Yices2 in SMT-COMP 2026

Bruno Dutertre, Aman Goel, Stéphane Graham-Lengrand,
Thomas Hader, Ahmed Irfan, Dejan Jovanović, Enrico Lipparini,
Ian A. Mason, Karthik Nukala, Harald Ruess

Computer Science Laboratory, SRI International

1 Introduction

Yices2 [3] is an open-source (GPLv3) SMT solver developed and distributed by SRI International. It can be downloaded at `http://yices.csl.sri.com` and on our GitHub repository at `https://github.com/SRI-CSL/yices2`. The solver supports linear and non-linear arithmetic, bit-vectors, finite fields, uninterpreted functions, and arrays.

Yices2 uses the standard CDCL(T) [9] architecture and a variant of the Nelson-Oppen method for combining decision procedures. Details are presented in [3]. The solver also includes a Model-Construction Satisfiability Calculus (MCSat) [2, 7, 4, 6, 5] implementation. By default, MCSat is used for all theories that require non-linear arithmetic and CDCL(T) is used for everything else. Quantifier reasoning is supported for the UF theory, via E-graph matching and model-based instantiation. For bitvector solving, Yices2 can use third-party SAT solvers as bit-blasting backend, called "delegates". Currently, it supports four SAT solvers: CaDiCaL [1], CryptoMiniSat [10], Kissat [1], and `y2sat`, a core SAT solver within the Yices2 code base.

2 What's new?

The main changes since SMT-COMP 2025 are in the clause-deletion heuristics of both the CDCL(T) and MCSat engines, in the modularity of the bit-blasting backend, in the MCSat local search for nonlinear arithmetic, in MCSat reasoning over arrays and tuples, and in the Yices2 API.

- We replaced the Minisat-style learned-clause deletion strategy with a CaDiCaL-style scheme, in both the CDCL(T) core and the MCSat Boolean plugin: clause-database reductions are now triggered on a conflict-based schedule that is independent of the size of the input problem, and learned clauses that participated in recent conflicts are protected from deletion for at least one reduction round. In MCSat, the new schedule additionally takes effect under partial restarts, where the previous strategy would not trigger reductions at all.

- For bitvector solving, the delegate bit-blasting backends can now be used in incremental mode and push-pop mode.
- MCSat also gained better support for arrays and tuples, including more robust model reconstruction for array and function values. The thread-safe mode now also covers MCSat. We also continued to improve the MCSat local-search procedure [8] for nonlinear arithmetic, which guides the search using promising candidate assignments.
- The Yices API has also been extended with support for finite-field arithmetic and a more comprehensive set of primitives for constructing and inspecting models and model values. Thread safety is also now more robust.
- Finally, both the CDCL(T) and MCSat engines now print array values in SMT-LIB2 format. This allows us to enter the array logics in the model-validation track this year. Algebraic model values are also printed according to the SMT-competition format.

3 Competition Version

For SMT-COMP 2026, we are participating with the latest development version of Yices2 (`https://github.com/SRI-CSL/yices2/tree/smtcomp2026`). This version will compete in all supported logics and divisions, including the incremental, model-validation, and unsat-core tracks. Additionally, we are submitting a portfolio solver in the parallel track for the nonlinear arithmetic logics.

This year, we use CaDiCaL as the backend SAT solver for QF_BV in the single-query and incremental tracks, via the `--delegate=cadical` option.

For the QF_NIA logic, we enable the MCSat local-search feature (`--mcsat-l2o`) in the single-query, model-validation, and parallel tracks.

The set of logics we run on is largely unchanged from last year in the single-query, incremental, and unsat-core tracks. In the model-validation track, we additionally enter the array logics (QF_ABV, QF_ALIA, QF_ANIA, QF_AUFBV, QF_AUFBVLIA, QF_AUFBVNIA, QF_AUFLIA, QF_AUFNIA, and QF_AX), which we did not submit in 2025. In the parallel track, we now restrict the portfolio to the nonlinear arithmetic logics, whereas in 2025 we ran it on essentially all logics.

References

[1] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL, Gimsatul, IsaSAT and Kissat entering the SAT Competition 2024. In Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions*, volume B-2024-1 of *Department of Computer Science Report Series B*, pages 8–10. University of Helsinki, 2024.

[2] Leonardo Mendonça de Moura and Dejan Jovanovic. A model-constructing satisfiability calculus. In Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni, editors, *Ver-*

ification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings, volume 7737 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.

[3] Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, 2014.

[4] Stéphane Graham-Lengrand, Dejan Jovanovic, and Bruno Dutertre. Solving bitvectors with MCSAT: explanations from bits and pieces. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*, volume 12166 of *Lecture Notes in Computer Science*, pages 103–121. Springer, 2020.

[5] Thomas Hader, Daniela Kaufmann, Ahmed Irfan, Stéphane Graham-Lengrand, and Laura Kovács. Mcsat-based finite field reasoning in the yices2 smt solver (short paper). In *International Joint Conference on Automated Reasoning*, pages 386–395. Springer, 2024.

[6] Ahmed Irfan and Stéphane Graham-Lengrand. Arrays Reasoning in MCSat. *SMT Workshop 2024*, 2024.

[7] Dejan Jovanovic, Clark Barrett, and Leonardo De Moura. The design and implementation of the model constructing satisfiability calculus. In *2013 Formal Methods in Computer-Aided Design*, pages 173–180. IEEE, 2013.

[8] Enrico Lipparini, Thomas Hader, Ahmed Irfan, and Stéphane Graham-Lengrand. Boosting mcsat modulo nonlinear integer arithmetic via local search. *CADE 2025*, To appear.

[9] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Abstract DPLL and abstract DPLL modulo theories. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14-18, 2005, Proceedings*, volume 3452 of *Lecture Notes in Computer Science*, pages 36–50. Springer, 2004.

[10] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009.