dynamic tasks. In addition, the system makes use of incremental SAT solving throughout the implementation. This means that a SAT solver is instantiated only once during a single run of the program, allowing for maintaining the state of the solver from one call to another.

In more detail, the μ-toksia system implements SAT-based algorithms for handling the dynamic tasks in the dynamic track. This is achieved by adapting the standard SAT encodings for classical tasks [3] for the dynamic track as follows. As in the classical case, we have variables x_a for each $a \in A$, with $x_a = \top$ iff a is included in the σ-extension encoded by the formula. In addition, for each change $\pm(a, b)$, we condition the relevant parts of the SAT encoding with a fresh variable $r_{a,b}$, interpreting $r_{a,b} = \top$ as including the attack (a, b) in the attack structure. This allows for employing incremental SAT solving for the dynamic track, using the assumptions interface of the SAT solver over the $r_{a,b}$ variables.

Additionally, we use the following optimizations for the dynamic track.

- If the exact same attack structure has already been encountered before, we output the answer obtained during that iteration.
- If at the start of an iteration, the previous SAT solver call was satisfiable, we check whether the assignment is still a valid assignment under the new assumptions, e.g., whether the extension obtained previously is still an extension for the new AF. This is achieved by including the extension as assumptions in a SAT call.
- If at the start of an iteration, the previous SAT solver call was unsatisfiable, we check whether the final conflict clause (unsatisfiable core) expressed in terms of the assumption variables (i.e., the attack structure) still evaluates to false given the current assumptions.

Semantics-dependent details are described in the following.

- *Grounded (GR).* As noted in [7], the grounded extension can be obtained by performing unit propagation on the SAT encoding for complete semantics. We implemented this approach instead of a dedicated polynomial-time algorithm. For acceptance, we simply check whether the positive literal corresponding to the argument has been propagated.

- *Complete (CO) and stable (ST).* Finding an extension under these semantics for a given AF is employed with a single SAT solver call. Similarly, credulous and skeptical acceptance can be decided by adding the unit clause (x_a) or $(\neg x_a)$ to the SAT encoding. Enumeration is implemented by adding a clause C blocking the current variable assignment of the SAT solver after each extension found. In the dynamic case, we instead add the clause $s_i \rightarrow C$, where s_i is a new Boolean variable with the meaning "we are at iteration i". During iteration i, we add the literals $\neg s_1, \ldots, \neg s_{i-1}, s_i$ as assumptions. (We note that skeptical acceptance under complete coincides with credulous acceptance under grounded, the system makes use of this.)
- *Preferred (PR), semi-stable (SST), and stage (STG).* For maximal semantics with higher computational complexity, a counterexample-guided abstraction refinement (CEGAR) procedure has been implemented, in the style of the AF solver CEGARTIX [6]. For the dynamic track (preferred only), clauses blocking subsets of the current extension are conditioned with s_i variables, as in the previous case. (We note that credulous acceptance under preferred coincides with credulous acceptance under complete. Additionally, if a stable extension exists, stable, semi-stable, and stage semantics coincide. The system makes use of these observations.)
- *Ideal (ID).* As noted in [4], the ideal extension is the maximal admissible extension which is not attacked by any admissible extension. We implemented this by computing the union of complete extensions with iterative SAT calls, removing the arguments which are attacked by an argument in the union, and then iteratively maximizing a complete extension within this set, again with iterative SAT.

μ-toksia includes Glucose (version 4.1) [2] as the core SAT engine. μ-toksia has been compiled using GCC (version 7.3.0) with optimization flag `-O3`. Additionally, Glucose has been compiled with incremental mode [1] enabled.

2 Supported Tasks and Semantics

μ-toksia supports all dynamic tasks as well as all classical tasks (by simply dropping the $r_{a,b}$ variables) of ICCMA 2019. That is, the following classical tasks are supported:

- credulous acceptance (**DC**-σ),
- skeptical acceptance (**DS**-σ),
- finding one extension (**SE**-σ),
- enumerating extensions (**EE**-σ),

under the semantics $\sigma \in \{\mathrm{CO}, \mathrm{PR}, \mathrm{ST}, \mathrm{SST}, \mathrm{STG}, \mathrm{GR}, \mathrm{ID}\}$, and the following dynamic tasks:

- credulous acceptance (**DC**-σ-**D**),
- skeptical acceptance (**DS**-σ-**D**),
- finding one extension (**SE**-σ-**D**),
- enumerating extensions (**EE**-σ-**D**),

under $\sigma \in \{\mathrm{CO}, \mathrm{PR}, \mathrm{ST}, \mathrm{GR}\}$.
Both `.apx` and `.tgf` formats for the input AF and the modification file are supported.

3 Docker Repository

The solver binary has been pushed to the Docker repository `andreasniskanen/iccma2019`.

References

[1] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, pages 309–317, 2013.

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

[3] Philippe Besnard and Sylvie Doutre. Checking the acceptability of a set of arguments. In *10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada, June 6-8, 2004, Proceedings*, pages 59–64, 2004.

[4] Martin Caminada. A labelling approach for ideal and stage semantics. *Argument & Computation*, 2(1):1–21, 2011.

[5] 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.

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

[7] Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly. CoQuiAAS: A constraint-based quick abstract argumentation solver. In *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Vietri sul Mare, Italy, November 9-11, 2015*, pages 928–935, 2015.