The Barcelogic SMT Solver (Tool Paper)*

Miquel Bofill†, Robert Nieuwenhuis*, Albert Oliveras*,
Enric Rodríguez-Carbonell* and Albert Rubio*

† Universitat de Girona.
* Technical University of Catalonia, Barcelona.

Abstract. This is the first system description of the Barcelogic SMT solver, which implements all techniques that our group has been developing over the last four years as well as state-of-the-art features developed by other research groups. We pay special attention to the theory solvers and to functionalities that are not common in SMT solvers.

1 Introduction

The importance of propositional SAT solvers in verification applications has been largely shown in the last few years. However, propositional logic is not very expressive and by encoding practical problems into SAT, sometimes important structural information is lost or substantial blow-up in the formula size is caused. A successful alternative to SAT is to consider more expressive logics that still have efficient solvers. For example, for reasoning about timed automata, it is very useful to consider *Difference Logic (DL)*, where atoms are of the form $x - y \leq k$, being x and y integer or real variables, and k a numeric constant; in hardware verification, when one wants to abstract away the concrete behavior of certain components, it is useful to consider the logic of *Equality with Uninterpreted Functions (EUF)*, where atoms are equalities between first-order terms; similarly, for software verification, one may need to reason about concrete data structures such as arrays, lists or queues. Hence, it becomes very natural to consider satisfiability modulo these concrete theories and deal with formulas that contain thousands of clauses like:

$$p \quad \vee \quad x - read(A, i) \leq y \quad \vee \quad f(write(A, j, i + 2)) = read(A, j) + 1$$

In general, the problem known as *Satisfiability Modulo Theories (SMT)* amounts to deciding the satisfiability of a typically ground formula modulo a background theory T. To achieve this goal, similarly to what is done in most state-of-the-art SMT solvers, Barcelogic combines a Boolean DPLL(X) engine, very similar in nature to a SAT solver, responsible for enumerating propositional models of the formula, with a theory solver $Solver_T$, responsible for checking that

* All authors partially supported by the by the project LogicTools-2 (TIN2007-68093-C02-01) funded by the Spanish Ministry of Science and Technology.

these models remain consistent with the theory T. The integration of DPLL(X) with the concrete solver $Solver_T$, produces what we call a DPLL(T) system.

In order to produce our Barcelogic SMT solver we have worked, together with some external colleagues, on developing and refining the DPLL(T) approach [NOT06,BNOT06], designing efficient solvers for distinct theories that comply with all the requirements of a $Solver_T$ [NO07,NO05] and in extending SMT solvers to give support for other uses rather than just checking the satisfiability of a formula [LNO06,NO06]. All these ideas have been incorporated in our Barcelogic system, hence producing a very efficient and robust SMT solver, as one can observe from its performance in previous editions of the SMT competition (see `http://www.smtcomp.org`).

2 System Description

In this section we discuss the main components in the Barcelogic SMT solver: the parser and preprocessor, the Boolean engine DPLL(X) and all theory solvers that allow Barcelogic to deal with EUF, DL, *Linear Arithmetic (LA)* and combinations of these theories. Finally, we sketch some additional capabilities.

2.1 Parser and Preprocessor

Given an input formula in SMT-LIB format [TR05], Barcelogic's parsing module performs two tasks. One of them is to detect which is the most efficient theory solver that is able to process all input atoms; the second task is to massage the formula so that it can be fed to the DPLL(X) engine: convert it to CNF, abstract all theory atoms taking into account T-equivalent atoms, apply Ackermann's reduction if EUF and an arithmetic theory are involved, and split the arithmetic equality constraints into conjunctions of inequalities, to name but a few.

2.2 The Boolean Engine

The DPLL(X) engine currently used is a slight modification of our Barcelogic SAT solver that took part in the 2006 SAT-Race (for detailed results, see `http://fmv.jku.at/sat-race-2006`). It is competitive with state-of-the-art SAT solvers and is indeed very similar to them, as it borrows most of the ideas present in zChaff [MMZ$^+$01] and MiniSAT [ES04]. Additionally, it implements several adaptive heuristics to find the right frequency of calls to $Solver_T$ (both for consistency checks and theory propagations) and has been extended to accommodate the splitting-on-demand technique presented in [BNOT06] where one needs to add both new literals and clauses on the fly.

2.3 Theory Solvers

The EUF Solver. The theory solver for EUF is an extension of the congruence closure algorithm presented in [NO07]. It is an incremental algorithm that pioneered the integration of integer offsets in a congruence closure algorithm and the efficient computation of small explanations of inconsistency.

The DL Solver. The solver for DL is an implementation of the algorithm proposed by Cotton and Maler in [SM06]. It is a negative-cycle-detection algorithm that allows one to compute exhaustive theory propagation in a very efficient way. In order to improve efficiency, the infinite-precision arithmetic library GMP [GMP] is only called if C++ native arithmetic types do not suffice to ensure correctness.

The LA Solver. The solver for *Linear Real Arithmetic (LRA)* implements a primal simplex algorithm based on [RS04] that allows incremental addition and deletion of constraints. However, thanks to the work done by the preprocessor, no (dis)equalities have to be processed, and hence the costly exhaustive implicit equality propagation is no longer necessary. Further, as our algorithm can handle linear programs in general form [Mar86], bounds are dealt with in a more efficient way than in [RS04]. Finally, both the *tableau* as well as the *revised* implementations of the algorithm are available for the solver to choose depending on, e.g., the condition number or the density of the problem.

As regards *Linear Integer Arithmetic (LIA)*, a branch-and-cut algorithm has been implemented. Branching is performed by means of the cooperation of the Boolean engine and the LRA solver following the splitting-on-demand architecture [BNOT06], instead of splitting inside a stand-alone LIA solver. In combination with branch-and-bound, a cutting-planes algorithm has also been implemented along the lines of [DdM06].

Given the remarkable amount of DL literals that is typically manipulated when solving a problem in LA, the aforementioned solvers are called in a layered fashion [BBC$^+$05]: a pre-filtering DL solver is used which checks the consistency of the DL fragment of the assignment, and also propagates theory information to DPLL(X).

2.4 Further Capabilities

Model generation. When a formula is found to be satisfiable, Barcelogic outputs a model as a conjunction of atoms in the SMT-LIB format. For the theories under consideration, this amounts to (i) a truth value for each Boolean variable, (ii) a concrete integer or real for each numeric variable, and (iii) a partial mapping for each uninterpreted function symbol in the formula. Note that since the formula is ground, only a finite number of function applications appear and hence a partial mapping suffices to provide a model of the formula.

Predicate abstraction. Predicate abstraction is a technique for automatically extracting finite-state abstractions for systems with potentially infinite state space. The core operation in predicate abstraction is, given (i) a set of predicates P that express properties of the system, and (ii) a formula F that symbolically represents a transition system or a set of states, to compute the best approximation of F using the predicates P. In [LNO06] we showed that by means of a careful enumeration of satisfying assignments, state-of-the-art SMT solvers can

be turned into very efficient predicate abstraction engines, obtaining important speedups wrt. previously existing techniques.

Max-SAT and Max-SMT. In several applications, one has a set of constraints which is known to be unsatisfiable in advance and wants to find an assignment that satisfies the maximum number of constraints. This is the so-called Max-SAT or Max-SMT problem, depending on whether the constraints are expressed as SAT or SMT. A further extension is the weighted version of these problems, where one assigns a weight, called the *violation cost*, to each constraint and wants to find the assignment that minimizes the sum of the costs of the unsatisfied constraints. In [NO06] we showed how SMT tools can be easily adapted to support this functionality and our Barcelogic SMT solver implements all the techniques described there.

References

[BBC+05] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani. The MathSAT 3 System. In CADE'05, LNCS 3632, pp. 315–321.

[BNOT06] C. Barrett, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Splitting on Demand in SAT Modulo Theories. In LPAR'06, LNCS 4246, pp. 512–526.

[DdM06] B. Dutertre and L. de Moura. A Fast Linear-Arithmetic Solver for DPLL(T). In CAV'06. LNCS 4144, pp. 81–94.

[ES04] N. Eén and N. Sörensson. An Extensible SAT-solver. In SAT'03, LNCS 2919, pp. 502–518.

[GMP] The GNU MP Bignum Library. Available at `http://gmplib.org/`.

[LNO06] S. K. Lahiri, R. Nieuwenhuis, and A. Oliveras. SMT Techniques for Fast Predicate Abstraction. In CAV'06, LNCS 4144, pp. 413–426.

[Mar86] I. Maros. Computational Techniques of the Simplex Method. Kluwer's International Series, 2003.

[MMZ+01] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *38th Design Automation Conference, DAC'01*, pages 530–535. ACM Press, 2001.

[NO05] R. Nieuwenhuis and A. Oliveras. DPLL(T) with Exhaustive Theory Propagation and its Application to Difference Logic. In CAV'05, LNCS 3576, pp.321–334.

[NO06] R. Nieuwenhuis and A. Oliveras. On SAT Modulo Theories and Optimization Problems. In SAT'06, LNCS 4121, pp. 156–169.

[NO07] R. Nieuwenhuis and A. Oliveras. Fast Congruence Closure and Extensions. *Information and Computation, IC*, 2005(4):557–580, 2007.

[NOT06] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM, JACM*, 53(6):937–977, 2006.

[RS04] H. Rueß and N. Shankar. Solving Linear Arithmetic Constraints. Technical Report CSL-SRI-04-01, SRI International, 2004.

[SM06] S.Cotton and O. Maler. Fast and Flexible Difference Constraint Propagation for DPLL(T). In SAT'06, LNCS 4121, pp. 170–183.

[TR05] C. Tinelli and S. Ranise. SMT-LIB: The Satisfiability Modulo Theories Library, 2005. http://goedel.cs.uiowa.edu/smtlib/.