

opensmt

(version 0.2)

R. Bruttomesso E. Pek N. Sharygina A. Tsitovich

Verification and Security Group

Università della Svizzera Italiana (USI)

August 3, 2009

opensmt

opensmt is an open-source SMT-Solver

opensmt is an open-source SMT-Solver

Motivations (other than doing research):

- to promote the use of SMT-Solvers in combination with other verification tools

opensmt is an open-source SMT-Solver

Motivations (other than doing research):

- to promote the use of SMT-Solvers in combination with other verification tools
- to provide a level of detail that goes beyond the scientific publication

opensmt is an open-source SMT-Solver

Motivations (other than doing research):

- to promote the use of SMT-Solvers in combination with other verification tools
- to provide a level of detail that goes beyond the scientific publication
- to promote the development of SMT-Solvers by providing a simple infrastructure for the addition of new theories

- explanations for theory-propagations for a theory T are computed on demand (as in [NO05]) but by reusing the consistency check procedure for T

- explanations for theory-propagations for a theory T are computed on demand (as in [NO05]) but by reusing the consistency check procedure for T
 - theory-solver is just required to *detect* theory-propagations without providing/storing an explanation

- explanations for theory-propagations for a theory T are computed on demand (as in [NO05]) but by reusing the consistency check procedure for T
 - theory-solver is just required to *detect* theory-propagations without providing/storing an explanation
 - it allows theory-propagation for hard theories such as QF_BV

- explanations for theory-propagations for a theory T are computed on demand (as in [NO05]) but by reusing the consistency check procedure for T
 - theory-solver is just required to *detect* theory-propagations without providing/storing an explanation
 - it allows theory-propagation for hard theories such as QF_BV
 - consistency check run lazily often produces better explanations

- explanations for theory-propagations for a theory T are computed on demand (as in [NO05]) but by reusing the consistency check procedure for T
 - theory-solver is just required to *detect* theory-propagations without providing/storing an explanation
 - it allows theory-propagation for hard theories such as QF_BV
 - consistency check run lazily often produces better explanations
- preliminary preprocessor for arithmetic SMT formulæ [Bru09]

- explanations for theory-propagations for a theory T are computed on demand (as in [NO05]) but by reusing the consistency check procedure for T
 - theory-solver is just required to *detect* theory-propagations without providing/storing an explanation
 - it allows theory-propagation for hard theories such as QF_BV
 - consistency check run lazily often produces better explanations
- preliminary preprocessor for arithmetic SMT formulæ [Bru09]
- an efficient and complete decision procedure for bit-vector extraction and concatenation [BS09]

- QF_UF: based on SIMPLIFY's cc algorithm [DNS05]

- QF_UF: based on SIMPLIFY's cc algorithm [DNS05]
- QF_IDL: based on S.Cotton and O.Maler [CM06] graph algorithm

- QF_UF: based on SIMPLIFY's cc algorithm [DNS05]
- QF_IDL: based on S.Cotton and O.Maler [CM06] graph algorithm
- QF_RDL: converted into QF_IDL at preprocessing

- QF_UF: based on SIMPLIFY's cc algorithm [DNS05]
- QF_IDL: based on S.Cotton and O.Maler [CM06] graph algorithm
- QF_RDL: converted into QF_IDL at preprocessing
- QF_LRA: based on B.Dutertre L.de Moura Simplex [DdM06]

- QF_UF: based on SIMPLIFY's cc algorithm [DNS05]
- QF_IDL: based on S.Cotton and O.Maler [CM06] graph algorithm
- QF_RDL: converted into QF_IDL at preprocessing
- QF_LRA: based on B.Dutertre L.de Moura Simplex [DdM06]
- QF_BV: ultimately based on bit-blasting

Expected performance

- decent performance for QF_UF, QF_IDL, QF_RDL
- QF_LRA, QF_BV still need a lot of tuning

Expected performance

- decent performance for QF_UF, QF_IDL, QF_RDL
- QF_LRA, QF_BV still need a lot of tuning

We thank David Monniaux for contributing a class for fast arbitrary precision numbers

Expected performance

- decent performance for QF_UF, QF_IDL, QF_RDL
- QF_LRA, QF_BV still need a lot of tuning

We thank David Monniaux for contributing a class for fast arbitrary precision numbers

Project page	http://verify.inf.unisi.ch/opensmt
Code Repository	http://code.google.com/p/opensmt
Discussion Group	http://groups.google.com/group/opensmt

Thanks



R. Bruttomesso.

An Extension of the Davis-Putnam Procedure and its Application to Preprocessing in SMT.

In *SMT*, 2009.



R. Bruttomesso and N. Sharygina.

A Scalable Decision Procedure for Fixed-Width Bit-Vectors.

In *ICCAD*, 2009.

to appear.



S. Cotton and O. Maler.

Fast and Flexible Difference Constraint Propagation for DPLL(T).

In *SAT'06*, pages 170–183, 2006.



B. Dutertre and L. M. de Moura.

A Fast Linear-Arithmetic Solver for DPLL(T).

In *CAV'06*, pages 81–94, 2006.



D. Detlefs, G. Nelson, and J. B. Saxe.

Simplify: a theorem prover for program checking.

Journal of ACM, 52(3):365–473, 2005.



R. Nieuwenhuis and A. Oliveras.

DPLL(T) with Exhaustive Theory Propagation and Its Application to Difference Logic.

In *CAV'05*, pages 321–334, 2005.