`lazybv2int` at the SMT Competition 2020

Yoni Zohar[1], Ahmed Irfan[1], Makai Mann[1], Andres Nötzli[1], Andrew Reynolds[2], and Clark Barrett[1]

[1] Stanford University
[2] The University of Iowa

Abstract

`lazybv2int` is a new prototype SMT-solver, that will participate in the incremental and non-incremental tracks of the QF_BV logic.

Overview. `lazybv2int` is a prototype SMT-solver for the theory of fixed-width bit-vectors and uninterpreted functions. This is the first year it will compete in SMT-COMP. It will participate in the incremental and non-incremental QF_BV tracks. The basic algorithm of the tool relies on a translation from bit-vectors to non-linear integer arithmetic with uninterpreted functions, followed by a CEGAR loop [7] that lazily instantiates bit-vector axioms over the translation. The idea of using integer reasoning for bit-vector solving is not new (see, e.g., [1,4]), however, it is worth revisiting due to recent improvements in solvers for non-linear integer arithmetic [5,8,10]. We expect this solver to perform better on benchmarks that involve arithmetic bit-vector constraints and large bit-widths because the encoding of arithmetic constraints is straightforward and independent of bit-width, as opposed to the encoding of bit-wise constraints which is less natural and hindered by larger bit-widths. The tool is open-source and is available at `https://github.com/yoni206/lazybv2int`.

Dependencies on Other Tools. To parse the input problem, `lazybv2int` employs MathSAT5's parser through an API [6]. To solve the translated arithmetic problem it uses CVC4 [3]. In some cases, MathSAT5 is called on an extension of the original bit-vector problem. The interface to both external solvers uses smt-switch [9]. According to the rules published by the organizers of SMT-COMP 2020, `lazybv2int` is a *wrapper tool* (see [2]) and not a *portfolio solver*, and thus is allowed to compete. This was confirmed in a private communication with the competition organizers.

Technical Details. `lazybv2int` works as follows. The input QF_BV formula φ is translated into a QF_UFNIA formula φ'. φ' is obtained from φ by eliminating bit-vector operators. For arithmetical operators, this is standard. The bit-wise operators other than bit-wise conjunction, left and right (logical) shift, and negation are first eliminated in a preprocessing stage using other bit-vector operators. Bit-wise negation has a standard arithmetical interpretation which is utilized. Bit-wise conjunction as well as shift operations are replaced by uninterpreted functions. The translated formula is solved using a CEGAR-loop that refines the current formula by adding lemmas. The procedure is complete because the lemma schemes that are used are complete. These include some basic properties of the abstracted operations (e.g., idempotence of bit-wise *and*), and in the worst case, they include a full expansion of the operation (using *ite* operations and summations). In each step of the loop, an under-approximation check is performed by MathSAT5's QF_BV solver: the current model μ' for φ' induces assumptions of the form $x = ToBV(\mu'[x'])$ for variables x of the original formula and their translations x'. These assumptions are added in order to solve φ. The assumptions do not include variables which appear in bit-wise *and* and bit-wise *shift* operators. Clearly, φ conjoined with the assumptions is an under-approximation of the original formula φ. So, if the result of the under-approximation is is SAT, then φ is satisfiable. In case the result of the additional check is unsatisfiable, then the disjunction of negated assumptions —only those assumptions that appear in the unsatisfiable core— is learned as an additional refinement lemma.

Conclusion. This is a prototype experimental tool that is aimed to serve as a playground for arithmetic-based techniques for bit-vector solving. Incorporating such techniques in a full-fledged solver is left for future work, and is planned for when these techniques are better understood and evaluated using this tool.

Acknowledgments. We would like to thank the CVC4 and MathSAT5 teams for allowing us to use their tools. In particular, we thank Alberto Griggio for clarifying the relevant aspects of the MathSAT5 license, and to Aina Niemetz and Mathias Preiner for helpful tips regarding benchmarking and evaluating bit-vector formulas. We also thank the competition organizers, Haniel Barbosa, Jochen Hoenicke, and Antti Hyvarinen for clarifying the status of this tool for the competition and willingly accepting it as a new participant.

References

[1] Peter Backeman, Philipp Rümmer, and Aleksandar Zeljic. Bit-vector interpolation and quantifier elimination by lazy reduction. In *FMCAD*, pages 1–10. IEEE, 2018.

[2] Haniel Barbosa, Jochen Hoenicke, and Antti Hyvarinen. 15th international satisfiability modulo theories competition (smt-comp 2020): Rules and procedures.

[3] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 171–177. Springer, 2011.

[4] Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Anders Franzén, Ziyad Hanna, Zurab Khasidashvili, Amit Palti, and Roberto Sebastiani. Encoding RTL constructs for mathsat: a preliminary report. *Electron. Notes Theor. Comput. Sci.*, 144(2):3–14, 2006.

[5] Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. Experimenting on solving nonlinear integer arithmetic with incremental linearization. In *SAT*, volume 10929 of *Lecture Notes in Computer Science*, pages 383–398. Springer, 2018.

[6] Alessandro Cimatti, Alberto Griggio, Bastiaan Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT Solver. In Nir Piterman and Scott Smolka, editors, *Proceedings of TACAS*, volume 7795 of *LNCS*. Springer, 2013.

[7] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2000.

[8] Dejan Jovanovic. Solving nonlinear integer arithmetic with MCSAT. In *VMCAI*, volume 10145 of *Lecture Notes in Computer Science*, pages 330–346. Springer, 2017.

[9] Makai Mann, Amalee Wilson, Cesare Tinelli, and Clark Barrett. smt-switch. https://github.com/makaimann/smt-switch/.

[10] Andrew Reynolds, Cesare Tinelli, Dejan Jovanovic, and Clark W. Barrett. Designing theory solvers with extensions. In *FroCoS*, volume 10483 of *Lecture Notes in Computer Science*, pages 22–40. Springer, 2017.