The OpenSMT Solver in SMT-COMP 2021

Masoud Asadzade, Martin Blicha, Antti E. J. Hyvärinen, and Natasha Sharygina

Università della Svizzera italiana (USI), Lugano, Switzerland

1 Overview

OpenSMT [7] is a T-DPLL based SMT solver [13] that has been developed at USI, 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), and real and integer difference logics (QF_RDL, QF_IDL). OpenSMT also supports some aspects of bit-vector logic (QF_BV).

In comparison to 2020, the 2021 competition entry features a dedicated difference logic solver based on Exhaustive Theory Propagation [12], contributed by Václav Luňák, and support for producing models for all fully supported logics (QF_UF, QF_LRA, QF_LIA, QF_RDL, QF_IDL). Additionally, the handling of ITE terms has been improved, resulting in a large performance benefit on QF_LIA benchmarks containing complex nested ITE terms.

OpenSMT features not exercised in the competition include support for a wide range of interpolation algorithms for propositional logic [2], linear real arithmetic [5], and uninterpreted functions [3] (available also in the incremental mode); an experimental lookahead-based search algorithm [8] as an alternative to the more standard CDCL algorithm; and features that support search-space partitioning in particular designed for parallel solving [9]. OpenSMT is now also able to efficiently produce certificates of unsatisfiability [14], although this feature has not yet been merged to the main repository.

2 Cloud Solver

The cloud version of OpenSMT, called SMTS, is designed to run in AWS infrastructure and runs on our parallelization infrastructure described in [11]. We have entered two versions of the solver to the experimental cloud track: *SMTS portfolio* which randomises the SAT solver by choosing 2% of the decision variables randomly; and *SMTS cube-and-conquer*, which uses the *parallelization tree* [9] approach to implement a version of search-space-partitioning.

3 External Code and Contributors

The SAT solver driving OpenSMT is based on the MiniSAT solver [6], 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, Charles University), Roberto Bruttomesso (Cybersecurity / Nozomi Networks), Antti E. J. Hyvärinen (USI), Václav Luňák (Charles University), Matteo Marescotti (USI), Rodrigo Benedito Otoni (USI), Edgar Pek (University of Illinois, Urbana-Champaign), Simone Fulvio Rollini (United Technologies Research Center), Parvin Sadigova (King's College London), 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. It has been used as an interpolation engine of the Sally model checker [10] which won the first and the second place in the transition systems category in the constrained Horn clause competition 2019 and 2020, respectively. Recently, is has been used as the basis for a new CHC solver Golem which won the second place in LRA-TS and LIA-Lin categories in CHC-COMP 2021. OpenSMT also forms the basis of the model checkers HiFrog [1] and UpProver [4].

5 Availability

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

- https://github.com/usi-verification-and-security/opensmt and
- http://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] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] Robert Nieuwenhuis and Albert Oliveras. Dpll(t) with exhaustive theory propagation and its application to difference logic. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification*, pages 321–334, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[13] 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.

[14] 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.