STP in the SMTCOMP 2018

Vijay Ganesh, Trevor Hansen, Mate Soos,
Dan Liew, Ryan Govostes, Norbert Manthey

1 Introduction

STP[1] is an efficient open source solver for QF_BV and arrays without extensionality. STP recursively simplifies bit-vector constraints, solves linear bit-vector equations, and then eagerly encodes them to CNF for solving. Array axioms are added as needed during an abstraction-refinement phase.

2 Development history

STP was originally developed by Vijay Ganesh under the supervision of Professor David Dill. Later releases were developed by Trevor Hansen under the supervision of Peter Schachte and Harald Søndergaard. STP handles arbitrary precision integers using Steffen Beyer's library. STP encodes into CNF via the and-inverter graph package ABC of Alan Mishchenko [2]. STP supports different SAT backends, by now MiniSat [3], CryptoMiniSat [4] and Riss [5].

3 SMT Competition Specifics

Three versions of STP are submitted to SMTCOMP 2016, two sequential versions as well as a multicore version. The used commit hashes of each of the tools is given in the COMMITs file in the starexec package.

While STP allows the SAT backends in an incremental way, where the backend is linked to STP itself, in the competition mode STP generates a SAT formula from the SMT input and calls a stand-alone SAT solver. Based on the answer of this SAT solver, the wrapper script then outputs "sat" or "unsat". Given this setup, arbitrary SAT solvers could be used as backend.

The two sequential versions of STP use CryptoMiniSat and Riss, respectively, in the default configurations. The multicore version uses CryptoMiniSat and passes all available computing units to CryptoMiniSat.

4 Recent Developments to STP

Since SMT-COMP 2015 we have made progress at cleaning up the STP repository and adding tests in particular fuzz-tests. The STP repository contains a

large number of automated testing, building and deployment scripts. Further, it contains a large test suite and more inner self-checks through asserts. STP is being actively developed on GitHub.

5 Recent Developments to the Underlying SAT Solvers

The underlying SAT solver, CryptoMiniSat, has been significantly improved by including state-of-the-art techniques from past SAT competition winners and it has been tuned for the SMT use case. The way a SAT solver is used in a typical SAT competition scenario, i.e. when the solve() function is called only once, is very different than when used in a typical SMT scenario. Returning solutions quickly and close to one another is not a requirement to win a SAT competition yet is essential when the SAT solver is used from within an SMT solver. Hence CryptoMiniSat has been tuned to worked well in this sphere as well as in the regular, SAT competition, use case.

Riss is a sequential SAT solver with the formula simplifier Coprocessor [6]. Riss received only little minor modifications in the recent past. The biggest change is its introduction as STP backend, as well as an improved re-implementation of the learned clause minimization technique [7] that was used by the winner of the SAT competition 2017.

Acknowledgements

We would like to thank everyone who submitted bug reports, pull requests, and other useful data such as test cases.

References

1. Ganesh, V.: Decision Procedures for Bit-Vectors, Arrays and Integers. PhD thesis, Computer Science Department, Standford University, CA, United States (2007)
2. Brayton, R., Mishchenko, A.: Abc: An academic industrial-strength verification tool. In: Proceedings of the 22Nd International Conference on Computer Aided Verification. CAV'10, Berlin, Heidelberg, Springer-Verlag (2010) 24–40
3. Niklas Sörensson, N.E.: GitHub repository for MiniSat (may 2018) https://github.com/niklasso/minisat.
4. Soos, M.: GitHub repository for CryptoMiniSat (may 2018) https://github.com/msoos/cryptominisat.
5. Manthey, N.: GitHub repository for Riss (may 2018) https://github.com/niklasso/minisat.
6. Manthey, N.: Coprocessor 2.0 – a flexible cnf simplifier. In Cimatti, A., Sebastiani, R., eds.: Theory and Applications of Satisfiability Testing – SAT 2012, Berlin, Heidelberg, Springer Berlin Heidelberg (2012) 436–441
7. Mao Luo, Chu-Min Li, F.X.F.M.Z.L.: An effective learnt clause minimization approach for cdcl sat solvers. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. (2017) 703–711