**Abstract**—We present FUDGE v3.2.8, an abstract argumentation solver that tightly integrates satisfiability solving technology to solve a series of abstract argumentation problems. While most of the encodings used by FUDGE v3.2.8 derive from standard translation approaches, FUDGE v3.2.8 makes use of sophisticated encodings to solve the skeptical reasoning problem wrt. preferred semantics and problems wrt. ideal semantics.

I. INTRODUCTION

An abstract argumentation framework $AF$ is a tuple $AF = (A, R)$ where $A$ is a (finite) set of arguments and $R$ is a relation $R \subseteq A \times A$ [6]. For two arguments $a, b \in A$ the relation $aRb$ means that argument $a$ attacks argument $b$. For a set $S \subseteq A$ we define

$$S^+ = \{a \in A \mid \exists b \in S, bRa\}$$

$$S^- = \{a \in A \mid \exists b \in S, aRb\}$$

We say that a set $S \subseteq A$ is conflict-free if for all $a, b \in S$ it is not the case that $aRb$. A set $S$ defends an argument $b \in A$ if for all $a$ with $aRb$ there is $c \in S$ with $cRa$. A conflict-free set $S$ is called admissible if $S$ defends all $a \in S$.

Different semantics [1] can be phrased by imposing constraints on admissible sets. In particular, set $E$

- is a complete (CO) extension iff it is admissible and for all $a \in A$, if $E$ defends $a$ then $a \in E$,
- is a grounded (GR) extension iff it is complete and minimal,
- is a stable (ST) extension iff it is conflict-free and $E \cup E^+ = A$,
- is a preferred (PR) extension iff it is admissible and maximal,
- is an ideal (ID) extension iff $E \subseteq E'$ for each preferred extension $E'$, $E$ is admissible, and $E$ is maximal,
- is a semi-stable (SST) extension $E$ is admissible and $E \cup E^+$ is maximal, and
- is a stage (SST) extension $E$ is conflict-free and $E \cup E^+$ is maximal.

All statements on minimality/maximality are meant to be with respect to set inclusion.

Given an abstract argumentation framework $AF = (A, R)$ and a semantics $\sigma \in \{CO, GR, ST, PR, ID, SST, STG\}$ we are interested in the following computational problems:

- SE-$\sigma$: Given $AF$, compute some $\sigma$-extension.
- DC-$\sigma$: Given $AF$ and an argument $a$, decide whether $a$ is in at least one $\sigma$-extension of $AF$.
- DS-$\sigma$: Given $AF$ and an argument $a$, decide whether $a$ is in all $\sigma$-extensions of $AF$.

Note that DC-$\sigma$ and DS-$\sigma$ are equivalent for $\sigma \in \{GR, ID\}$ as those extensions are uniquely defined [1]. For these, we will only consider DS-$\sigma$.

The FUDGE v3.2.8 solver supports solving the above-mentioned computational problems wrt. all $\sigma \in \{CO, GR, ST, PR, ID, SST, STG\}$. In the remainder of this system description, we give a brief overview on the architecture of FUDGE v3.2.8 (Section II) and conclude in Section III.

II. ARCHITECTURE

FUDGE v3.2.8 follows the standard reduction-based approach to solve the above-mentioned reasoning problems [2], [4] with the target formalism being the satisfiability problem SAT [3]. For example, given the problem SE-ST and an input argumentation framework $AF = (A, R)$, first, for each argument $a \in A$, we create propositional variables $in_a$ and $out_a$, with the meaning that $in_a$ (resp. $out_a$) is true in a satisfying assignment iff the argument $a$ is in (resp. attacked by) the stable extension to be found. Then

$$\Phi_1(AF) = \bigwedge_{a \in A} (\neg in_a \lor \neg out_a)$$

and

$$\Phi_2(AF) = \bigwedge_{a \in A} (out_a \Leftrightarrow \bigvee_{(b, a) \in R} in_b)$$

model the basic intuition behind these variables and, in particular, ensure conflict-freeness of the modelled extension. The constraint that all arguments not included in the extension must be attacked can be modelled by

$$\Phi_3(AF) = \bigwedge_{a \in A} (in_a \lor out_a)$$

Then the formula $\Phi_1(AF) \land \Phi_2(AF) \land \Phi_3(AF)$ is satisfiable iff $AF$ has a stable extension and a stable extension can be easily extracted from a satisfying assignment of $\Phi_1(AF) \land \Phi_2(AF) \land \Phi_3(AF)$. All reasoning problems on the first level of the polynomial hierarchy [7] can be solved in a similar manner.
Particularly challenging problems are those wrt. preferred semantics as, in particular, \(DS-PR\) is \(\Pi^P_2\)-complete \[7\]. To solve that problem, we use the approach of \[8\]. This approach relies on the following observation\[1\]

**Theorem 1** (\[8\]). \(a \in A\) is skeptically accepted wrt. preferred semantics iff

1) there is an admissible set \(S\) with \(a \in S\) and
2) for every admissible set \(S'\) with \(a \in S\) and every admissible set \(S''\) with \(S' \cap S'' \neq \emptyset\), there is an admissible set \(S''\) with \(S' \cup \{a\} \subseteq S''\).

The above theorem states that we can decide skeptical acceptance wrt. preferred semantics by considering only those admissible sets that attack an admissible set containing the argument in question. As an admissibility check can be solved by a satisfiability check, similarly as above, the above insight leads to an algorithm that can solve \(DS-PR\) without actually computing preferred extensions.

Coming to ideal semantics, it is worth recalling \[5, Theorem 3.3\].

**Theorem 2** (\[5, Theorem 3.3\]). An admissible set of arguments \(S\) is ideal iff for each argument \(a\) attacking \(S\) there exists no admissible set of arguments containing \(a\).

The interesting aspect of \[5, Theorem 3.3\] is that ideal semantics, although defined based on skeptical acceptance wrt. preferred semantics, does not rely on that notion. Moreover, in \[8\] we also prove that starting from the set of arguments which are not attacked by an admissible set, its largest admissible set is the ideal extension. We can therefore tweak the machinery we created for computing skeptical acceptance wrt. preferred semantics to compute the ideal extension too. The complete algorithm is presented in \[8\].

**FUDGE v3.2.8** is written in C++ and uses the satisfiability solver CaDiCaL 1.3.\[2\]. **FUDGE v3.2.8** improves over its previous version v2.4 submitted at ICCMA’21 by using a different method for calling the SAT solver, as well as streamlined encodings, and support for the semi-stable and stage semantics.

III. **Summary**

We presented **FUDGE v3.2.8** a reduction-based solver for various problems in abstract argumentation. **FUDGE v3.2.8** leverages on a mix of standard and novel SAT encodings to solve reasoning problems, with the aim of avoiding the costly maximisation step that is characteristic of some of the abstract argumentation problems. The source code of **FUDGE v3.2.8** is available at [https://github.com/aig-hagen/taas-fudge](https://github.com/aig-hagen/taas-fudge)

**REFERENCES**


---

\[1\] Define \(S'RS\) iff there is \(a \in S'\) and \(b \in S\) with \((a, b) \in R\).

[http://fmv.jku.at/cadical/](http://fmv.jku.at/cadical/)