Boolector at the SMT Competition 2014

Aina Niemetz, Mathias Preiner, and Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University, Linz, Austria

***Abstract*—This paper serves as solver description for our SMT solver Boolector entering the SMT Competition 2014 in three different configurations. We only list important differences to the earlier version of Boolector that participated in the SMT Competition 2012 [2]. For further information we refer to [2] or source code.**

Overview

This year's version of Boolector considerably differs from the version submitted to the SMT competition in 2012. First and foremost, we introduced a new lemmas on demand decision procecure DP_λ for lambdas as described in [6], which replaces the decision procedure for arrays introduced in [5]. Internally, we now represent all array operations and arrays as lambda terms and uninterpreted functions. Further, we introduced several improvements to a number of internal components of Boolector. Finally, we use an internal version of our SAT solver Lingeling, which is close to the version submitted to the SAT competition 2014, as back-end solver.

Note that DP_λ does not yet support extensionality on arrays. Thus, for benchmarks in the QF_ABV track, we use an older internal version of Boolector to handle extensionality on arrays in case that the input file is still extensional after rewriting.

In the following, we discuss the most notable improvements compared to the competition version of 2012.

Improvements

We made several improvements to the SMT2 parser and added support for SMT2 macros (*define-fun* commands).

Boolector now implements a new model generation algorithm to fix the performance drop of older versions in case model generation is enabled. The overhead of the new model generation algorithm is now negligible.

Similar to *cloning* in Lingeling [3][2], Boolector now also supports cloning and provides a *clone* function to generate an exactly identical copy of a Boolector instance.

We made a lot of improvements in terms of testing and debugging Boolector. Among others, we introduced API call tracing (recording/replaying API call sequences that trigger erroneous behaviour), model-based testing (c.f. [1]), and an internal validation of models for satisfiable instances after each SAT call. Due to these new testing and debugging techniques, we identified and fixed several bugs in Boolector since the previous competition in 2012, including several fixes in the rewriting engine and incremental API.

We further fixed the previously disabled *unconstrained* optimization, which is enabled by default in the current competition configurations.

Finally, we implemented two new optimization techniques based on don't care reasoning to speed-up our lemmas on demand procedure.

Configurations

This year, one configuration of Boolector ("Boolector") competes in the QF_BV track, whereas two different configurations "Boolector (dual propgation)" and "Boolector (justification)", which enable don't care reasoning to optimize the lemmas on demand procedure, compete in the QF_ABV track.

QF_BV configuration: Boolector

Since the current version of Boolector handles *define-fun* commands lazily (as described in [6]), we enabled *full beta reduction* to eagerly eliminate macros. Further, we disabled the *simpdelay* feature of Lingeling (which was previously enabled in version 1.5.118 of Boolector).

QF_ABV configuration 1: Boolector (dual propagation)

For this configuration we enabled our first optimization technique, which employs don't care reasoning and is based on *dual propagation*. We expect this configuration to be a bit slower than the *Boolector (justification)* configuration due to the overhead produced by the dual propagation solver.

QF_ABV configuration 2: Boolector (justification)

For this configuration we enabled the second don't care reasoning optimization technique, which is based on *justification*.

Copyright

Boolector has been originally developed by Robert Brummayer and Armin Biere at the FMV institute. Since 2009 it was maintained and extended by Armin Biere. Since 2012 it is maintained and extended by Armin Biere, Aina Niemetz, and Mathias Preiner.

License

For the competition version of Boolector we use the same license scheme as introduced last year for our SAT solver Lingeling [4]. It allows the use of the software for academic, research and evaluation purposes. It further prohibits the use of the software in other competitions or similar events without explicit written permission. Please refer to the actual license, which comes with the source code, for more details.

https://doi.org/10.35011/fmvtr.2014-1

Technical Report 14/1, June 2014, FMV Reports Series
Institute for Formal Models and Verification, Johannes Kepler University
Altenbergerstr. 69, 4040 Linz, Austria

REFERENCES

[1] Cyrille Artho, Armin Biere, and Martina Seidl. Model-Based Testing for Verification Back-Ends. In Margus Veanes and Luca Viganò, editors, *TAP*, volume 7942 of *Lecture Notes in Computer Science*, pages 39–55. Springer, 2013.

[2] A. Biere. Boolector Entering the SMT Competition 2012. Technical report, FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, 2012.

[3] A. Biere. Lingeling and Friends Entering the SAT Challenge 2012. In *Proc. of SAT Challenge 2012: Solver and Benchmark Descriptions*, volume B-2012-2 of *Department of Computer Science Series of Publications B, University of Helsinki*, pages 33–34, 2012.

[4] A. Biere. Lingeling, Plingeling and Treengeling entering the SAT Competition 2013. In A. Belov, M. Heule, and M. Järvisalo, editors, *Proc. of SAT Competition 2013*, volume B-2013-1 of *Department of Computer Science Series of Publications B, University of Helsinki*, pages 51–52, 2013.

[5] Robert Brummayer and Armin Biere. Lemmas on Demand for the Extensional Theory of Arrays. *JSAT*, 6(1-3):165–201, 2009.

[6] Mathias Preiner, Aina Niemetz, and Armin Biere. Lemmas on Demand for Lambdas. In Malay K. Ganai and Alper Sen, editors, *DIFTS@FMCAD*, volume 1130 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.