# SMT-COMP 2020

## 15th International Satisfiability Modulo Theories Competition

Haniel Barbosa, Jochen Hoenicke, Antti Hyvärinen, July 7, 2020

# 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

# New this year

An attempt at containing the workload

- Soft **limits** on number of solvers per team
- Select **less benchmarks**
- Set **lower timeout**
- Give **more time to run** the actual competition

**Extension** of the Model Validation track

A ban for **portfolio-style** solvers

# SMT-Solvers

Determine **satisfiability** of instances from first-order logic fragments

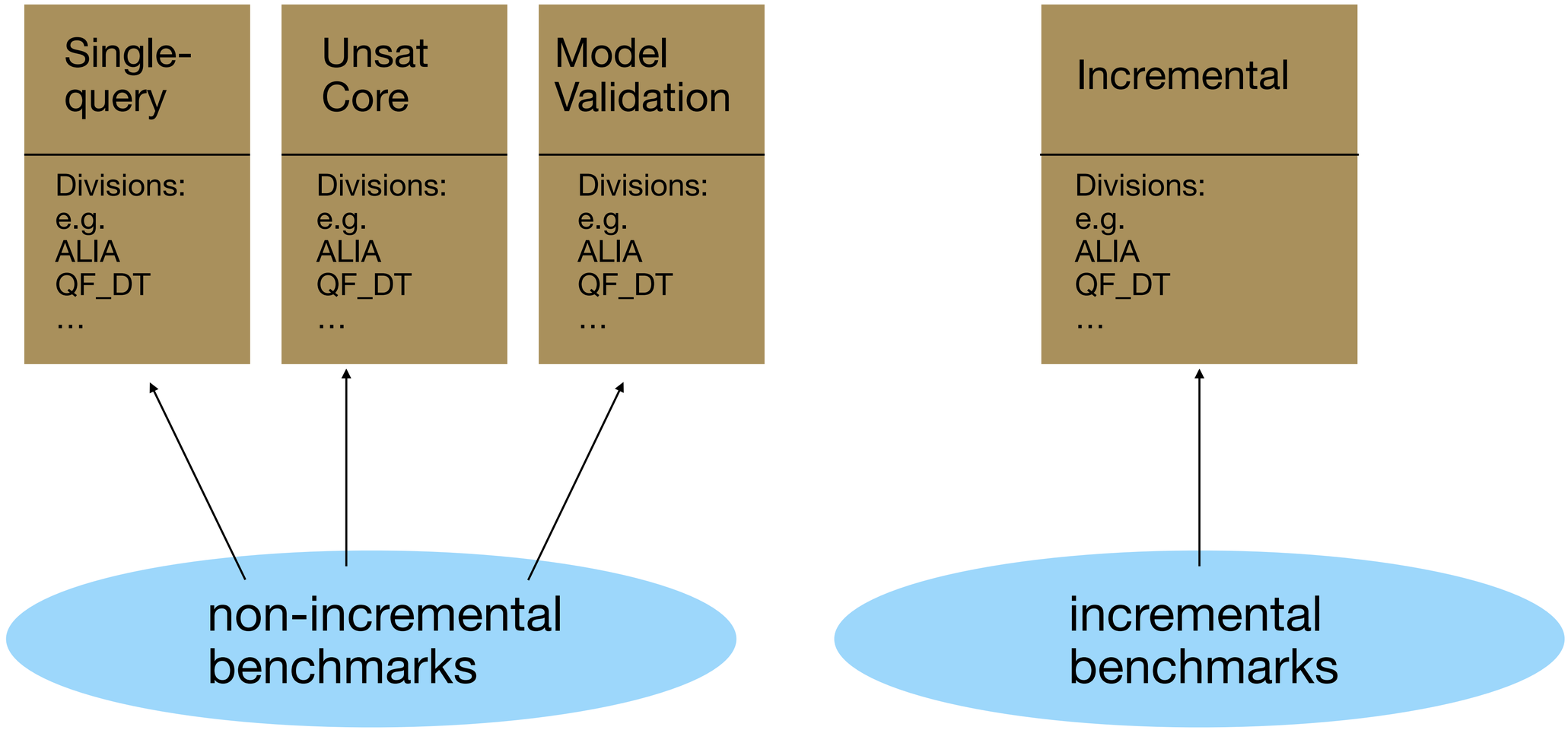
- benchmarks in **SMT-LIB** format

This year participants:

- CDCL(T)-based “classic” SMT solvers
- Automated Theorem Provers
- Wrapper Tools: call at least one other SMT solver

This year no derived tool submissions

# Competition overview



# Tracks

**Single-Query Track** (previously Main Track)

- One single `check-sat` command, no push/pop commands
- Remove all benchmarks solved by all solvers in 2018 or 2019 in ≤ 1 s
- Partly randomised selection of at least 40% of instances
- Time limit 1200s (20 min)

**Incremental Track** (previously Application Track)

- Multiple `check-sat` commands, and push/pop commands
- Solvers are executed on benchmarks via the trace executor
- Partly randomised selection of at least 40% of instances
- Time limit 1200s (20 min)

# Tracks

## Unsat Core Track

- One single `check-sat` command, ≥ 2 assert commands
- Benchmarks with status **UNSAT**
- Extract unsat core as set of top-level assertions
- Partly randomised selection of at least 40% of instances
- Time limit 1200s (20 min)

## Model Validation Track

- One single `check-sat` command
- Selection of benchmarks with status **SAT**
- Produce full, correct, well-formed model in SMT-LIB format*
- for division QF_BV, experimental for QF_IDL, QF_LIA, QF_LIRA, QF_LRA, QF_RDL
- Time limit 1200s (20 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

We do not run non-competitive divisions

# Solver Presentations by Participants

# Alt-Ergo @ SmtComp 2020

Alt-Ergo:

- Shostak based SMT solver
- Specialised in quantified formulas and program verification

Developers:

- Sylvain Conchon, Albin Coquereau, Mattias Roux and Guillaume Bury

News:

- Overall performance optimisation

OCaml PRO

# Semi-Deciding QF_NIA with AProVE via Bit-Blasting

J. Giesl, M. Brockschmidt, F. Frohn, C. Fuhs, J. Hensel, P. Schneider-Kamp, T. Ströder, R. Thiemann


... mainly a (non-)termination and complexity bounds prover, but also ...

- SMT-LIB 2 front-end for QF_NIA
- use bit-blasting for binary arithmetic, back-end: **MiniSat** [Eén, Sörensson, *SAT '03*]
- bit-length for unknowns **fixed**
- bit-length for constants, sums, products etc. **as needed** (unlike QF_BV)
- SAT encoding: [Fuhs, Giesl, Middeldorp, Schneider-Kamp, Thiemann, Zankl, *SAT '07*]

$\Rightarrow$ back-end for proof techniques for termination/complexity bounds

- approach for SMT-COMP
  - start with small search space
  - **if** MiniSat says **satisfiable**: return with model
  - **else**: retry with larger search space until satisfiable (or out of resources)
- NB: small solver id **1229** (next higher id at SMT-COMP '20: **28001**), scored highest in 2015

# Bitwuzla at the SMT-COMP'20

Aina Niemetz, Mathias Preiner

## Bitwuzla

. . . is the successor of Boolector.

## New Features

- Floating-points: word-blasting with SymFPU
- Unsat cores for all supported quantifer-free logics

## Improvements

- Improved propagation-based local search for bit-vectors
- CaDiCaL version 1.2.1 as default SAT back end for all logics

## Tracks/Divisions

| | |
|---|---|
| **Single Query:** | BV, QF_{A,BV,FP,UF}$^+$ |
| **Incremental Track:** | QF_{A,UF,BV,FP}$^+$ |
| **Unsat Core:** | QF_{A,BV,FP,UF}$^+$ |
| **Model Validation:** | QF_BV |

`https://bitwuzla.github.io`

# COLIBRI(2020)

Goal participate to all `QF_*FP*` division:

- support N-dimension arrays
- better integraton of multi-theories other than QF_FP

Wins of COLIBRI:

- QF_BVFPLRA: unsat
- QF_ABVFPLRA: 24s
- QF_FP: 24s (Bravo Bitwusla !)
- QF_FPLRA: all category. Nothing proved after the first 24s

# CVC4 at the SMT Competition 2020

Clark Barrett, Haniel Barbosa, Martin Brain, Ahmed Irfan, Makai Mann, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, Amalee Wilson, Yoni Zohar

## Divisions

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

## New Features/Improvements

- String solver:
  - More aggressive context-dependent simplifications
  - Better reductions of extended functions
  - Lazy processing of regular expression intersections
- Eager bit-blasting solver:
  - New version of CaDiCaL
  - Ackermannization of uninterpreted sorts
- Arithmetic solver:
  - Linear integer arithmetic: Branch-and-bound modified to use ternary splitting lemmas inspired by unit-cube tests
  - Non-linear arithmetic: Backward implication of tangent lemma adopted from MathSAT-na-ext

## Configurations

- Similar to last year
- Model-Validation Track uses same configuration as Single Query Track but disables simplification of unconstrained terms

# `lazybv2int` at the SMT Competition 2020

Yoni Zohar, Ahmed Irfan, Makai Mann, Andres Nötzli, Andrew Reynolds, Clark Barrett

New participant in QF_BV incremental and non-incremental tracks

## Highlights

- Prototype solver
- Wrapper tool – uses smt-switch
- Uses MathSAT5 and CVC4
- Playground for integer-based solving

## Approach

- Translation from BV to integers
- Arithmetic operations are easy
- Shift operations: UF
- Bit-wise operations: UF

## Eager Mode

- Shifts:
  - Replace with a UF
  - Equate with a big ITE
  - ITE simulates exponentiation
- Bitwise Operators:
  - Replace with a UF
  - Equate with sum or "compare bits"

## Lazy Mode

- Shifts:
  - Replace with a UF
  - Lazily instantiate axioms
  - Worst case: ITE
- Bitwise Operators:
  - Replace with a UF
  - Lazily instantiate axioms
  - Check candidate model with BV solver
  - Worst case: sum/bits

# OpenSMT

A CDCL(T)-based SMT solver

Supports **QF_UF**, **QF_LRA**, and to some extent **QF_LIA**, **QF_BV**

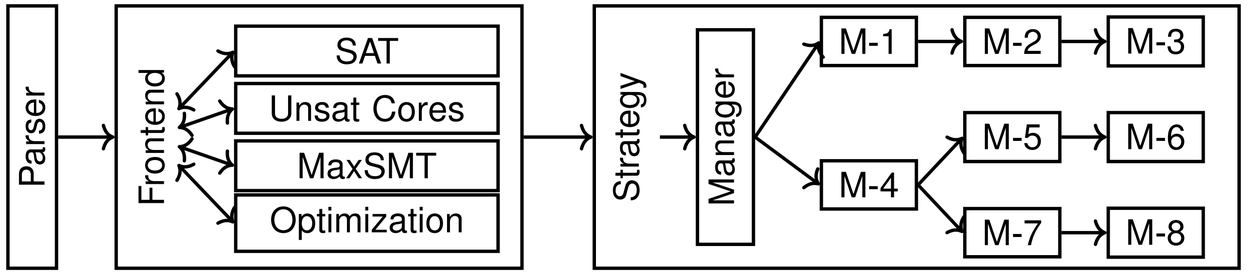
New in 2020:

- Performance improvements for **QF_LRA**:
  - Theory guided choice of polarity for decision vars
  - Lazy tableau maintenance (quasi-basic vars)
  - Single consistent backup assignment
- Models in SMT-LIB2
- Interpolation in incremental mode

http://verify.inf.usi.ch/opensmt
https://github.com/usi-verification-and-security/opensmt

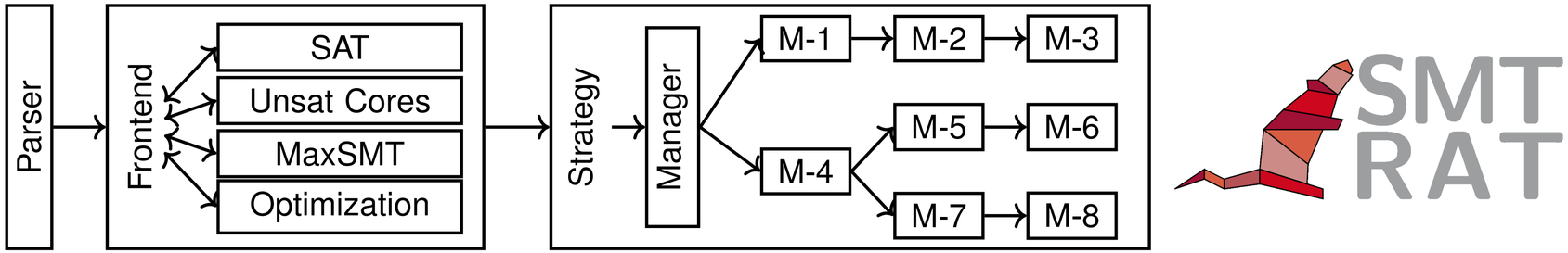
RWTH AACHEN UNIVERSITY

# SMT-RAT 20.04




RWTH AACHEN UNIVERSITY

# SMT-RAT 20.04



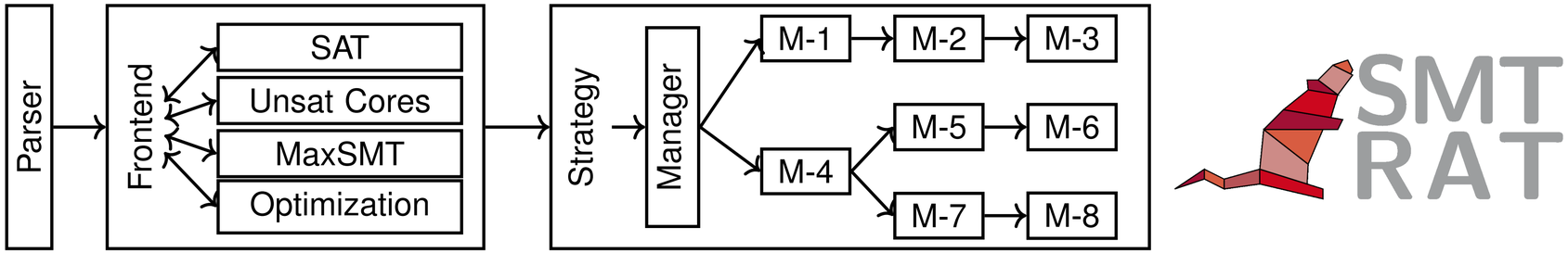




- `SMT-RAT` strategy
  - SAT solver: `minisat` adapted for less-lazy SMT solving
  - non-linear arithmetic: subtropical satisfiability, interval constraint propagation, virtual substitution and cylindrical algebraic decomposition
  - non-linear integer arithmetic: bit-blasting and branch&bound

RWTH AACHEN UNIVERSITY

# SMT-RAT 20.04



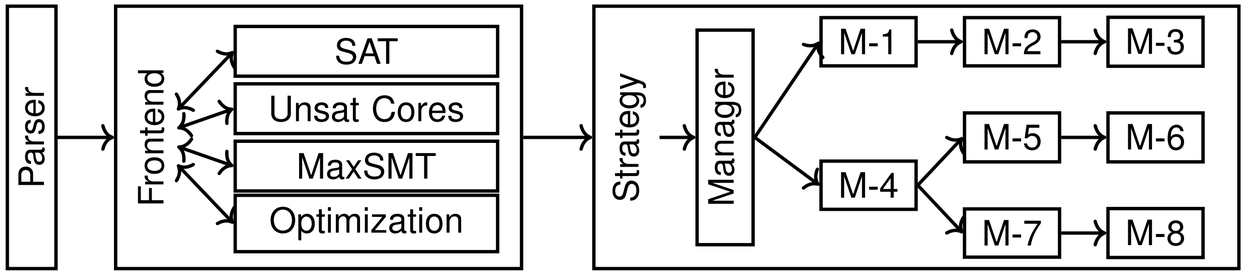




- `SMT-RAT` strategy
  - SAT solver: `minisat` adapted for less-lazy SMT solving
  - non-linear arithmetic: subtropical satisfiability, interval constraint propagation, virtual substitution and cylindrical algebraic decomposition
  - non-linear integer arithmetic: bit-blasting and branch&bound
- `SMT-RAT-MCSAT` strategy: MCSAT module based on `minisat`
  - Fourier-Motzkin, interval constraint propagation, virtual substitution, one-cell construction, NLSAT-style model based projections

RWTH AACHEN UNIVERSITY

# SMT-RAT 20.04





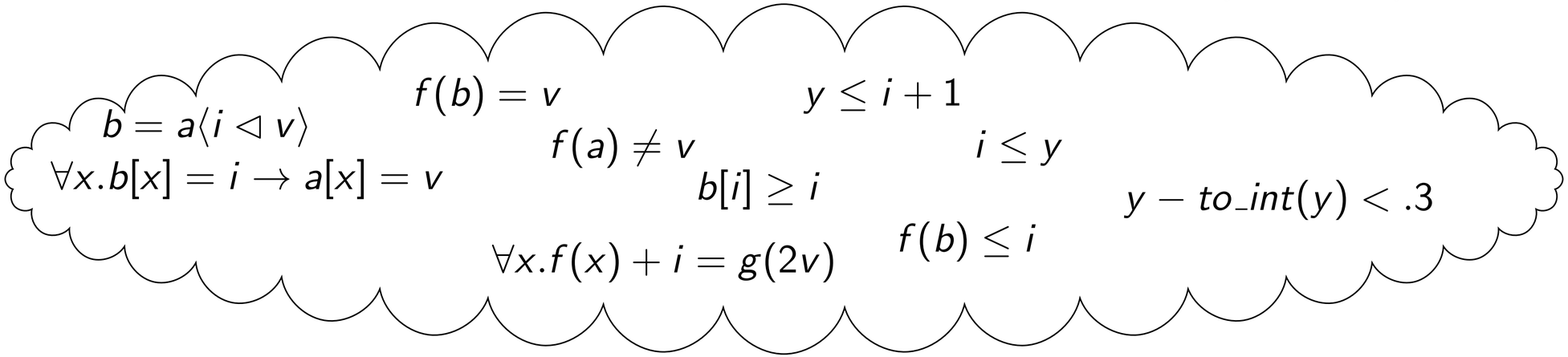
- `SMT-RAT` strategy
  - SAT solver: `minisat` adapted for less-lazy SMT solving
  - non-linear arithmetic: subtropical satisfiability, interval constraint propagation, virtual substitution and cylindrical algebraic decomposition
  - non-linear integer arithmetic: bit-blasting and branch&bound
- `SMT-RAT-MCSAT` strategy: MCSAT module based on `minisat`
  - Fourier-Motzkin, interval constraint propagation, virtual substitution, one-cell construction, NLSAT-style model based projections
- `SMT-RAT-CAlC` strategy: less-lazy SMT module implementing the novel method cylidrical algebraic coverings (CAlC)

# SMTInterpol

Jürgen Christ, Jochen Hoenicke, Tanja Schindler

UNI FREIBURG

- CDCL(T) based SMT solver (sat / unsat)
- for Arrays, Uninterpreted Functions, Linear Integer and Real Arithmetic (AUFLIRA)




- that supports model/proof production and *interpolation*, and

get-model | get-proof | get-unsat-core | get-interpolants

- (since 2019) has support for quantifiers (E-matching based unit/conflict propagation, MBQI).

Open Source (LGPL)

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

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


**The adapter that adds quantifier support to your SMT solver!**

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

- Origin: Quantifier elimination of the Ultimate Automizer software verifier
- Supported theories: Arrays, Bitvectors, Floats, Integers, Reals

2020 competition candidate:

UltimateEliminator+MathSAT-5.6.3

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

# Vampire 4.5

*Reger, Suda, Voronkov, Kovács, Kotelnikov, Robillard, Riener, Rawson, Gleiss, Rath, Hozzova, Schoisswohl*

https://vprover.github.io

Entering SMTCOMP (single query) since 2016
SMT Logics: A, DT, LIA, LRA, NIA, NRA, UF
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

Haniel Barbosa, Daniel El Ouraoui, 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

- New instantiation heuristics
  - Restrict number of instances per quantified formula
  - Restrict the total number of skolem constants
- Automatically generated option scheduler
  - Generated using integer programming
  - Non-competing variant for 24s timeout: *veriT+vite*
- Improved support for higher-order logic
- Better proof production
- Instantiation (Learned instance selection, improved conflicting instantiation)

# Yices 2 in SMTCOMP 2020

## 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

## Entered in all Supported Logics and Divisions

- Single Query: Quantifier-free logics including linear and nonlinear arithmetic, bitvectors, and combination with UF and Arrays.
- Model Validation: All logic supported
- Incremental: Same logics
- Unsat Core: Same logics minus non-linear arithmetic

# New Developments in 2020

## Main Solver

- New backend SAT solver
- More bit-vector simplification and rewriting
- Improvements to MCSAT-BV (presented at IJCAR)
- Improvements to MCSAT (better handling of equalities, interval reasoning)
- Better SMT-LIB Models

## Language Bindings

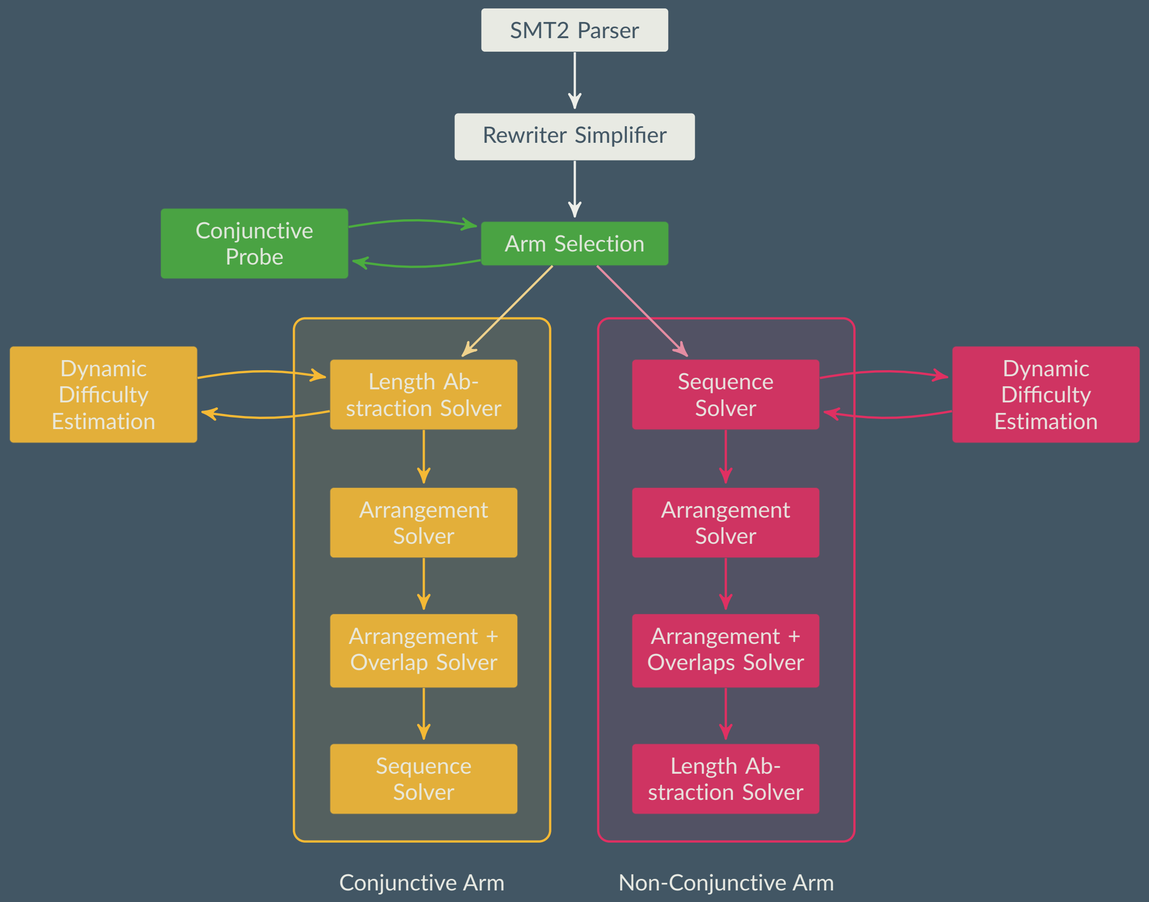
- Python, Ocaml, Go (on GitHub) + Java (to be released)

## Experimental/In Progress

- Support for quantifiers
- Interpolants via MCSAT

# Z3 | str | 4
# A Two-Armed String Solver

Murphy Berzish, Mitja Kulczynski, Federico Mora, Dirk Nowotka, Vijay Ganesh




- Two-armed solver incorporating three sub-solvers:
  - Z3str3
  - LAS (Length Abstraction Solver)
  - Z3seq (Microsoft Research sequence solver)
- Arm selection is done via a syntax-driven “probe” that predicts which arm will have the best runtime
- LAS is a novel CEGAR-style algorithm that solves abstractions and refinements of integer constraints implied by string equations
- Z3str3 leverages a bit-vector reduction to combine the efficiency of an unfolding-based strategy with the ability to reason about unbounded string terms
- Sequence solver and LAS augmented with “dynamic difficulty estimation” heuristic
- Full system description at `https://z3str4.github.io/smtcomp.pdf`

# Non-Competitive Solvers

**Total**: 24 (SQ), 17 (Inc), 15 (UC), 7 (MV)

Submitted by organisers:

- Z3-4.8.8
- MathSAT 5.6.3
- Division winners from last year (27 solvers)

Submitted by participants:

- Late submission (veriT+vite)
- Fixed solvers (Bitwuzla, MinkeyRink, SMTInterpol, Yices2)

# Benchmark Selection

Remove **easy benchmarks**

- SQ: all benchmarks solved by all solvers in ≤ 1 s in 2018 or 2019

Ensure **minimum number of instances** in a division

- $n \leq 300$: all instances
- $300 < n \leq 750$: 300 instances
- $n > 750$: 40% of the instances

Guarantee inclusion of **new benchmarks** (at least one per family)

Select benchmarks uniformly at **random**

# Scores

**Single Query**: number of correctly solved instances

**Incremental**: number of correctly solved `check-sat` calls

**Unsat Core**: reduction number of top-level assertions

**Model Validation**: number of correctly solved instances with validated models

# Scores

**Sequential score** (SQ, MV): time limit applied to CPU time

**Parallel score** (all): time limit applied to wall-clock time

**SAT score** (SQ): parallel score for satisfiable instances

**UNSAT score** (SQ): parallel score for unsatisfiable instances

**24s score** (SQ): parallel score for time limit of 24s

# Competition-Wide Recognitions

Two competition-wide rankings

**Biggest lead**

- in terms of score over the solver in the next 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

**Teams:** 16 (-7)

**StarExec Stats:** 943’564 jobs (-79’258), 9.9 CPU years (-11.5)

| | Solvers | | Divisions | | Benchmarks | | |
|---|---|---|---|---|---|---|---|
| Track | Total | C/NC | Total | C/Exp | C | Selected | Total |
| SQ | 42 (-9) | 20/22 | 67 (+10) | 67/0 | 89’910 (+25’754) | 89’924 (+107) | 353’790 (+26’749) |
| Inc | 24 (+2) | 9/15 | 26 (-3) | 26/0 | 9’817 (+2’982) | 10’296 (+2’729) | 23’519 (+9’489) |
| UC | 18 (+4) | 5/13 | 40 (+2) | 40/0 | 42’417 (+12’609) | 44’539 (+198) | 107’357 (+28’655) |
| MV | 13 (+3) | 7/6 | 6 (+5) | 1/5 | 5752 (+5752) | 8’284 (+1093) | 20’550 (+6’168) |

C — competitive, NC — non-competitive, Exp — experimental

# Results


SMT-COMP

The International Satisfiability Modulo Theories (SMT) Competition.

**GitHub**

**Home**
**Introduction**
**Benchmark Submission**
**Publications**
**SMT-LIB**
**Competitions by Year**

SMT-COMP 2020

**Rules**
**Benchmarks**
**Tools**
**Specs**
**Participants**
**Results**


## SMT-COMP 2020 Results

### Competition-Wide Recognitions

**Largest Contribution Ranking**

- Incremental Track
- Model Validation Track
- Single Query Track
- Unsat Core Track

**Biggest Lead Ranking**

- Incremental Track
- Model Validation Track
- Single Query Track
- Unsat Core Track

### Tracks Summary

- Incremental Track
- Model Validation Track
- Single Query Track
- Unsat Core Track

### Disagreements

Solvers disagreed on some benchmarks marked as unknown in the following tracks and divisions.

- **Single Query Track**
  - FP
  - LIA
  - QF_ABVFPLRA
  - QF_S

# Competition-Wide Recognitions

# Largest contribution

| | 1st Place | 2nd Place | 3rd Place |
|---|---|---|---|
| **Single Query** | | | |
| seq | CVC4 | Yices2 | Vampire |
| par | CVC4 | Yices2 | Vampire |
| sat | CVC4 | SMTInterpol | Vampire |
| unsat | Yices2 | CVC4 | Vampire |
| 24s | Yices2 | CVC4 | Vampire |
| **Incremental** | | | |
| par | CVC4 | Yices2 | Bitwuzla |
| **Unsat Core** | | | |
| seq | CVC4 | Yices2 | Bitwuzla |
| par | CVC4 | Yices2 | Bitwuzla |

# Largest contribution

| | 1st Place | 2nd Place | 3rd Place |
|---|---|---|---|
| **Single Query** | | | |
| seq | CVC4 (UFNIA) | Yices2 (QF_NIA) | Vampire (UF) |
| par | CVC4 (UFNIA) | Yices2 (QF_NIA) | Vampire (UF) |
| sat | CVC4 (UFDTLIRA) | SMTInterpol (UFLIA) | Vampire (UF) |
| unsat | Yices2 (QF_NIA) | CVC4 (UFNIA) | Vampire (UF) |
| 24s | Yices2 (QF_NIA) | CVC4 (UFNIA) | Vampire (UF) |
| **Incremental** | | | |
| par | CVC4 (UF) | Yices2 (QF_UFLIA) | Bitwuzla (QF_ABV) |
| **Unsat Core** | | | |
| seq | CVC4 (AUFLIRA) | Yices2 (QF_AX) | Bitwuzla (QF_BV) |
| par | CVC4 (AUFLIRA) | Yices2 (QF_AX) | Bitwuzla (QF_BV) |

# Biggest Lead

| | 1st Place | 2nd Place | 3rd Place |
|---|---|---|---|
| **Single Query** | | | |
| seq | CVC4 | Yices2 | SMT-RAT |
| par | CVC4 | Yices2 | SMT-RAT |
| sat | CVC4 | Yices2 | SMTInterpol |
| unsat | CVC4 | Vampire | Yices2 |
| 24s | CVC4 | Vampire | Yices2 |
| **Incremental** | | | |
| par | CVC4 | Yices2 | Bitwuzla |
| **Unsat Core** | | | |
| seq | CVC4 | Yices2 | Bitwuzla |
| par | CVC4 | Yices2 | Bitwuzla |

# Biggest Lead

| | 1st Place | 2nd Place | 3rd Place |
|---|---|---|---|
| **Single Query** | | | |
| seq | CVC4 (AUFBVDTLIA) | Yices2 (QF_UFNRA) | SMT-RAT (QF_NIRA) |
| par | CVC4 (AUFBVDTLIA) | Yices2 (QF_UFNRA) | SMT-RAT (QF_NIRA) |
| sat | CVC4 (AUFBVDTLIA) | Yices2 (QF_UFNRA) | SMTInterpol (ALIA) |
| unsat | CVC4 (UFFPDTLIRA) | Vampire (NIA) | Yices2 (QF_AUFBV) |
| 24s | CVC4 (UFFPDTLIRA) | Vampire (UFDTLIA) | Yices2 (QF_AUFBV) |
| **Incremental** | | | |
| par | CVC4 (UFNIA) | Yices2 (QF_AUFLIA) | Bitwuzla (QF_BVFP) |
| **Unsat Core** | | | |
| seq | CVC4 (UFDT) | Yices2 (QF_AX) | Bitwuzla (QF_FP) |
| par | CVC4 (UFDT) | Yices2 (QF_AX) | Bitwuzla (QF_FP) |

# Model Validation

| | 1st Place | 2nd Place | 3rd Place |
|---|---|---|---|
| **seq** | Bitwuzla | CVC4 | STP + MergeSAT |
| **par** | Bitwuzla | STP + CMS | CVC4 |

# New entrants

In 2020 the new solvers are

- Bitwuzla:
  - Aina Niemetz, Mathias Preiner
- LazyBV2Int:
  - Ahmed Irfan, Makai Mann, Andres Nötzli, Andrew Reynolds, Yoni Zohar, and Clark Barrett
- veriT+vite
  - Haniel Barbosa, Daniel El Ouraoui, Pascal Fontaine, Hans-Jörg Schurr
- Z3str4
  - Murphy Berzish, Mitja Kulczynski, Federico Mora, Vijay Ganesh, Dirk Nowotka

# Benchmark contributors

The 2020 competition new benchmarks were provided by

- David Deharbe
- Matthias Heizmann
- Johannes Kanig
- Andres Nötzli
- Dennis Yurichev

# Solver disagreements

Two solvers **disagree** on a benchmark with unknown status:

- One says SAT and the other UNSAT

This is particularly problematic if one of the solvers was not found unsound on a benchmark with known status in the same division

This year we had disagreements

- on 149 benchmarks (115 SQ, 34 Inc, 44 with an unsound solver)
- on 8 divisions (6 SQ, 2 Inc)
- Involving 15 solvers (some of them from last year)

**Last year there were no disagreements**

# Discussion

**Model Validation Solvers** are not standard compliant

**Two DL extensions**

- First was necessary because we were not ready
- Second caused some harm, we plan to address this in the rules

**Running fixed solvers on all divisions** where they were submitted

- We found this very insightful

**Inclusion of Par4** from last year

- Portfolio solvers were not allowed this year, so was this unfair?

# Workload of organising SMT-COMP

| | |
|---|---|
| meetings | 15 |
| git commits | 400 |
| Slack discussions | numerous |
| Python lines | 4'592 |
| Shell script lines | 1'055 |
| Liquid HTML lines | 1'369 |

# Acknowledgements

Aina Niemetz and Mathias Preiner (insights and support)

Aaron Stump (StarExec)