# SMT-COMP 2019

## 14th International Satisfiability Modulo Theories Competition

---

Liana Hadarean  Antti Hyvärinen  Aina Niemetz  Giles Reger

SMT Workshop, July 7-8, 2019, Lisbon, Portugal

# SMT-COMP

⟶ annual competition for SMT solvers

⟶ on (a selection of) benchmarks from SMT-LIB

- first held in 2005
- 2013: evaluation instead of competition
- since 2014: hosted by StarExec

**Goals**

- encourage scientific advances in SMT solvers
- stimulate community to explore shared challenges
- promote tools and their usage
- engage and include new members of the community
- support the SMT-LIB project to promote and develop the SMT-LIB format and collect relevant benchmarks

# Participants

**SMT solver:** determine (un)satisfiability of benchmarks from SMT-LIB

- **SMT Solvers** in the 'classical' sense
- **Wrapper Tools:** call one or more other SMT solvers
- **Derived Tools:** based on and extends another SMT solver
- **Automated Theorem Provers** (e.g., Vampire)

$\longrightarrow$ New system description mandatory

$\longrightarrow$ New naming convention for derived tools

# Tracks

- **Single Query Track** (previously: Main Track)
  - one **single** `check-sat` command, no `push`/`pop` commands
  - New **remove** benchmarks solved by all solvers in 2018 in $\leq 1s$
  - New **selection** of benchmarks
  - New **time limit:** 2400s (40 min)

- **Incremental Track** (previously: Application Track)
  - **multiple** `check-sat` and `push`/`pop` commands
  - solvers are executed on benchmarks via **trace executor**
  - New **selection** of benchmarks
  - New **keep** benchmarks with first `check-sat` status `unknown`
  - New execute solver **beyond** first status `unknown` `check-sat` call
  - **time limit:** 2400s (40 min)

# Tracks

- **Unsat Core Track**
  - one **single** `check-sat` command, **multiple** `assert` commands
  - benchmarks with **status** `unsat`
  - extract **unsat core** as set of top-level assertions
  - New **remove** benchmarks with a single `assert` command
  - New **selection** of benchmarks
  - **time limit:** 2400s (40 min)

# Tracks

- New: **Challenge Track**
  - two subtracks: non-incremental and incremental
  - benchmarks that were **nominated** by their submitters for this track
  - **time limit:** 43200s (12 hours)

- New: **Model Validation Track** (experimental)
  - one **single** `check-sat` command,
  - **selection** of benchmarks with **status** `sat`
  - produce full, correct, well-formed **model** in SMT-LIB format
  - **only** for division QF_BV
  - **time limit:** 2400s (40 min)

# Divisions

⟶ **Tracks** are split into divisions

⟶ **Divisions** correspond to logics in SMT-LIB

- solvers are submitted to divisions in a track
- **winners** are declared
  - per division and track
  - with respect to different scoring schemes per track
- New do not run **non-competitive** divisions

# Benchmark Selection

- 2015-2018: **all** eligible benchmarks in a division
  - $\longrightarrow$ results more predictable
  - $\longrightarrow$ more of an evaluation than a competition
  - $\longrightarrow$ Main Track (2018):
    - 78% solved by all participating solvers
    - 71% solved in $\leq 1s$
    - in 7 out of 46 divisions $>$ 99% solved by all solvers

- New alternative benchmark selection
  - **remove** easy/uninteresting benchmarks
    - SQ: all benchmarks solved by all solvers in $\leq 1s$ in 2018
    - UC: all benchmarks with only a single assertion
  - **cap** number of instances in a division
    - $n \leq 300$: all instances
    - $300 < n \leq 600$: 300 instances
    - $n > 600$: 50% of the logic
  - guarantee inclusion of **new** benchmarks (at least one per family)
  - select benchmarks randomly using a uniform distribution

# Single Query and Unsat Core Track Scoring

- 2016-2018: **weighted** with respect to benchmark family size
  - ⟶ **goal:** de-emphasize large benchmark families
  - ⟶ fairly complicated, not necessarily intuitive
  - ⟶ complicates comparing paper and competition results

- **Competition report** for 2015-2018 (under review):
  - ⟶ families **no significant impact** on the (weighted) scores
    - problems with scoring script (2016-2018)
    - incorrect interpretation of benchmark family
    - **after fix:** only one change (2017 AUFNIRA: CVC4 over Vampire)
  - ⟶ **unweighted:** only 7 out of 139 winners in 2016-2018 change

- New drop weighted scoring, use **unweighted** scheme from 2015

# Scores

- **Single Query, Challenge (non-incremental):**
  number of correctly solved instances
- **Incremental, Challenge (incremental):**
  number of correctly solved `check-sat` calls
- **Unsat Core:**
  reduction in terms of top-level assertions
- **Model Validation:**
  number of correctly solved instances with validated models

# Scores

- **sequential score** (SQ, CHSQ, UC, MV)
  time limit applied to CPU time
- **parallel score** (all)
  time limit applied to wall-clock time
- New **sat score** (SQ, CHSQ)
  parallel score for satisfiable instances
- New **unsat score** (SQ, CHSQ)
  parallel score for unsatisfiable instances
- New **24s score** (SQ, CHSQ)
  parallel score for time limit of 24s

# Competition-Wide Recognitions

- 2014-2018:
  - **competition-wide scores** as weighted sum of division scores
  - emphasis on number of entered divisions

- New replace with **two new competition-wide rankings**

  $\longrightarrow$ focus on measures that make sense to compare between divisions

  $\longrightarrow$ for all scores in a track

- **biggest lead**
  - in terms of **score** over the solver in the second place
  - **tie:** ranked by biggest lead in CPU/wall-clock time

- **largest contribution**
  - ranked by contribution to **virtual best solver** in terms of **score**
  - **tie:** ranked by largest contribution in terms of CPU/wall-clock time

# Competition Overview

| | Solvers | | Divisions | | Benchmarks | | |
|---|---|---|---|---|---|---|---|
| Track | Total | C/NC | Total | C/NC/Exp | C | Selected | Total |
| SQ | 51 (+27) | 37/14 | 57 (+7) | 49/6/2 | 64156 | 89817 | 327041 |
| Inc | 22 (+16) | 14/8 | 29 (+8) | 24/5/0 | 6835 | 7567 | 14030 |
| CHSQ | 21 (+21) | 15/6 | 3 (+3) | 3/0/0 | 29 | 29 | 29 |
| CHInc | 12 (+12) | 7/5 | 3 (+3) | 3/0/0 | 22 | 22 | 22 |
| UC | 14 (+9) | 8/6 | 38 (-6) | 33/5/0 | 29808 | 44341 | 136012 |
| MV | 10 (+10) | 10/0 | 1 (+1) | 1/0/0 | 7191 | 7191 | 14382 |

C . . . Competitive NC . . . Non-Competitive Exp . . . Experimental

**Teams**: 23 (+6)

**StarExec Stats**: 21.4 years CPU time; 1,022,802 job pairs

# Non-Competitive Solvers

**Total:** **14** (SQ), **8** (Inc), **6** (CHSQ), **5** (CHINC), **6** (UC)

- submitted by **organizers**
  - Z3 4.8.4
  - best solvers 2018 (SQ: 9, Inc: 5, CHSQ: 3, CHINC: 3, UC: 5)
- submitted by **participants**
  - 2 derived tools (Boolector-ReasonLS, CVC4-SymBreak)
  - 3 fixed solver versions (1 x CVC4, 2 x STP)

# Solver Presentations

Boolector, COLIBRI, CVC4, MathSAT, OpenSMT, SPASS-SATT,
Vampire, VeriT Yices

# Boolector at the SMT-COMP'19

Aina Niemetz, Mathias Preiner, Armin Biere

## Tracks/Divisions

**Single Query:** BV, QF_ABV, QF_AUFBV, QF_BV, QF_UFBV
**Incremental:** QF_ABV, QF_AUFBV, QF_BV, QF_UFBV
**Challenge:** QF_ABV, QF_AUFBV, QF_BV
**Model Validation:** QF_BV

## Improvements

- Incremental improvements to avoid redundant clauses in SAT solver
- SAT race 2019 version of CaDiCaL for all logics and tracks
  - ▶ now default SAT engine for incremental and non-incremental
- GMP for faster BV implementation (improving LS engines)
- CryptoMiniSat support

## Configurations

- *Boolector*: Combination of prop.-based local search + bit-blasting
  - ▶ Local search for QF_BV and BV
- *Poolector*: Portfolio of four parallel (non-incremental) Boolector configurations:
  - ▶ CaDiCaL, Lingeling, CryptoMiniSat, and SLS (for QF_BV)

`https://boolector.github.io`


# COLIBRI

CEA LIST | Bruno Marre, F.Bobot, Zakaria Chihani


# COLIBRI(2019)

**QF_FP**: Since last year small bug fix and improvements

- Forgot to participate to QF_FPLRA
- Focused on 25s

# CVC4 at the SMT Competition 2019

Clark Barrett, Haniel Barbosa, Martin Brain, Tim King, Makai Mann, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Cesare Tinelli, Yoni Zohar

## Divisions

This year's configuration of CVC4 enters all divisions in all tracks.

## New Features/Improvements

- Eager bit-blasting solver:
  - New version of CaDiCaL with support for incremental solving
  - Support for incremental eager bit-blasting with CaDiCaL as backend (`QF_BV`)
  - Not using ABC anymore
  - Fewer consistency lemmas in Ackermannization preprocessing pass
- String solver: better heuristics, more aggressive rewriting, more efficient reductions of extended operators
- Floating-point solver: new version of SymFPU (primarily bug fixes)

## Configurations

- Industry Challenge Track and Model-Validation Track: Same configurations as Single Query Track
- Unsat-Core Track: Fixed last year's configuration that had errors on `QF_UFBV`

# OpenSMT

A relatively small DPLL(T)-based SMT Solver

Developed at University of Lugano, Switzerland

Supports QF_UF, QF_LRA, and to some extent QF_BV

Lookahead-Based SMT

Theory refinement

Interpolation (esp. in LRA)

Integration to model checkers HiFrog and Sally

2018-2019: Performance improvements, better defined development process

Available from http://verify.inf.usi.ch/opensmt

http://www.spass-prover.org/spass-satt


## Developers:

Martin Bromberger, Mathias Fleury, Simon Schwarz, Christoph Weidenbach

## Ground Linear Arithmetic Solver:

- newest tool in the SPASS Workbench
- combines our theory solver SPASS-IQ and our unnamed SAT solver
- supports QF_LIA, QF_LRA, (and QF_LIRA)
- complete but efficient theory solver [IJCAR2018]
- uses fast cube tests [IJCAR2016, FMSD2017]
- SAT decisions based on theory solver information
- uses many more well-known techniques for linear arithmetic

http://www.spass-prover.org/spass-satt

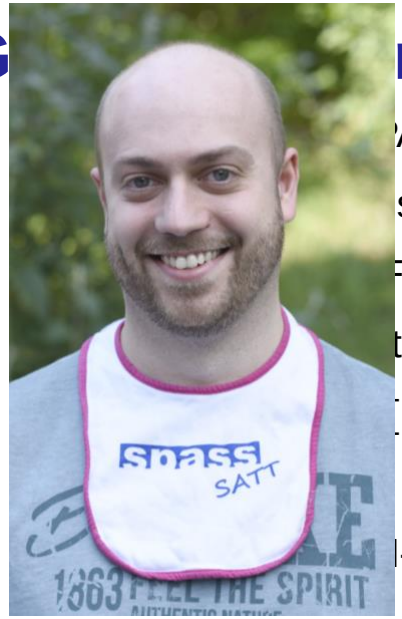
# SPASS SATT

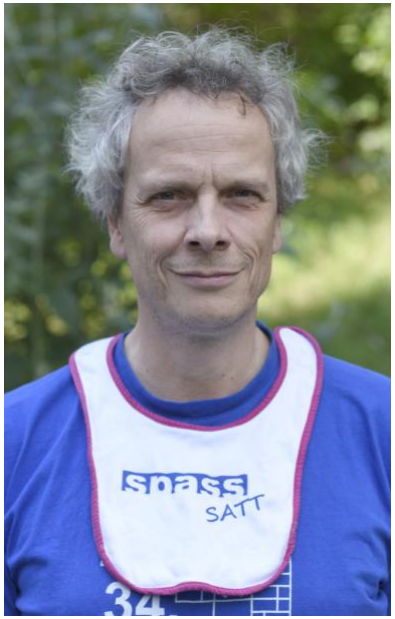
## Developers:

Martin Bromberger, Mathias Fleury, Simon Schwarz, Christoph Weidenbach

## G r o

- PAS
- so r u
- F_L
- t th 8]
- IJC
- or tio
- l-kr ea





# Vampire 4.4-SMT

Giles Reger[1], Martin Suda[2], Andrei Voronkov[15], Evgeny Kotelnikov[3], Simon Robillard[3], Laura Kovács[4], and Martin Riener[1]

SMT Comp 2019

July 8, Lisbon, Portugal

[1]University of Manchester, Manchester, UK

[2]Czech Technical University in Prague, Czech Republic

[3]Chalmers University of Technology, Gothenburg, Sweden

[4]Institute for Information Systems, Vienna University of Technology, Austria

[5]Easychair

# Features

- Superposition based First Order Resolution Prover
- Finite Model Finding
- Inst-gen
- Redundancy elimination
- Splitting via AVATAR
- Sine axiom selection
- Induction
- CASC since 1999

# SMT Related Features

- SMT Logics: A, DT, LIA, LRA, NIA, NRA, UF
- Single Queries
- SMT since 2016

- Theory axioms
- AVATAR modulo theories (ground splitting via Z3)
- Unification with abstraction
- Theory instantiation

# Available online

https://vprover.github.io

https://github.com/vprover/vampire

# veriT

http://www.veriT-solver.org

Haniel Barbosa, David Déharbe,
Daniel El Ouraoui, Pascal Fontaine, and Hans-Jörg Schurr

Loria, INRIA, Université de Lorraine (France), ClearSy

What is new (not yet in the SMT-COMP version):

- cleaning, efficiency improvements
- $\lambda$-free Higher-order
- improved quantifier handling (ML, instantiation, superposition)
- better proofs

Goals:

- clean, small SMT for UF(N|L)IRA with quantifiers and proofs
- for verification platforms B, TLA+

# Yices 2 in SMTCOMP 2019

## 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) + MC-SAT
- https://github.com/SRI-CSL/yices2
- https://yices.csl.sri.com

## New in 2019

- Models in SMT-LIB2 format
- Improved bitblasting-based solver
- MC-SAT for bitvectors
- Thread-safe

# Bitblasting-Based Solver

## Bitblasting in Yices 2

- implemented in 2009 + extended with many simplifications and rewriting rules
- uses a relatively simple CDCL solver (no preprocessing, simple heuristics)
- incremental

## New developments

- support for third-party SAT-solvers (as long as provide the right API)
- currently supported:
  - CaDiCal (Armin Biere)
  - CryptoMiniSAT (Mate Soos)
- We also have developed a new, more performant CDCL-based SAT solver to replace the default

# MC-SAT for Bitvectors

## MC-SAT

- alternative to CDCL(T)
- in Yices: used primarily for non-linear arithmetic (+ UF)

## New developments

- extended MC-SAT to QF_BV: our goal is to support word-level reasoning
  - BDDs for representing sets of values
  - specialized reasoning components for two QF_BV fragments:
    - concatenation + extraction + equalities
    - (simple) linear-arithmetic
  - unsat cores + bit-blasting outside these framents
- still work in progress, very fast on some examples

# MathSAT5 (Nonlinear)
## at the SMT Competition 2019

Ahmed Irfan[1], Alessandro Cimatti[2],
Alberto Griggio[2], Roberto Sebastiani[3]

[1] Stanford University, USA
[2] Fondazione Bruno Kessler, Italy
[3] University of Trento, Italy

– SMT Competition 2019, Lisbon, Portugal –

# MathSAT5 (Nonlinear)

## MathSAT5, a DPLL(T) solver

- supports most SMT-LIB theories + functionalities (e.g unsat cores, interpolation, ALLSMT)
- supports nonlinear arithmetic on reals & integers + transcendental functions (sin(), exp())
  - based on incremental linearization: abstraction/refinement to SMT(QF_UFLA)
    - multiplication, sin() and exp() modeled by uninterpreted functions
    - incrementally axiomatized on demand by linear constraints

## Participation and Configurations

- Categories:
  - Single query track: QF_ANIA, QF_AUFNIA, QF_NIA, QF_NIRA, QF_NRA, QF_UFNIA, QF_UFNRA.
  - Incremental track: QF_ANIA, QF_AUFBVNIA, QF_NIA, QF_UFNIA.
  - Unsat Core track: QF_ANIA, QF_AUFNIA, QF_NIA, QF_NIRA, QF_NRA, QF_UFNIA, QF_UFNRA.
- Submitted versions:
  - MathSAT default: public release version 5.5.4 +minor fixes, $\approx$ as described in our SAT'18 paper
  - MathSAT-na-ext: MathSAT default
    - + use of lazier strategy for the instantiation of linearization lemmas;
    - + try to minimize the Boolean assignment that are given to theory solvers;
    - + use bi-implication tangent lemmas:
    - + linearization lemmas learnt only temporarily

# Results

SMT-COMP

The International Satisfiability Modulo Theories (SMT) Competition.

GitHub

Home
Introduction
Rules
Benchmarks
Tools
Specs
Participants
Results
Benchmark submission
Solver submission
Previous

SMT-LIB

## SMT-COMP 2019 Results

### Competition-Wide Recognitions

**Largest Contribution Ranking**

- Challenge Track (incremental)
- Challenge Track (non-incremental)
- Incremental Track
- Model Validation Track (experimental)
- Single Query Track
- Unsat Core Track

**Biggest Lead Ranking**

- Challenge Track (incremental)
- Challenge Track (non-incremental)
- Incremental Track
- Model Validation Track (experimental)
- Single Query Track
- Unsat Core Track

### Tracks Summary

- Challenge Track (incremental)
- Challenge Track (non-incremental)
- Incremental Track
- Model Validation Track (experimental)
- Single Query Track
- Unsat Core Track

### Divisions

- **ABVFP**
  - Incremental Track
  - Single Query Track

# Trophies

# Trophies: Largest Contribution

| **Single Query** | **1st Place** | **2nd Place** |
|---|---|---|
| seq | **CVC4** (QF_NIA) | **Vampire** (UF) |
| par | **CVC4** (QF_NIA) | **Vampire** (UF) |
| sat | **Par4** (AUFLIRA) | **SMTInterpol** (UFLIA) |
| unsat | **Par4** (UFNIA) | **Vampire** (UF) |
| 24s | **Vampire** (UF) | **Par4** (UFNIA) |
| **Incremental** | **1st Place** | **2nd Place** |
| par | **CVC4** (UFLRA) | **Boolector** (QF_BV) |
| **Unsat Core** | **1st Place** | **2nd Place** |
| seq | **CVC4** (AUFLIRA) | **MathSAT** (QF_NIA) |
| par | **CVC4** (AUFLIRA) | **MathSAT** (QF_NIA) |
| **Challenge** | **1st Place** | **2nd Place** |
| par | **Yices** (QF_AUFBV) | **Boolector** (QF_ABV) |

# Trophies: Biggest Lead

| **Single Query** | **1st Place** | **2nd Place** |
|---|---|---|
| seq | **CVC4** (FP) | **Par4** (UFBV) |
| par | **CVC4** (FP) | **Par4** (UFBV) |
| sat | **CVC4** (AUFDTLIA) | **Par4** (AUFLIRA) |
| unsat | **CVC4** (BVFP) | **SMT-RAT** (QF_NIRA) |
| 24s | **CVC4** (BVFP) | **Par4** (UFBV) |
| **Incremental** | **1st Place** | **2nd Place** |
| par | **CVC4** (ANIA) | **Yices** (QF_AUFBV) |
| **Unsat Core** | **1st Place** | **2nd Place** |
| seq | **CVC4** (UFLIA) | **Yices** (QF_AX) |
| par | **CVC4** (UFLIA) | **Yices** (QF_AX) |
| **Challenge** | **1st Place** | **2nd Place** |
| par | **Yices** (QF_AUFBV) | **Boolector** (QF_ABV) |

# Discussion

- **time limit**
  - increased back to 2400s (from 1200s 2017-2018) in SQ track
  - only $-3953$ instances if cut off at **1200s** (sequential score)
  - $\sim 50\%$ of the timeouts in **quantified** divisions

$\longrightarrow$ run selected challenging benchmarks in the **challenge track**

$\longrightarrow$ **decrease** time limit (maybe even further) for **other tracks**

$\longrightarrow$ shorter time limit for **quantified divisions**?
(typically: solved within short time or "never")

# Discussion

- **divisions**
  - size of competitions is getting out of hand
  - this year we didn't run non-competitive divisions

  $\longrightarrow$ **don't run** if less than 3? 4? competitive participants?

- **parallel score**
  - StarExec only offers 4 cores per job
  - not interesting for real parallelism

  $\longrightarrow$ **future plans:** dedicated **parallel track**

  $\longrightarrow$ would require to move away from StarExec

# Discussion

- **portfolio wrapper tools**
  - wrapper tools allowed to participate without restrictions
  - problems with portfolio (not author of the wrapped solvers)
    - → win with simple script and work of other teams
    - → negative/unfair impact on competition-wide rankings
    - → progress of non-portfolio tools harder to distinguish
  - disallowing wrapper tools entirely is problematic (example: Vampire)

⟶ **disallow** portfolio with wrapped solvers from other teams?

⟶ only allow **non-competitive** submission?

⟶ at least **exclude** them from competition-wide recognitions

⟶ **similar issues** with SATzilla-style systems

# Ackknowledgements


- Mathias Preiner (benchmark selection and scoring scripts)
- Aaron Stump (StarExec)
- Andres Nötzli (trace executor extension)
- Marco Gario and Andrea Micheli (PySMT)
- Martin Riener (certificates/trophies logistics)