YicesQS 2026, an extension of Yices for quantified satisfiability

Stéphane Graham-Lengrand

SRI International, USA

YicesQS is a solver derived from Yices 2, an open-source SMT solver developed and distributed by SRI International. It extends Yices to supports quantifiers for complete theories, and is unrelated to the support of quantifiers in Yices for UF. Its core algorithm is a generalization of counterexample-guided quantifier instantiation (CEGQI) [Dut15] that can be seen as a form of lazy quantifier elimination. YicesQS submits quantifier-free queries to one of Yices's core solvers CDCL(T) or MCSAT [dMJ13]. It is written in OCaml and uses the OCaml bindings for the Yices C API; it indirectly relies on the libpoly library for arithmetic. Since the 2022 SMT competition, YicesQS enters logics BV, NRA, NIA, LRA, and LIA (single-query tracks). The exact commits for 2026 are as follows:

https://github.com/disteph/yicesQS	commit e5107457f
https://github.com/SRI-CSL/yices2_ocaml_bindings	commit 3c33185ad
https://github.com/SRI-CSL/yices2	commit f67499dc0
https://github.com/SRI-CSL/libpoly_ocaml_bindings	commit 43018ddec

and the exact command is

```
./main.exe -auto_portfolio 1200 $1
```

Algorithmic approach. The main algorithm in YicesQS is QSMA [BGV25]. It does not modify the structure of quantifiers in formulas: it does not prenexify formulas and, more importantly, it does not skolemize them to avoid introducing uninterpreted function symbols. This departs from the standard architecture for quantifier support consisting of keeping a set of universally quantified clauses, to be grounded and sent to a core SMT-solver for ground clauses. Instead, YicesQS sees a formula as a game in the tradition of Bjørner & Janota's *Playing with Quantified Satisfaction* [BJ15] and earlier work on QBF. YicesQS's algorithm is designed to answer queries of the following form:

"Given a formula $A(\overline{z},\overline{x})$ and a model $\mathfrak{M}_{\overline{z}}$ on $\overline{z}$, produce either
- SAT($U(\overline{z})$), with $U(\overline{z})$ under-approx. of $\exists\overline{x}\ A(\overline{z},\overline{x})$ satisfied by $\mathfrak{M}_{\overline{z}}$; or
- UNSAT($O(\overline{z})$), with $O(\overline{z})$ over-approx. of $\exists\overline{x}\ A(\overline{z},\overline{x})$ not satisfied by $\mathfrak{M}_{\overline{z}}$;

where under-approximations and over-approximations are quantifier-free."

To answer such queries, YicesQS calls Yices's feature *satisfiability modulo a model*, while the production of under- and over-approximations leverages *model-based over-approximation* (MBO) and *model-based under-approximation* (MBU). When the input formula is in the exists-forall fragment, the algorithm degenerates to the CEGQI one used in Yices' $\exists\forall$ solver [Dut15] using quantifier-free solving and MBU. MBO, a form of which is used within Yices's MCSAT solver to solve quantifier-free problems [JD21], also becomes useful with more quantifier alternations than $\exists\forall$. It generalizes to non-Boolean inputs the notion of UNSAT cores, which has been used in the *quantified-problems-as-games* approach [BJ15].

LIA. *Solving Modulo a Model*: We leverage Yices's CDCL(T) solver, where the theory solver is based on the simplex, as follows. The partial input model is represented

as double inequalities between integer variables and their values, and given to Yices as assumptions. *MBO*: We simply use UNSAT cores produced by Yices's CDCL(T) solver. *MBU*: We compute an implicant cube for the formula to approximate, and use a model-based projection approach specific to Presburger arithmetic.

NRA, NIA, LRA. *Solving Modulo a Model*: We use Yices's MCSAT solver; the MCSAT approach is inspired by CDCL, building a tentative solution model on a trail where Boolean and theory assignments are decided, propagated, and backtracked upon. MCSAT can natively take a partial model as input [JD21]. *MBO*: We use the MBOs natively produced by Yices's MCSAT solver: a lemma that is learnt at decision level 0, defeating the input model, constitutes an MBO. Learnt lemmas arise from theory-specific mechanisms for explaining conflicts, which in the case of arithmetic is leveraging Cylindrical Algebraic Decomposition (CAD) [JdM12,Jov17]. *MBU*: We compute one or more implicant cubes for the formula to approximate (1 for LRA, all of them for NRA and NIA); the final MBU is the disjunction of the MBUs for each cube. For a given cube, we mainly use two forms of model-based projection. For real arithmetic (NRA, LRA) we use a projection based on CAD. For integer arithmetic (NIA), we use generalization-by-substitution (a generic MBU already used in Yices's $\exists\forall$ solver [Dut15]) to eliminate some of the variables until the substituted cube is in Presburger arithmetic and we can apply the same projection as for LIA, or all of the variables to eliminate have been substituted. A few arithmetic constructs, such as division, modulo, floor, ceiling, abs, ITE, are not supported natively by either projection, so we first compute a preliminary MBU of the original formula that does not contain such constructs but does not eliminate any variable. For the special case of models where denominators in the formula evaluate to 0, we resort to generalization-by-substitution again, which may cause YicesQS to not terminate on NRA (it is undecidable in presence of division), just like it may not terminate on NIA.

Bitvectors (BV). As described for NRA, NIA, LRA, MCSAT can be used for *Solving modulo a Model* and MBO for BV [GLJD20]. As described for LIA, CDCL(T) can also be used for that, modeling the input model as either equality or inequality assumptions, and then fetching an UNSAT core. In the CDCL(T) solver, bitblasting handles BV reasoning and several incremental SAT-solvers can be used as backends: Cadical, Cryptominisat, and Yices itself. For the competition we run a sequential portfolio consisting of CDCL(T)+Yices for 700s (we do not use Cadical or Cryptominisat), then MCSAT. In both cases, MBU uses invertibility conditions from Niemetz et al. [NPR+18], including ϵ-terms, in combination with generalization-by-substitution. For the BV theory, the cegqi solver from [NPR+18] is probably the closest to YicesQS.

Changes since 2025. The following items are new:

- LIA: using double inequalities for CDCL(T), Presburger model-based projection;
- Arithmetic MBUs: Handling advanced arithmetic constructs differently than via basic generalization-by-substitution; then taking as final MBU the disjunction of cube-specific MBUs for all cube implicants (NIA, NRA); the attempt in NIA to reduce the problem to Presburger arithmetic with repeated substitutions;
- The possibility (unused) to call external SAT-solvers for bitblasted problems;
- A more advanced sequential portfolio infrastructure;
- Changes in Yices that affect YicesQS: Local search [LHIGL25] and new caching heuristics [HIGL25] in MCSAT, new clause deletion heuristics.

References

BGV25. M. P. Bonacina, S. Graham-Lengrand, and C. Vauthier. The QSMA algorithm for quantifiers in SMT. *J. of Automated Reasoning*, 69(2):13, 2025. https://doi.org/10.1007/S10817-025-09727-8 1

BJ15. N. Bjorner and M. Janota. Playing with quantified satisfaction. In M. Davis, A. Fehnker, A. McIver, and A. Voronkov, editors, *Proc. of the the 20th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'15)*, volume 9450 of *LNCS*. Springer-Verlag, 2015. https://doi.org/10.1007/978-3-662-48899-7 1

dMJ13. L. M. de Moura and D. Jovanovic. A model-constructing satisfiability calculus. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *Proc. of the 14th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)*, volume 7737 of *LNCS*, pages 1–12. Springer-Verlag, 2013. https://doi.org/10.1007/978-3-642-35873-9_1 1

Dut15. B. Dutertre. Solving exists/forall problems with Yices. In *Proc. SMT-13*, 2015. 1, 2

GLJD20. S. Graham-Lengrand, D. Jovanović, and B. Dutertre. Solving bitvectors with MCSAT: explanations from bits and pieces. In N. Peltier and V. Sofronie-Stokkermans, editors, *Proc. of the 10th Int. Joint Conf. on Automated Reasoning (IJCAR'20)*, volume 12166(1) of *LNCS*, pages 103–121. Springer-Verlag, 2020. 2

HIGL25. T. Hader, A. Irfan, and S. Graham-Lengrand. Decision heuristics in mcsat. In R. Piskac and Z. Rakamaric, editors, *Proc. of the 37th Int. Conf. on Computer Aided Verification (CAV'25)*, LNCS. Springer-Verlag, 2025. 2

JD21. D. Jovanovic and B. Dutertre. Interpolation and model checking for nonlinear arithmetic. In A. Silva and K. R. M. Leino, editors, *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part II*, volume 12760 of *Lecture Notes in Computer Science*, pages 266–288. Springer, 2021. https://doi.org/10.1007/978-3-030-81688-9_13 1, 2

JdM12. D. Jovanović and L. de Moura. Solving non-linear arithmetic. In B. Gramlich, D. Miller, and U. Sattler, editors, *Proc. of the 6th Int. Joint Conf. on Automated Reasoning (IJCAR'12)*, volume 7364 of *LNCS*, pages 339–354. Springer-Verlag, 2012. 2

Jov17. D. Jovanović. Solving nonlinear integer arithmetic with MCSAT. In A. Bouajjani and D. Monniaux, editors, *Proc. of the 18th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'17)*, volume 10145 of *LNCS*, pages 330–346. Springer-Verlag, 2017. https://doi.org/10.1007/978-3-319-52234-0_18 2

LHIGL25. E. Lipparini, T. Hader, A. Irfan, and S. Graham-Lengrand. Boosting mcsat modulo nonlinear integer arithmetic via local search. In C. Barrett and U. Waldmann, editors, *Proc. of the 30th Int. Conf. on Automated Deduction (CADE'25)*. Springer-Verlag, 2025. 2

NPR+18. A. Niemetz, M. Preiner, A. Reynolds, C. W. Barrett, and C. Tinelli. Solving quantified bit-vectors using invertibility conditions. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of FLoC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pages 236–255. Springer, 2018. https://doi.org/10.1007/978-3-319-96142-2_16 2