Abstract Argumentation in Prover9 and Mace4

Adrian Groza[1], Liana Toderean[1], Eliza Olariu[1], Emanuel Baba[1], Oana Grigoras[1], and Bogdan Bele[1]

Technical University of Cluj-Napoca, Cluj-Napoca, Romania.
adrian.groza@cs.utcluj.ro, liana.toderean@student.utcluj.ro,
oana.avasi@student.utcluj.ro, emanuel.baba@student.utcluj.ro,
bogdan.bele@student.utcluj.ro

Abstract

Our approach relies on the search capabilities of Prover9 and Mace4. We aim to tune some search parameters of Prover9 and Mace4 for the specific task of labelling abstract argumentation frameworks. We aim to exploit the Knuth-Bedinx ordering in Prover9, hint clauses, first-order-formula reductions, or semantic guidance. We are targeting the complete semantics.

1 Research hypotheses

In line with Kinyon et al. [4], our task was to investigate ways of fine tuning the Prover9 and Mace4 [5] search algorithms for the specific task of finding extensions in abstract argumentation frameworks. The approach was to change the default lexicographic term ordering in Prover9 as follows:

First, we aim to exploit the KBO (Knuth-Bendix Ordering) that allows to weigh the symbols or set the symbol precedence. The proposed weights should favor arguments which have the highest number of attack relations, either *in* or *out*.

Second, we aim to introduce various hint clauses. These hint clauses adapted for the abstract argumentation frameworks can be used to guide Prover9's search.

Third, we aim to use the semantic guidance capability of Prover9. Semantic guidance uses finite interpretations to guide the search for a proof.

Given the model explosion in case MACE4 has a large domain size, our expectation is that our approach may be effective for the task of finding a single extension/model.

2 System description

We rely on the work of McCune on Mace4 and Prover9 [5, 6]. The coding has been done by 3^{rd} year students at Computer Science Department with Technical University of Cluj-Napoca). Fig. 1 depicts the top level architecture of our system called Sportacus. The solver is available at https://cloud.docker.com/repository/docker/sportacus/app. We target the *complete semantics*: SE, DC, DS.

These semantics have been directly formalised in First Order Logic. The set of In arguments must be conflict-free, it means that any x argument can be *in* only if there is no attack relation with another *in* argument y. We also define the `Defeated(x)` and `Not_Defended(x)` predicates. An argument `x` is defeated if it was attacked by an In argument. If there is a `y` that attacks `x` and `y` is not attacked by an In argument that means `x` is not defended.

The input file is pre-processed as follows. First, we explicitly add the not attack relations. This is required for Mace4 runs under the open world assumption. Second, we add the predicate `arg(a, nr)`, to specify the number of attacks starting from the argument `a`.

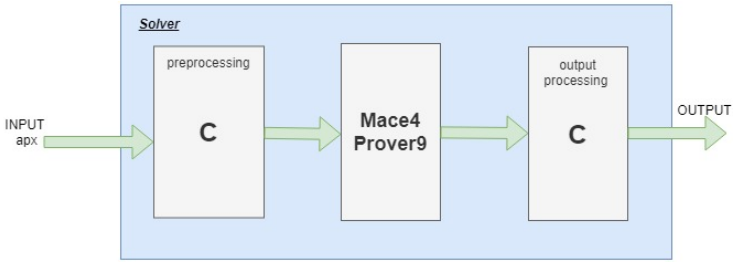

Figure 1: Top level architecture of Sportacus.

For the KBO strategy, the weight of an argument depends on the number of arguments that it attacks. The domain size for Mace4 has to be the number of arguments. We used the labelling argumentation semantics [1]. We defined the labelled function $\mathcal{IN}$ with two possible values: 0 (for label *Out*) and 1 for label *In*.

To improve the Mace4's performance, we set some arguments In or Out in the pre-processing phase. The argument that attacks the most is considered In (the arguments are ordered descending by the number of attacks) and the satisfiability is checked using Mace4. If the test fails, that argument is changed to Out and the next argument is set In. These steps are repeated a finite number of times (depending on the number of arguments), or until a solution is found. If no extension was found, the argument that was considered In last is set Out and then it is tested again. If the last test fails, we assume there is no extension.

3 Related work

A related approach from the coding perspective is the work of [2] and [3]. Here the solution is based on answer set programming, while in our case on resolution mechanism of Prover9 and model finding capabilities of Mace4. Recently, Kinyon has investigated in [4] various ideas to improve the search capabilities of Prover9.

References

[1] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The knowledge engineering review*, 26(4):365–410, 2011.

[2] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Aspartix: Implementing argumentation frameworks using answer-set programming. In *International Conference on Logic Programming*, pages 734–738. Springer, 2008.

[3] Sarah A Gaggl, Norbert Manthey, Alessandro Ronca, Johannes P Wallner, and Stefan Woltran. Improved answer-set programming encodings for abstract argumentation. *Theory and Practice of Logic Programming*, 15(4-5):434–448, 2015.

[4] Michael Kinyon. Proof simplification and automated theorem proving. *Philosophical Transactions of the Royal Society A*, 377(2140):20180034, 2019.

[5] William McCune. Mace4 reference manual and guide. *arXiv preprint cs/0310055*, 2003.

[6] William McCune. Prover9 and mace4, 2005.