SAT-IT: the Interactive SAT Tracer (extended abstract)

Marc Cané
Department of Computer Science, Applied Mathematics and Statistics, University of Girona, Spain

Jordi Coll ✉
Institut d'Investigació en Intel·ligència Artificial, CSIC, Bellaterra, Spain

Marc Rojo
Department of Computer Science, Applied Mathematics and Statistics, University of Girona, Spain

Mateu Villaret ✉
Department of Computer Science, Applied Mathematics and Statistics, University of Girona, Spain

Abstract

In this paper we present the Interactive SAT Tracer (SAT-IT), a tool to assist teaching and research in SAT solving. This is an extended abstract of [3].

2012 ACM Subject Classification Theory of computation → Constraint and logic programming

Keywords and phrases SAT, CDCL, teaching tool

Digital Object Identifier 10.4230/LIPIcs.WTCP.2023.5

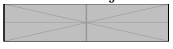
Supplementary Material *Software and Documentation*: `https://imae.udg.edu/Recerca/LAI/`

Funding *Jordi Coll*: Grants PID2019-111544GB-C21 , TED2021-129319B-I00 and PID2022-139835NB-C21, funded by MCIN/AEI /10.13039/501100011033.
Mateu Villaret: Grant PID2021-122274OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by ERDF A way of making Europe

The Interactive SAT Tracer

The Boolean Satisfiability problem (SAT) is the paradigmatic NP-complete problem [4, 8]. This is the problem of deciding whether a Boolean propositional formula can be satisfied. This problem is not only relevant for being the first problem that was shown to be NP-complete, but his popularity and research interest have been continuously increasing during the last decades for its applicability as a problem-solving paradigm. The key factors in the success of SAT are the facility to translate a plethora of hard constraint satisfaction and optimisation problems to SAT formulas, and the development of extremely efficient algorithms for solving such formulas. Despite the enormous theoretical computational complexity of SAT, nowadays we have efficient methods capable of solving huge formulas coming from problems of industrial interest such as circuit verification [6], planning [7], scheduling or timetabling [5, 2], to name a few.

Except for very specific domains, there is one clearly predominating algorithm to solve SAT: the Conflict-Driven Clause Learning algorithm (CDCL). The essence of CDCL is a combination of search and inference. It consists of a (non-chronological) backtracking scheme that explores a search tree to find a solution if any exists, enhanced with the Unit Propagation (UP) rule and which uses the Resolution rule to learn new clauses from dead ends of the search tree. Therefore, understanding the basics of SAT-solving requires to get familiar with the previously mentioned inference rules and their integration into a SAT solving algorithm. In this regard, in [9] there was presented a very compact and precise rule-based framework to describe SAT solving algorithms.



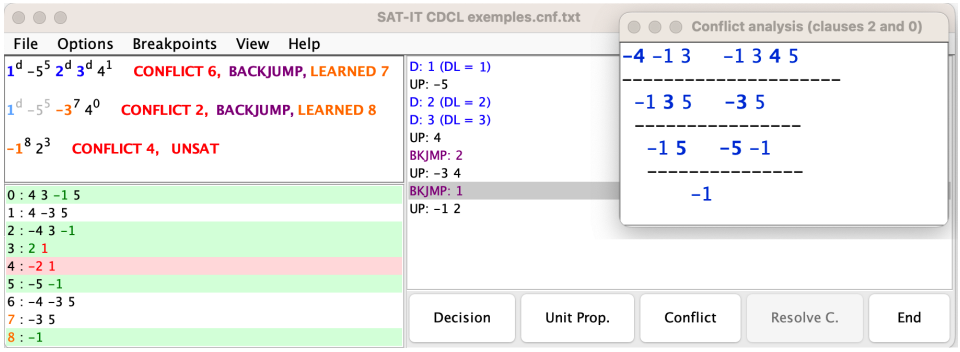


Figure 1 Main view of SAT-IT, with a completed execution of the CDCL algorithm, and the pop-up window showing the conflict analysis for the second backjump.

Inspired by the previously mentioned framework, in this work we present the Interactive SAT Tracer (SAT-IT), a visual and interactive tool to monitor and illustrate the basic algorithms for SAT solving. In order to facilitate the learning of SAT solving techniques to users that start without a background knowledge, we consider three algorithms with progressively increasing sophistication: simple backtracking, its extension with UP, so called Davis–Putnam–Logemann–Loveland (DPLL), and its further extension with conflict-driven clause learning, i.e. CDCL. In contrast to the teaching tool LearnSAT [1] our tool provides an interactive environment with some graphical support. The motivation of our tool is to provide an environment where the user can see the full trace of a SAT solving process in a compact but detailed way and understand the reasons why every variable assignment, backtrack or backjump occurs. Further information is included in the tool, such as the sequence of resolution rules involved in each conflict analysis, what clauses have been learnt, or what are the inspected literals involved in the 2-watched literals scheme for implementing UP. Moreover, the user is able to control the solving process evolution at the desired pace and choose what variables should be used for branchings (or decisions). The system displays a full log of the SAT solving process that allows to see which has been the total progress of the execution until the current point, and users can go back to any previous point of the solving process to reinspect or to try "what if" scenarios. In particular, one could see which variables are unit-propagated after some particular assignments. This could serve the user to get some practical insights of some properties of the encodings such as correctness or generalized arc consistency enforcement by UP.

SAT-IT is publicly available[1]. This tool allows the user to work with CNFs in DIMACS format using any of the three considered algorithms. We find an example execution of CDCL in Figure 1 considering the following clauses and where decisions are done choosing the first unassigned variable in the order $1, 2, \ldots, 5$, and always the positive literal:

$$C_0 : x_3 \vee x_4 \vee \neg x_1 \vee x_5 \quad C_2 : x_3 \vee \neg x_4 \vee \neg x_1 \quad C_4 : x_1 \vee \neg x_2 \quad C_6 : \neg x_3 \vee \neg x_4 \vee x_5$$
$$C_1 : \neg x_3 \vee x_4 \vee x_5 \quad C_3 : x_1 \vee x_2 \quad C_5 : \neg x_1 \vee \neg x_5$$

We plan to improve SAT-IT by including more features of CDCL SAT solvers such as restarts, learnt clause removal and bounded variable elimination, by providing graphic representation of unsatisfiability proofs, and by supporting MaxSAT solving algorithms.

[1] `https://imae.udg.edu/Recerca/LAI/`

References

1 Mordechai Moti Ben-Ari. Learnsat: a sat solver for education. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 403–407. Springer, 2013.

2 Miquel Bofill, Jordi Coll, Marc Garcia, Jesús Giráldez-Cru, Gilles Pesant, Josep Suy, and Mateu Villaret. Constraint solving approaches to the business-to-business meeting scheduling problem. *J. Artif. Intell. Res.*, 74:263–301, 2022. doi:10.1613/jair.1.12670.

3 Marc Cané, Jordi Coll, Marc Rojo, and Mateu Villaret. SAT-IT: the Interactive SAT Tracer. In *Proceedings of the 25th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2023*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2023 (in press).

4 Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, 1971.

5 Emir Demirovic, Nysret Musliu, and Felix Winter. Modeling and solving staff scheduling with partial weighted maxsat. *Ann. Oper. Res.*, 275(1):79–99, 2019. doi:10.1007/s10479-017-2693-y.

6 Daniela Kaufmann, Armin Biere, and Manuel Kauers. Verifying large multipliers by combining sat and computer algebra. In *2019 Formal Methods in Computer Aided Design (FMCAD)*, pages 28–36, 2019. doi:10.23919/FMCAD.2019.8894250.

7 Henry A Kautz, Bart Selman, et al. Planning as satisfiability. In *ECAI*, volume 92, pages 359–363. Citeseer, 1992.

8 Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

9 Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(T). *Journal of the ACM*, 53(6):937–977, 2006. doi:10.1145/1217856.1217859.