SMTInterpol

Version 2.0

Jürgen Christ Jochen Hoenicke Alexander Nutz
University of Freiburg
{christj,hoenicke,nutz}@informatik.uni-freiburg.de

Introduction

SMTInterpol [CHN12] is a proof-producing and interpolating SMT-solver written in Java. It is available from http://ultimate.informatik.uni-freiburg.de/smtinterpol under the GNU Lesser General Public License (LGPL) version 3.0. The solver reads input in SMTLIB format. It includes a parser for version 1.2, and a parser for the current version. All required and some optional commands of the SMTLIB standard are supported. SMTInterpol supports the quantifier-free combination of uninterpreted functions and linear (real and integer) arithmetic, i. e., the SMTLIB logics QF_UF, QF_LIA, QF_LRA, QF_UFLIA, and QF_UFLRA. For all these logics, SMTInterpol supports the computation of inductive sequences of Craig interpolants, which are used by several interpolation-based model checkers [HHP09, HHP10, EHP12].

All formulas are stored in a central term repository. The repository type-checks the formulas. Asserted formulas are converted to CNF using Plaisted–Greenbaum encoding [PG86]. The core of the solver is a CDCL engine that is connected to multiple theories. The engine uses these theories during constraint propagation, backtracking, and consistency checking.

For uninterpreted functions and predicates, we use a theory solver based on the congruence closure algorithm. An extension to arrays and quantifiers via e-matching is under development. For linear arithmetic, we use a theory solver based on the Simplex algorithm [DdM06]. It always computes the strongest bounds that can be derived for a variable and uses them during satisfiability checks. If a conflict cannot be explained using known literals, the solver derives new literals and uses them in conflict explanation. Disequalities are used to strengthen bounds, or are delayed until final checks. The solver supports integer arithmetic using a variant of the cuts from proof technique [DDA09] together with a branch-and-bound engine.

SMTInterpol uses a variant of model-based theory combination [dMB08]. The linear arithmetic solver does not propagate equalities between shared variables but introduces them as decision points. The model mutation algorithm resolves disequalities and tries to create as many distinct equivalence classes as possible.

Interpolation

SMTInterpol produces inductive sequences of interpolants for the SMTLIB logics QF_UF, QF_LRA, QF_UFLRA, QF_LIA, and QF_UFLIA. Since the integer logics defined in the SMTLIB standard are not closed under interpolation, SMTInterpol extends these logics with the division and modulo operators with constant divisor.

The architecture of the interpolation engine roughly follows the DPLL(T) paradigm: A *core interpolator* produces *partial interpolants* for the resolution steps while theory specific interpolators produce partial interpolants for T-lemmas. In the presence of *mixed literals*, i.e., literals that use symbols from more than one block of the interpolation problem, an approach loosely based on the method of Yorsh et al. [YM05] is used. The basic idea of the approach used in SMTInterpol is to virtually purify each mixed literal using an auxiliary variable, to restrict the places where the variable may occur in partial

interpolants, and to use special resolution rules to eliminate the variable when the mixed literal is used as a pivot. In essence, for convex theories, this approach can be seen as a lazy version of the method of Yorsh et al. The approach also works for non-convex theories using disjunctions in the interpolants.

New Developments

Compared to the version that participated in the SMT competition in 2011, several performance improvements have been implemented. These include clause minimization techniques and a new pivot strategy in the linear arithmetic solver. Additionally, the assertion stack management has been reworked to be more stable.

For unsatisfiable formulas, SMTInterpol supports the optional SMTLIB command `get-unsat-core`. This command extracts an unsatisfiable core from a proof tree. This technique does not guarantee minimality of the returned core. The optional command `get-proof` was already supported in last year's version. Additionally, the non-standard command `get-interpolants` can be used to compute an inductive sequence of Craig interpolants. The interpolation engine is complete for all logics supported by SMTInterpol.

For satisfiable formulas, SMTInterpol supports the optional SMTLIB command `get-value` and the non-standard command `get-model`. These commands can be used to inspect the model produced by SMTInterpol. For uninterpreted sorts, SMTInterpol generates a finite sort interpretation. The domain of this interpretation contains input terms instead of abstract values (see the SMTLIB standard).

References

[CHN12] Jürgen Christ, Jochen Hoenicke, and Alexander Nutz. SMTInterpol: An interpolating SMT solver. In *SPIN*, pages 248–254, 2012.

[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.

[EHP12] Evren Ermis, Jochen Hoenicke, and Andreas Podelski. Splitting via interpolants. In *VMCAI*, pages 186–201, 2012.

[HHP09] Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Refinement of trace abstraction. In *SAS'09*, number 5673 in LNCS, pages 69–85. Springer, 2009.

[HHP10] Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Nested interpolants. In *POPL'10*, pages 471–482. ACM, 2010.

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

[YM05] Greta Yorsh and Madanlal Musuvathi. A combination method for generating interpolants. In *CADE*, pages 353–368, 2005.