PORTSAT: A Portfolio of SAT Solvers for Abstract Argumentation

Sylvain Declercq
Université Paris Cité
Paris, France
syl.dec@live.fr

Quentin Januel Capellini
Sorbonne Université
Paris, France
quentinjanuel.pro@gmail.com

Christophe Yang
Université Paris Cité
Paris, France
christopheyang@live.fr

Jérôme Delobelle
Université Paris Cité, LIPADE
Paris, France
jerome.delobelle@u-paris.fr

Jean-Guy Mailly
Université Paris Cité, LIPADE
Paris, France
jean-guy.mailly@u-paris.fr

***Abstract*—We describe PORTSAT, a portfolio of SAT solvers which can solve several classical reasoning tasks in abstract argumentation, namely** DC-CO**,** DC-ST**,** DS-PR**,** DS-ST**,** SE-PR **and** SE-ST**.**

***Index Terms*—Abstract argumentation, SAT solvers, portfolio**

I. Background: Semantics and Problems

We consider abstract argumentation frameworks (AFs) [5]. An AF is a directed graph $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ where $\mathcal{A}$ are the *arguments* and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is the *attack relation*. We focus on several classical extension-based semantics, *i.e.* functions σ s.t. $\sigma(\mathcal{F}) \subseteq 2^{\mathcal{A}}$. The semantics that we consider require an extension $S \subseteq \mathcal{A}$ to be *conflict-free* (*i.e.* $\forall a, b \in S$, $(a, b) \notin \mathcal{R}$) and admissible (*i.e.* S is conflict-free and defend all its elements, meaning that $\forall a \in S$, $\forall b \in \mathcal{A}$ s.t. $(b, a) \in \mathcal{R}$, $\exists c \in S$ s.t. $(c, b) \in \mathcal{R}$). The semantics are:

- Complete (CO): S is an extension iff it is an admissible set which does not defend any argument outside of S,
- Preferred (PR): S is an extension iff it is a $\subseteq$-maximal CO-extension,
- Stable (ST): S is an extension iff it is a conflict-free s.t. $\forall b \in \mathcal{A} \setminus S$, $\exists a \in S$ s.t. $(a, b) \in \mathcal{R}$.

We consider all the reasoning tasks for the Main Track of ICCMA 2023:[1]:

- DC-σ: Given $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ and $a \in \mathcal{A}$, does a belong to some σ-extension of $\mathcal{F}$? If yes, provide a σ-extension containing a.
- DS-σ: Given $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ and $a \in \mathcal{A}$, does a belong to each σ-extension of $\mathcal{F}$? If no, provide a σ-extension not containing a.
- SE-σ: Given $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$, provide a σ-extension of $\mathcal{F}$.

Our solver can provide answers to six problems: DC-CO, DC-ST, DS-PR, DS-ST, SE-PR and SE-ST.[2]

[1]https://iccma2023.github.io/tracks.html

[2]DC-PR is ignored because it is equivalent to DC-CO; DS-CO and SE-CO are ignored because they are equivalent to the same problems for the grounded semantics, which is polynomially computable.

II. System

The solver PORTSAT has been implemented in Rust.[3] It uses a portfolio of SAT solvers to solve the various request to NP-complete oracles. The source code and documentation are available online.[4]

A. SAT Encoding

The propositional encoding for the various semantics are inspired by the classical approach from [2]. The formula Φ_{ST} from Figure 1 is such that its set of models correspond to the stable extensions of an AF $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$. So one can solve the problem SE-ST by asking a SAT solver to provide a model of it, and DC-ST (resp. DS-ST) by asking for a model of $\Phi_{\mathrm{ST}} \wedge a$ (resp. $\Phi_{\mathrm{ST}} \wedge \neg a$).

$$\bigwedge_{a_i \in \mathcal{A}} \left(a_i \vee \bigvee_{(a_j, a_i) \in \mathcal{R}} a_j \right) \wedge \bigwedge_{a_i \in \mathcal{A}} \left[\bigwedge_{(a_j, a_i) \in \mathcal{R}} (\neg a_i \vee \neg a_j) \right]$$

Fig. 1. Φ_{ST}

Similarly, the models of the formula Φ_{CO} from Figure 2 correspond to the complete extensions of $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$, so a model of $\Phi_{\mathrm{CO}} \wedge a$ provides an answer to DC-CO.

$$\begin{array}{c} \bigwedge_{a_i \in A} (\neg a_i \vee \neg P_{a_i}) \wedge \\ \bigwedge_{a_i \in A} \left(a_i \vee \bigvee_{a_j \in A, (a_j, a_i) \in R} \neg P_{a_j} \right) \wedge \\ \bigwedge_{a_i \in A} \bigwedge_{a_j \in A, (a_j, a_i) \in R} \left(\neg a_i \vee P_{a_j} \right) \wedge \\ \bigwedge_{a_i \in A} \left(\neg P_{a_i} \vee \bigvee_{a_j \in A, (a_j, a_i) \in R} a_j \right) \wedge \\ \bigwedge_{a_i \in A} \bigwedge_{a_j \in A, (a_j, a_i) \in R} (P_{a_i} \vee \neg a_j) \end{array}$$

Fig. 2. Φ_{CO}

For the preferred semantics, we compute the largest complete extension with Φ_{CO} for solving SE-PR, and we

[3]https://www.rust-lang.org

[4]https://github.com/QuentinJanuel/PORTSAT

enumerate-and-check all the complete extensions to obtain all the preferred ones, and from there we can check whether an argument belongs to all of them to solve DS-PR.

B. SAT Solvers

a) MiniSat: MiniSat is a major reference in the field of Boolean Satisfiability [8]. It is a minimalistic open-source SAT solver, that was successful during the SAT competitions following its release. Many other SAT solvers have been based on Minisat, which makes its API a common tool in the SAT community. Like most of the modern SAT solvers, it is based on the DPLL/CDCL approach [4], [11].

b) ManySat: ManySat is a portfolio of variations of the DPLL algorithm [9], which runs in parallel these various SAT solvers to benefit from their respective strengths.

c) MapleSAT: MapleSAT [10] uses MiniSat combined with machine learning techniques instead of "classical" heuristics.

d) Glucose: Glucose is (again) a SAT solver based on Minisat, whose main feature is the concept of "glue clauses" [1], a special type of clauses which have an important role during the search. PORTSAT actually includes two versions of Glucose, with or without preprocessing. The version with preprocessing generally outperforms the other version on "hard" problems, but the latter seems to be more efficient on "easy" instances.

III. Conclusion

Future work include the addition of other SAT solvers to the portfolio, the implementation of better algorithms for solving the current list of problems (in particular, a CEGAR-style approach [7] for solving DS-PR would probably outperform the current naive implementation) and the addition of other semantics (*e.g.* semi-stable [3], stage [12], ideal [6]).

References

[1] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solvers. In C. Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 399–404, 2009.

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

[3] M. W. A. Caminada, W. A. Carnielli, and P. E. Dunne. Semi-stable semantics. *J. Log. Comput.*, 22(5):1207–1254, 2012.

[4] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[5] P. M. 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.

[6] P. M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artif. Intell.*, 171(10-15):642–674, 2007.

[7] W. Dvorák, M. Järvisalo, J. P. Wallner, and S. Woltran. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.*, 206:53–78, 2014.

[8] N. Eén and N. Sörensson. An extensible sat-solver. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[9] Y. Hamadi, S. Jabbour, and L. Sais. Manysat: a parallel SAT solver. *J. Satisf. Boolean Model. Comput.*, 6(4):245–262, 2009.

[10] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki. Learning rate based branching heuristic for SAT solvers. In N. Creignou and D. L. Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 123–140. Springer, 2016.

[11] J. P. M. Silva and K. A. Sakallah. GRASP - a new search algorithm for satisfiability. In R. A. Rutenbar and R. H. J. M. Otten, editors, *Proceedings of the 1996 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 1996, San Jose, CA, USA, November 10-14, 1996*, pages 220–227. IEEE Computer Society / ACM, 1996.

[12] B. Verheij. Two approaches to dialectical argumentation: Admissible sets and argumentation stages. In *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC'96)*, pages 357–368, 1996.