

Solving Constraints over Bit-Vectors with SAT-based Model Checking

Extended Abstract

Yakir Vizel¹, Alexander Nadel², and Sharad Malik¹

¹ Princeton University, USA

² Intel Development Center, Haifa, Israel

Introduction

We present **BVMC**, a novel decision procedure for constraints over fixed-width bit-vectors, which is based on propositional model checking (MC).

Nowadays, Satisfiability Modulo Theory (SMT) [6] solvers for the quantifier-free fixed-width bit-vector (QF_BV) logic are widely used, especially when bit-precise reasoning is required. One subset of QF_BV, which is particularly important in formal verification of software (SW), is quantifier-free linear arithmetic over *integers modulo* 2^N (LIA_N). This paper presents an efficient decision procedure, **BVMC**, suitable for solving LIA_N .

Formal verification of SW is one of the main forces driving SMT research. SW verification usually involves reasoning about arithmetic constraints, and in particular, linear arithmetic constraints over integers modulo 2^N for some $N \in \mathbb{N}$. This is due to the fact that SW uses a finite representation for integers. More precisely, arithmetic operations over integers are interpreted over the ring $\mathbb{Z}/2^N\mathbb{Z}$ (“machine arithmetic”) rather than over the ring \mathbb{Z} . As a result, efficient bit-precise reasoning is highly desired.

In order to capture the semantics of linear arithmetic over $\mathbb{Z}/2^N\mathbb{Z}$ (LIA_N), SMT solvers for the theory of fixed-width bit-vectors (BV solvers) are often used. BV solvers, however, are not efficient when the bit-vectors are wide. Namely, when the value of 2^N is large (e.g. $N = 128$), solving linear arithmetic constraints over $\mathbb{Z}/2^N\mathbb{Z}$ becomes intractable for BV solvers. This inefficiency is mainly due to the way BV solvers are implemented: in most cases, the formula is reduced to a propositional formula using *bit-blasting*. Therefore, as N increases, so is the complexity of the resulting SAT formula. One way to overcome this inefficiency is by applying a LIA solver. Unlike BV solvers, LIA solvers reason about linear arithmetic over \mathbb{Z} . While LIA solvers are more efficient than that of BV solvers for this task, they are less precise. This imprecision comes from the different semantics of LIA and LIA_N . Namely, arithmetic operations over \mathbb{Z} cannot result in an “overflow” (i.e. wrap-around). In the context of SW verification, this may lead to unsound results. Hence, an efficient LIA_N solver, which this paper presents, should be extremely useful for SW verification.

Our Approach

Our novel decision procedure **BVMC** is based on a reduction of the input formula to a safety verification problem. Namely, a formula φ in LIA_N is transformed to a transition system T such that the satisfiability of φ corresponds to whether T is **SAFE** or **UNSAFE**. The key to our reduction lies in treating bit-vectors as unbounded streams of bits over time. More precisely, for each input bit-vector, the least significant bit (LSB) corresponds to time 0 in the corresponding stream, and the k -th bit corresponds to the bit received at time k . The structure of T captures the constraints between the bit-vector variables that appear in φ . To determine if T is **SAFE** or **UNSAFE**, **BVMC** uses SAT-based model checking (SATMC) [10].

One possible way to reason about T is by using Bounded Model Checking (BMC) [2], an efficient SATMC algorithm that can show T is **UNSAFE**. Considering our reduction, if BMC finds a counterexample of length N in T (T is **UNSAFE**), then φ is satisfiable over $\mathbb{Z}/2^N\mathbb{Z}$. If no counterexample of length N exists in T , then φ is unsatisfiable over $\mathbb{Z}/2^N\mathbb{Z}$. This can be used as a decision procedure for LIA_N . However, the performance of such an approach is usually not better than that of BV solvers [7]. BMC can either find a counterexample of length N , or prove that counterexample of length N does not exist. In that sense, in the context of **BVMC**, it can only reason about LIA_N for a given N . In fact, this approach is somewhat “equivalent” to how modern eager BV solvers are implemented. Consequently, this approach is, in general, not superior to solving the bit-blasted formula.

Unlike BMC, modern SATMC algorithms [8,3,9] use *generalization* in order to show that no counterexample, of any length, exists, and by that they can prove a transition system is **SAFE**. **BVMC** takes advantage of this generalization mechanism. In case **BVMC** finds φ to be unsatisfiable over $\mathbb{Z}/2^k\mathbb{Z}$, SATMC’s generalization mechanism is applied to show φ is unsatisfiable over $\mathbb{Z}/2^N\mathbb{Z}$ for every $N > k$. For the case a counterexample of length k is found, we have implemented an efficient procedure in **BVMC** that tries to extend the counterexample to some target N (where $N > k$) and by that show φ is satisfiable over $\mathbb{Z}/2^N\mathbb{Z}$. When such a counterexample cannot be extended, **BVMC** blocks it and continues the search until either a new counterexample is found (possibly longer) or until unsatisfiability is established.

Evaluation

As discussed above, our goal in designing **BVMC** is to support **QF_BV**. Currently, we implemented a prototype which supports all bit-wise operators, as well as all operators required to support LIA_N . For evaluation, we transformed the **QF_LIA** subset of the **SMT-COMP’16** benchmark to **QF_BV** using fixed-width bit-vectors of sizes 32, 64, and 128. We then compared **BVMC** to Boolector³ [4], and Z3⁴ [5].

³ Version 2.4.1

⁴ Version 4.5.1

BVMC solved *the most satisfiable* instances out of the three, even for a width as low as 32. Moreover, it was able to solve many more instances, that were not solved by neither Boolector nor Z3.

Table 1. Number of solved instances for LIA_N . *Total* stands for the total number of test cases in that benchmark. The difference is due to the fact that not all LIA test cases can be represented in LIA_N for certain values of N .

Benchmark	Total	Status	BVMC	Boolector	Z3	Virtual Best
LIA ₅ (32bit)	2647	SAT	1475	1257	1373	1539
		UNSAT	784	988	881	995
LIA ₆ (64bit)	2784	SAT	1630	1340	1448	1781
		UNSAT	680	1017	889	1023
LIA ₇ (128bit)	2742	SAT	1565	1233	1347	1734
		UNSAT	637	1013	861	1020

Table 1 shows the number of solved instances for the different experiments of LIA_N . As can be seen from the table, BVMC has a big advantage specifically on satisfiable instances, for all values of N . BVMC constructs a satisfying assignment, incrementally, starting from the LSB. We believe this is the main reason for the performance advantage of BVMC over the other methods. Figures 1-3 further emphasize the performance advantage of BVMC on satisfiable instances. Moreover, we can see the performance advantage of BVMC grows as the width of bit-vectors grows.

It is important to note that the approaches are complementary as many test cases are solved by BVMC and not by Boolector, and vice-versa. Overall, BVMC solves 205 test cases not solved neither by Boolector nor Z3 for $N = 5$. For $N = 6$ and for $N = 7$, BVMC solves 288 and 331 test cases that are not solvable by the other solvers. When compared to Boolector, for $N = 5$, BVMC solves 370 test cases not solved by Boolector, and Boolector solves 331 test cases not solved by BVMC. For $N = 6$ and $N = 7$, BVMC solves 427 and 496 test cases not solved by Boolector, while Boolector solves 482 and 501 test cases not solved by BVMC. In the case of Z3, for $N = 5, 6, 7$, BVMC solves 324, 329 and 397 cases not solved by Z3, while Z3 solves 265, 337, 349 cases not solved by BVMC.

Related and Future Work

A closely related line of work appears in [1,7], where a reduction from a fragment of BV, restricted to addition, shift by one and equality, to propositional model checking has been introduced (as a by-product of studying the complexity of bit-vector logic). The proposed method has been implemented and shown to outperform traditional SMT solvers on crafted BV benchmarks, restricted to the aforementioned BV fragment. Unlike the transformation applied by BVMC, the modeling suggested in [1] only supports fixed-width bit-vectors, making SATMC algorithms inefficient. As a result, BDD-based model checking algorithms were

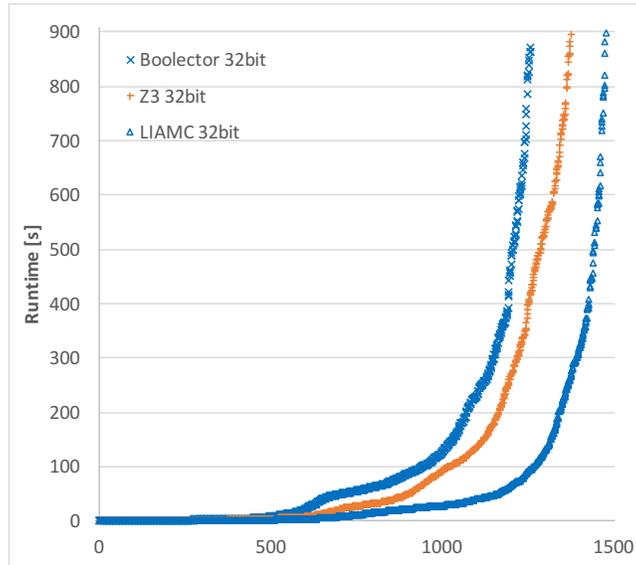


Fig. 1. $\mathbb{Z}/2^5\mathbb{Z}$: Trend for satisfiable instances (32 bit).

found to be the most efficient experimentally [1]. BVMC shows how SATMC can be applied efficiently even for the subset supported by [1]⁵. In addition, our approach can handle a more extensive set of operators, which makes it applicable to arbitrary formulas in LIA_N .

Our future work in this direction includes the following:

- Extend our method to fully support QF_BV, and
- Implement dedicated SATMC algorithms that can efficiently solve transition systems originating from LIA_N and from QF_BV.

References

1. A. B. Andreas Fröhlich, Gergely Kovásznai. Efficiently solving bit-vector problems using model checkers. In *11th International Workshop on Satisfiability Modulo Theories*, 2013.
2. A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 58:117–148, 2003.
3. A. R. Bradley. SAT-Based Model Checking without Unrolling. In *VMCAI*, pages 70–87, 2011.
4. R. Brummayer and A. Biere. Boolector: An efficient SMT solver for bit-vectors and arrays. In *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009*, pages 174–177, 2009.

⁵ Note that throughout our experiments, the transition systems include more than thousands of state elements, making BDD-based MC intractable.

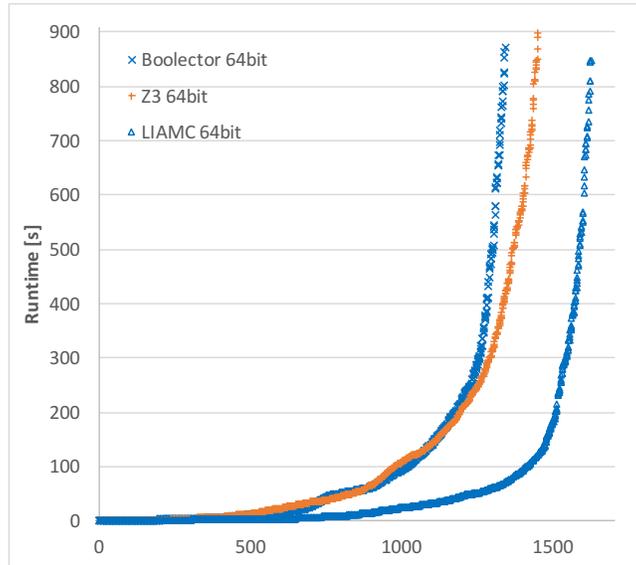


Fig. 2. $\mathbb{Z}/2^6\mathbb{Z}$: Trend for satisfiable instances (64 bit).

5. L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008*, pages 337–340, 2008.
6. L. M. de Moura and N. Bjørner. Satisfiability modulo theories: introduction and applications. *Commun. ACM*, 54(9):69–77, 2011.
7. G. Kovásznai, A. Fröhlich, and A. Biere. Complexity of fixed-size bit-vector logics. *Theory Comput. Syst.*, 59(2):323–376, 2016.
8. K. L. McMillan. Interpolation and SAT-Based Model Checking. In *CAV*, pages 1–13, 2003.
9. Y. Vizel and A. Gurfinkel. Interpolating property directed reachability. In *CAV*, pages 260–276, 2014.
10. Y. Vizel, G. Weissenbacher, and S. Malik. Boolean satisfiability solvers and their applications in model checking. *Proceedings of the IEEE*, 103(11):2021–2035, 2015.

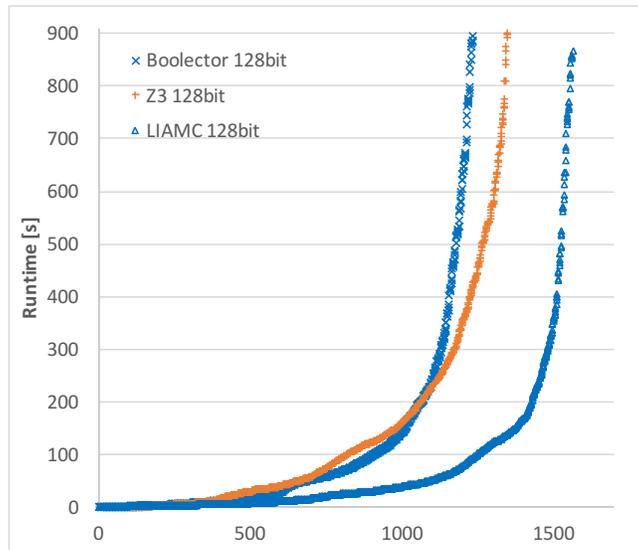


Fig. 3. $\mathbb{Z}/2^7\mathbb{Z}$: Trend for satisfiable instances (128 bit).