Verifying Bit-vector Invertibility Conditions in Coq

Arjun Viswanathan[†], Burak Ekici[‡], Yoni Zohar[§], Clark Barrett[§] and Cesare Tinelli[†]

[†]*Department of Computer Science*, *University of Iowa*, Iowa City, USA
[‡]*Department of Computer Science*, *University of Innsbruck*, Innsbruck, Austria
[§]*Department of Computer Science*, *Stanford University*, Stanford, USA

***Abstract*—This report describes ongoing work of verifying invertibility conditions for the theory of fixed-width bit-vectors, that are used to solve quantified bit-vector formulas in the Satisfiability Modulo Theories (SMT) solver CVC4. This work complements the verification efforts of previous work by proving a subset of these invertibility conditions in the Coq proof assistant.**

***Index Terms*—bit-vectors, Coq, invertibility conditions**

I. Introduction

The theory of bit-vectors can be used to model problems in various applications such as hardware circuit analysis [1], bounded model checking [2], and symbolic execution [3]. Most of these applications require reasoning about quantified formulas over bit-vectors. Few SMT solvers can deal with this fragment, one of which is CVC4. CVC4 uses a quantifier instantiation technique to reason about quantified formulas. As presented in [4], a quantifier instantiation method using invertibility conditions benefits from a smaller number of instantiations, resulting in a more efficient solver. An invertibility condition for a literal specifies the conditions under which that literal is solvable. For instance, $x + s = t$ is unconditionally solvable for x, where x, s, and t are bit-vectors of the same sort and $+$ is bit-vector addition. A general solution or *inverse* for x is $t - s$, and since x is always invertible, the invertibility condition for the literal $x + s = t$ is simply $\top$, or true. This is represented by the equivalence $x + s = t \iff \top$. A more interesting case is that of bit-wise conjunction ($\&$), which is represented by the equivalence $x \ \& \ s = t \iff t \ \& \ s = t$. Reference [4] found 160 of these invertibility equivalences for a representative set of operators and predicates from the bit-vector theory of the SMT-LIB 2 standard (in the previous two examples, the operators are $+$ and $\&$, and the predicate is equality, or $=$) and verified them using SMT solvers, for bit-widths up to 65. The correctness of the quantifier instantiation technique that uses these equivalences, however, assumes the equivalences to be valid in the theory of bit-vectors for any bit-width n. The challenge of verifying these equivalences for bit-vectors of arbitrary bit-widths comes from SMT-solvers' inability to express bit-vectors over arbitrary bit-widths. Reference [5] encoded these problems over the theory of non-linear arithmetic with uninterpreted functions. The corresponding verification attempt was still unable to prove over a quarter of the equivalences. We complement these works by proving a representative subset of invertibility equivalences in the Coq proof assistant. We extended a bit-vector library in Coq [6], and proved 18 invertibility equivalences.

II. Invertibility Equivalence Proofs

We assume the usual terminology of many-sorted first-order logic with equality (see, e.g., [7] for more details), and consider the SMT-LIB 2 [8] [1] theory of bit-vectors, Σ_{BV}. In general, we write $\psi[x_0, ..., x_n]$ to denote a formula whose free variables are from the set $\{x_0, ..., x_n\}$. An *invertibility condition* for a variable x in a Σ_{BV}-literal $\ell[x, s, t]$ is a formula $IC[s, t]$ such that $\forall s.\, \forall t.\, IC[s, t] \Leftrightarrow \exists x.\ell[x, s, t]$ is valid in the theory of fixed-width bit-vectors. In what follows, we denote by Σ_0 the sub-signature of Σ_{BV} containing the predicate symbols $<_u$, $>_u$, $\leq_u$, $\geq_u$ (corresponding to strong and weak unsigned comparisons between bit-vectors, respectively), as well as the function symbols $+$ (bit-vector addition), $\&$, $|$, $\sim$ (bit-wise conjunction, disjunction and negation), $-$ (2's complement unary negation), and $\ll$, $\gg$ and $\gg_a$ (left shift, and logical and arithmetical right shifts). We also denote by Σ_1 the extension of Σ_0 with the predicate symbols $<_s$, $>_s$, $\leq_s$, and $\geq_s$ (corresponding to strong and weak signed comparisons between bit-vectors, respectively), as well as the function symbols $-$, $\cdot$, $\div$, mod (corresponding to subtraction, multiplication, division and remainder), and $\circ$ (concatenation).

Reference [4] defines invertibility conditions for a representative set of literals ℓ having a single occurrence of x, that involve the bit-vector operators of Σ_1. The soundness of the technique proposed in that work relies on the correctness of the invertibility conditions. The signature Σ_{BV} of the SMT-LIB 2 theory of fixed-width bit-vectors includes a unique sort for each positive integer n, which we denote by $\sigma_{[n]}$. Thus, every literal $\ell[x, s, t]$ and its corresponding invertibility condition $IC[s, t]$ induce the *invertibility equivalence*

$$\forall s : \sigma_{[n]}.\, \forall t : \sigma_{[n]}.\, IC[s, t] \Leftrightarrow \exists x : \sigma_{[n]}.\ell[x, s, t] \ .$$

which one needs to prove valid for *all* $n > 0$. Reference [4] was able to prove these equivalences for values of n from 1 to 65.

Reference [5] proves over half of the 160 equivalences for arbitrary bit-widths using SMT-solvers by encoding the equivalences into theories which the solvers could deal with.

We focused mainly on proving those equivalences that [5] failed to prove. We chose Σ_0 as a representative subset of

[1] The SMT-LIB 2 theory is defined at http://www.smt-lib.org/theories.shtml

$\ell[x]$	$=$	$\neq$	$<_u$	$>_u$	$\leq_u$	$\geq_u$
$-x \bowtie t$	✓✓	✓	✓	✓	✓	✓
$\sim x \bowtie t$	✓✓	✓	✓	✓	✓	✓
$x \,\&\, s \bowtie t$	✓	✓	✓	✓	✓	✓
$x \mid s \bowtie t$	✓	✓	✓	✓	✓	✓
$x \ll s \bowtie t$	✓	✓	✓	✓	✓	✓
$s \ll x \bowtie t$	✓✓	✓	✓	✓	✓	✓
$x \gg s \bowtie t$	✓✓	✓	✓	×	✓	✓
$s \gg x \bowtie t$	✓✓	✓	✓	✓	✓	✓
$x \gg_a s \bowtie t$	✓	✓	✓	✓	✓	✓
$s \gg_a x \bowtie t$	✓✓	✓	✓	✓	✓	✓
$x + s \bowtie t$	✓✓	✓	✓	✓	✓	✓

TABLE I
RESULTS OF PROVING INVERTIBILITY EQUIVALENCES FOR LITERALS OVER Σ_0.

Σ_1, and proved 18 of the equivalences, 11 of which were unproved by [5]. Our results are summarized in Table I. There, ✓ means that the invertibility equivalence was successfully verified in Coq but not in [5], while ✓ means the opposite; ✓✓ means that the invertibility equivalence was verified using both approaches, and × that it was verified with neither. Meta-symbol $\bowtie$ ranges over the operators in the table's top row.

III. Library and Proof Details

We used a library originally developed for the SMTCoq tool [6], a Coq plugin that dispatches proof goals to SMT-solvers. Although there are other libraries such as Coq's bit-vector library [9], the Bedrock Bit Vectors Library [10], and the SSRBit library [11], we choose this library because it was developed for SMT-LIB 2 bit-vectors, and as a result, had many relevant lemmas for our proofs already available. We extended this library with the $\gg_a$ operator, and the predicates $\leq_u$ and $\geq_u$. In using the library for our proofs, we also enriched it with various additional lemmas.

The bit-vector library represents bit-vectors as lists of Booleans, dependent on a natural number, representing their size. This dependent bit-vector type is constructed from an underlying non-dependent representation. This separation makes it easier to expand the library - one can represent operators and lemmas in the non-dependent representation, before using the library's functor to transform it into the required dependent type [2].

We were also assisted by CoqHammer, a tool that solves proof goals by learning from previous proofs, and by outsourcing proofs of certain subgoals to external automated provers.

IV. Conclusion

We present here our work in an ongoing project of using invertibility conditions to facilitate a quantifier instantiation technique that is deployed in the CVC4 SMT solver. Our contribution was to prove a subset of these invertibility conditions in the Coq proof assistant for a general bit-width.

V. Future Work

This work was accepted and is to be presented at PxTP 2019 [12] as an extended abstract. In the future, we plan to expand this library with operators such as division and modulus, and prove a larger set of invertibility conditions. The next goal would be to extend SMTCoq so it can certify proofs over the newly added operators of the Coq library. Another potential direction of future work is to recreate the library in the Lean Theorem Prover [13].

[2]Both the library and the proofs of invertibility equivalences can be found at https://github.com/ekiciburak/bitvector/tree/pxtp2019

References

[1] A. Gupta and A. L. Fisher, "Representation and symbolic manipulation of linearly inductive boolean functions," in *Proceedings of the 1993 IEEE/ACM International Conference on Computer-aided Design*, ser. ICCAD '93. Los Alamitos, CA, USA: IEEE Computer Society Press, 1993, pp. 192–199. [Online]. Available: http://dl.acm.org.stanford.idm.oclc.org/citation.cfm?id=259794.259827

[2] A. Armando, J. Mantovani, and L. Platania, "Bounded model checking of software using smt solvers instead of sat solvers," *International Journal on Software Tools for Technology Transfer*, vol. 11, pp. 69–83, 2008.

[3] C. Cadar, V. Ganesh, P. M. Pawlowski, D. L. Dill, and D. R. Engler, "Exe: Automatically generating inputs of death," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 322–335. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180445

[4] A. Niemetz, M. Preiner, A. Reynolds, C. Barrett, and C. Tinelli, "Solving quantified bit-vectors using invertibility conditions," in *Proceedings of the 30th International Conference on Computer Aided Verification, CAV 2018, Oxford, UK*, ser. Lecture Notes in Computer Science, H. Chockler and G. Weissenbacher, Eds., vol. 10982. Springer, 2018, pp. 236–255. [Online]. Available: https://doi.org/10.1007/978-3-319-96142-2_16

[5] A. Niemetz, M. Preiner, A. R. Y. Zohar, C. Barrett, and C. Tinelli, "Towards bit-width-independent proofs in SMT solvers," in *Proceedings of the 27th International Conference on Automated Deduction, CADE-27, Natal, Brazil*, ser. Lecture Notes in Computer Science, P. Fontaine, Ed. Springer, 2019, (to appear).

[6] B. Ekici, A. Mebsout, C. Tinelli, C. Keller, G. Katz, A. Reynolds, and C. Barrett, "Smtcoq: A plug-in for integrating SMT solvers into coq," in *Proceedings of 29th International Conference on Computer Aided Verification (CAV 2017)*, ser. Lecture Notes in Computer Science, vol. 10427. Springer, 2017, pp. 126–133.

[7] H. B. Enderton, "Chapter two - first-order logic," in *A Mathematical Introduction to Logic (Second Edition)*, second edition ed., H. B. Enderton, Ed. Boston: Academic Press, 2001, pp. 67 – 181.

[8] C. Barrett, A. Stump, and C. Tinelli, "The SMT-LIB Standard: Version 2.0," in *Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, UK)*, A. Gupta and D. Kroening, Eds., 2010.

[9] J. Duprat, "Library coq.bool.bvector." [Online]. Available: https://coq.inria.fr/library/Coq.Bool.Bvector.html

[10] T. Chajed, H. Chen, A. Chlipala, J. Choi, A. Erbsen, J. Gross, S. Gruetter, F. Kaashoek, A. Konradi, G. Malecha, D. Oe, M. Vijayaraghavan, N. Zeldovich, and D. Ziegler, "Bedrock bit vectors library." [Online]. Available: https://github.com/mit-plv/bbv

[11] A. Blot, P.-E. Dagand, , and J. Lawall, "Bit sequences and bit sets library." [Online]. Available: https://github.com/pedagand/ssrbit

[12] B. Ekici, A. Viswanathan, Y. Zohar, C. Barrett, and C. Tinelli, "Verifying bit-vector invertibility conditions in coq," in *Proof eXchange for Theorem Proving, PxTP@CADE 2019, Natal, Brazil*, ser. EPTCS, 2019, (to appear).

[13] L. M. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer, "The lean theorem prover (system description)." in *CADE*, ser. Lecture Notes in Computer Science, A. P. Felty and A. Middeldorp, Eds., vol. 9195. Springer, 2015, pp. 378–388. [Online]. Available: http://dblp.uni-trier.de/db/conf/cade/cade2015.html#MouraKADR15