SMTInterpol

Version 2.5-671-g6d0a7c6e

Jochen Hoenicke and Tanja Schindler

University of Freiburg
{hoenicke,schindle}@informatik.uni-freiburg.de

May 25, 2020

Description

SMTInterpol is an SMT solver written in Java and available under LGPL v3. It supports the combination of the theories of uninterpreted functions, linear arithmetic over integers and reals, and arrays. Furthermore it can produce models, proofs, unsatisfiable cores, and interpolants. The solver reads input in SMT-LIB format. It includes parsers for DIMACS, AIGER, and SMT-LIB version 1.2 and 2.5.

The solver is based on the well-known DPLL(T)/CDCL framework [GHN+04]. It uses variants of standard algorithms for CNF conversion [PG86] and congruence closure [NO05]. The solver for linear arithmetic is based on Simplex [DdM06], the sum-of-infeasibility algorithm [KBD13], and branch-and-cut for integer arithmetic [CH15a, DDA09]. The array decision procedure is based on weak equivalences [CH15b] and includes an extension for constant arrays [HS19]. Theory combination is performed based on partial models produced by the theory solvers [dMB08].

In the current release, the solver for quantified formulas was extensively revised. The core of the new approach is an incremental search for instances of quantified clauses that result in a conflict or a propagation. Instead of using E-matching as an instantiation heuristic, the E-graph is used to detect and evaluate candidate instances before adding them to the ground problem. The solver decides the finite almost uninterpreted fragment [GdM09].

The main focus of SMTInterpol is the incremental track. This track simulates the typical application of SMTInterpol where a user asks multiple queries. As an extension SMTInterpol supports quantifier-free interpolation for the supported theories [CH16, HS18, HS19]. This makes it useful as a backend for software verification tools. In particular, Ultimate Automizer[1] and CPAchecker[2], the winners of the SV-COMP 2016–2020, use SMTInterpol.

Competition Version

The version submitted to the SMT-COMP 2020 includes a revised solver for quantified formulas. Proof production for quantified formulas is only partially implemented, and while this version supports unsat core extraction, it does not support interpolation in the presence of quantified formulas. The solver is conservative and returns unknown for satisfiable formulas that are not in the supported fragment.

[1] https://ultimate.informatik.uni-freiburg.de/
[2] https://cpachecker.sosy-lab.org/

Webpage and Sources

Further information about SMTInterpol can be found at

http://ultimate.informatik.uni-freiburg.de/smtinterpol/

The sources are available via GitHub

https://github.com/ultimate-pa/smtinterpol

Authors

The code was written by Jürgen Christ, Matthias Heizmann, Jochen Hoenicke, Alexander Nutz, Markus Pomrehn, Pascal Raiola, and Tanja Schindler.

Logics, Tracks and Magic Number

SMTInterpol participates in the single query track, the incremental track, the unsat core track, and the model validation track. It supports all combinations of uninterpreted functions, linear arithmetic, and arrays, and participates in the following logic divisions:

ALIA, AUFLIA, AUFLIRA, LIA, LRA, QF_ALIA, QF_AUFLIA, QF_AX, QF_IDL, QF_LIA, QF_LIRA, QF_LRA, QF_RDL, QF_UF, QF_UFIDL, QF_UFLIA, QF_UFLRA, UF, UFIDL, UFLIA, UFLRA.

Magic Number: 192 843 011

References

[CH15a] Jürgen Christ and Jochen Hoenicke. Cutting the mix. In *CAV*, pages 37–52, 2015.

[CH15b] Jürgen Christ and Jochen Hoenicke. Weakly equivalent arrays. In *FROCOS*, pages 119–134, 2015.

[CH16] Jürgen Christ and Jochen Hoenicke. Proof tree preserving tree interpolation. *J. Autom. Reasoning*, 57(1):67–95, 2016.

[DDA09] Isil Dillig, Thomas Dillig, and Alex Aiken. Cuts from proofs: A complete and practical technique for solving linear inequalities over integers. In *CAV*, pages 233–247, 2009.

[DdM06] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for DPLL(T). In *CAV*, pages 81–94, 2006.

[dMB08] Leonardo de Moura and Nikolaj Bjørner. Model-based theory combination. *Electr. Notes Theor. Comput. Sci.*, 198(2):37–49, 2008.

[GdM09] Yeting Ge and Leonardo Mendonça de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.

[GHN+04] Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. DPLL(T): fast decision procedures. In *CAV*, volume 3114 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2004.

[HS18] Jochen Hoenicke and Tanja Schindler. Efficient interpolation for the theory of arrays. In *IJCAR*, volume 10900 of *Lecture Notes in Computer Science*, pages 549–565. Springer, 2018.

[HS19] Jochen Hoenicke and Tanja Schindler. Solving and interpolating constant arrays based on weak equivalences. In *VMCAI*, volume 11388 of *Lecture Notes in Computer Science*, pages 297–317. Springer, 2019.

[KBD13] Tim King, Clark Barrett, and Bruno Dutertre. Simplex with sum of infeasibilities for SMT. In *FMCAD*, pages 189–196. IEEE, 2013.

[NO05] Robert Nieuwenhuis and Albert Oliveras. Proof-producing congruence closure. In *RTA*, pages 453–468. Springer, 2005.

[PG86] David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *J. Symb. Comput.*, 2(3):293–304, 1986.