Bitwuzla at the SMT-COMP 2022

Aina Niemetz
Stanford University

Mathias Preiner
Stanford University

***Abstract*—In this paper, we present Bitwuzla, our Satisfiability Modulo Theories (SMT) solver for the theories of bit-vectors, floating-points, arrays and uninterpreted functions and their combinations. We discuss selected features and provide details of its configuration and participation in the 2022 edition of the annual SMT competition.**

I. Introduction

Bitwuzla is a Satisfiability Modulo Theories (SMT) solver for the theories of fixed-size bit-vectors, floating-point arithmetic, arrays and uninterpreted functions and their combinations. Its name is derived from an Austrian dialect expression that can be translated as “someone who tinkers with bits”.

Bitwuzla implements a lemmas on demand procedure for logics with arrays and uninterpreted functions that generalizes the lemmas on demand for arrays approach from [11] to non-recursive first-order lambda terms [22, 23]. For quantifier-free bit-vectors, it supports the classic bit-blasting approach [15], different approaches to local search [16, 19, 20, 21], and a sequential combination of both. For floating-point logics, Bitwuzla includes SymFPU [9], a C++ library of bit-vector encodings of floating-point operations. Bitwuzla supports unsat core extraction for all supported quantifier-free logics. Further, for the first time, this year Bitwuzla provides experimental support for quantifiers for all theories and their combinations that are supported in the quantifier-free fragment.

This paper serves as system description for Bitwuzla as entered into the SMT competition 2022 [4]. Bitwuzla is licensed under the MIT license. Source code, releases and more information is available on the Bitwuzla website [1].

II. Features

A. Arrays and Uninterpreted Functions

Bitwuzla generalizes the lemmas on demand for extensional arrays approach [11] to non-recursive first-order lambda terms [22, 23], which enables compact representations for operations such as memset and memcpy [24] and constant arrays. It further supports dual propagation-based and justification-based optimization techniques for lemmas on demand, where the overhead for consistency checking is reduced by extracting partial candidate models via don’t care reasoning on full candidate models [18].

B. Quantifier-Free Bit-Vectors

Bitwuzla implements two orthogonal strategies for solving quantifier-free bit-vector constraints: the classic bit-blasting approach employed by most state-of-the-art bit-vector solvers, and local search. Since local search procedures are only able to determine satisfiability, Bitwuzla allows to combine local search with bit-blasting in a sequential portfolio setting, where the local search procedure is run until a certain limit is reached, before falling back to the bit-blasting engine.

Local Search for Quantifier-Free Bit-Vectors. Bitwuzla implements several local search strategies for quantifier-free bit-vectors. It supports the stochastic local search (SLS) approach presented in [13], an improved variant where SLS is augmented with a propagation-based strategy [21], and, most importantly, a novel generalization of the complete propagation-based local search procedure presented in [20] to ternary values [17]. This generalization addresses the main weakness of the propagation-based local search strategy [16, 20], its obliviousness to bits that can be simplified to constant values. The local search engines can now also be combined with the lemmas on demand engine and quantified bit-vectors.

Bit-Blasting. Bitwuzla implements bit-blasting in two phases. Initially, it generates an And-Inverter Graph (AIG) circuit representation of the simplified input formula and then applies AIG-level rewriting [10]. The rewritten AIG representation is then converted into Conjunctive Normal Form (CNF) and sent to one of following SAT back ends: MiniSat [12], PicoSAT [6], Lingeling [7], CaDiCaL [8], CryptoMiniSat [27], Kissat [2], or Gimsatul [3].

Bitwuzla uses CaDiCaL version rel-1.5.2 as default SAT back end. It further utilizes Lingeling for preprocessing the Boolean skeleton of the input formula.

C. Floating-Points

For the theory of floating-points, Bitwuzla implements an eager translation of the simplified input formula to the theory of bit-vectors. This approach is sometimes also referred to as *word-blasting*. To translate floating-point expressions to the word-level, Bitwuzla integrates SymFPU [9], a C++ library of bit-vector encodings of floating-point operations. SymFPU uses templated types for Booleans, (un)signed bit-vectors, rounding modes and floating-point formats, which allows to plug it in as a back end while utilizing solver-specific representations. It is also integrated in the SMT solver CVC4 [5].

D. Unsat Cores

Bitwuzla implements unsat core extraction via solving under assumptions [12]. When unsat core extraction is enabled, all assertions in the formula are assumed in the SAT back end. If given input formula is unsatisfiable, Bitwuzla returns all unsatisfiable (failed) assumptions as unsat core. Unsat Core extraction is not yet supported for quantified formulas.

E. Quantifiers

Bitwuzla now has experimental support for quantifiers for all logics that are also supported in the quantifier-free fragment. It implements variants of counterexample-guided quantifier instantiation [26] and model-based quantifier instantiation [14], as well as techniques from [25].

III. Configurations

Bitwuzla participates in the single query, incremental, unsat core, and model validation tracks in the following divisions:

- *Single Query Track (SQ):*
  ABV, ABVFP, ABVFPLRA, AUFBV, AUFBVFP, BV, BVFP, BVFPLRA, FP, FPLRA, QF_ABV, QF_ABVFP, QF_ABVFPLRA, QF_AUFBV, QF_AUFBVFP, QF_BV, QF_BVFP, QF_BVFPLRA, QF_FP, QF_FPLRA, QF_UFBV, QF_UFFP, UFBV, UFBVFP
- *Incremental Track (INC):*
  ABVFPLRA, BV, BVFP, BVFPLRA, QF_ABV, QF_ABVFP, QF_ABVFPLRA, QF_AUFBV, QF_BV, QF_BVFP, QF_BVFPLRA, QF_FP, QF_FPLRA, QF_UFBV, QF_UFFP
- *Unsat Core Track (UC):*
  QF_ABV, QF_ABVFP, QF_ABVFPLRA, QF_AUFBV, QF_AUFBVFP, QF_BV, QF_BVFP, QF_BVFPLRA, QF_FP, QF_FPLRA, QF_UFBV, QF_UFFP
- *Model Validation Track (MV):*
  QF_BV, QF_BVFP, QF_BVFPLRA, QF_FP, QF_FPLRA, QF_UFBV, QF_UFFP

For the QF_BV division in the SQ and MV track, Bitwuzla uses a sequential combination of bit-blasting and propagation-based local search with a limit of 10k propagation steps and 2M model update steps.

IV. License

Bitwuzla is licensed under the MIT license. For more details, refer to the actual license text, which is distributed with the source code.

References

[1] Bitwuzla website. https://bitwuzla.github.io, 2021.

[2] Kissat. https://github.com/arminbiere/kissat, 2021.

[3] Gimsatul. https://github.com/arminbiere/gimsatul, 2022.

[4] SMT-COMP 2022 website. https://www.smt-comp.org/2022, 2022.

[5] C. W. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli. CVC4. In *CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *LNCS*, pages 171–177. Springer, 2011.

[6] A. Biere. Picosat essentials. *JSAT*, 4(2-4):75–97, 2008.

[7] A. Biere. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2018. In *SAT Competition 2018 – Solver and Benchmark Descriptions*, 2018. To appear.

[8] A. Biere. CaDiCaL at the SAT Race 2019. In *Proc. of SAT Race 2019 – Solver and Benchmark Descriptions*, volume B-2019-1 of *Department of Computer Science Series of Publications B*, pages 8–9. University of Helsinki, 2019.

[9] M. Brain, F. Schanda, and Y. Sun. Building better bit-blasting for floating-point problems. In *TACAS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I*, volume 11427 of *LNCS*, pages 79–98. Springer, 2019.

[10] R. Brummayer and A. Biere. Local Two-Level And-Inverter Graph Minimization without Blowup. In *2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS'06), Mikulov, Czechia, October 2006, Proceedings*, 2006.

[11] R. Brummayer and A. Biere. Lemmas on demand for the extensional theory of arrays. *JSAT*, 6(1-3):165–201, 2009.

[12] N. Eén and N. Sörensson. An extensible sat-solver. In *SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.

[13] A. Fröhlich, A. Biere, C. M. Wintersteiger, and Y. Hamadi. Stochastic local search for satisfiability modulo theories. In *AAAI 2015, January 25-30, 2015, Austin, Texas, USA.*, pages 1136–1143. AAAI Press, 2015.

[14] Y. Ge and L. M. de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.

[15] D. Kroening and O. Strichman. *Decision Procedures - An Algorithmic Point of View*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2008.

[16] A. Niemetz. *Bit-Precise Reasoning Beyond Bit-Blasting*. PhD thesis, Informatik, Johannes Kepler University Linz, 2017.

[17] A. Niemetz and M. Preiner. Ternary propagation-based local search for more bit-precise reasoning. In *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*, pages 214–224. IEEE, 2020.

[18] A. Niemetz, M. Preiner, and A. Biere. Turbo-charging lemmas on demand with don't care reasoning. In *FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 179–186. IEEE, 2014.

[19] A. Niemetz, M. Preiner, and A. Biere. Precise and complete propagation based local search for satisfiability modulo theories. In *CAV (1)*, volume 9779 of *LNCS*, pages 199–217. Springer, 2016.

[20] A. Niemetz, M. Preiner, and A. Biere. Propagation based local search for bit-precise reasoning. *Formal Methods in System Design*, 51(3):608–636, 2017.

[21] A. Niemetz, M. Preiner, A. Biere, and A. Fröhlich. Improving local search for bit-vector logics in SMT with path propagation. In *DIFTS@FMCAD, Austin, TX, USA, September 26-27, 2015.*, pages 1–10, 2015.

[22] M. Preiner. *Lambdas, Arrays and Quantifiers*. PhD thesis, Informatik, Johannes Kepler University Linz, 2017.

[23] M. Preiner, A. Niemetz, and A. Biere. Lemmas on demand for lambdas. In *DIFTS@FMCAD 2013, Portland, OR, USA*, volume 1130 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[24] M. Preiner, A. Niemetz, and A. Biere. Better lemmas with lambda extraction. In *FMCAD 2015, Austin, Texas, USA, September 27-30, 2015*, pages 128–135. IEEE, 2015.

[25] M. Preiner, A. Niemetz, and A. Biere. Counterexample-guided model synthesis. In *TACAS (1)*, volume 10205 of *LCNS*, pages 264–280, 2017.

[26] A. Reynolds, M. Deters, V. Kuncak, C. Tinelli, and C. W. Barrett. Counterexample-guided quantifier instantiation for synthesis in SMT. In D. Kroening and C. S. Pasareanu, editors, *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, volume 9207 of *Lecture Notes in Computer Science*, pages 198–216. Springer, 2015.

[27] M. Soos, K. Nohl, and C. Castelluccia. Extending SAT solvers to cryptographic problems. In *SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009.