Yices 1.0: An Efficient SMT Solver

*SMT-COMP’06*

Leonardo de Moura (joint work with Bruno Dutertre)

\{demoura, bruno\}@csl.sri.com.

Computer Science Laboratory

SRI International

Menlo Park, CA
Introduction

- Yices is an SMT Solver developed at SRI International.
- It is used in SAL, PVS, and CALO.
- It is a complete reimplementation of SRI’s previous SMT solvers.
  - It has a new architecture, and uses new algorithms.
  - Counterexamples and Unsatisfiable Cores.
  - Incremental: push, pop, and retract.
  - Weighted MaxSAT/MaxSMT.
- Supports all theories in SMT-COMP.
Supported Features

- Uninterpreted functions
- Linear real and integer arithmetic
- Extensional arrays
- Fixed-size bit-vectors
- Quantifiers
- Scalar types
- Recursive datatypes, tuples, records
- Lambda expressions
- Dependent types
It is “impossible” to build an efficient SAT solver (and SMT solver) for arbitrary formulas.

Ignore hand-made and random benchmarks.

“The breakthrough is SAT solving happened after industrial benchmarks started to be used.”

Randy Bryant

“What is the hardest part in the implementation of a theorem prover? Ans: Testing/Benchmarking”

Greg Nelson
The new architecture integrates:

- a modern DPLL-based SAT solver,
- a core theory solver that handles equalities and uninterpreted functions,
- satellite theories (for arithmetic, arrays, bit-vectors, etc.).
- It should be easy to extract the model.

Yices uses an extension of the standard Nelson-Oppen combination method.

The core and satellite theories communicate via offset equalities

\[ x = y + k. \]
Yices can be used as a regular SAT solver (it can read DIMACS files).

Uses ideas from top performing SAT solvers: MiniSAT, Siege, zChaff.

Supports the creation of clauses and boolean variables during the search.

It is tightly integrated with the core theory solver.

Supports user defined constraints. Examples:

- Linear pseudo-boolean constraint (used in MaxSMT).
- Bridge between bit-vector terms and boolean variables used in bit-blasting.
DPLL-based SAT solver (cont.)

- Explanations for assigned literals:
  - Clause (like any SAT solver).
  - Generic explanation.
    - Antecedents can be computed only when they are needed.
    - Very convenient for implementing new theories.
    - Avoids flooding the SAT solver with useless clauses.
  - Processes the case-splits produced by satellite theories:
    - Bit-vector
    - Linear integer arithmetic
    - Array
Core Theory Solver

- Core theory solver handles (offset) equalities and uninterpreted functions.
  - Offset equalities \( \sim \) less communication overhead.
  - Offset equalities \( \sim \) less shared variables.

- The algorithm used in the core is similar to the one used in the Simplify theorem prover.

- Extensions for producing precise explanations and for handling offset equalities.

- Exhaustive theory propagation (equalities & disequalities).
  - \( x_1 = \ldots = x_n \neq y_m = \ldots = y_1 \sim x_1 \neq y_1 \)

- Satellite theories are attached to the core.

- It is very easy to add new satellite theories.
Equality propagation

- Satellite theories are not required to propagate all implied equalities.
- Yices case splits on (offset) equalities between shared variables to achieve completeness.
- Each theory is responsible for creating the required case-splits.
- Simple filters are used to minimize the number of case-splits.
  - Example: suppose the core contains four terms $f(x_1, x_2)$, $f(x_3, x_4)$, $g(x_5)$, and $g(x_6)$, and $x_1$ to $x_6$ are shared variables.
  - Case splitting on $x_1 = x_3$, $x_2 = x_4$ and $x_5 = x_6$ is sufficient.
Linear arithmetic

- Novel Simplex-based algorithm (see CAV’06 paper).
  - Efficient backtracking and theory propagation.
  - New approach for solving strict inequalities \( (t > 0) \).
  - Presimplification step.
- Integer arithmetic: Gomory Cuts, Branch & Bound, and GCD Test.
- Arbitrary precision arithmetic.

- On sparse problems, this solver is competitive with tools specialized for difference logic.
- For dense difference-logic problems, Yices uses a specialized algorithm based on incremental Floyd-Warshall.
Yices creates the clause $x \neq y \lor f(x) = f(y)$ whenever the congruence rule $x = y \leadsto f(x) = f(y)$ is used to deduce a conflict.

Yices can perform the propagation $f(x) \neq f(y) \leadsto x \neq y$, which is missed by traditional congruence-closure algorithms.

This propagation rule has a dramatic performance benefit on many problems.

Avoids flooding the SAT solver with unnecessary instances.

DPLL solver clause-deletion heuristics can safely remove any of the dynamically created instances since they are not required for completeness.
Function (Array) Theory

- Yices (like PVS) does not make a distinction between arrays and functions.
- Function theory handles: function updates, lambda expressions, and extensionality.
- Lazy instantiation of theory axioms.
  - $\forall f, i, v. \ select(store(f, i, v), i) = v$
  - $\forall f, i, j, v. i = j \lor \ select(store(f, i, v), j) = select(f, j)$
  - $\forall f, g. f = g \lor \exists k. \ select(f, k) \neq select(g, k)$
Lazy reduction to uninterpreted functions.

- $f \sim g$ means $f$ and $g$ are in the same equivalence class.
- $\text{store}(f, i, v) \sim \text{select}(\text{store}(f, i, v), i) = v$
- $g \sim \text{store}(f, i, v), \text{select}(g, j) \sim$
  
  $i = j \lor \text{select}(\text{store}(f, i, v), j) = \text{select}(f, j)$
- $g \sim f, \text{store}(f, i, v), \text{select}(g, j) \sim$
  
  $i = j \lor \text{select}(\text{store}(f, i, v), j) = \text{select}(f, j)$
- $f \not\equiv g \sim$ for a fresh $k$
  
  $\text{select}(f, k) \neq \text{select}(g, k) \land \text{typepred}(k)$

- A similar approach is used to implement tuples, records and recursive datatypes.
Bit-vector Theory

- It is implemented as a satellite theory.
- So, core theory handles equalities and uninterpreted functions.
- Straightforward implementation:
  - Simplification rules.
  - Bit-blasting for all bit-vector operators but equality.
  - “Bridge” between bit-vector terms and the boolean variables.
Quantifiers

- Main approach: egraph matching (Simplify)
  - Extension for offset equalities and terms.
  - Several triggers (multi-patterns) for each universally quantified expression.
  - The triggers are fired using a heuristic that gives preference to the most conservative ones.
- Fourier Motzkin elimination to simplify quantified expressions.
- Instantiation heuristic based on:
  
  *What’s Decidable About Arrays?*,
  
  A. R. Bradley, Z. Manna, and H. B. Sipma, VMCAI’06.
Yices is an efficient and flexible SMT solver.
- Yices supports all theories in SMT-COMP and much more.
- It is being used in SAL, PVS, and CALO.
- Fixed all bugs in Yices 0.1.
- Tested on all (42167) SMT-LIB benchmarks with 10 different random seeds.
- Yices is not ICS.
- Yices is freely available for end-users.
  - http://yices.csl.sri.com
- Yices tutorial: AFM workshop (Tomorrow - August 21)