SMTInterpol

Version 2.5-514-gea29b303

Jochen Hoenicke and Tanja Schindler

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

June 17, 2019

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]. The solver uses variants of standard algorithms for CNF conversion [PG86], congruence closure [NO05], Simplex [DdM06] and branch-and-cut for integer arithmetic [CH15a, DDA09]. The array decision procedure is based on weak equivalences [CH15b]. Theory combination is performed based on partial models produced by the theory solvers [dMB08].

This release uses the "sum-of-infeasibility" algorithm [KBD13] for linear arithmetic. The array theory was extended by constant arrays [HS19]. New in the current release is an experimental solver for quantified formulas. It is based on model-based quantifier instantiation, in particular, it solves the almost uninterpreted fragment [GdM09]. This approach is combined with ideas from conflict-based quantifier instantiation [RTdM14]. A distinguishing feature is that it does not use pattern-based E-matching at all, but uses the E-graph to identify potential conflict and unit clauses.

The main focus of SMTInterpol is the incremental track. This track simulates the typical application of SMTInterpol where a user asks multiple queries. The main focus of the development team of SMTInterpol is the interpolation engine [CH16, HS18]. 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 2016–2019, use SMTInterpol.

Competition Version

The version submitted to the SMT-COMP 2019 is an experimental release with the new solver for quantified formulas. The algorithms to generate proofs and models are not adapted to the new quantifier theory. Therefore, some features like unsat cores do not work yet. 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, and the unsat core track. In the single-query and incremental track it supports all combinations of uninterpreted functions, linear arithmetic, and arrays: 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. In the unsat core track, SMTInterpol participates only in the quantifier-free logics: QF_ALIA, QF_AUFLIA, QF_AX, QF_IDL, QF_LIA, QF_LIRA, QF_LRA, QF_RDL, QF_UF, QF_UFIDL, QF_UFLIA, QF_UFLRA.

Magic Number: 983 571 724

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.

[RTdM14] Andrew Reynolds, Cesare Tinelli, and Leonardo Mendonça de Moura. Finding conflicting instances of quantified formulas in SMT. In *FMCAD*, pages 195–202. IEEE, 2014.