CoQuiAAS v3.0
ICCMA 2019 Solver Description [*]

Jean-Marie Lagniez[1], Emmanuel Lonca[1], and Jean-Guy Mailly[2]

[1] Centre de Recherche en Informatique de Lens – Univ. Artois & CNRS
{lagniez,lonca}@cril.fr
[2] LIPADE – Univ. Paris Descartes
jean-guy.mailly@parisdescartes.fr

Abstract

This paper presents how we have extended the existing software CoQuiAAS to handle the new challenges proposed for the ICCMA 2019. The main idea behind CoQuiAAS is to use different Constraint Programming techniques to develop a software library dedicated to argumentative reasoning. More specifically, we use Boolean Satisfiability (SAT) and Maximal Satisfiable Sets (MSS) extraction to solve all the different computational problems from ICCMA 2019. Our library offers the advantages to be efficient, generic and easily adaptable.

This solver description is clearly inspired by the one redacted for CoQuiAAS v2.0 (the version involved in ICCMA 2017). Some of the content of the previous solver description has been reused as is, especially in the introductory section.

1 Introduction

Computational problems related to abstract argumentation frameworks (AFs) [3] are interesting from a theoretical point of view (since argumentation can be used to encode a wide family of non-monotonic inference relations) and from a practical point of view (applications of argumentation exist in various fields like e-democracy, automated negotiation or reasoning from inconsistent knowledge). An AF can be seen as a directed graph where nodes represent arguments and edges represent attacks between these arguments. The meaning of such a graph is determined by an acceptability semantics, which indicates how to select a set of arguments which can be jointly accepted; such a set of arguments is then called an extension.

Previous iterations of ICCMA were restricted to *static* frameworks, in the sense that softwares had to solve sequences of distinct problems. This year's competition introduces *dynamics* in argumentation frameworks [1]: given an initial set of arguments and a problem, solvers must solve the problem for multiple sets of attacks, where two successive sets differ by adding or removing a single attack.

In the following, we first present how we modified CoQuiAAS in order to make it dynamic. Then, we describe the other slight adaptations we made to CoQuiAAS, in order to fit adjustments required by the 2019 edition of the competition. We conclude by giving information needed to download the docker container that can be used to run CoQuiAAS.

2 Encoding of Dynamics

CoQuiAAS still exploits the encodings described in [5] to turn out argumentation frameworks into propositional formulas. As a reminder, these encodings are based on the ones of [2] except

[*]This work has been supported by the project CPER Data from the region “Hauts-de-France”.

$$\begin{array}{ll}
(1) & \bigwedge_{a_i \in A} (\neg a_i \vee \neg P_{a_i}) \\
(2) & \bigwedge_{a_i \in A} (a_i \vee \bigvee_{a_j \in A | (a_j, a_i) \in R} \neg P_{a_j}) \\
(3) & \bigwedge_{a_i \in A} (\bigwedge_{a_j \in A | (a_j, a_i) \in R} (\neg a_i \vee P_{a_j})) \\
(4) & \bigwedge_{a_i \in A} (\neg P_{a_i} \vee \bigvee_{a_j \in A | (a_j, a_i) \in R} a_j) \\
(5) & \bigwedge_{a_i \in A} [\bigwedge_{a_j \in A | (a_j, a_i) \in R} (P_{a_i} \vee \neg a_j)]
\end{array}$$

Figure 1: Complete semantics encoding of CoQuiAAS.

additional variables P_a defined as the disjunction of attackers of $a \in A$ are introduced for complete semantics based problems in order to reduce space complexity of the encodings. Figure 1 shows the encoding for an argumentation framework $F = \langle A, R \rangle$.

For this competition, dynamics is described by the initial attack set and a set of changes (additions and deletions of one attack) provided at the start of the solver. Contrary to some articles of the state of the art which do not have any information about forthcoming changes, we can exploit the full knowledge of the dynamics by producing a formula in which we can activate/deactivate attacks using assumptions.

First, we partition the set of attacks in two:

- R^H is the set of "hard" attacks which are present at each iteration;
- R^D is the set of "dynamic" attacks.

Then, for each dynamic attack $(a_j, a_i) \in R^D$, we define three new variables:

- the assumption variable h_{a_j,a_i} to be set to *true* (resp. *false*) to activate (resp. deactivate) the attacks;
- the *attacker replacement variable* $a'_j \equiv a_j \wedge h_{a_j,a_i}$ used to replace a_j when (a_j, a_i) is involved;
- the *attacker disjunction replacement variable* $P'_j \equiv P_j \vee \neg h_{a_j,a_i}$ used to replace P_j when (a_j, a_i) is involved.

Figure 2 describes how the problem is encoded using this additional information. Although we do not demonstrate here the correctness of the encoding, here is a sketch of proof:

- setting h_{a_j,a_i} to *true* builds an equivalence relation between a'_j and a_j; this way, the attack is considered as a static attack in points (4) and (5). Setting h_{a_j,a_i} to *false* makes a'_j *false*, letting the formula in the state it would have been if the attack was not present for point (4) and (5).
- setting h_{a_j,a_i} to *true* builds an equivalence relation between P'_j and P_j; this way, the attack is considered as a static attack in points (2) and (3). Setting h_{a_j,a_i} to *false* makes P'_j *true*, letting the formula in the state it would have been if the attack was not present for point (2) and (3).

Concerning the stable dynamic encoding, we proceed in the same way, except we only consider the replacement of a_j by a'_j for each $(a_j, a_i) \in R^D$.

$$\bigwedge_{a_i \in A} (\neg a_i \vee \neg P_{a_i}) \tag{1}$$

$$\bigwedge_{a_i \in A} [a_i \vee (\bigvee_{a_j \in A | (a_j, a_i) \in R^H} \neg P_{a_j}) \vee (\bigvee_{a_j \in A | (a_j, a_i) \in R^D} \neg P'_{a_j})] \tag{2}$$

$$\bigwedge_{a_i \in A} [\bigwedge_{a_j \in A | (a_j, a_i) \in R^H} (\neg a_i \vee P_{a_j})] \wedge [\bigwedge_{a_j \in A | (a_j, a_i) \in R^D} (\neg a_i \vee P'_{a_j})] \tag{3}$$

$$\bigwedge_{a_i \in A} [\neg P_{a_i} \vee (\bigvee_{a_j \in A | (a_j, a_i) \in R^H} a_j) \vee (\bigvee_{a_j \in A | (a_j, a_i) \in R^D} a'_j)] \tag{4}$$

$$\bigwedge_{a_i \in A} [(\bigwedge_{a_j \in A | (a_j, a_i) \in R^H} (P_{a_i} \vee \neg a_j)) \wedge (\bigwedge_{a_j \in A | (a_j, a_i) \in R^D} (P_{a_i} \vee \neg a'_j))] \tag{5}$$

$$\bigwedge_{a_j \in A | \exists (a_j, a_i) \in R^D} [a'_j \Leftrightarrow a_j \wedge h_{a_j, a_i}] \tag{6}$$

$$\bigwedge_{a_j \in A | \exists (a_j, a_i) \in R^D} [P'_j \Leftrightarrow P_j \vee \neg h_{a_j, a_i}] \tag{7}$$

Figure 2: Dynamic complete semantics encoding of CoQuiAAS.

3 Other adjustments

In addition to the dockerization of CoQuiAAS, we have made some little changes that are reported below. First, we adapted the solver output to fit the requirements of this competition. We added a CLI option for CoQuiAAS to use the former format for compatibility reasons. In addition to the format, we have changed the manner the solver outputs the extensions, delegating to a thread the display of each extension as soon as it has been computed (in the previous version, all the output was made after the computations were finished).

Concerning the (co)MSS extractor used by CoQuiAAS, we took advantage of the recent advances proposed in [4] to include it in CoQuiAAS and set it as the default MSS solver. The previous behavior (invoking an external solver) is still available.

4 Participation

We enter the competition with CoQuiAAS v3.0 on all static and dynamic tracks opened at the ICCMA 2019 competition. It is available as a Docker container from `https://cloud.docker.com/repository/docker/lonca/coquiaas_iccma2019`.

We finally thank warmly the organizers for the time and the energy they use to make this competition possible.

References

[1] Gianvincenzo Alfano, Sergio Greco, and Francesco Parisi. Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 49–55. ijcai.org, 2017.

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

[3] 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.

[4] Éric Grégoire, Yacine Izza, and Jean-Marie Lagniez. Boosting mcses enumeration. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 1309–1315. ijcai.org, 2018.

[5] 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. IEEE Computer Society, 2015.