Computing Preferred Extensions for Abstract Argumentation

Xindi Zhang, Shaowei Cai*

[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
{zhangxd,caisw,chenzh}@ios.ac.cn

***Abstract*—In this document, we describe how to generate SAT instances though a general SAT-based abstract argumentation solver called MiniAF. We only focus on solving the reasoning tasks of preferred semantic on the ICCMA benchmarks.**

I. Introduction

SAT-based method is one of the most popular formal argumentation approaches for solving reasoning tasks of the abstract argumentation framework [1]. In this document, we use a general solver called MiniAF [6] to solve tasks from International Competition on Computational Models of Argumentation(ICCMA). For clearer understanding, we repeating the encoding method in this document.

II. Basic concepts

A given abstract argumentation framework(AF) $AF = (A, R)$ can be represented by a directed graph, where A is a set of arguments and $R \subseteq A \times A$ is the relation. For two arguments $a, b \in A$, the relation aRb means that a attacks b (which can be represented by $a \rightarrow b$ as well), and we denote $a^- = \{b|bRa\}$. A set $S \subseteq A$ defends an argument $b \in A$ if for all a with aRb there is $c \in S$ with cRa. Semantic σ represents a kind of property over a set of arguments, and a σ-extension $E \subseteq A$ is a argument set with the property σ. There are definitions of some important semantic extensions.

- An extension E is $conflict\text{-}free$ (cf) iif there are no arguments $a, b \in E$ with aRb;
- An extension E is $admissible$ (adm) iif E is cf and E defends every $a \in E$;
- An extension E is $complete$ (co) iif E is adm and if E defends a then $a \in E$;
- An extension E is $preferred$ (prf) iif E is maximal co.

Given a semantic $\sigma \in \{co, prf\}$ and an AF $AF = (A, R)$, an argument $a \in A$ is $skeptically\ accepted$ in AF if a is contained in every σ-extensions, is $credulously\ accepted$ in AF if a is contained in some σ-extensions. There are some tasks base on a given AF $AF = (A, R)$ and an argument a.

- EE-σ: Enumerate all extensions $E \subseteq A$ that are σ-extensions;
- SE-σ: Return an extension $E \subseteq A$ that is a σ-extension;
- DC-σ: Decide if a is credulously accepted under σ;
- DS-σ: Decide if a is skeptically accepted under σ.

III. Labelling Encoding Method

This section introduces an equivalent way to define different types of semantics by labelling encoding method [4], [6]. Given a set of arguments A, a $labelling$ L mapping each argument $a \in A$ to $\{in, out, undec\}$, which means that a is accepted, rejected or the status is undecided, respectively. The set of all $labellings$ for a given $AF = (A, R)$ is denoted as $\zeta(AF)$.

$L \in \zeta(AF)$ is called a $complete\ labelling$ (co-L) iif for any $a \in A$ holds:

- $L(a) = in \Leftrightarrow \forall b \in a^-, L(b) = out$;
- $L(a) = out \Leftrightarrow \exists b \in a^-, L(b) = in.$

A co-L $L \in \zeta(AF)$ is equal to a prf-extension iif L maximize the set of arguments labelled in.

IV. Encoding for Complete Semantics

This section gives the classic encoding method for complete semantics [2] used in MiniAF [6]. Given an AF $AF = (A, R)$ with $|A| = k$, and $\Phi : \{1, ..., k\} \rightarrow A$ is an indexing bijection.

At first, we define three symbols I_i, O_i, U_i for each argument $a \in AF$ with indexing i, and for each argument $a \in AF$, a can labelled exact one type label.

$$\bigwedge_{i \in \{i,\dots,k\}} ((I_i \vee O_i \vee U_i) \wedge (\neg I_i \vee \neg O_i) \wedge (\neg I_i \vee \neg U_i) \wedge (\neg O_i \vee \neg U_i)) \tag{1}$$

By the definition, for each argument $a \in AF$ without any attackers, a should be labelled in.

$$\bigwedge_{\{i|\Phi(i)^- = \emptyset\}} (I_i \wedge O_i \wedge U_i) \tag{2}$$

Then, for each argument $a \in AF$ with at least one attacker: $L(a) = in \Rightarrow \forall b \in a^-, L(b) = out$; $L(a) = in \Leftarrow \forall b \in a^-, L(b) = out.$

$$\bigwedge_{\{i|\Phi(i)^- \neq \emptyset\}} \Big(\bigwedge_{\{j|\Phi(j) \rightarrow \Phi(i)\}} \neg I_i \vee O_j \Big) \tag{3}$$

$$\bigwedge_{\{i|\Phi(i)^- \neq \emptyset\}} \Big(I_i \vee \Big(\bigvee_{\{j|\Phi(j) \rightarrow \Phi(i)\}} \neg O_j \Big) \Big) \tag{4}$$

This work was supported by Beijing Academy of Artificial Intelligence (BAAI), and Youth Innovation Promotion Association, Chinese Academy of Sciences [No. 2017150].

* Corresponding author

At last, for each argument $a \in AF$ with at least one attacker: $L(a) = out \Rightarrow \exists b \in a^-, L(b) = in$; $L(a) = out \Leftarrow \exists b \in a^-, L(b) = in$.

$$\bigwedge_{\{i|\Phi(i)^- \neq \emptyset\}} \Big(\neg O_i \vee \Big(\bigvee_{\{j|\Phi(j)\to\Phi(i)\}} \neg I_j\Big)\Big) \tag{5}$$

$$\bigwedge_{\{i|\Phi(i)^- \neq \emptyset\}} \Big(\bigwedge_{\{j|\Phi(j)\to\Phi(i)\}} I_j \vee O_i\Big) \tag{6}$$

All the above formulas (1)-(6) make up a conjunctive normal form (CNF) Π, which can be solved by a given SAT solver. At last, to enumerate all extensions, MiniAF excluding previous model s by add a formula $\neg s$ to Pi after each time a model is found by the SAT solver, until the SAT solver return that there are no more model (UNSAT).

V. Preferred Semantics and Related Tasks

MiniAF uses an improved PrefSAT algorithm [3] for computing $preferred\ labellings$ (prf-L). The algorithm iterates over a set of co-Ls to identify the preferred ones and optimizes the process by set inclusion to maximise co-Ls.

To decide the credulous acceptance of an argument a, the CNF Π is updated to $\Pi \wedge I_{\Phi^{-1}(a)}$. To check the skeptically acceptance of an argument a, MiniAF subsequently enumerates all prf-Ls util it finds a labelling with $L(a) \neq in$.

VI. Benchmark selection

We use MiniAF to solve the tasks of EE-prf, DS-prf and DC-prf on the benchmarks from ICCMA-17, ICCMA-19 which can be downloaded from http://argumentationcompetition.org/. Following the definition of 'interesting instance' that one should not be solved by MiniSat [5] in a minute and should be solved by our own solver within 1 hour, We select some interesting instances from the intermediate results of MiniAF, which are in the format of "$.cnf$".

References

[1] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial intelligence*, 93(1-2):63–101, 1997.

[2] F. Cerutti, P. E. Dunne, M. Giacomin, and M. Vallati. Computing preferred extensions in abstract argumentation: A sat-based approach. In *International Workshop on Theorie and Applications of Formal Argumentation*, pages 176–193, 2013.

[3] F. Cerutti, M. Vallati, and M. Giacomin. An efficient java-based solver for abstract argumentation frameworks: jargsemsat. *International Journal on Artificial Intelligence Tools*, 26(02):1750002, 2017.

[4] G. Charwat, W. Dvořák, S. A. Gaggl, J. P. Wallner, and S. Woltran. Methods for solving reasoning problems in abstract argumentation–a survey. *Artificial intelligence*, 220:28–63, 2015.

[5] N. Eén and N. Sörensson. An extensible sat-solver. In *International conference on theory and applications of satisfiability testing*, pages 502–518, 2003.

[6] J. Klein and M. Thimm. Revisiting sat techniques for abstract argumentation. *Computational Models of Argument: Proceedings of COMMA 2020*, 326:251, 2020.