SMT-COMP 2022
17th International Satisfiability Modulo Theory Competition

Haniel Barbosa   Jochen Hoenicke   François Bobot

Universidade Federal de Minas Gerais, Brazil

Albert-Ludwigs-Universität Freiburg, Germany

CEA List, France

Aug 11, 2022

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

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
        - model validation
        - proof checking
    - 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

391 363 instances (+9680)
with 1 query each
in 81 logics (+2).

Incremental

43 285 instances (+1)
with 33 998 935 queries (+141)
in 39 logics.

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

391 363 instances (+9680)
with 1 query each
in 81 logics (+2).

Incremental

43 285 instances (+1)
with 33 998 935 queries (+141)
in 39 logics.

Selected Non-incremental

206 932 instances

Selected Incremental

22 300 instances

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

391 363 instances (+9680)
with 1 query each
in 81 logics (+2).

Incremental

43 285 instances (+1)
with 33 998 935 queries (+141)
in 39 logics.

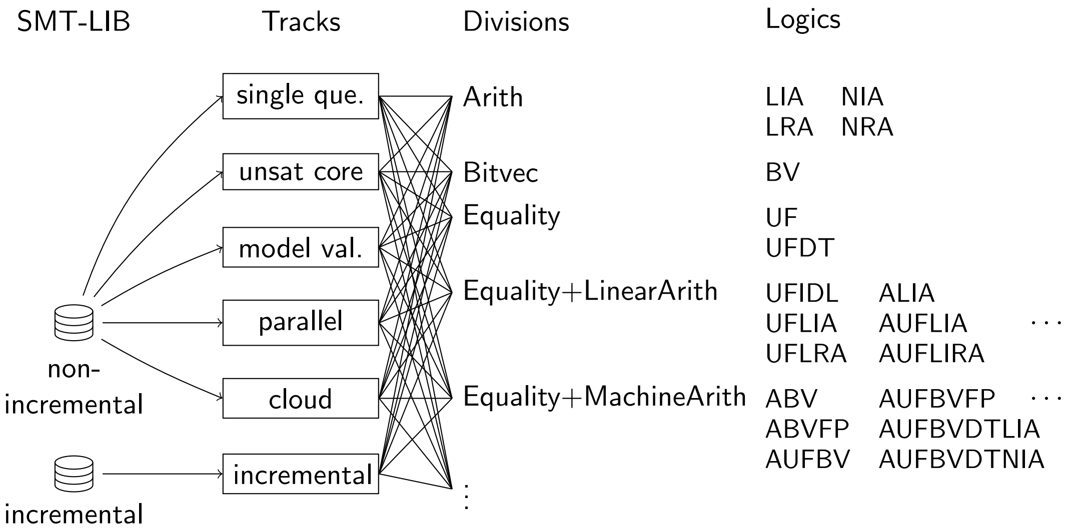
Selected Non-incremental

206 932 instances
We mistakenly ignored 9466 new benchmarks

Selected Incremental

22 300 instances

Competition Overview



SMT-COMP Tracks (traditional)

Single Query (SQ) 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 (experimental)

Model Validation

- Division with quantifier-free floating-point logics
- Model validation with Dolmen (thanks to Gillaume Bury and François Bobot)

Cloud and Parallel Track (sponsored by AWS, led by Mike Whalen)

- Solve a large problem over the cloud (or a big computer)
  - 100 machines, 1600 cores, 6400 GB of memory (cloud)
  - 64 cores, 256 GB of memory (parallel)
- Solver answers sat/unsat/unknown

SMT-COMP Tracks (experimental)

Model Validation

- Division with quantifier-free floating-point logics
- Model validation with Dolmen (thanks to Gillaume Bury and François Bobot)

Cloud and Parallel Track (sponsored by AWS, led by Mike Whalen)

- Solve a large problem over the cloud (or a big computer)
  - 100 machines, 1600 cores, 6400 GB of memory (cloud)
  - 64 cores, 256 GB of memory (parallel)
- Solver answers sat/unsat/unknown

Proof Exhibition Track

- Solver submitted together with a checker for unsatisfiability proofs
- No predefined format or checker
- No ranking
- Qualitative assessment

SMT-COMP Tracks (experimental)

Model Validation

- Division with quantifier-free floating-point logics
- Model validation with Dolmen (thanks to Gillaume Bury and François Bobot)

Cloud and Parallel Track (sponsored by AWS, led by Mike Whalen)

- Solve a large problem over the cloud (or a big computer)
  - 100 machines, 1600 cores, 6400 GB of memory (cloud)
  - 64 cores, 256 GB of memory (parallel)
- Solver answers sat/unsat/unknown

Proof Exhibition Track

- Solver submitted together with a checker for unsatisfiability proofs
- No predefined format or checker
- No ranking
- Qualitative assessment

This year the sat/unsat results from sound solvers in SQ were used to include benchmarks on the MV, UC and PE tracks.

Tracks, Solvers, Divisions, and Benchmarks

Teams: 21 (+3)

Track	Solvers	Divisions	Benchmarks
Single Query	22(+3)	19(+1)	93 945
Incremental	8(+1)	17(+2)	22 300
Unsat Core	6(-1)	18(+1)	57 245
Model Validation	8(+1)	7(+ 1 exp.)	32 766
Proof Exhibition	4	18 exp.	57 245
Parallel	4(+1)	14 exp.	400
Cloud	4(-1)	14 exp.	400

Number in parenthesis shows changes from 2021

Participants

SMT-COMP 2022 participants rely on multiple reasoning frameworks:

- CDCL(T)
- mcSAT
- saturation
- automata
- finite domain
- CP
- local search
- besides wrappers extending the scope of existing solvers

Six new solvers participated:

- NRA-LS (Liu et al.)
- OSTRICH (Chen et al.)
- Yices-ismt (Jia et al.)
- Z3++ (Cai et al.)
- solsmt (Reitwiessner and Soos)

Solver Presentation

Bitwuzla at the SMT-COMP'22

Aina Niemetz, Mathias Preiner

Tracks/Divisions

Single Query: QF_$\{A,BV,FP,FPLRA,UF\}^+$, $\{A,BV,FP,FPLRA,UF\}^+$
Incremental: QF_$\{A,BV,FP,FPLRA,UF\}^+$, $\{A,BV,FP,FPLRA,UF\}^+$
Unsat Core: QF_$\{A,BV,FP,FPLRA,UF\}^+$
Model Validation: QF_BV, QF_BVFP, QF_BVFPLRA, QF_FP, QF_FPLRA, QF_UFBV, QF_UFFP

Hightlights

- Quantifiers support for all supported theories (**new**)
- Sequential combination of bit-blasting and propagation based local search for QF_BV (SQ, MV)
- Uses CaDiCaL version 1.5.2 as SAT backend

`https://bitwuzla.github.io`

Stanford | Center for Automated Reasoning

COLIBRI Bruno Marre, François Bobot

CP solver:

- Domains (Intervals, modulo, FP and Bitvector representation)
- Local propagators for every constraints
- Constraint rewriting
- Global propagators (Simplex,...)
- Decision by domain splitting or value

- FP, Modulo arithmetic, reals, BV
- <=25s QF_FP

- Fuzzing with non-standard operators?
  - computer division
  - Encoding!

CVC5 at the SMT Competition 2022

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

Versatile and Industrial-Strength

- Support for all standardized SMT-LIB theories, user-friendly API
- Features beyond SMT solving (synthesis, proofs, ...)
- This year: Focus on robustness, features

Configurations

cvc5 entered all divisions in all tracks.

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

Proof Exhibition Track

- Two configurations: Internal proof checker and LFSC (cvc5-lfsc)
- Default
  - Uses the internal proof format: directed acyclic graphs of proof rule applications
  - Proofs are checked during construction
- cvc5-lfsc
  - Uses cvc5's LFSC back end
  - Proof and proof signatures in LFSC
  - Proof checker ensures that proof is well-formed w.r.t. the signature

Components

- Divide-and-conquer algorithm
- cvc5 as the base solver
- A splitter based on cvc5
- An MPI-based architecture for scheduling

Splitter

- Uses existing infrastructure and smarts of cvc5
- Intercepts calls to theory solvers after configurable number of checks
- Collects subset of literals $l_1, \ldots, l_m$ from the current decision trail
- Blocks $\neg(l_1 \wedge \ldots \wedge l_m)$
- Generates n-th cube $l_1 \wedge \ldots \wedge l_m \wedge \neg C_1 \wedge \ldots \wedge C_{n-1}$
- If less than two partitions are made, tries with other parameters

NRA-LS: Applying Local Search on Non-linear Real Arithmetic

Minghao Liu, Fuqi Jia, Rui Han, Yu Zhang, Pei Huang, Feifei Ma, Jian Zhang

- New entrant to SMT-COMP, participating in QF_NRA single query track this year

Highlights

- A local search solver that wraps `CVC5-1.0.0`
- To solve satisfiable instances with high-order polynomials

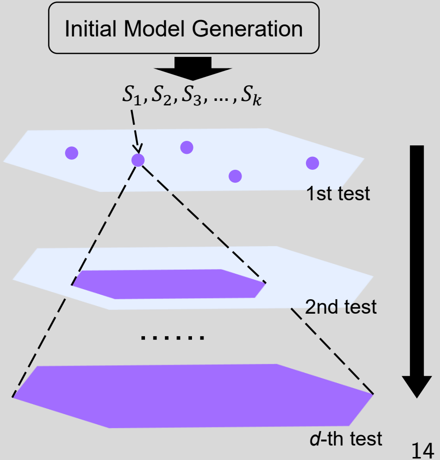
Approach

- Initial model generation
- Iterative satisfiability testing of sub-problems
- Time slots assignment

Resources

- System description and project are available at

  `https://github.com/minghao-liu/NRA-LS`



OpenSMT 2022

Efficient, interpolating solver for linear arithmetic, uninterpreted functions, and arrays

Written in C++17, used by the Horn solver Golem

New logics:
- Combination logics QF_UFLIA and QF_UFLRA (model-based theory combination)
- Arrays QF_AX (from SMTInterpol)

Performance improvements
- Arithmetic: Cuts-from-proofs, memory efficiency
- SAT solver: phase saving, glucose-style learned clauses management

Parallel and cloud solver (SMTS)
- Based on the partition-tree approach
- Search-space partitioning
- Clause sharing between arbitrary partitions
- Major rewrite from last year
- Two versions: portfolio and cube-and-conquer

Proof track
- Custom theory-specific trail format with drat-based SAT proofs

The OSTRICH String Solver

Straight-Line Formulas

- $S \quad ::= \quad x := f(\bar{x}) \mid \textbf{assert}(R(\bar{x})) \mid S; S$
- Soundness + completeness guarantee for SL formulas
- Complex string functions, including replaceall and transducers

Backwards-propagation

- Main algorithm for SL formulas
- $z = x \circ y \;;\; \textbf{assert}(z \in L)$
- $\textbf{assert}(x \in L_1) \;;\; \textbf{assert}(y \in L_2)$

Extensions

$$x0y = 11z$$

$$\begin{array}{ccc} |x| < |1| & |x| = |1| & |x| > |1| \\ {\color{red} 0y = 11z} & {\color{red} 0y = 1z} & x = 1x' \wedge x'0y = 1z \end{array}$$

SMTInterpol

Jochen Hoenicke, Tanja Schindler, . . .

UNI FREIBURG

Interpolating SMT solver

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

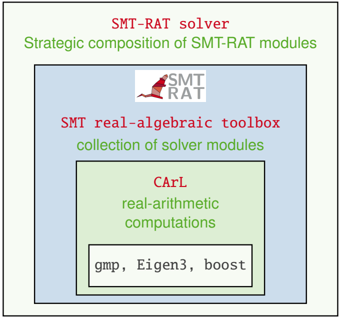
  and Data Types
- supports quantifiers
- produces models, proofs, and unsat cores
- computes sequence and tree interpolants

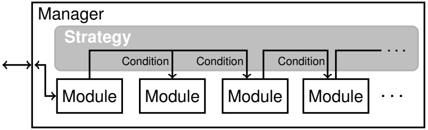
SMTInterpol at SMT-COMP 2022

- with detailed proof production
- proof check in single query/unsat core
- quantified data type logics

Try it in your browser:

`https://tinyurl.com/smtinterpol`





CArL library: basic arithmetic datatypes and computations [Sapientia'18, NFM'11, CAI'11]

Basic modules
- Preprocessing/simplifying modules
- SAT solver
- CNF converter

Non-algebraic decision procedures
- Bit-blasting
- Equalities and uninterpreted functions
- Bit-vectors
- Interval constraint propagation
- Pseudo-Boolean formulas

Algebraic decision procedures
- Gauß+Fourier-Motzkin
- Simplex [ISSAC'21]
- Linearization
- Gröbner bases [CAI'13]
- MCSAT (FM,VS,CAD) [2xSC^2'19]
- Subtropical satisfiability
- Virtual substitution [SC^2'17, FCT'11; PhD Corzilius]
- Cylindrical algebraic decomposition [JLAMP'21, SYNASC'21, SC^2'21, CADE-24, JSC'20, SC^2'20, SC^2'17; PhDs Loup, Kremer, Nalbach]
- Generalized branch-and-bound [CASC'16]
- Cube tests

SolSMT

tiny SMT solver inside the Solidity compiler

Christian Reitwiessner, Mate Soos
Ethereum Foundation

Solidity: Most widely used language for Ethereum smart contracts

not to be confused with **SolCMC**: interface to other SMT solvers for program verification

SolSMT: tiny integrated SMT solver used in optimizer

- remove redundant overflow checks, determine non-overlapping memory access, etc
- any bug in SMT solver can lead to bug in program
- needs to be fully deterministic and platform-independent for reproducibility
- implements QF_LRA using CDCL(T)
- inspired by MiniSat1.14 plus Dutertre-deMoura for LRA
- written in ~3k lines of C++

plan to add proof generation and checking for SAT and theory

https://github.com/ethereum/solidity branch=smtComp

STP at SMT-COMP22 QF_BV

- STP is an eager bit-blasing solver for QF_BV, and also QF_ABV but without extensionality.
- Our focus is on analysis at the bit-vector level, for example with unsigned-intervals, then applying sharing-aware rewrites. Because of this we go well on inefficiently-encoded problems.
- Development is hosted on Github.

veriT

Bruno Andreotti[♘], 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

- Focus on proofs
  - Participates in the proof exhibition
  - Uses the Alethe format
  - Proofs are checked with an high-performance proof checker written in Rust
  - Many small fixes and clarifications

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

Yices-ismt

Fuqi Jia, Rui Han, Minghao Liu, Cunjing Ge, Pei Huang, Feifei Ma, Jian Zhang.

- New entrant to SMT-COMP, participating in QF_NIA single query track this year.
- A wrapper solver: Yices2 + ismt.
- ismt: Preprocessor + "The Three Musketeers".
  - **Decider**: decide abstraction assignment → Interval Assignment.
  - **Searcher**: search in the given space → Bit-Blasting.
  - **Resolver**: resolve conflict from failure → Failed Interval Lemma ψ.
- Yices-ismt: Yices2(ϕ) → ismt (ϕ) → Yices ($\phi \wedge \psi$).
  - ψ rule out failed space.
- https://github.com/MRVAPOR/Yices-ismt

Dependencies
- *Yices 2.6.2*
- *Libpoly v0.1,11*
- *CaDiCal 1.5.2*

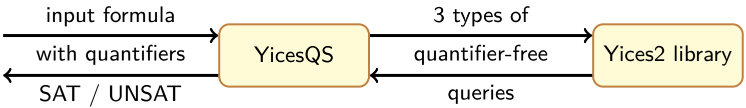
YicesQS, an extension of Yices2 for quantifiers (SMT-comp 2022)

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

YicesQS implements a 2-player game ($\forall$ player vs $\exists$ player) playing on a quantified input formula F. Our recursive 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.

2022: YicesQS entered logics *NRA*, *NIA*[NEW], *LRA*[NEW], *LIA*[NEW] and *BV*, & 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 invertibility conditions for BV, CAD projections + ϵ-terms for algebraic reals[NEW] for arithmetic)
- *Model interpolation* (here again relying on MCSAT, incl. CAD for arithmetic)



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

Z3++

https://z3-plus-plus.github.io/

Shaowei Cai, Bohan Li, Jinkun Lin, Zhonghan Wang, Bohua Zhan, Xindi Zhang, Mengyu Zhao
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China

QF_BV:

Off-the-shelf SAT solvers;
Word-level and bit-level rewriting rules

QF_LIA, QF_IDL, QF_NIA:

A local search solver dedicated for integer arithmetic logic

QF_NRA:

A feasible region consistency checker;
Sample-cell projection in NLSAT

Other participants

- Q3B
- SMT-RAT
- SMTS
- UltimateEliminatior+MathSAT
- Vampire
- Z3str4

Non-Competitive Solvers

Submitted by organisers

- z3-4.8.17
- MathSAT 5.6.8
- Best solvers, per division, from previous years (23 Solvers)

Submitted by participants

- Fixed solvers (OpenSMT, STP, Yices-ismt, Z3++,smtinterpol)

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, sequential score

- Bitwuzla: FPArith, QF_Bitvec, QF_Equality+Bitvec, QF_FPArith
- cvc5: Arith, Bitvec, Equality, Equality+LinearArith, Equality+MachineArith, Equality+NonLinearArith, QF_Datatypes, QF_Equality+NonLinearArith, QF_NonLinearIntArith, QF_NonLinearRealArith, QF_Strings
- OpenSMT: QF_LinearIntArith
- SMTInterpol: QF_Equality+LinearArith
- Yices2: QF_Equality, QF_LinearRealArith

Division Winners

Single Query, sequential score

- Bitwuzla: FPArith, QF_Bitvec, QF_Equality+Bitvec, QF_FPArith
- cvc5: Arith, Bitvec, Equality, Equality+LinearArith, Equality+MachineArith, Equality+NonLinearArith, QF_Datatypes, QF_Equality+NonLinearArith, QF_NonLinearIntArith, QF_NonLinearRealArith, QF_Strings
- OpenSMT: QF_LinearIntArith
- SMTInterpol: QF_Equality+LinearArith
- Yices2: QF_Equality, QF_LinearRealArith

Unsat Core

- cvc5: Arith, Bitvec, Equality+LinearArith, Equality+MachineArith,Equality+NonLinearArith,Equality(Seq), QF_Datatypes
- Vampire: Equality(Par)
- Bitwuzla: QF_EqualityBitvec, QF_FPArith
- Yices2: QF_Bitvec, QF_Equality+LinearArith, QF_Equality, QF_LinearIntArith, QF_LinearRealArith
- SMTInterpol: QF_Equality+NonLinearArith
- UltimateEliminator+MathSAT: FPArith

Division Winners

Incremental

- cvc5: Arith, Bitvec, Equality+LinearArith,EqualityNonLinearArith,Equality
- UltimateEliminator+MathSAT: Equality+MachineArith
- Bitwuzla: FPArith, QF_FPArith
- Yices2: QF_Bitvec, QF_Equality+Bitvec, QF_Equality, QF_LinearIntArith
- SMTInterpol: QF_Equality+LinearArith, QF_Equality+NonLinearArith, QF_NonLinearIntArith
- OpenSMT: QF_LinearRealArith

Division Winners

Incremental

- cvc5: Arith, Bitvec, Equality+LinearArith,EqualityNonLinearArith,Equality
- UltimateEliminator+MathSAT: Equality+MachineArith
- Bitwuzla: FPArith, QF_FPArith
- Yices2: QF_Bitvec, QF_Equality+Bitvec, QF_Equality, QF_LinearIntArith
- SMTInterpol: QF_Equality+LinearArith, QF_Equality+NonLinearArith, QF_NonLinearIntArith
- OpenSMT: QF_LinearRealArith

Model Validation (competitive only)

- Bitwuzla: QF_Bitvec, QF_Equality+Bitvec,
- smtinterpol: QF_Equality+LinearArith
- Yices2: QF_Equality
- Z3++: QF_LinearIntArith
- OpenSMT: QF_LinearRealArith
- experimental: QF_FPArith : Bitwuzla, cvc5

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
par	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
sat	cvc5	(Eq+LA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
unsat	cvc5	(Eq+MA)	Z3++	(QF_NonLIA)	OSTRICH	(QF_Strings)
24	Vampire	(Eq+NA)	Vampire	(Equality)	Yices2	(QF_LinIA)

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
par	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
sat	cvc5	(Eq+LA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
unsat	cvc5	(Eq+MA)	Z3++	(QF_NonLIA)	OSTRICH	(QF_Strings)
24	Vampire	(Eq+NA)	Vampire	(Equality)	Yices2	(QF_LinIA)
Incremental						
par	cvc5	(Eq+NA)	Yices2	(QF_Eq+LA)	SMTInterpol	(QF_Eq+NA)

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
par	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
sat	cvc5	(Eq+LA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
unsat	cvc5	(Eq+MA)	Z3++	(QF_NonLIA)	OSTRICH	(QF_Strings)
24	Vampire	(Eq+NA)	Vampire	(Equality)	Yices2	(QF_LinIA)
Incremental						
par	cvc5	(Eq+NA)	Yices2	(QF_Eq+LA)	SMTInterpol	(QF_Eq+NA)
Unsat Core						
seq	cvc5	(Eq+LA)	Bitwuzla	(QF_Eq+Bitvec)		
par	cvc5	(Eq+NA)	Bitwuzla	(QF_Eq+Bitvec)		

Largest contribution

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
par	cvc5	(Eq+MA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
sat	cvc5	(Eq+LA)	YicesQS	(Arith)	Bitwuzla	(FPArith)
unsat	cvc5	(Eq+MA)	Z3++	(QF_NonLIA)	OSTRICH	(QF_Strings)
24	Vampire	(Eq+NA)	Vampire	(Equality)	Yices2	(QF_LinIA)
Incremental						
par	cvc5	(Eq+NA)	Yices2	(QF_Eq+LA)	SMTInterpol	(QF_Eq+NA)
Unsat Core						
seq	cvc5	(Eq+LA)	Bitwuzla	(QF_Eq+Bitvec)		
par	cvc5	(Eq+NA)	Bitwuzla	(QF_Eq+Bitvec)		
Model Validation						
seq	Z3++	(QF_LinIA)	Bitwuzla	(QF_Bitvec)	smtinterpol	(QF_Eq+LIA)
par	Z3++	(QF_LinIA)	Bitwuzla	(QF_Bitvec)	smtinterpol	(QF_Eq+LIA)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
par	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
sat	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Z3++	(QF_NonRA)
unsat	cvc5	(QF_DT)	Z3++	(QF_NonIA)	Bitwuzla	(FPArith)
24	cvc5	(Eq+MA)	smtinterpol	(QF_DT)	Vampire	(Equality)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
par	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
sat	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Z3++	(QF_NonRA)
unsat	cvc5	(QF_DT)	Z3++	(QF_NonIA)	Bitwuzla	(FPArith)
24	cvc5	(Eq+MA)	smtinterpol	(QF_DT)	Vampire	(Equality)
Incremental						
par	SMTInterpol	(QF_NIA)	Yices2	(QF_LinIA)	cvc5	(Eq+NonLA)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
par	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
sat	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Z3++	(QF_NonRA)
unsat	cvc5	(QF_DT)	Z3++	(QF_NonIA)	Bitwuzla	(FPArith)
24	cvc5	(Eq+MA)	smtinterpol	(QF_DT)	Vampire	(Equality)
Incremental						
par	SMTInterpol	(QF_NIA)	Yices2	(QF_LinIA)	cvc5	(Eq+NonLA)
Unsat Core						
seq	cvc5	(Eq+MA)	Yices2	(QF_Eq+LA)	smtinterpol	(QF_Eq+NA)
par	cvc5	(Eq+MA)	Yices2	(QF_Eq+LA)	smtinterpol	(QF_Eq+NA)

Biggest Lead

	1st Place		2nd Place		3rd Place	
Single Query						
seq	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
par	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Yices2	(QF_LinRA)
sat	cvc5	(QF_DT)	Bitwuzla	(FPArith)	Z3++	(QF_NonRA)
unsat	cvc5	(QF_DT)	Z3++	(QF_NonIA)	Bitwuzla	(FPArith)
24	cvc5	(Eq+MA)	smtinterpol	(QF_DT)	Vampire	(Equality)
Incremental						
par	SMTInterpol	(QF_NIA)	Yices2	(QF_LinIA)	cvc5	(Eq+NonLA)
Unsat Core						
seq	cvc5	(Eq+MA)	Yices2	(QF_Eq+LA)	smtinterpol	(QF_Eq+NA)
par	cvc5	(Eq+MA)	Yices2	(QF_Eq+LA)	smtinterpol	(QF_Eq+NA)
Model Validation						
seq	Z3++	(QF_LinIA)	smtinterpol	(QF_LinA)	OpenSMT	(QF_LinRA)
par	Z3++	(QF_LinIA)	smtinterpol	(QF_LinA)	OpenSMT	(QF_LinRA)

Checking Disagreements

- 111 451 instances of 391 363 have no status
- Only 18 benchmarks with disagreements (AUFDTLIA, QF_NIA)
- We manually resolved the disagreements
  - Authors confirmed solver unsoundness
  - Sound solvers agreed on result
- We had only three solvers with soundness issues: SMTIinterpol, Z3++, yices-ismt
  - Down from 10 last year

Checking Disagreements

- 111 451 instances of 391 363 have no status
- Only 18 benchmarks with disagreements (AUFDTLIA, QF_NIA)
- We manually resolved the disagreements
  - Authors confirmed solver unsoundness
  - Sound solvers agreed on result
- We had only three solvers with soundness issues: SMTIinterpol, Z3++, yices-ismt
  - Down from 10 last year (but note that this may be because of our issue with ignoring most of the new benchmarks in the selection)

Plans for SMT-COMP 2022

- Move to Dolmen as single model validator
- Dolmen worked well in QF_FP (minor issue with empty output, simple to fix)
- pysmt timed out in two cases (locally with more time one of them succeeded)
- pysmt could not understand the model in 9 cases
- Dolmen is easy to extend to new theories
- Easy to integrate with correct-by-construction code extracted from Coq
- Should be more efficient for handling function call

Plans for SMT-COMP 2022

- Proof *validation* track
- Hopefully proof exhibition this year will help
- We have to analyze the data still
  - Job only finished last week (took 18 days to run)
  - 830gb

SMT-COMP organizing committee

Three people organize the SMT-COMP. In 2022:

- Haniel Barbosa
- Jochen Hoenicke
- François Bobot

Both Haniel and Jochen have been organizers for three-years. One of them should be replaced.

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

Acknowledgements

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

Benchmark contributors

In 2022 new benchmarks were contributed by:

- Alex Coffin
- Alex Ozdemir
- Ali Uncu, James Davenport and Matthew England
- Bohan Li
- Elizabeth Polgreen
- Fuqi Jia
- Johann-Tobias Aaron and Raphael Schäg
- Matthew England and Miguel Del Rio Almajano
- Nicolas Amat
- Yannick Moy
- Yoni Zohar

Thanks

to all participants

Thanks

to all participants

and to you for listening