Using Term Rewriting to Solve Bit-Vector Arithmetic Problems (Poster Presentation)

Iago Abal[1], Alcino Cunha[1], Joe Hurd[2], and Jorge Sousa Pinto[1]

[1] HASLab / INESC TEC & Universidade do Minho, Braga, Portugal
[2] Galois, Inc., Portland, OR, USA

Among many theories supported by SMT solvers, the theory of *finite-precision bit-vector arithmetic* is one of the most useful, for both hardware and software systems verification. This theory is also particularly useful for some specific domains such as cryptography, in which algorithms are naturally expressed in terms of bit-vectors. Cryptol is an example of a domain-specific language (DSL) and toolset for cryptography developed by Galois, Inc.; providing an SMT backend that relies on bit-vector decision procedures to certify the correctness of cryptographic specifications [3]. Most of these decision procedures use *bit-blasting* to reduce a bit-vector problem into pure propositional SAT. Unfortunately bit-blasting does not scale very well, especially in the presence of operators like multiplication or division. For example, the equality $x_{[n]}^2 - 1_{[n]} = (x_{[n]} + 1_{[n]}) \times (x_{[n]} - 1_{[n]})$ is a simple consequence of distributivity and associativity laws; but even for small values of n the bit-level representation of this formula is so huge that it is intractable by current SAT solvers. The main reason for this is the loss of high-level algebraic structure present in the original decision problem. The point here is that one can exploit algebraic properties concerning the domain of bit-vectors to rewrite this problem into an equisatisfiable, but computationally less hard, problem. For instance, the above equality can be proved *valid* as follows (subscripts are omitted for clarity): $x^2 - 1 = (x + 1) \times (x - 1) \quad \equiv \{\text{distributivity} \times 3;\ \text{associativity}\} \quad x^2 - 1 = x^2 + x - x - 1 \quad \equiv \{\text{inverse; right identity}\} \quad x^2 - 1 = x^2 - 1 \quad \equiv \{\text{reflexivity}\} \quad \mathit{true}$. Modern SMT solvers already include a simplification phase that performs some rewriting on the input problem prior to bit-blasting [4]. Nevertheless, SMT solvers have to deal with a wide range of application domains, and hence the set of rewrite rules employed for simplification inevitably excludes many rules that are useful for some particular domains but may be inconvenient for others.

The present work was motivated by the difficulties reported by the Galois Cryptol team in achieving automatic equivalence checking for public-key cryptography (PKC). PKC is particularly hard because it involves multiplication and modular exponentiation on long bit-vectors. Hence, the bit-level representation of any PKC algorithm is usually so huge that such equivalence problems are too hard for current SAT solvers, unless a significant amount of rewriting is performed before bit-blasting. SMT solvers employing high-level rewriting-based techniques have been shown to be promising, but they are still insufficiently powerful to handle hard problems, such as those resulting from PKC.

This problem may be addressed by combining custom rewrite patterns, somehow encapsulating domain-specific proof strategies, with standard bit-vector decision procedures. Our first attempt consisted in extending SMT specifications with algebraic properties provided in the form of *quantified formulas*, expecting the SMT solver to use them as rewrite rules. Unfortunately, we have found that most of the times SMT solvers do not use these rules effectively, and even become quite unpredictable in the presence of universal quantifiers. After this failed attempt, we prototyped a rewriting system in Maude [1] that focuses on simplifying PKC equivalence problems. Employing a set of 200 handcrafted rewrite rules and a very simple rewriting strategy enabled us to achieve quite promising results. For instance, this system proved the correctness of a 16-bit peasant multiplier and SHA-1 implementations in a few seconds, while the 3.2 version of Z3 [2] times out (16 hours) for the peasant case and quickly runs out of memory (2 GB) solving the SHA-1 one. Using this rewriting system as a preprocessing step for Z3 we also achieved good speedups for some equivalence problems, such as a speedup of 2 for an 8-bit modular exponentiation algorithm.

Even though there is still considerable work to be done in order to reach a reasonable degree of automation for PKC equivalence checking, the above results show the potential of the term-rewriting approach. In the same way that proof assistants allow defining custom tactics to encapsulate specific proof techniques, our intention is to encode those proof tactics as rewrite patterns in the context of SMT solving. This allows simplifications that drastically reduce the size of the input problem before bit-blasting, leading to better overall performance. Ideally, SMT solvers should allow easy customization of their solving strategies with such rules —we are aware of some recent work in this direction. It is worth noting that we are not relying on complex combinations of rewriting strategies, which would make our approach more fragile and less scalable. Finally, Maude turned out to be a good platform for experimentation, but it significantly restricts the strategies that we could employ and presents some limitations with respect to achieving perfect subterm sharing. Thus we are presently working on a framework to specify custom rewriting-based simplifications for fixed-size bit-vector arithmetic, that should allow us to overtake the above limitations.

References

1. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: The Maude 2.0 System. In: Nieuwenhuis, R. (ed.) Rewriting Techniques and Applications. pp. 76–87. No. 2706 in LNCS, Springer-Verlag (June 2003)
2. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer-Verlag, Berlin, Heidelberg (2008)
3. Erkök, L., Matthews, J.: Pragmatic equivalence and safety checking in Cryptol. In: Proceedings of the 3rd workshop on Programming Languages meets Program Verification. pp. 73–82. PLPV '09, ACM, New York, NY, USA (2008)
4. Franzen, A.: Efficient Solving of the Satisfiability Modulo Bit-Vectors Problem and Some Extensions to SMT. Ph.D. thesis, University of Trento (March 2010)