

Leveraging SMT: Using SMT Solvers to Improve Verification; Using Verification to Improve SMT Solvers

Clark Barrett

Christopher L. Conway

October 30, 2010

Abstract

Solvers for the Satisfiability Modulo Theories (SMT) problem are making rapid progress. However, many verification tools aren't making use of the full power of modern SMT solvers. We believe that the verification community could be benefiting more from the work of the SMT community; at the same time, the SMT community could benefit from a more active and engaged verification user community.

1 Introduction

The last five years have shown rapid progress in solvers for the Satisfiability Modulo Theories (SMT) problem. Since the initiation of the Satisfiability Modulo Theories Competition (SMT-COMP) in 2005 [2], we have seen a rapid increase in the number of tools available and in the performance of existing tools. Users may now choose from more than a dozen tools that are free for research use, including a handful of free, open source tools. Support for theories like arrays, bit vectors and even nonlinear arithmetic is increasingly common. Complete decision procedures have been discovered for increasingly large fragments of first-order logic [1, 9]. New algorithms for theory combination have enlarged the set of theories that can be combined modularly [10] and improved performance [8, 11]. Finally, standardization around the SMT-LIB input format [4] allows users the convenience of experimenting with different tools.

We believe that verification tools could be benefiting more from the full power of modern SMT solvers. A core goal of the NYU ACSys group's research program has been to apply the CVC3 SMT solver to hard verification problems. We've had success applying SMT to translation validation [3]. More recently, the flexibility of CVC3 has allowed us to support high-level and low-level programming languages; static analysis and deductive verification in the Cascade verification framework [5]. Our recent work on datatype verification [6] would not have been possible without the combination of the theories of bit vectors, arrays, uninterpreted functions, and datatypes offered by CVC3.

Just as important as the benefit available to verification tools from SMT solvers is the potential benefit to SMT solvers from the active engagement of the verification community. New benchmarks tell us what kinds of practical problems are easy and what kinds of practical problems demand more research. Examining a wide variety of problems where current techniques fail may lead us to discover new decision procedures and combination techniques.

2 Challenges

What are the challenges facing us in increasing the impact of SMT on verification?

The first major challenge is the proprietary culture that has taken hold in the SMT community. Though there are beginning to be competitive SMT solvers (such as CVC3 [7] and OpenSMT [12]), that are free and open source, the majority of tools are closed source. This is in contrast to the SAT community where source

code releases are the norm - a fact that has arguably been largely responsible for the continued success of that field. The emphasis on closed-source tools has slowed the rapid advance of the state of the art, as competing tool developers are forced to discover and rediscover the same techniques independently. Although relations between tool developers are for the most part collegial and open in an informal sense, new implementation ideas spread slowly by word of mouth rather than with the immediacy of frequent source code releases.

The second major challenge is the large gap between SMT implementers and users, which makes SMT solvers more difficult to use than they otherwise might be. In particular, we note the following difficulties faced by users of SMT solvers:

- The SMT-LIB input language is defined with a level of mathematical rigor that many users will find intimidating and confusing. It is particularly rigid in the definition of fragmentary logics, like difference logic and linear arithmetic.
- Quantifier performance has not improved nearly as much as users would like or might expect, leading to frustration when seemingly simple queries return incomplete results. Quantifier heuristics can be sensitive to the form of the input in ways which are frankly confusing to users.
- The user typically has to manage the connection between the elements of the problem domain (e.g., program variables) and the terms of any generated SMT formula. There is a particular problem when generating counterexamples: the term produced for an assignment in an SMT model needs to be translated back into the problem domain, removing artifacts of the SMT encoding; the term will often appear in the SMT solver's internal normal form (e.g., " $-1 + 1 * x - 2 * y = 0$ ") instead of a more intuitive external representation (e.g., " $x = 2y + 1$ ").

None of these challenges are insurmountable—they are mostly technical and can be overcome by the care and attention of dedicated implementers.

References

- [1] Henny B. Sipma Aaron R. Bradley, Zohar Manna. What's decidable about arrays? In *Verification, Model-Checking, and Abstract-Interpretation*, 2006.
- [2] C. Barrett, L. de Moura, and A. Stump. SMT-COMP: Satisfiability Modulo Theories Competition. In K. Etessami and S. Rajamani, editors, *Computer Aided Verification*, pages 20–23, 2005.
- [3] Clark Barrett, Yi Fang, Ben Goldberg, Ying Hu, Amir Pnueli, and Lenore Zuck. TVOC: A translation validator for optimizing compilers. In *Computer Aided Verification*, pages 291–295, 2005.
- [4] Clark Barrett, Aaron Stump, and Cesare Tinelli. The SMT-LIB Standard: Version 2.0. In *SMT Workshop*, 2010.
- [5] Cascade. <http://cs.nyu.edu/acsys/cascade/>.
- [6] Christopher L. Conway and Clark Barrett. Verifying low-level implementations of high-level datatypes. In *Computer Aided Verification*, pages 306–320, 2010.
- [7] CVC3. <http://cs.nyu.edu/acsys/cvc3/>.
- [8] Leonardo de Moura and Nikolaj Bjørner. Model-based theory combination. In *SMT Workshop*, 2007.
- [9] Yeting Ge and Leonardo de Moura. Complete instantiation for quantified formulas in Satisfiability Modulo Theories. In *Computer-Aided Verification*, pages 306–320, 2009.
- [10] Dejan Jovanović and Clark Barrett. Polite theories revisited. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 402–416, 2010.

- [11] Dejan Jovanović and Clark Barrett. Sharing is caring. In *SMT Workshop*, 2010.
- [12] OpenSMT. <http://verify.inf.unisi.ch/opensmt/>.