

System Description: MathSAT 4

<http://mathsat.itc.it>

Roberto Bruttomesso¹, Alessandro Cimatti¹, Anders Franzén¹, Alberto Griggio², and Roberto Sebastiani²

¹ FBK-IRST, Via Sommarive 18, 38050 Povo, Trento, Italy. {bruttomesso,cimatti,franzen}@itc.it

² Università di Trento, Via Sommarive 14, 38050 Povo, Trento, Italy. {griggio,rseba}@dit.unitn.it

MATHSAT is built on top of the standard “online” lazy integration schema used in many SMT tools (see, e.g., [1, 9]). In short: after some preprocessing to the input formula ϕ , a DPLL-based SAT solver is used as an enumerator of (possibly partial) truth assignments for (the boolean abstraction of) ϕ ; the consistency in \mathcal{T} of (the set of atomic constraints corresponding to) each assignment is checked by a solver \mathcal{T} -SOLVER. This is done until either one \mathcal{T} -consistent assignment is found, or all assignments have been checked.

MATHSAT 4 is an almost-from-scratch re-implementation of MATHSAT 3 [3], written entirely in C++. The basic architecture is the same, but some features are still missing.

The Preprocessor

A preprocessing step is performed to convert the input formula in CNF (if it is not already) and to encode equivalent \mathcal{T} -literals into a unique representation. Moreover, if the input formula contains constraints on mixed theory $\mathcal{EUF} \cup \mathcal{T}$, then the preprocessor eliminates uninterpreted functions by applying Ackermann’s reduction³. Finally, MATHSAT applies static learning [3], i.e. it adds to the formula small clauses representing \mathcal{T} -valid lemmas (e.g. transitivity constraints) which can fasten the boolean reasoning process.

The SAT Solver

The boolean enumerator is built upon the MINISAT solver [8], version 2.0.

The Theory Solvers

A \mathcal{T} -SOLVER gets in input a set of quantifier-free constraints μ and checks whether μ is \mathcal{T} -satisfiable or not. In the first case, it also tries to perform deductions in the form $\mu' \models_{\mathcal{T}} l$, where $\mu' \subseteq \mu$ and l is a literal representing a truth assignment to a not-yet-assigned atom occurring in the input formula. In the second case, it returns the (possibly minimal) sub-assignment $\mu' \subseteq \mu$ which caused the inconsistency (*conflict set*).

MATHSAT 4 supports four different \mathcal{T} -SOLVERS:

³ The Delayed Theory Combination scheme [2] of MATHSAT 3.4 has not been ported yet to MATHSAT 4.

The \mathcal{EUF} solver is based on the congruence closure algorithm as presented in [6], with generation of explanations performed like in [10]. The solver has been extended to handle numeric constants and (partially) numeric relations, to support addition of new atoms at run-time, and to perform the dynamic Ackermann's expansion technique of [5].

The Difference Logic solver is an implementation of the procedure described in [4].

The $\mathcal{LA}(\mathbb{R})$ solver is based on [7], with an extension to handle disequalities directly.

Finally, the $\mathcal{LA}(\mathbb{Z})$ solver is based on branch and bound [3].

When more than one theory solver is activated, *layering* is used [3]: faster, more general solvers are run first, and slower, more specialized ones are invoked only if the early ones do not detect an inconsistency. Layering is particularly effective on difficult theories, like $\mathcal{LA}(\mathbb{R})$ and especially $\mathcal{LA}(\mathbb{Z})$.

SMT-COMP problem divisions

MATHSAT 4 will compete in the QF_UF, QF_IDL, QF_RDL, QF_UFIDL, QF_LRA, QF_LIA and QF_UFLIA divisions. As regards performance, we expect it to be generally faster than MATHSAT 3.4.

Seed number: 54507995

References

1. G. Audemard, P. Bertoli, A. Cimatti, A. Kornilowicz, and R. Sebastiani. A SAT Based Approach for Solving Formulas over Boolean and Linear Mathematical Propositions. In *CADE-18*, volume 2392 of *LNAI*, pages 195–210. Springer, 2002.
2. M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, S. Ranise, P. van Rossum, and R. Sebastiani. Efficient theory combination via boolean search. *Inf. Comput.*, 204(10):1493–1525, 2006.
3. M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. van Rossum, S. Schulz, and R. Sebastiani. MathSAT: Tight Integration of SAT and Mathematical Decision Procedures. *J. Autom. Reasoning*, 35(1-3):265–293, 2005.
4. S. Cotton and O. Maler. Fast and Flexible Difference Constraint Propagation for DPLL(T). In *Proc. of SAT*, volume 4121 of *LNCS*, pages 170–183, 2006.
5. L. de Moura and N. Bjorner. Model-based theory combination. In *Proc. of SMT'07 workshop*. To appear.
6. D. Detlefs, G. Nelson, and J. 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 *Proc. of CAV*, pages 81–94, 2006.
8. N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. of SAT*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.
9. C. Flanagan, R. Joshi, X. Ou, and J.B. Saxe. Theorem Proving using Lazy Proof Explication. In *CAV 2003*, volume 2725 of *LNCS*, pages 355–367. Springer, 2003.
10. R. Nieuwenhuis and A. Oliveras. Proof-producing congruence closure. In *Proc. of RTA*, pages 453–468, 2005.