STP in the SMT-COMP 2016

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

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.

Two versions of STP are submitted to SMTCOMP 2016. Both are nearly identical to revision 3785148da15919 of STP's publicly available source code repository [2]. The version with "-minisat-v16" uses MiniSat [3] revision 37dc6c67 set as default solver. The version with "-cryptominisat4-v16" contains CryptoMiniSat4 [4] revision cdba1990ac0cbf set as the default solver.

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 [5].

3 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.

4 Recent Developments to the Underlying SAT Solver

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.

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. Ganesh, V., Hansen, T., Liew, D., Govostes, R., Soos, M.: GitHub repository for STP (june 2015) https://github.com/stp/stp.
3. Niklas Sörensson, N.E.: GitHub repository for MiniSat (june 2015) https://github.com/niklasso/minisat.
4. Soos, M.: GitHub repository for CryptoMiniSat (june 2015) https://github.com/msoos/cryptominisat.
5. 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