SMT-COMP 2021
16th International Satisfiability Modulo Theory Competition

Haniel Barbosa Jochen Hoenicke Antti Hyvärinen

Universidade Federal de Minas Gerais, Brazil

Albert-Ludwigs-Universität Freiburg, Germany

Universita della Svizzera italiana, Switzerland

July 18, 2021

SMT-COMP

Annual competition for SMT solvers
on (a selection of) benchmarks from SMT-LIB

History

2005	first competition
2013	evaluation instead of competition
2014	since then hosted by StarExec
2021	16th competition

Goals:

- spur development of SMT solver implementations
- promote SMT solvers and their usage
- support the SMT-LIB project
  - to promote and develop the SMT-LIB format
  - to collect relevant benchmarks
- engage and include new members

SMT Solvers and SMT-LIB

SMT Solver

- checks formulas in SMT-LIB format for satisfiability modulo theories

SMT-LIB is

1. a language in which benchmarks are written
2. a community effort to collect benchmarks

Non-incremental

381 683 instances (+5082)
with 1 query each
in 79 logics (+9).

Incremental

43 284 instances (+19 073)
with 33 998 794 queries (+554 499)
in 35 logics (+2).

SMT-LIB Logics

QF_IDL
QF_LIA
QF_LIRA
QF_RDL
QF_LRA
QF_NIA
QF_NIRA
QF_NRA
QF_AX
QF_DT
QF_UF
QF_UFDT
QF_BV
QF_S

QF_ALIA
QF_AUFLIA
QF_UFIDL
QF_UFLIA
QF_UFDTLIRA
QF_UFLRA
QF_ANIA
QF_AUFNIA
QF_UFNIA
QF_UFNRA
QF_ABV
QF_AUFBV
QF_UFBV
QF_SLIA

QF_ABVFP
QF_ABVFPLRA
QF_AUFBVFP
QF_BVFP
QF_BVFPLRA
QF_FP
QF_FPLRA
QF_UFFP
QF_UFFPDTLIRA

QF_AUFBVLIA
QF_AUFBVNIA
QF_UFBVLIA
QF_SNIA

LIA
LRA
NIA
NRA

UF
UFDT

BV

BVFP
BVFPLRA
FP
FPLRA

ALIA
AUFLIA
AUFLIRA
AUFDTLIA
AUFDTLIRA
ANIA
AUFNIA
AUFNIRA
AUFDTNIRA
ABV
ABVFP
ABVFPLRA
AUFBV
AUFBVFP
AUFBVDTLIA
AUFBVDTNIA

UFDTLIA
UFDTLIRA
UFIDL
UFLIA
UFLRA
UFDTNIA
UFDTNIRA
UFNIA
UFNRA
AUFFPDTLIRA
AUFFPDTNIRA
UFBV
UFBVFP
UFBVLIA
UFFPDTLIRA
UFFPDTNIRA

SMT-LIB Logics

Quantifier-free

QF_IDL
QF_LIA
QF_LIRA
QF_RDL
QF_LRA
QF_NIA
QF_NIRA
QF_NRA
QF_AX
QF_DT
QF_UF
QF_UFDT
QF_BV
QF_S

QF_ALIA
QF_AUFLIA
QF_UFIDL
QF_UFLIA
QF_UFDTLIRA
QF_UFLRA
QF_ANIA
QF_AUFNIA
QF_UFNIA
QF_UFNRA
QF_ABV
QF_AUFBV
QF_UFBV
QF_SLIA

QF_ABVFP
QF_ABVFPLRA
QF_AUFBVFP
QF_BVFP
QF_BVFPLRA
QF_FP
QF_FPLRA
QF_UFFP
QF_UFFPDTLIRA

QF_AUFBVLIA
QF_AUFBVNIA
QF_UFBVLIA
QF_SNIA

Quantified

LIA
LRA
NIA
NRA

UF
UFDT

BV

BVFP
BVFPLRA
FP
FPLRA

ALIA
AUFLIA
AUFLIRA
AUFDTLIA
AUFDTLIRA
ANIA
AUFNIA
AUFNIRA
AUFDTNIRA
ABV
ABVFP
ABVFPLRA
AUFBV
AUFBVFP
AUFBVDTLIA
AUFBVDTNIA

UFDTLIA
UFDTLIRA
UFIDL
UFLIA
UFLRA
UFDTNIA
UFDTNIRA
UFNIA
UFNRA
AUFFPDTLIRA
AUFFPDTNIRA
UFBV
UFBVFP
UFBVLIA
UFFPDTLIRA
UFFPDTNIRA

Quantifier **F**ree **A**rray **U**ninterpreted **F**unction **B**it**V**ector **F**loating**P**oint **D**ata**T**ype **S**trings
Nonlinear/**L**inear **I**nteger **R**eal **A**rithmetic **D**ifference **L**ogic

SMT-LIB Logics

Quantifier-free

QF_IDL
QF_LIA
QF_LIRA

QF_RDL
QF_LRA

QF_NIA
QF_NIRA

QF_NRA

QF_AX
QF_DT
QF_UF
QF_UFDT

QF_BV

QF_ALIA
QF_AUFLIA
QF_UFIDL
QF_UFLIA
QF_UFDTLIRA
QF_UFLRA

QF_ANIA
QF_AUFNIA
QF_UFNIA
QF_UFNRA

QF_ABV
QF_AUFBV
QF_UFBV

QF_ABVFP
QF_ABVFPLRA
QF_AUFBVFP
QF_BVFP
QF_BVFPLRA
QF_FP
QF_FPLRA
QF_UFFP
QF_UFFPDTLIRA

QF_AUFBVLIA
QF_AUFBVNIA
QF_UFBVLIA

QF_S QF_SLIA QF_SNIA

Quantified

LIA
LRA
NIA
NRA

UF
UFDT

BV

BVFP
BVFPLRA
FP
FPLRA

ALIA
AUFLIA
AUFLIRA
AUFDTLIA
AUFDTLIRA

UFDTLIA
UFDTLIRA
UFIDL
UFLIA
UFLRA

ANIA
AUFNIA
AUFNIRA
AUFDTNIRA

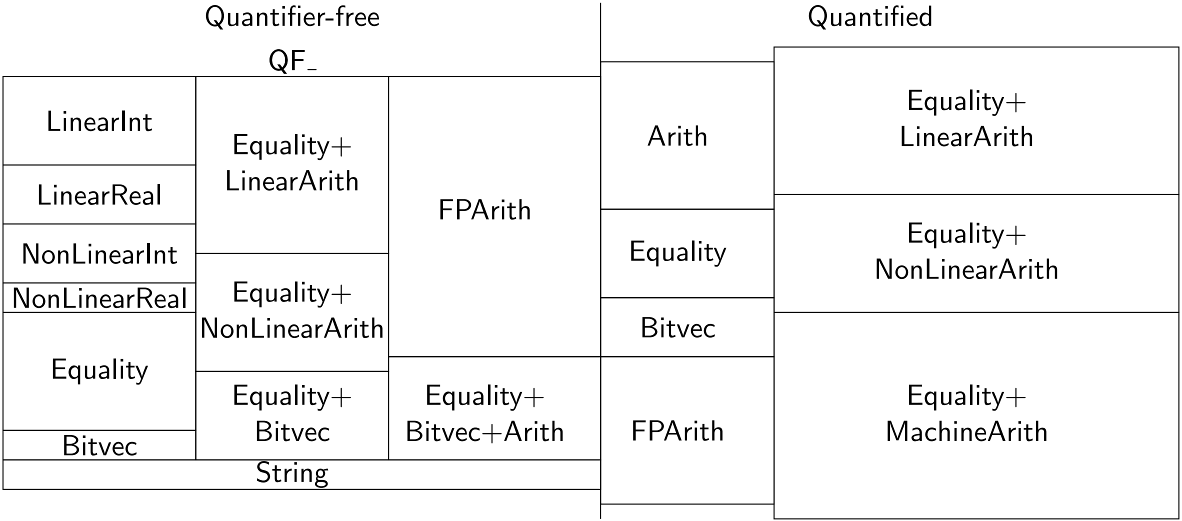
UFDTNIA
UFDTNIRA
UFNIA
UFNRA

ABV
ABVFP
ABVFPLRA
AUFBV
AUFBVFP
AUFBVDTLIA
AUFBVDTNIA

AUFFPDTLIRA
AUFFPDTNIRA
UFBV
UFBVFP
UFBVLIA
UFFPDTLIRA
UFFPDTNIRA

Quantifier **F**ree **A**rray **U**ninterpreted **F**unction **B**it**V**ector **F**loating**P**oint **D**ata**T**ype **S**trings
Nonlinear/**L**inear **I**nteger **R**eal **A**rithmetic **D**ifference **L**ogic

SMT-LIB Logics



Quantifier-free

QF_

LinearInt

LinearReal

NonLinearInt

NonLinearReal

Equality

Bitvec

Equality+ LinearArith

Equality+ NonLinearArith

Equality+ Bitvec

FPArith

Equality+ Bitvec+Arith

String

Quantified

Arith

Equality

Bitvec

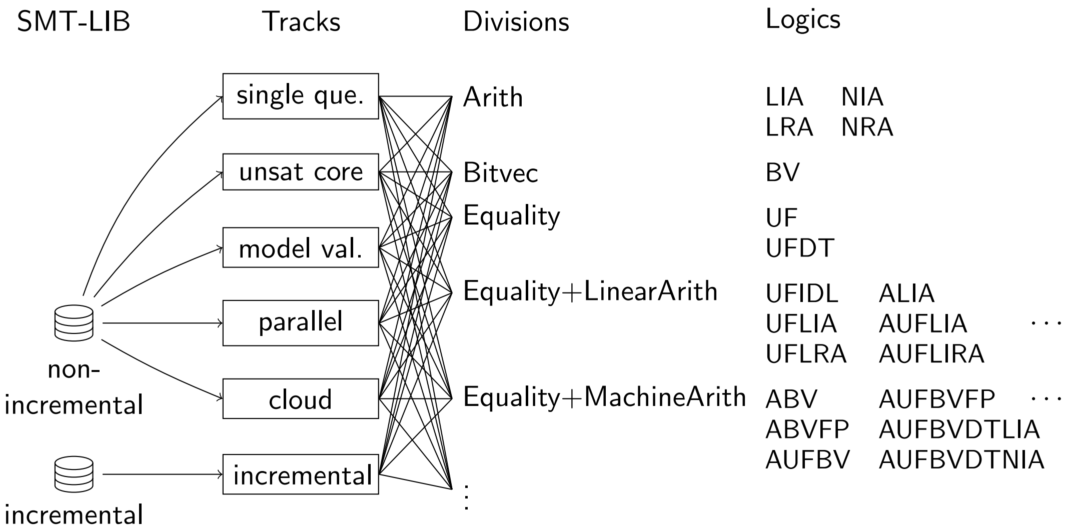
FPArith

Equality+ LinearArith

Equality+ NonLinearArith

Equality+ MachineArith

Competition Overview



SMT-LIB

Tracks

Divisions

Logics

single que.

unsat core

model val.

parallel

cloud

incremental

non-incremental

incremental

Arith

Bitvec

Equality

Equality+LinearArith

Equality+MachineArith

⋮

LIA	NIA	
LRA	NRA	
BV		
UF		
UFDT		
UFIDL	ALIA	
UFLIA	AUFLIA	···
UFLRA	AUFLIRA	
ABV	AUFBVFP	···
ABVFP	AUFBVDTLIA	
AUFBV	AUFBVDTNIA	

SMT-COMP Tracks (traditional)

Single Query Track

- Determine satisfiability of one problem
- Solver answers sat/unsat/unknown

Unsat Core Track

- Find small unsatisfiable subset of input.
- Solver answers unsat + list of formulas.

Model Validation Track

- Find a model for a satisfiable problem.
- Solver answers sat + value for each non-logical symbol.

Incremental Track

- Solve many small problems interactively.
- Solver acks commands and answers sat/unsat for each check.

SMT-COMP Tracks (new)

SMT-COMP 2021 has two new experimental tracks (sponsored by AWS).

Parallel Track

- Solve a large problem on a big computer
    - 64 cores, 256 GB of memory
- Solver answers sat/unsat/unknown

Cloud Track

- Solve a large problem on a network of computers
    - 100 machines, 1600 cores, 6400 GB of memory
- Solver answers sat/unsat/unknown

Tracks, Solvers, Divisions, and Benchmarks

Teams: 18 (+2)

Track	Solvers	Divisions	Benchmarks
Single Query	19(-1)	18(-49)	101300/381683
Incremental	7(-2)	15(-11)	22233/43284
Unsat Core	7(+2)	17(-23)	55463/108188
Model Validation	7(=)	3(+2) + 3 exp.	13301/21251
Parallel	3	14 exp.	413/20705
Cloud	5	14 exp.	405/20669

Number in parenthesis shows changes from 2020

Participants

SMT-COMP 2021 participants:

- classic CDCL(T)-based SMT solvers
- mcSAT-based solvers
- automated theorem provers
- finite domain solver
- local search techniques
- wrapper extending the scope of existing solvers

Four new solvers participated:

- iProver (Konstantin Korovin, Andre Duarte, Edvard K Holden)
- mc2 (Simon Cruanes, Guillaume Bury)
- YicesLS (Bohan Li, Shaowei Cai, Xindi Zhang)
- YicesQS (Stéphane Graham-Lengrand)

Solver Presentation

Bitwuzla at the SMT-COMP'21

Aina Niemetz, Mathias Preiner

Tracks/Divisions

Single Query:	QF_{A,BV,FP,FPLRA,UF}$^{+}$
Incremental:	QF_{A,BV,FP,FPLRA,UF}$^{+}$
Unsat Core:	QF_{A,BV,FP,FPLRA,UF}$^{+}$
Model Validation:	QF_BV, QF_UFBV

News

- Code now available at `https://github.com/bitwuzla/bitwuzla`
- New API for C, Python, and OCaml[1]
- Floating-points: Real to FP support (for FPLRA logics)
- Bit-vectors: CaDiCaL version sc2021 as default SAT backend for all logics
- Lots of improvements/refactoring going on behind the scenes

`https://bitwuzla.github.io`

[1] Thanks to Frédéric Recoules for the OCaml bindings

COLIBRI(2021)

- Use dolmen for parsing (presented tomorrow): through a Prolog ↔ OCaml bridge
- We didn't secured enough time for preparing the competition, so we botched the submission
- CP solvers usually only handle finite domains, the extension to infinite domains for `Int` is too difficult to maintain so we are reimplementing the solver as Colibri2

cvc5 at the SMT Competition 2021

C. Barrett, H. Barbosa, M. Brain, G. Kremer, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz,
A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, Y. Zohar

~~CVC4~~ cvc5

- Preview of upcoming release in fall 2021
- Support for all standardized SMT-LIB theories
- User-friendly API, significant refactoring of internals
- Complete rewrite of proof module

New Features/Improvements

- New subsolver for non-linear arithmetic based on cylindrical algebraic coverings using libpoly
- New bit-vector solver, integrating efficient SAT solvers, e.g., CaDiCaL, with CDCL($\mathcal{T}$)
- Syntax-guided quantifier instantiation
- New decision heuristic with optional prioritization of assertions involved in conflicts

Configurations

cvc5 entered all divisions in all tracks.

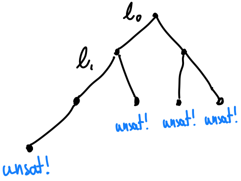
- Single query track: Sequential portfolio
- Unsat-core track: Based on new proof module and assumptions in the SAT solver

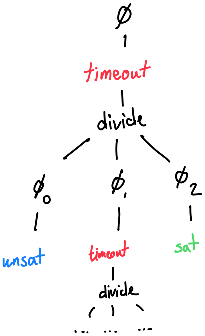
Follow the development: `https://cvc5.github.io/`

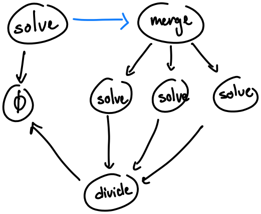
gg-cvc5

Barrett, Noetzli, Ozdemir, Reynolds, Wilson, Wu

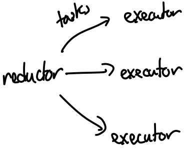
DPLL(T) → Divide-and-Conquer Search → Dependency Graph → gg







⚠ work-in-progress



Credits:
GitHub.com/cvc5
GitHub.com/gg-project
pygg

iProver v3.5 (Konstantin Korovin, André Duarte, Edvard K. Holden)

iProver supports all combinations of: quantifiers, uninterpreted functions, data types, linear and non-linear arithmetic.

- Quantified reasoning: model-guided Inst-Gen + superposition + resolution calculi.
  - Saturation algorithm: priority queues, discrimination trees, feature vector indexing.
  - Simplifications: forward/backward: demodulation, light normalisation, subsumption, global subsumption and subsumption resolution, AC ground joinability, AC normalisaion.
  - Preprocessing: predicate elimination, splitting, semantic filtering, subtyping and definition elimination.
- Ground reasoning: MiniSAT, Z3
- Clausification and Theory Axioms: Vampire
- Heuristic optimisation and scheduling using machine learning: HOS-ML

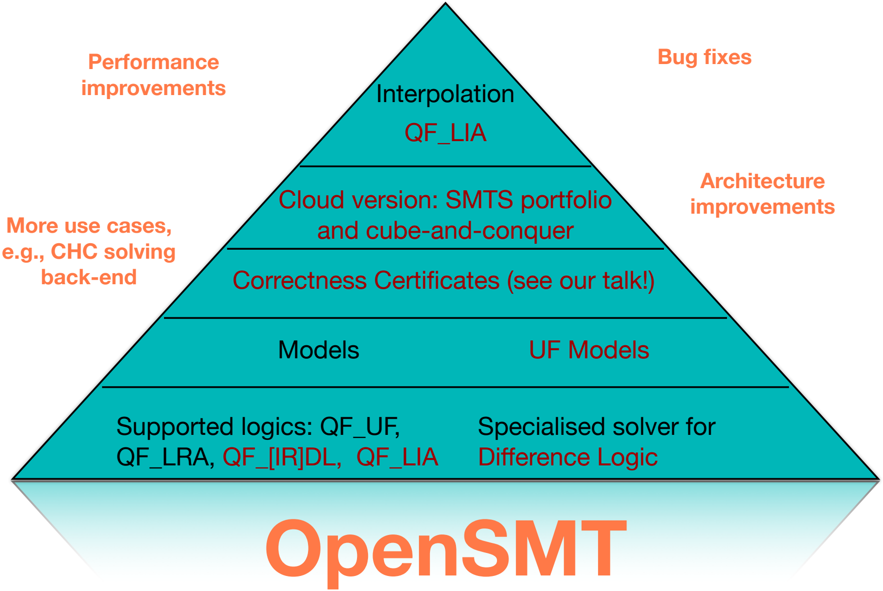
iProver is implemented in OCaml. `https://www.cs.man.ac.uk/~korovink/iprover`

mc² — a mcSAT solver

- **Implementation of mcSAT in OCaml (descendant of msat/alt-ergo zero)**
- **< 7kloc total**
- **Theories:**
  - Uninterpreted functions
  - LRA (conflict-driven Fourier Motzkin)
  - Boolean formulas via Tseitin encoding
- **https://github.com/c-cube/mc2 (Apache license)**
- **Basic calculus, not much in way of simplifications**
  - Naturally good at diamond problems
  - Decent performance on QF_UFLRA (hyp: theory combination is cheap in mcSAT?)
- **Currently not incremental**

Performance improvements

Bug fixes



Interpolation

QF_LIA

Cloud version: SMTS portfolio and cube-and-conquer

Correctness Certificates (see our talk!)

Models

UF Models

Supported logics: QF_UF, QF_LRA, QF_[IR]DL, QF_LIA

Specialised solver for Difference Logic

OpenSMT

Architecture improvements

More use cases, e.g., CHC solving back-end

SMTInterpol and SMTInterpol-remus

Jürgen Christ, Leonard Fichtner, Jochen Hoenicke, Moritz Mohr, Tanja Schindler

Interpolating SMT solver

- based on CDCL(T)
- for Arrays, Uninterpreted Functions, Linear Integer and Real Arithmetic
    - plus `div` and `mod` with constants,

  and DataTypes
- supports quantifiers
- produces models, proofs, and unsat cores
- computes sequence and tree interpolants

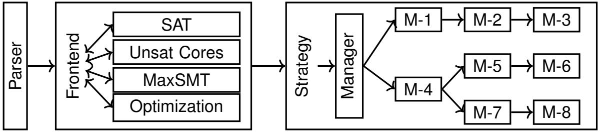
SMTInterpol at SMT-COMP 2021

- with proof check mode enabled
- experimentally participated in some Nonlinear Arithmetic divisions
- variant SMTInterpol-remus with unsat core enumeration

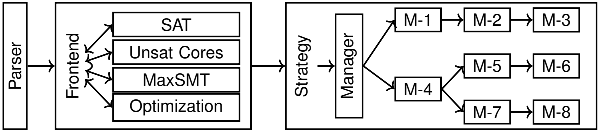
`https://github.com/ultimate-pa/smtinterpol`

`https://ultimate.informatik.uni-freiburg.de/smtinterpol`

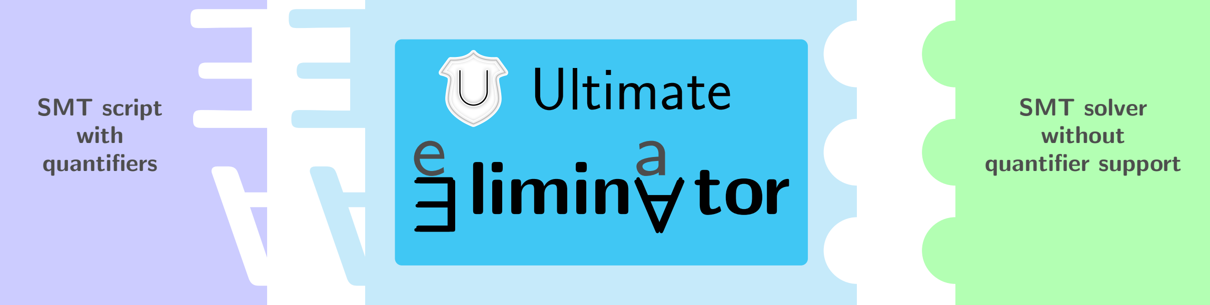
SMT-RAT 21.05


Parser
Frontend
SAT
Unsat Cores
MaxSMT
Optimization
Strategy
Manager
M-1
M-2
M-3
M-4
M-5
M-6
M-7
M-8

SMT-RAT 21.05



- `SMT-RAT` strategy
  - SAT solver: `minisat` adapted for less-lazy SMT solving
  - non-linear arithmetic: subtropical satisfiability, interval constraint propagation, virtual substitution and cylindrical algebraic decomposition
  - non-linear integer arithmetic: bit-blasting and branch&bound
- `SMT-RAT-MCSAT` strategy
  - MCSAT module based on `minisat`
  - Fourier-Motzkin, interval constraint propagation, virtual substitution, one-cell construction, NLSAT-style model based projections
  - novel variant of the one-cell construction (not published yet)



SMT script with quantifiers

Ultimate Eliminator

SMT solver without quantifier support

The wrapper that adds quantifier support to your SMT solver!

`https://ultimate.informatik.uni-freiburg.de/eliminator/`

2021 competition candidate:

ULTIMATEELIMINATOR+MATHSAT-5.6.6

Max Barth, Daniel Dietsch, Leonard Fichtner,
Matthias Heizmann, Andreas Podelski

Vampire 4.6

Reger, Suda, Voronkov, Kotelnikov, Kovacs, Riener, Rawson, Gleiss, Rath, Bhayat, Schoisswohl, Hozzova and Hajdu

https://vprover.github.io

Single query since 2016. Trying unsat-core and parallel/cloud.
SMT Logics: A, DT, LIA, LRA, NIA, NRA, UF (all with Q)
Uses a portfolio of strategies and wraps Z3 for ground reasoning.

General Approach is proof search using the Superposition and Resolution Calculus (also using finite model finding in UF)

Theory Reasoning:

- Theory axioms and Evaluation
- AVATAR modulo theories (ground splitting via Z3)
- Theory instantiation (using Z3)
- Induction on Datatypes

veriT

`www.veriT-solver.org`

Haniel Barbosa[♘], Pascal Fontaine[♔], *Hans-Jörg Schurr*[♙]

[♘]Department of Computer Science, Universidade Federal de Minas Gerais (UFMG)
[♙]CNRS, Inria, and the University of Lorraine, Nancy, France
[♔]Université de Liège, Belgium

- Automatically generated hybrid schedule
  1. optimal 24 s schedule
  2. optimal 1176 s schedule for the remaining problems
- Integration with Isabelle/HOL
  - Full proof reconstruction ships with Isabelle 2021
  - The proof format is coming of age as *Alethe*
- Unification-based preprocessing

Yices 2 in SMTCOMP 2021

Yices 2

- Supports linear and non-linear arithmetic, arrays, UF, bitvectors
- Supports incremental solving and unsat cores
- Includes two types of solvers: classic CDCL(T) + MCSAT
- https://github.com/SRI-CSL/yices2
- https://yices.csl.sri.com

New in 2021

- Quantifier reasoning: model-based quantifier instantiation + E-graph matching (thanks to Aman Goel)
- MCSAT extensions
  - Solving modulo a model
  - Interpolant for MCSAT-supported theories

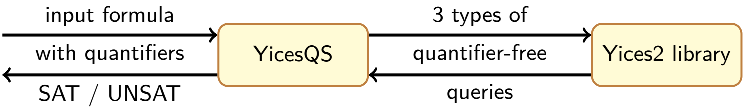
YicesQS, an extension of Yices2 for quantifiers (SMT-comp 2021)

Stéphane Graham-Lengrand https://github.com/disteph/yicesQS

YicesQS implements a 2-player game (∀ player vs ∃ player) playing on a quantified input formula F. Our generalization of counter-example-guided quantifier instantiation (CEGQI) produces a quantifier-free satisfiable under-approximation of F or a quantifier-free unsatisfiable over-approximation of F.

YicesQS entered logics *NRA* and *BV* (first entry of Yices in quantified logics),
& generally targets complete theories with procedures for answering 3 types of quantifier-free queries:

- *Satisfiability modulo assignment / modulo a model* (here relying on MCSAT)
- *Model generalization*
  (here using CAD projections for NRA, invertibility conditions for BV)
- *Model interpolation* (here again relying on MCSAT)



input formula
with quantifiers
SAT / UNSAT
YicesQS
3 types of
quantifier-free
queries
Yices2 library

YicesQS is written in OCaml, using Yices2 as a library via its OCaml bindings.
https://github.com/SRI-CSL/yices2
https://github.com/SRI-CSL/yices2_ocaml_bindings

State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences

YicesLS

Bohan Li, Shaowei Cai, Xindi Zhang

A Local Search SMT solver designed for QF-IDL with Wrapped Solver: Yices-2.6.2

Feature:

breaking through DPLL(T) framework, first local search solvers for SMT on theories with non-Boolean variables

Main components:

a local search framework, novel operators, scoring functions, and the operation selection heuristics

Division: QF-IDL

Track: Single Query & Model Validation

https://github.com/DouglasLee001/YicesLS

Non-Competitive Solvers

Submitted by Organisers

- z3-4.8.11
- MathSAT 5.6.6
- Par4 (for Parallel Track)
- Division winners from last year (32 Solvers)

Submitted by Participants

- Fixed solvers (Bitwuzla, COLIBRI, cvc5, iProver, OpenSMT, Vampire)

New in 2021

- model validation track extended
- manually checking and resolving disagreements
- multiple logics per division
- we will create an artifact

Model Validation Track

Solvers print model for each constant/uninterpreted function

```
sat
(model
  (define-fun x15 () Int 5)
  (define-fun x24 () x135 (as @1 x135))
  (define-fun x10 ((?X1 Int)) Int (ite (and (= ?X1 10)) 2 (ite...
```

- model validator based on pySMT
- all models of the competitors were accepted
- validator is still inefficient ($>$ 15 minutes)

Many thanks to Andrea Micheli

Checking Disagreements

- 111285 instances of 381683 have no status
- we checked disagreements between solvers
- 209 non-incremental instances and 65 incremental instances
- only 28 had known sat/unsat status
- tactics to resolve status:
  - similar disagreements on instances with known status
  - confirmed with fixed version of participants
  - analyzed model from solver claiming sat
  - majority vote
  - in rare cases manually analyzed instances
- solvers found unsound: (Bitwuzla, COLIBRI, cvc5, iProver, MathSAT, OpenSMT, Par4, UltimateEliminator+MathSAT, Vampire, Z3str4)

Divisions

Tracks are split into divisions.

- before 2021: Division = SMT-LIB logic
- 2021: 19 Divisions subdivided in 84 logics

This implied more changes:

- solver authors select supported logics

  ⇒ solvers may run on only a part of a division.
- fewer winners (1–5 winners per division)
- two solvers won a logic but not a division
  - YicesQS won NRA
  - YicesLS won QF_IDL

Statistics

Solver Size

- range from 930 kB to 196 MB (compressed)
- total 436 MB
- unsat core post-processor: 361 MB

Statistics

Solver Size

- range from 930 kB to 196 MB (compressed)
- total 436 MB
- unsat core post-processor: 361 MB

Job statistics

- Total Job Size: ∼ 85 GB
  - 35 GB YicesLS temporary files
  - 29 GB z3 models
- 1 371 593 pairs (+428 000)
- 16.3 CPU years (+6.4), 9.22 computer years on StarExec
- excludes result processors, StarExec overhead, glitches

Scoring

Computing scores:

- Single Query/Parallel/Cloud: number of solved instances
- Incremental: number of solved queries
- Unsat Core: number of top-level assertions removed
- Model Validation: number of solved instances with correct models

Error scores:

- All Tracks: given for sat reply for unsat instance, or vice versa
- Unsat Core: given if returned core is satisfiable.
- Model Validation: given if given model evaluates formula to false

Error scores are draconian.

Score and Ranking

In each track we collect different scores:

- Sequential score (SQ, UC, MV): all time limits apply to cpu time
- Parallel score (all): all time limits apply to wallclock time
- SAT score (SQ): parallel score for satisfiable instances
- UNSAT score (SQ): parallel score for unsatisfiable instances
- 24s (SQ): parallel score with time limit of 24s

Division ranking (for each score)

- For each division, one winner is declared

Two competition-wide rankings (for each score)

- Biggest lead: division winner with most score difference to second place
- Largest contribution: improvement each solver provided to a virtual best solver

Division Winners

Division Winners

Single Query

- Bitwuzla: QF_Bitvec, QF_Equality+Bitvec
- cvc5: Arith, Bitvec, Equality, Equality+LinearArith, Equality+MachineArith, Equality+NonLinearArith, FPArith, QF_Equality, QF_Equality+LinearArith, QF_Equality+NonLinearArith, QF_FPArith, QF_LinearIntArith, QF_LinearRealArith, QF_NonLinearIntArith, QF_NonLinearRealArith, QF_Strings
- iProver: Equality+NonLinearArith
- SMTInterpol: QF_Equality+LinearArith
- UltimateEliminator+MathSAT: Equality+NonLinearArith
- Vampire: Arith, Equality, Equality+NonLinearArith
- Yices2: QF_Bitvec, QF_Equality, QF_LinearIntArith, QF_LinearRealArith, QF_NonLinearIntArith

Division Winners

Single Query

- Bitwuzla: QF_Bitvec, QF_Equality+Bitvec
- cvc5: Arith, Bitvec, Equality, Equality+LinearArith, Equality+MachineArith, Equality+NonLinearArith, FPArith, QF_Equality, QF_Equality+LinearArith, QF_Equality+NonLinearArith, QF_FPArith, QF_LinearIntArith, QF_LinearRealArith, QF_NonLinearIntArith, QF_NonLinearRealArith, QF_Strings
- iProver: Equality+NonLinearArith
- SMTInterpol: QF_Equality+LinearArith
- UltimateEliminator+MathSAT: Equality+NonLinearArith
- Vampire: Arith, Equality, Equality+NonLinearArith
- Yices2: QF_Bitvec, QF_Equality, QF_LinearIntArith, QF_LinearRealArith, QF_NonLinearIntArith

Unsat Core

- Bitwuzla: QF_Bitvec, QF_Equality+Bitvec, QF_FPArith
- cvc5: Arith, Bitvec, Equality, Equality+LinearArith, Equality+MachineArith, Equality+NonLinearArith, FPArith, QF_Equality, QF_Equality+NonLinearArith, QF_LinearIntrArith, QF_NonLinearIntArith, QF_NonLinearRealArith
- Yices2: QF_Equality+LinearArith, QF_LinearRealArith

Division Winners

Incremental

- cvc5: Arith, Bitvec, Equality, Equality+LinearArith, Equality+NonLinearArith, FPArith, QF_Equality, QF_Equality+LinearArith, QF_FPArith
- OpenSMT: QF_LinearRealArith
- SMTInterpol: QF_Equality+NonLinearArith,QF_NonLinearIntArith
- STP: QF_Bitvec
- Yices2: QF_Equality+Bitvec,QF_LinearIntArith

Division Winners

Incremental

- cvc5: Arith, Bitvec, Equality, Equality+LinearArith, Equality+NonLinearArith, FPArith, QF_Equality, QF_Equality+LinearArith, QF_FPArith
- OpenSMT: QF_LinearRealArith
- SMTInterpol: QF_Equality+NonLinearArith,QF_NonLinearIntArith
- STP: QF_Bitvec
- Yices2: QF_Equality+Bitvec,QF_LinearIntArith

Model Validation (competitive only)

- Bitwuzla: QF_Bitvec
- cvc5: QF_LinearIntArith
- Yices2: QF_LinearRealArith

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_NIA)
par	iProver	(Eq+NA)	Vampire	(Eq)	cvc5	(Eq+LA)
sat	cvc5	(Eq+LA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+NA)	Yices2	(QF_NIA)	Vampire	(Eq)
24	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_LIA)

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_NIA)
par	iProver	(Eq+NA)	Vampire	(Eq)	cvc5	(Eq+LA)
sat	cvc5	(Eq+LA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+NA)	Yices2	(QF_NIA)	Vampire	(Eq)
24	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_LIA)
Incremental						
par	cvc5	(Eq)	Yices2	(QF_Eq+LA)	SMTInterpol	(QF_Eq+NA)

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_NIA)
par	iProver	(Eq+NA)	Vampire	(Eq)	cvc5	(Eq+LA)
sat	cvc5	(Eq+LA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+NA)	Yices2	(QF_NIA)	Vampire	(Eq)
24	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_LIA)
Incremental						
par	cvc5	(Eq)	Yices2	(QF_Eq+LA)	SMTInterpol	(QF_Eq+NA)
Unsat Core						
seq	cvc5	(Eq+LA)	Yices2	(QF_Eq+LA)		
par	cvc5	(Eq+LA)	Yices2	(QF_Eq+LA)		

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_NIA)
par	iProver	(Eq+NA)	Vampire	(Eq)	cvc5	(Eq+LA)
sat	cvc5	(Eq+LA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+NA)	Yices2	(QF_NIA)	Vampire	(Eq)
24	Vampire	(Eq+NA)	cvc5	(Eq+LA)	Yices2	(QF_LIA)
Incremental						
par	cvc5	(Eq)	Yices2	(QF_Eq+LA)	SMTInterpol	(QF_Eq+NA)
Unsat Core						
seq	cvc5	(Eq+LA)	Yices2	(QF_Eq+LA)		
par	cvc5	(Eq+LA)	Yices2	(QF_Eq+LA)		
Model Validation						
seq	cvc5	(QF_LIA)	Bitwuzla	(QF_BV)	Yices2	(QF_LRA)
par	cvc5	(QF_LIA)	Bitwuzla	(QF_BV)	Yices2	(QF_LRA)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	Vampire	(Eq+NA)	Bitwuzla	(QF_Eq+BV)
par	cvc5	(Eq+MA)	iProver	(Eq+NA)	Vampire	(Eq)
sat	cvc5	(Eq+MA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+MA)	Yices2	(QF_NIA)	Vampire	(Arith)
24	cvc5	(Eq+MA)	Yices2	(QF_LIA)	Vampire	(Eq+NA)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	Vampire	(Eq+NA)	Bitwuzla	(QF_Eq+BV)
par	cvc5	(Eq+MA)	iProver	(Eq+NA)	Vampire	(Eq)
sat	cvc5	(Eq+MA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+MA)	Yices2	(QF_NIA)	Vampire	(Arith)
24	cvc5	(Eq+MA)	Yices2	(QF_LIA)	Vampire	(Eq+NA)
Incremental						
par	cvc5	(BV)	SMTInterpol	(QF_NIA)	Yices2	(QF_LIA)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	Vampire	(Eq+NA)	Bitwuzla	(QF_Eq+BV)
par	cvc5	(Eq+MA)	iProver	(Eq+NA)	Vampire	(Eq)
sat	cvc5	(Eq+MA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+MA)	Yices2	(QF_NIA)	Vampire	(Arith)
24	cvc5	(Eq+MA)	Yices2	(QF_LIA)	Vampire	(Eq+NA)
Incremental						
par	cvc5	(BV)	SMTInterpol	(QF_NIA)	Yices2	(QF_LIA)
Unsat Core						
seq	cvc5	(QF_NRA)	Yices2	(QF_Eq+LA)	Bitwuzla	(QF_Eq+BV)
par	cvc5	(QF_NRA)	Yices2	(QF_Eq+LA)	Bitwuzla	(QF_Eq+BV)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	Vampire	(Eq+NA)	Bitwuzla	(QF_Eq+BV)
par	cvc5	(Eq+MA)	iProver	(Eq+NA)	Vampire	(Eq)
sat	cvc5	(Eq+MA)	UltimateElim	(Eq+NA)	Vampire	(Eq)
unsat	cvc5	(Eq+MA)	Yices2	(QF_NIA)	Vampire	(Arith)
24	cvc5	(Eq+MA)	Yices2	(QF_LIA)	Vampire	(Eq+NA)
Incremental						
par	cvc5	(BV)	SMTInterpol	(QF_NIA)	Yices2	(QF_LIA)
Unsat Core						
seq	cvc5	(QF_NRA)	Yices2	(QF_Eq+LA)	Bitwuzla	(QF_Eq+BV)
par	cvc5	(QF_NRA)	Yices2	(QF_Eq+LA)	Bitwuzla	(QF_Eq+BV)
Model Validation						
seq	cvc5	(QF_LIA)	Yices2	(QF_LRA)	Bitwuzla	(QF_BV)
par	cvc5	(QF_LIA)	Yices2	(QF_LRA)	Bitwuzla	(QF_BV)

Plans for SMT-COMP 2022

- Extend Model Validator to new logics?
- Run Model Validator on unknown benchmarks?
- New Proof Validation Track

Proof Validation Track

We plan to introduce a proof validation track

1. define solver independent proof format
    - probably resolution based
    - atoms are SMT-LIB formulas
    - fine-grain proofs
2. implement proof validator
3. run solvers on unsat and unknown benchmarks

Contact us, if you're interested in SMT-LIB proofs.

SMT-COMP organizing committee

Three people organize the SMT-COMP. In 2021:

- Haniel Barbosa
- Jochen Hoenicke
- Antti Hyvärinen

Antti's three-year term is ending now.

We need a successor for next year's competition. Contact us if you would like to volunteer!

Acknowledgements

- Andrea Micheli: Model Validator
- Clark Barrett, Pascal Fontaine, Aina Niemetz, Mathias Preiner, Hans-Jörg Schurr: SMT-LIB benchmarks
- Aaron Stump: StarExec support
- Mike Whalen, Jonathan Eidelman: Cloud/Parallel Track

starexec

powered by aws

Benchmark contributors

In 2021 new benchmarks were contributed by:

- Andrew V. Jones
- Nuno Lopes
- Pierre Bouvier
- Alexey Vishnyakov, Andrey Fedotov, Daniil Kuts, Alexander Novikov, Darya Parygina, Eli Kobrin, Vlada Logunova, Pavel Belecky, Shamil Kurmangaleev
- Hernán Ponce de León
- Anastasiia Izycheva, Eva Darulova
- Da Shen, Yuliya Lierler
- Maria A Schett, Julian Nagele
- Mathias Preiner, Aina Niemetz
- Clark Barrett
- Johannes Schoisswohl
- Jackson Melchert, Nestan Tsiskaridze
- Wei-Cheng Wu
- Matteo Favaro, Peter Garba
- Tjark Weber

Thanks

to all participants

Thanks

to all participants

and to you for listening