

Sateen: Sat Enumeration Engine for SMT-COMP'07 ^{*}

Hyondeuk Kim, HoonSang Jin, and Fabio Somenzi

University of Colorado at Boulder

{Hyondeuk.Kim,HoonSang.Jin,Fabio}@Colorado.EDU

1 Description

Sateen is a satisfiability solver that combines a propositional reasoning engine with theory-specific procedures. It uses the lazy approach that relies on incremental refinements of a propositional abstraction of the given formula during the enumeration of its solutions.

Sateen deals with *Integer Difference Logic* (IDL), in which arithmetic atomic formulae constrain the difference between the values of pairs of integer variables. It is sufficient to rewrite each *equality* constraint (of the form $x - y = n$) as the conjunction of two inequalities. However, if an equality constraint is negated, then the conjunction turns into a disjunction, which requires case splitting in the enumeration of the propositional solutions. In contrast, Sateen is based on the approach of [3], which does not decompose equalities and their negations; rather, it converts the problem of checking satisfiability of a conjunction of arithmetic atomic formulae into a set of propositional satisfiability checks—whose cardinality is bounded by the number of strongly connected components (SCC) of a suitable constraint graph.

Sateen has been upgraded to the new version of the propositional solver, CirCUs [2]. Sateen generates solutions to the Boolean variables and the numerical variables in the original formula if the problem is satisfiable. Sateen incrementally checks infeasibility of IDL constraints. If the partial assignments from propositional solver are satisfiable, Sateen finds implied atomic variables through theory propagation. In contrast to other solvers, Sateen performs equality theory propagation that searches for implied atomic variables from zero-slack SCCs. To check the feasibility of IDL constraints that contain disequality constraints, it generates a maximal clique with bounds for each variable. In the final step, finite instantiation is performed for complete assignments. UNSAT Core is returned as a proof of infeasibility if the encoded SAT instance is unsatisfiable.

2 Problem Divisions

Quantified Free Integer Difference Logic.

3 Programming Language(s)

Sateen is written in C. An ANSI C compiler and GNU make are required to build it.

^{*} Supported by SRC contract 2006-TJ-1365.

4 Software Architecture of the System

Sateen consists of a new version of propositional solver and a theory solver that decides whether a conjunction of literals in the theory (e.g., a conjunction of integer difference constraints) is satisfiable. The theory solver produces a model if the constraints are satisfiable and a proof of unsatisfiability otherwise. The proof is a set of unsatisfiable constraints—typically a subset of the constraints passed to it.

The propositional SAT solver part is characterized by:

- All SAT Enumeration [2]
- UNSAT Core generation

The current theory solver part is based on layering [1]. It relies in particular on five layers:

- Negative cycle detection
- Zero-slack strongly connected components check
- Clique detection
- Theory propagation
- Finite instantiation

References

- [1] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani. An incremental and layered procedure for the satisfiability of linear arithmetic logic. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'05)*, pages 317–333, Edinburgh, UK, Apr. 2005. LNCS 3440.
- [2] H. Jin, H. Han, and F. Somenzi. Efficient conflict analysis for finding all satisfying assignments of a Boolean circuit. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'05)*, pages 287–300, Apr. 2005. LNCS 3440.
- [3] H. Kim and F. Somenzi. Finite instantiations for integer difference logic. In *Formal Methods in Computer Aided Design (FMCAD'06)*, pages 31–38, San Jose, CA, Nov. 2006.