The OpenSMT Solver in SMT-COMP 2024

Tomáš Kolárik[1], Martin Blicha[1,3], Konstantin I. Britikov[1],
Antti E. J. Hyvärinen[2], Rodrigo Otoni[1], and
Natasha Sharygina[1]

[1] Università della Svizzera italiana (USI), Lugano, Switzerland
[2] Certora
[3] Ethereum Foundation

1 Overview

OpenSMT [12] is a T-DPLL based SMT solver [17] that has been developed at USI in Switzerland since 2008. The solver is written in `C++` and currently supports the quantifier-free logics of equality with uninterpreted functions (QF_UF), linear real and integer arithmetic (QF_LRA, QF_LIA), arrays (QF_AX) and their combinations (QF_UFLRA, QF_UFLIA, QF_ALIA, QF_ALRA, QF_AUFLIA, QF_AUFLRA). It has a specialized solver for real and integer difference logics (QF_RDL, QF_IDL).

In comparison to 2023, the 2024 competition entry adds an experimental support for unsat cores. We also re-established our participation in the parallel and cloud track (see below).

OpenSMT features not exercised in the competition include support for a wide range of interpolation algorithms for propositional logic [2], linear real arithmetic [8], and uninterpreted functions [3] (available also in the incremental mode); an experimental lookahead-based search algorithm [13] as an alternative to the more standard CDCL algorithm; and features that support search-space partitioning in particular designed for parallel solving [14]. OpenSMT is now also able to efficiently produce proofs of unsatisfiability [18], although this feature is not merged to the main repository.

2 Cloud and Parallel Solver

The parallel version of OpenSMT, called SMTS, runs on our parallelization infrastructure described in [16]. The system is a client-server architecture that communicates with a custom protocol over TCP/IP. Parallel solving features include the complete implementation of the partition tree protocol with clause sharing. On a high level, the parallel solver partitions the instance dynamically on-demand and allows clauses to be shared between solvers working on different instances whenever this is allowed based on the information in the solvers. The algorithm is called *cube-and-conquer* [14].

We support also the cloud version of OpenSMT, running the solver e.g. in the AWS infrastructure.

3 External Code and Contributors

The SAT solver driving OpenSMT is based on the MiniSAT solver [9], and the rational number implementation is inspired by a library written by David Monniaux. Several people have directly contributed to the OpenSMT code. In alphabetical order, the major contributors are Leonardo Alt (Ethereum Foundation), Sepideh Asadi (USI), Masoud Asadzade (USI), Martin Blicha (USI, Ethereum Foundation), Konstantin I. Britikov (USI), Roberto Bruttomesso (Cybersecurity / Nozomi Networks), Antti E. J. Hyvärinen (Certora), Andrew Jones (Vector), Tomáš Kolárik (USI), Václav Luňák (Charles University), Matteo Marescotti (Meta), Rodrigo Benedito Otoni (USI), Edgar Pek (University of Illinois, Urbana-Champaign), Simone Fulvio Rollini (United Technologies Research Center), Parvin Sadigova (King's College London), Mate Soos (Ethereum Foundation), Michal Tarina and Aliaksei Tsitovich (Sonova). The solver is being developed in Natasha Sharygina's software verification group at USI.

4 Utilization

OpenSMT is used in a range of projects as a back-end solver. Most notably, it is a basis for a new CHC solver Golem [7] which scored among the top solvers in LIA-Lin, LIA-Nonlin, and LRA-TS tracks in the last five editions of CHC-COMP [19, 11, 4, 5, 10]. OpenSMT also forms the basis of the model checkers HiFrog [1] and UpProver [6]. It was also used as an interpolation engine for the Sally model checker [15].

5 Availability

The source code repository and more information on the solver is available at

- https://github.com/usi-verification-and-security/opensmt and
- https://verify.inf.usi.ch/opensmt

References

[1] Leonardo Alt, Sepideh Asadi, Hana Chockler, Karine Even-Mendoza, Grigory Fedyukovich, Antti E. J. Hyvärinen, and Natasha Sharygina. HiFrog: SMT-based function summarization for software verification. In *Proc. TACAS 2017*, volume 10206 of *LNCS*, pages 207–213. Springer, 2017.

[2] Leonardo Alt, Grigory Fedyukovich, Antti E. J. Hyvärinen, and Natasha Sharygina. A proof-sensitive approach for small propositional interpolants. In *Proc. VSTTE 2015*, volume 9593 of *LNCS*, pages 1–18. Springer, 2016.

[3] Leonardo Alt, Antti Eero Johannes Hyvärinen, Sepideh Asadi, and Natasha Sharygina. Duality-based interpolation for quantifier-free equalities and uninterpreted functions. In *Proc. FMCAD 2017*, pages 39–46. IEEE, 2017.

[4] Emanuele De Angelis and Hari Govind V K. CHC-COMP 2022: Competition report. *Electronic Proceedings in Theoretical Computer Science*, 373:44–62, nov 2022.

[5] Emanuele De Angelis and Hari Govind V K. CHC-COMP 2023 report. https://chc-comp.github.io/2023/CHC-COMP%202023%20Report%20-%20HCSV.pdf, 2023.

[6] Sepideh Asadi, Martin Blicha, Antti E. J. Hyvärinen, Grigory Fedyukovich, and Natasha Sharygina. Incremental verification by smt-based summary repair. In *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*, pages 77–82. IEEE, 2020.

[7] Martin Blicha, Konstantin Britikov, and Natasha Sharygina. The golem horn solver. In Constantin Enea and Akash Lal, editors, *Computer Aided Verification*, pages 209–223, Cham, 2023. Springer Nature Switzerland.

[8] Martin Blicha, Antti E. J. Hyvärinen, Jan Kofron, and Natasha Sharygina. Decomposing Farkas interpolants. In *Proc. TACAS 2019*, volume 11427 of *LNCS*, pages 3–20. Springer, 2019.

[9] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proc. SAT 2004*, volume 2919 of *LNCS*, pages 502–518. Springer, 2004.

[10] Gidon Ernst. CHC-COMP 2024 report. `https://chc-comp.github.io/CHC-COMP%202024%20Report%20-%20HCSV.pdf`, 2024.

[11] Grigory Fedyukovich and Philipp Rümmer. Competition report: CHC-COMP-21. In Hossein Hojjat and Bishoksan Kafle, editors, *Proceedings 8th Workshop on Horn Clauses for Verification and Synthesis, HCVS@ETAPS 2021, Virtual, 28th March 2021*, volume 344 of *EPTCS*, pages 91–108, 2021.

[12] Antti E. J. Hyvärinen, Matteo Marescotti, Leonardo Alt, and Natasha Sharygina. OpenSMT2: An SMT solver for multi-core and cloud computing. In *Proc. SAT 2016*, volume 9710 of *LNCS*, pages 547–553. Springer, 2016.

[13] Antti E. J. Hyvärinen, Matteo Marescotti, Parvin Sadigova, Hana Chockler, and Natasha Sharygina. Lookahead-based SMT solving. In *Proc. LPAR-22*, volume 57 of *EPiC Series in Computing*, pages 418–434. EasyChair, 2018.

[14] Antti E. J. Hyvärinen, Matteo Marescotti, and Natasha Sharygina. Search-space partitioning for parallelizing SMT solvers. In *Proc. SAT 2015*, volume 9340 of *LNCS*, pages 369–386. Springer, 2015.

[15] Dejan Jovanovic and Bruno Dutertre. Property-directed k-induction. In *Proc. FMCAD 2016*, pages 85–92. IEEE, 2016.

[16] Matteo Marescotti, Antti E. J. Hyvärinen, and Natasha Sharygina. SMTS: distributed, visualized constraint solving. In *Proc. LPAR-22*, volume 57 of *EPiC Series in Computing*, pages 534–542. EasyChair, 2018.

[17] 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.

[18] Rodrigo Otoni, Martin Blicha, Patrick Eugster, Antti E. J. Hyvärinen, and Natasha Sharygina. Theory-specific proof steps witnessing correctness of SMT executions. In *Proc. DAC 2021*. IEEE, 2021. To appear.

[19] Philipp Rümmer. Competition report: CHC-COMP-20. *Electronic Proceedings in Theoretical Computer Science*, 320:197–219, 08 2020.