

SMT-COMP 2006 entry: CVC3

1 Introduction

CVC3 is a tool for determining the satisfiability of a first order formula modulo a first order theory (or a combination of such theories). CVC3 is the latest in the Cooperating Validity Checker family of tools, building on its predecessors, CVC [9] and CVC Lite [1].

CVC3 is implemented in C++. It follows the Extended Abstract DPLL Modulo Theories architecture [2] which combines a propositional SAT solver with a theory solver. The algorithm used for the theory solver is a Nelson-Oppen [8] style combination of theories and the implementation is based closely on an algorithm described in Clark Barrett's Ph.D. thesis [4].

2 DPLL Engine

CVC3 provides a thin abstract API for connecting to a generic DPLL engine. Currently, the API is instantiated with a basic DPLL engine using old version of zchaff [7]. We expect to extend this capability so that there is a choice of several SAT solvers as DPLL engines for CVC3.

3 Built-in Theories

Empty. The simplest supported theory is the empty theory over an expanded signature. This is also often called the theory of equality with uninterpreted functions. This theory is decided using a basic congruence closure algorithm.

Arrays. CVC3 includes a theory of abstract arrays with two operations, *read* and *write* which can be used to read from a location in an abstract array or to create a new array by *writing* a new value to a location in an existing array. The implementation is based on an algorithm described in Jeremy Levitt's PhD thesis [6].

Datatypes. A new feature in CVC3 is a rich theory of datatypes [3]. The theory supports mutually recursive datatypes with multiple constructors and selectors. CVC3 formulas can also include simple aggregate datatypes like records and tuples. These are handled with a simple decision procedure for a set of operations used to create, read from, and write to these datatypes (much like the array operations).

Arithmetic. As with its predecessors, CVC3 can decide the theory of linear arithmetic over any combination of the reals and the integers. In addition, CVC3 has some limited ability to reason about non-linear arithmetic.

Bit-Vectors. CVC3 can decide a fairly comprehensive set of bit-vector operations. The decision procedure is based on a combination of rewriting and reduction to propositional satisfiability.

4 Quantifiers

CVC3 includes a greatly enhanced module for reasoning about quantifiers. Quantified formulas are either skolemized (if they are existential) or treated as abstract propositional variables (if they are universal). Universally quantified formulas are then instantiated using an algorithm based on the matching algorithms pioneered by the theorem prover Simplify [5]. CVC3's quantifier instantiation techniques are much better than those used in CVC Lite, and can outperform Simplify on many examples.

5 Proofs

For maximum confidence, CVC3 has the capability to produce a proof when a formula is valid. The proof can be checked by an independent proof-checker.

6 SMT-COMP

Problem Divisions. CVC3 will be competing in all divisions.

Expected Performance. CVC3 is typically about twice as fast as CVC Lite on problems from last year's SMT-COMP divisions. Thus, we expect its performance in those divisions to be better than CVC Lite was, but not as good as the competition. Our main efforts have been on improving quantifiers and bit-vectors. We expect CVC3 to be competitive in the QF_UFBV32, AUFLIA, and AUFLIRA divisions.

7 People

CVC3 has been developed primarily by Clark Barrett at New York University, building on the existing implementation of CVC Lite which was done as a joint project of Stanford University and New York University (see last year's system description for a list of contributing CVC Lite developers). The new quantifier module was developed by Yeting Ge, a graduate student at New York University. Recently, Cesare Tinelli and his group at the University of Iowa joined the CVC3 development effort. Two Iowa students, Alexander Fuchs and George Hagan have made contributions.

References

- [1] C. Barrett and S. Berezin. CVC Lite: A new implementation of the cooperating validity checker. In R. Alur and D. A. Peled, editors, *Proceedings of the 16th International Conference on Computer Aided Verification (CAV '04)*, volume 3114 of *Lecture Notes in Computer Science*, pages 515–518. Springer-Verlag, July 2004. Boston, Massachusetts.
- [2] C. Barrett, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Splitting on demand in sat modulo theories. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR '06)*, Nov. 2006. Phnom Penh, Cambodia, to appear.
- [3] C. Barrett, I. Shikanian, and C. Tinelli. An abstract decision procedure for satisfiability in the theory of recursive data types. In *Proceedings of the 3rd*

Workshop on Pragmatics of Decision Procedures in Automated Reasoning (PDPAR '06), Aug. 2006. Seattle, Washington, USA, to appear.

- [4] C. W. Barrett. *Checking Validity of Quantifier-Free Formulas in Combinations of First-Order Theories*. PhD thesis, Stanford University, Jan. 2003. Stanford, California.
- [5] D. L. Detlefs, G. Nelson, and J. B. Saxe. Simplify: A theorem-prover for program checking. Technical Report HPL-2003-148, HP System Research Center, 2003.
- [6] J. Levitt. *Formal Verification Techniques for Digital Systems*. PhD thesis, Stanford University, 1999.
- [7] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, June 2001.
- [8] G. Nelson and D. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–57, 1979.
- [9] A. Stump, C. W. Barrett, and D. L. Dill. CVC: A cooperating validity checker. In E. Brinksma and K. G. Larsen, editors, *Proceedings of the 14th International Conference on Computer Aided Verification (CAV '02)*, volume 2404 of *Lecture Notes in Computer Science*, pages 500–504. Springer-Verlag, July 2002. Copenhagen, Denmark.