SMTInterpol

Version 2.5-1381-g0e9bd0bf

Max Barth[1], Jochen Hoenicke[2], and Tanja Schindler[3]

[1] LMU Munich, max.barth@lmu.de
[2] Certora, jochen@certora.com
[3] University of Basel, tanja.schindler@unibas.ch

Description

SMTInterpol is an SMT solver written in Java and available under LGPL v3. It supports the combination of the theories of uninterpreted functions, arithmetic over integers and reals, arrays, and datatypes. 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 2.6.

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]. The decision procedure for data types is based on the rules presented in [BST07]. The solver for quantified formulas performs an incremental search for conflicting and unit-propagating instances of quantified formulas [HS21] which is complemented with a version of enumerative instantiation [RBF18] to ensure completeness for the finite almost uninterpreted fragment [GdM09]. Theory combination is performed based on partial models produced by the theory solvers [dMB08]. SMTInterpol comes with a proof production [HS22] that is based on resolution using a minimal set of axioms.

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 interpolation for all supported theories [CH16, HS18, HS19, HHS21]. It computes quantifier-free interpolants for quantifier-free input formulas. 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 2016–2024, used SMTInterpol.

SMTInterpol aims for producing witnesses (proofs and models). The competition version will check these witnesses before reporting the result to ensure soundness. Only sat results for quantified logics are not checked.

Competition Version

The version of SMTInterpol submitted to the SMT-COMP 2024 contains preliminary support for non-linear benchmarks. Multiplication of arbitrary terms is handled by the pre-processor. However, the linear solver will treat products of variables as unconstrained variables. If a model is found, the solver reports unknown if a non-linear multiplication is part of the tableau.

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

We also have an experimental bit-vector solver based on a translation to integer arithmetic. The bit-vector rewrite rules are not checked by the proof checker so bugs may lead to wrong unsat results.

Authors, Logics, and Tracks

The code was written by Max Barth, Leon Cacace, Jürgen Christ, Daniel Dietsch, Leonard Fichtner, Joanna Greulich, Elisabeth Henkel, Matthias Heizmann, Jochen Hoenicke, Moritz Mohr, Alexander Nutz, Markus Pomrehn, Pascal Raiola, and Tanja Schindler. Further information about SMTInterpol can be found at

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

The sources are available via GitHub

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

SMTInterpol participates in all tracks: the single query track, the incremental track, the unsat core track, and the model validation track. It supports all combinations of uninterpreted functions, arithmetic, arrays, datatypes, bit-vectors, and quantified formulas. However, the model generator currently does not support quantifiers. SMTInterpol participates in the logics matched by `(QF_)?(AX?)?(BV)?(UF)?(DT)?([IR]DL|[NL][IR]*A)?`[3]. In the unsat core track it will not participates in logics containing BV.

References

[BST07] Clark W. Barrett, Igor Shikanian, and Cesare Tinelli. An abstract decision procedure for a theory of inductive data types. *J. Satisf. Boolean Model. Comput.*, 3(1-2):21–46, 2007.

[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.

[HHS21] Elisabeth Henkel, Jochen Hoenicke, and Tanja Schindler. Proof tree preserving sequence interpolation of quantified formulas in the theory of equality. In *SMT*, volume 2908 of *CEUR Workshop Proceedings*, pages 3–16. CEUR-WS.org, 2021.

[3] SMTInterpol will mostly ignore non-linear arithmetic, and report unknown on benchmarks that need non-linear reasoning. It fully supports modulo and division by constants, though.

[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.

[HS21] Jochen Hoenicke and Tanja Schindler. Incremental search for conflict and unit instances of quantified formulas with e-matching. In *VMCAI*, volume 12597 of *Lecture Notes in Computer Science*, pages 534–555. Springer, 2021.

[HS22] Jochen Hoenicke and Tanja Schindler. A simple proof format for SMT. In *SMT 2022*, CEUR Workshop Proceedings. CEUR-WS.org, 2022. to appear.

[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.

[RBF18] Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine. Revisiting enumerative instantiation. In *TACAS (2)*, volume 10806 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 2018.