Verifying Bit-vector Invertibility Conditions in Coq

Arjun Viswanathan
University of Iowa
Iowa City, USA
`arjun-viswanathan@uiowa.edu`

Burak Ekici
University of Innsbruck
Innsbruck, Austria
`burak.ekici@uibk.ac.at`

This work complements [1] and [2] - [1] describes a set of equivalences over bit-vector operations, that are used in the CVC4 SMT solver to implement a quantifier instantiation technique for bit-vector constraints. The LHS of these equivalences consist of quantified bit-vector formulas, and the RHS represents a quantifier- free bit-vector formula called an invertibility condition (ICs). These ICs are generated by a combination of manual efforts and an SMT-based synthesis solver, for fixed-size bit-vectors upto a certain length. [2] verifies a majority of these equivalences for bit-vectors of arbitrary bit-width using SMT solvers. It does this by translating the formulas over bit-vectors of non-fixed bit-width into formulas in a logic supported by SMT solvers that includes non-linear integer arithmetic, uninterpreted functions, and universal quantification. Our work verifies a subset of the equivalences that weren't verified in [2] using the Coq proof assistant. We use a Coq library that represents SMTLib-style bit-vectors as lists of Booleans to prove these equivalences.

1 Introduction

Reasoning logically about bit-vectors is useful for many applications in hardware and software. While SMT solvers are able to do this for bit-vectors of fixed width, they struggle to do the same for bit-vectors of non-fixed width. Proof assistants such as Coq are useful tools to produce guarantees of the latter kind.

We use a Coq bit-vector library that represents SMTLib-style bit-vectors as Boolean lists to certify a set of equivalences over bit-vectors that contain formulas called invertibility conditions (ICs). ICs are introduced in [1] to help SMT solvers find smart instantiations of quantified bit-vector formulas that improve the efficiency of solving. We will call these equivalences invertibility equivalences (IEs) for convenience. An IE is an equivalence, that transforms a quantified formula over a particular bit-vector operation into a quantifier-free formula that specifies the conditions under which the bit-vector operation is invertible.

A simple example of an inverse is subtraction for addition. $\forall x, x+s=t$ is always invertible for x. The *inverse* of x is $t-s$, and the *condition* under which this formula is invertible (or the IC of this formula) is simply $\top$.

[1] provides 162 such IEs for 9 bit-vector operations over 10 predicates (the predicate used in the example above is equality). These IEs are generated for bit-vectors of fixed-width and verified for bit widths up to 65. The challenge in verifying them for arbitrary bit-widths comes from the inability of SMT solvers to quantify over the length of a bit-vector. [2] addresses this challenge by translating these IEs for arbitrary-length bit-vectors into quantified formulas over the thories of non-linear arithmetic, and uninterpreted functions - a combination that SMT-solvers can reason about. They still failed to verify over a quarter of the IEs for non-fixed bit-widths using this technique.

In our work, we approach the task of verifying these IEs by proving them in the Coq proof assistant. We use a Coq library developed to support SMTLib-style bit-vectors. We extend the already rich library to support some additional operations in the IEs and prove a representative subset of the IEs that weren't

Submitted to:
PxTP 2019

verifiable using the translation in [2]. This work gives us confidence that proof assistants can support theorem provers in verification tasks by addressing some of the challenges that they face.

2 Invertibility Equivalences (IEs) and Invertibility Conditions (ICs)

We describe here in some formal detail what IEs and ICs are. We give a high-level understanding of how they are used to deal with quantified bit-vector formulas in the CVC4 SMT solver.

We present the IEs from [1], talk about the ones that are verified in [2], and mention the ones that we were able to verify.

3 Verification of IEs Using SMT Solvers

We present a high-level summary of how [2] translate these IEs to be able to verify them using SMT solvers and talk about some of the challenges they faced.

4 Library Description

We describe the Coq library that we use here and credit its original authors and application. We talk about alternative libraries that could've been considered and why this one is more suitable. We specify the bit-vector operators and predicates from the IEs that this library supports, describe our efforts to extend its support and mention some challenges. We describe our efforts to organize this library and make it accessible as one that supports SMTLib-style bit-vectors.

5 Low-Level Proof Details

We discuss some of the techniques and tactics used for proving IEs. We specify when CoqHammer was used and why it was needed. We talk about which proofs we found more challenging, and why.

6 Conclusion and Future Work

We conclude our work here. We talk about the directions in which we can extend this work. We discuss how we can extend the library to be able to represent more of the IEs. We also discuss which of the rest of the unverified IEs may be easier to prove.

Bitvectors - which are arrays of bits of 0s and 1s - are represented in the library as Boolean lists.

The library

References

[1] Aina Niemetz, Mathias Preiner, Andrew Reynolds, Clark Barrett & Cesare Tinelli (2018): *Solving Quantified Bit-Vectors Using Invertibility Conditions*. In: *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, pp. 236–255, doi:10.1007/978-3-319-96142-2_16. Available at https://doi.org/10.1007/978-3-319-96142-2_16.

[2] Aina Niemetz, Mathias Preiner, Andrew Reynolds Yoni Zohar, Clark Barrett & Cesare Tinelli: *Towards Bit-Width-Independent Proofs in SMT Solvers*.