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

Andreas Niskanen Matti Järvisalo

HIIT, Department of Computer Science, University of Helsinki, Finland

ECAI 2020

Motivation

Argumentation

- Active and vibrant area of modern AI research
- Central KR formalism for reasoning in abstract argumentation: *argumentation frameworks (AFs)*

Dynamic Argumentation Frameworks

- In addition to a fixed AF, a sequence of changes to the attack structure of the AF is provided
- "Dynamic track" in the 3rd International Competition on Computational Models of Argumentation (ICCMA'19)
  - Can we answer the same query (e.g. argument acceptance) on all AFs defined via the sequence of changes efficiently?

Contributions

What?

Design algorithms for dynamic argumentation frameworks

- Covering all tasks in the dynamic track of ICCMA'19: credulous and skeptical acceptance, single extension, and extension enumeration under complete, stable, preferred, and grounded semantics

How?

Employ incremental Boolean satisfiability (SAT) solving

- A SAT solver is instantiated only once during the run of the algorithm
- Make efficient use of the assumptions interface of the SAT solver

μ–TOKSIA System

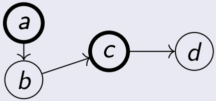
- Winner of every track in ICCMA'19
- Available online in open source at `https://bitbucket.org/andreasniskanen/mu-toksia`

Abstract Argumentation Frameworks

Argumentation Framework (AF)

A directed graph $F = (A, R)$, where

- A is the set of **arguments**
- $R \subseteq A \times A$ is the **attack relation**
  - $a \to b$ means argument a attacks argument b



Semantics

Define sets of jointly accepted arguments called **extensions**

- Required to be **conflict-free** (independent sets)
- Additional desired properties (e.g. self-defense, subset-maximality)
  - complete, preferred, stable, ...

Acceptance of argument $a \in A$ via extensions

- **credulously** accepted if contained in **some** extension
- **skeptically** accepted if contained in **all** extensions

Dynamic Argumentation Frameworks

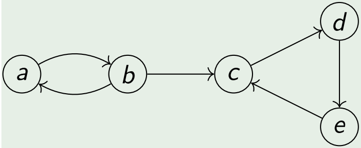
A dynamic AF consists of an AF $F = (A, R)$ and a sequence of **changes**

- a change is either an addition or removal of an attack $(a, b) \in A \times A$

Defines a sequence of attack structures $R_0 = R, R_1, \ldots, R_n$

- **dynamic** attacks are contained in some but not every R_i, $i = 0, \ldots, n$
- **static** attacks are contained in every R_i, $i = 0, \ldots, n$

Example



Changes $-(b, c)$, $+(c, b)$

- dynamic attacks: (b, c) and (c, b)
- static attacks: every attack except (b, c) and (c, b)

Note: $\{a\}$ remains a preferred extension

- skeptical acceptance of b by checking existence of preferred extension $\{a\}$ obtained from the original AF?

Dynamic Argumentation Frameworks

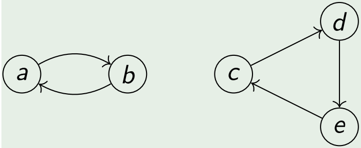
A dynamic AF consists of an AF $F = (A, R)$ and a sequence of **changes**

- a change is either an addition or removal of an attack $(a, b) \in A \times A$

Defines a sequence of attack structures $R_0 = R, R_1, \ldots, R_n$

- **dynamic** attacks are contained in some but not every R_i, $i = 0, \ldots, n$
- **static** attacks are contained in every R_i, $i = 0, \ldots, n$

Example



Changes $-(b, c)$, $+(c, b)$

- dynamic attacks: (b, c) and (c, b)
- static attacks: every attack except (b, c) and (c, b)

Note: $\{a\}$ remains a preferred extension

- skeptical acceptance of b by checking existence of preferred extension $\{a\}$ obtained from the original AF?

Dynamic Argumentation Frameworks

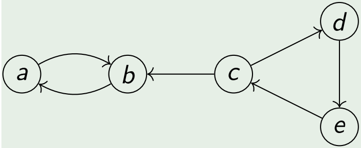
A dynamic AF consists of an AF $F = (A, R)$ and a sequence of **changes**

- a change is either an addition or removal of an attack $(a, b) \in A \times A$

Defines a sequence of attack structures $R_0 = R, R_1, \ldots, R_n$

- **dynamic** attacks are contained in some but not every R_i, $i = 0, \ldots, n$
- **static** attacks are contained in every R_i, $i = 0, \ldots, n$

Example



Changes $-(b, c)$, $+(c, b)$

- dynamic attacks: (b, c) and (c, b)
- static attacks: every attack except (b, c) and (c, b)

Note: $\{a\}$ remains a preferred extension

- skeptical acceptance of b by checking existence of preferred extension $\{a\}$ obtained from the original AF?

SAT Encodings for Dynamic AFs

Boolean Variables

$r_{a,b}$ for each dynamic attack (a, b)

- assigned true iff (a, b) occurs in the current AF

x_a for each argument $a \in A$

- assigned true iff $a \in E$ for some extension E of the current AF

Boolean Formulas

For semantics $\sigma \in \{cf, adm, com, stb\}$ and a dynamic AF F^χ, defining

$$\textsc{Att}(F_i) = \bigwedge_{(a,b)\in R_i} r_{a,b} \wedge \bigwedge_{(a,b)\notin R_i} \neg r_{a,b},$$

formula $\phi_\sigma(F^\chi) \wedge \textsc{Att}(F_i)$ encodes the σ-extensions of the AF F_i.

SAT-based Algorithms: Acceptance

Variables $r_{a,b}$ play a crucial role as **assumptions** passed to the SAT solver

Acceptance of $a \in A$ under Complete and Stable Semantics

At each iteration $i = 0, \ldots, n$, query a SAT solver with input formula $\phi_\sigma(F^\chi) \wedge q$, where

- $q = x_a$ for credulous acceptance
- $q = \neg x_a$ for skeptical acceptance

using assumptions

$$\textsc{Att}(F_i) = \bigwedge_{(a,b) \in R_i} r_{a,b} \wedge \bigwedge_{(a,b) \notin R_i} \neg r_{a,b}.$$

Skeptical Acceptance under Preferred Semantics

Assumptions on attacks similarly, adapting the procedure for the "static" acceptance problem implemented in the AF solver $\textsc{Cegartix}$

SAT-based Algorithms: Optimizations

Positive Check

If at iteration $i = 1, \ldots, n$, argument $a \in A$ **was** credulously accepted in the previous AF F_{i-1}, we have a witnessing extension
$\rightarrow$ check whether it still is an extension in F_i

Negative check

If at iteration $i = 1, \ldots, n$, argument $a \in A$ **was not** credulously accepted in the previous AF F_{i-1}, the previous call was unsatisfiable
$\rightarrow$ check whether the literal corresponding to the i-th change belongs to the **unsatisfiable core** reported by the SAT solver

Skeptical acceptance dually

- Positive check if $a \in A$ **was not** skeptically accepted
- Negative check if $a \in A$ **was** skeptically accepted (not for preferred)

SAT-based Algorithms: Enumeration

Algorithms for acceptance under complete and stable semantics easily adapted to extension enumeration via

- dropping the unit clause x_a (or $\neg x_a$),
- at each iteration i, using assumptions $\neg b_0, \neg b_1, \ldots, b_i$, calling the solver, and after each extension E found adding blocking clauses

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

until unsatisfiability for that iteration.

Preferred semantics: additionally a subset-maximization procedure

Implementation and Benchmarks

μ–TOKSIA

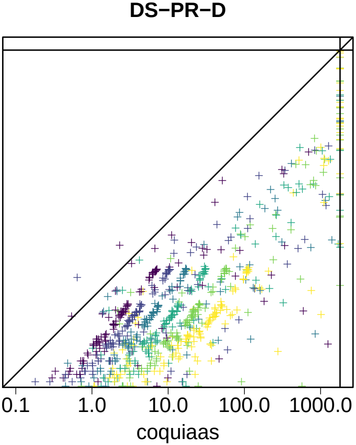
- GLUCOSE as the underlying SAT solver
- Available online in open source at `https://bitbucket.org/andreasniskanen/mu-toksia`

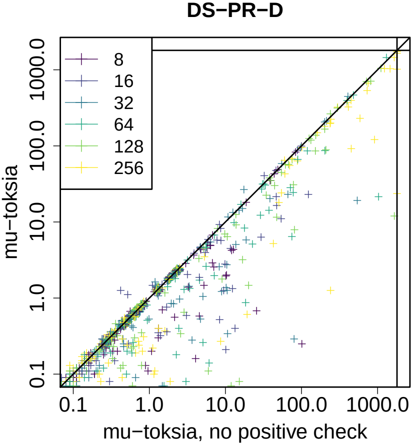
Benchmark Setup

- Per-instance 1800-second time limit and 64-GB memory limit
- ICCMA'19 used 8 changes in the sequence of changes
  - Extend to $16, 32, \ldots, 256$ by appending more changes at random
- NP-hard acceptance tasks considered in ICCMA'19
  - credulous acceptance under complete and stable
  - skeptical acceptance under stable and preferred

Experimental Evaluation

Skeptical acceptance under preferred semantics:
Left: μ–TOKSIA vs. COQUIAAS
Right: impact of "positive check"





Conclusions

Paper Summary

- Provided **SAT-based algorithms** for reasoning over dynamic AFs
  - Covering all reasoning tasks introduced in ICCMA'19
  - Based on incremental SAT solving using the assumptions interface
- **Empirical evaluation:** state-of-the-art approach

μ–TOKSIA

- Winner of every track in ICCMA'19
- Available online in open source at
  `https://bitbucket.org/andreasniskanen/mu-toksia`