Boolector at the SMT Competition 2015

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 2015 in three different configurations. We only list important differences to the earlier version of Boolector that participated in the SMT Competition 2014 [2]. For further information we refer to [3] or source code.**

Overview

This year's version of Boolector incorporates several improvements and additions compared to the version that entered the SMT competition in 2014.

In the previous year, we had to rely on an older version of Boolector in case of array extensionality, which last year's version of Boolector did not support. For this year's version we added support for array extensionality to Boolector's lemmas on demand for lambdas engine. We further improved the lemmas on demand engine and added optimizations for lemma generation, which improve the overall performance of the solver. Finally, we use an internal version of our SAT solver Lingeling, which is close to the version submitted to the SAT race 2015, as back-end solver.

In the following, we discuss the most notable improvements compared to the competition version of 2014.

Improvements

We added support for push/pop commands to the SMT2 parser, which enables Boolector to enter the QF_BV division of the application track this year.

We further added support for extensional arrays and implemented two optimizations, which improve the lemma generation of the lemmas on demand engine for lambdas. The first optimization tries to extract array patterns that can be succinctly represented by means of lambda terms, which is expected to improve Boolector's performance particularly on instances from symbolic execution. The second optimization improves lemma generation by reducing the number of SAT calls to the underlying SAT solver. In previous versions of Boolector, every lemma generated entailed one SAT solver call. With this optimization we now generate all lemmas for the current candidate model prior to the next SAT solver call, which considerably improves the runtime of Boolector.

Boolector now officially supports uninterpreted functions with bit vector sorts, which enables Boolector to enter the QF_UFBV and QF_AUFBV divisions this year.

Configurations

This year, we submit three configurations of Boolector which are defined as follows.

Boolector (QF_BV)

Since the current version of Boolector handles *define-fun* commands lazily (as described in [4]), we enabled *full beta reduction* to eagerly eliminate macros. We further enabled *unconstrained optimization* for this configuration.

Boolector (QF_AUFBV)

This configuration of Boolector will enter the QF_ABV, QF_UFBV, and QF_AUFBV divisions of the main track. For this configuration we enabled *unconstrained optimization* and disabled *slice elimination* and *lazy bit blasting (lazy synthesize)*. The two optimizations to our lemmas on demand engine will affect this configuration, which is expected to perform much better compared to last year's version in terms of runtime and number of solved instances.

Boolector (QF_BV incremental)

This configuration is the incremental version of the QF_BV configuration with *skeleton preprocessing* disabled.

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 in 2013 for our SAT solver Lingeling [1]. 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.

References

[1] 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.

[2] Aina Niemetz, Mathias Preiner, and Armin Biere. Boolector at the SMT competition 2014. Technical report, FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, 2014.

[3] Aina Niemetz, Mathias Preiner, and Armin Biere. Boolector 2.0. *JSAT*, 9:53–58, 2015.

[4] 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.

Technical Report 15/1, June 2015, FMV Reports Series
Institute for Formal Models and Verification, Johannes Kepler University
Altenbergerstr. 69, 4040 Linz, Austria