Boolector at the SMT Competition 2019

Aina Niemetz[1], Mathias Preiner[1], and Armin Biere[2]

[1]Stanford University
[2]Johannes Kepler University, Linz, Austria

***Abstract*—This paper serves as system description for our SMT solver Boolector as entered into the SMT Competition 2019. We only list important differences from the version of Boolector that entered the SMT Competition 2018 [13]. For further and more detailed information, we refer to [9, 10, 15, 16], the Boolector website [2] or the source code on GitHub [1].**

Overview

Boolector [10, 15] is an Satisfiability Modulo Theories (SMT) solver for the theory of fixed-size bit-vectors with arrays and uninterpreted functions and natively handles non-recursive first-order lambda terms [16, 17]. It further supports quantified bit-vectors [16, 18] and two different local search strategies for quantifier-free bit-vector formulas that don't rely on but can be combined with bit-blasting [9, 11, 12, 14] in a sequential portfolio setting. Boolector supports the SAT solvers CaDiCaL [5], CryptoMiniSat [19], Lingeling [6], PicoSAT [4] and MiniSat [7] as back-end.

New Features / Improvements

The version of Boolector entering SMT-COMP 2019 is an improved and extended version of the version that entered the SMT competition in 2018. Notable extensions and improvements are listed below.

GMP as Back-End for Bit-Vector Implementation. Boolector now supports The Gnu Multiple Precision Arithmetic Library (GMP) [3] as back-end for bit-vector constants. This considerably improves performance of the local search engines for quantifier-free bit-vectors. Boolector provides two local search engines for solving quantifier-free bit-vectors. One is based on score-based stochastic local search (SLS) [14], which, starting from an initial assignment, iteratively moves towards a satisfying assignment by flipping bits of the input assignments. The second approach is based on propagation-based local search [11, 12], which, again starting from an initial assignment, iteratively moves towards a satisfying assignment by propagating target values from the outputs towards the inputs. Both heavily rely on computations involving bit-vector values, both when generating and updating assignment values, and the latter when down-propagating target values. Previously, Boolector used a custom bit-vector implementation that suffered from performance degradation with increasing bit-widths for bit-widths greater than 64. With GMP as the back-end, we greatly improve the performance of Boolector's local search engines, as the performance of bit-vector computations is now almost independent from the bit-width.

Incremental Solving with CaDiCaL. Since 2017, Boolector supports CaDiCaL [5] as SAT back-end for non-incremental quantifier-free bit-vector problems. Until this year, CaDiCaL did not support incremental solving and thus, could not be used for solving problems with arrays or uninterpreted functions as well as incremental problems. Hence, last year, Lingeling [6] was configured as the default SAT back-end for all divisions except the non-incremental quantifier-free bit-vectors divisions QF_BV in the Main track, for which CaDiCaL was the default SAT engine. This year, CaDiCaL has been extended to support incremental solving [8] and is now the default SAT engine for all tracks and divisions.

CryptoMiniSat Support. Boolector now additionally supports the SAT solver CrypotMiniSat [19] as SAT solver back-end, which can optionally be configured to run in a multi-threaded setting.

Incremental Solving Improvements. Preprocessing in Boolector is by default applied as aggressive as possible on the whole input formula for non-incremental and incremental problems. This had the consequence that for incremental problems, already bit-blasted parts of the formula were rewritten and bit-blasted again. In many cases, this swamped the SAT solver with redundant clauses, which resulted in a considerable performance degradation. With this year's competition version, Boolector now applies preprocessing more carefully to avoid rewriting already bit-blasted terms.

Configurations

The following three Boolector configurations are submitted to the SMT competition 2019.

1) Boolector

This configuration uses CaDiCaL (version sr2019, submitted to the SAT race 2019) as SAT back-end. It further combines bit-blasting with propagation-based local search techniques from [11, 12] in a sequential portfolio setting for pure bit-vector problems. The local search engine is limited to run for 10k propagations before falling back to bit-blasting.

Divisions: QF_ABV, QF_UFBV, QF_AUFBV, QF_BV, BV
Tracks: Single-Query, Industry-Challenge, Model-Validation

2) Boolector (incremental)

This configuration also uses CaDiCaL (version sr2019, submitted to the SAT race 2019) as SAT back-end, which now also supports incremental solving. It further enables the incremental solving optimization discussed above and is expected to perform considerable better than versions of Boolector submitted in previous years.

Divisions: QF_ABV, QF_UFBV, QF_AUFBV, QF_BV
Tracks: Incremental, Industry-Challenge

3) Poolector

Poolector is a wrapper tool written in Python ($\sim$100 LOC) that runs four different Boolector configurations in a parallel portfolio setting. The fastest configuration wins, i.e., as soon as one Boolector configuration reports *satisfiable* or *unsatisfiable*, all other instances are terminated.

Portfolio configuration (1) is identical to the Boolector configuration submitted to the non-incremental track (with CaDiCaL version sr2019). Two portfolio configurations are variations of (1) with the SAT solvers (2) Lingeling (version bcj), and (3) CryptoMiniSat (version 5.6.3). Portfolio configuration (4) employs the stochastic local search [14] techniques discussed above. Note that configuration (1) uses the propagation-based local search approach in a sequential portfolio setting, whereas (2) and (3) disable it.

Divisions: QF_ABV, QF_UFBV, QF_AUFBV, QF_BV, BV
Tracks: Single-Query, Industry-Challenge

COPYRIGHT

Boolector was initially developed by Armin Biere and Robert Brummayer from 2007–2009. From 2009-2012 it was maintained and extended by Armin Biere. Since 2012 it is maintained and extended by Armin Biere, Aina Niemetz, and Mathias Preiner.

LICENSE

Since May 2018, Boolector is available on GitHub [1] and licensed under the MIT license. For more details, refer to the actual license text, which is distributed with the source code.

REFERENCES

[1] Boolector source code. https://github.com/boolector/boolector.
[2] Boolector website. https://boolector.github.io.
[3] The GNU Multiple Precision Arithmetic Library. https://gmplib.org/.
[4] Armin Biere. Picosat essentials. *JSAT*, 4(2-4):75–97, 2008.
[5] Armin Biere. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2017. In Tomáš Balyo, Marijn Heule, and Matti Järvisalo, editors, *SAT Competition 2017 – Solver and Benchmark Descriptions*, volume B-2017-1 of *Department of Computer Science Series of Publications B*, pages 14–15. University of Helsinki, 2017.
[6] Armin Biere. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2018. In *SAT Competition 2018 – Solver and Benchmark Descriptions*, 2018. To appear.
[7] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.
[8] Katalin Fazekas, Armin Biere, and Christoph Scholl. Incremental Inprocessing in SAT Solving. In *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, Lecture Notes in Computer Science, to appear, 2019.
[9] Aina Niemetz. *Bit-Precise Reasoning Beyond Bit-Blasting*. PhD thesis, Informatik, Johannes Kepler University Linz, 2017.
[10] Aina Niemetz, Mathias Preiner, and Armin Biere. Boolector 2.0. *JSAT*, 9:53–58, 2015.
[11] Aina Niemetz, Mathias Preiner, and Armin Biere. Precise and complete propagation based local search for satisfiability modulo theories. In Swarat Chaudhuri and Azadeh Farzan, editors, *CAV (1)*, volume 9779 of *Lecture Notes in Computer Science*, pages 199–217. Springer, 2016.
[12] Aina Niemetz, Mathias Preiner, and Armin Biere. Propagation based local search for bit-precise reasoning. *Formal Methods in System Design*, 51(3):608–636, 2017.
[13] Aina Niemetz, Mathias Preiner, and Armin Biere. Boolector at the SMT competition 2018. Technical report, FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, 2018.
[14] Aina Niemetz, Mathias Preiner, Armin Biere, and Andreas Fröhlich. Improving local search for bit-vector logics in SMT with path propagation. In *Proceedings of the Fourth International Workshop on Design and Implementation of Formal Tools and Systems, Austin, TX, USA, September 26-27, 2015.*, pages 1–10, 2015.
[15] Aina Niemetz, Mathias Preiner, Clifford Wolf, and Armin Biere. BTOR2, BtorMC and Boolector 3.0. In *CAV*, Lecture Notes in Computer Science. Springer, 2018. To appear.
[16] Mathias Preiner. *Lambdas, Arrays and Quantifiers*. PhD thesis, Informatik, Johannes Kepler University Linz, 2017.
[17] Mathias Preiner, Aina Niemetz, and Armin Biere. Lemmas on demand for lambdas. In *DIFTS@FMCAD*, volume 1130 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.
[18] Mathias Preiner, Aina Niemetz, and Armin Biere. Counterexample-guided model synthesis. In *TACAS (1)*, volume 10205 of *Lecture Notes in Computer Science*, pages 264–280, 2017.
[19] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009.