System Description: Z3 0.1

Nikolaj Bjørner and Leonardo de Moura

Microsoft Research, One Microsoft Way, Redmond, WA, 98074, USA
{nbjorner, leonardo}@microsoft.com

1 Introduction

Z3 is a new theorem prover being developed at Microsoft Research. Z3 supports linear real and integer arithmetic, fixed-size bit-vectors, extensional arrays, uninterpreted functions, and quantifiers. Z3 is still under development, but it has already been integrated with Spec#/Boogie [2, 5]. We are currently integrating Z3 with Pex [9], SAGE [8], and SLAM [1]. It can read problems in SMT-LIB and Simplify [6] formats. Z3 is available at: http://research.microsoft.com/projects/z3.

2 Algorithms and Implementation

Z3 integrates a modern DPLL-based SAT solver, a *core theory solver* that handles equalities and uninterpreted functions, *satellite solvers* (for arithmetic, arrays, etc.), and an *E-matching abstract machine* (for quantifiers). Z3 is implemented in C++.

Quantifiers Z3 uses a well known approach for quantifier reasoning that works over an E-graph to instantiate quantified variables. Z3 uses new algorithms that identify matches on E-graphs incrementally and efficiently. Experimental results show substantial performance improvements over existing state-of-the-art SMT solvers [3].

Quantifier instantiation has a side-effect of producing new clauses containing new atoms into the search space. Z3 garbage collects clauses, together with their atoms and terms, that were useless in closing branches.

Theory combination Traditional methods for combining theory solvers rely on capabilities of the solvers to produce all implied equalities or a pre-processing step that introduces additional literals into the search space. Z3 uses a new theory combination method that incrementally reconciles models maintained by each theory [4].

Don't cares DPLL(T) based solvers assign a boolean value to potentially all atoms appearing in a goal. In practice, several of these atoms are *don't cares.* Z3 ignores these atoms for expensive inference rules and theories, such as, quantifier instantiation.

Theories Z3 uses a linear arithmetic solver based on the algorithm used in Yices [7]. The array theory uses lazy instantiation of array axioms. The fixed-sized bit-vectors theory applies bit-blasting to all bit-vector operations, but equality.

3 Problem Divisions

Z3 will participate in all divisions of SMT-COMP'07.
Seed Number: 1

References

1. T. Ball and S. K. Rajamani. The SLAM project: debugging system software via static analysis. *SIGPLAN Not.*, 37(1):1–3, 2002.
2. M. Barnett, K. R. M. Leino, and W. Schulte. The Spec# programming system: An overview. In *CASSIS 2004*, LNCS 3362, pages 49–69. Springer, 2005.
3. L. de Moura and N. Bjørner. Efficient E-matching for SMT Solvers. In *CADE'07*. Springer-Verlag, 2007.
4. L. de Moura and N. Bjørner. Model-based Theory Combination. In *SMT'07*, 2007.
5. R. DeLine and K. R. M. Leino. BoogiePL: A typed procedural language for checking object-oriented programs. Technical Report 2005-70, Microsoft Research, 2005.
6. D. Detlefs, G. Nelson, and J. B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.
7. B. Dutertre and L. de Moura. A Fast Linear-Arithmetic Solver for DPLL(T). In *CAV'06*, LNCS 4144, pages 81–94. Springer-Verlag, 2006.
8. D. Molnar P. Godefroid, M. Levin. Automated Whitebox Fuzz Testing. Technical Report 2007-58, Microsoft Research, 2007.
9. N. Tillmann and W. Schulte. Unit tests reloaded: Parameterized unit testing with symbolic execution. *IEEE software*, 23:38–47, 2006. See `http://research.microsoft.com/pex`.