Journal on Satisfiability, Boolean Modeling, and Computation 11 (2019) 223-261

# The SMT Competition 2015–2018

**Tjark Weber** (competition chair, 2015–2018) tjark.weber@it.uu.se
*Uppsala University*
*Sweden*

**Sylvain Conchon** (co-organizer, 2015–2016) Sylvain.Conchon@lri.fr
*Paris-Sud University*
*France*

**David Déharbe** (co-organizer, 2015–2016) david.deharbe@clearsy.com
*ClearSy Systems Engineering*
*France*

**Matthias Heizmann** (co-organizer, 2016–2018) heizmann@informatik.uni-freiburg.de
*University of Freiburg*
*Germany*

**Aina Niemetz** (co-organizer, 2018) niemetz@cs.stanford.edu
*Stanford University*
*USA*

**Giles Reger** (co-organizer, 2017–2018) giles.reger@manchester.ac.uk
*University of Manchester*
*UK*

## Abstract

The International Satisfiability Modulo Theories Competition is an annual competition between Satisfiability Modulo Theories (SMT) solvers. The 2018 edition of the competition was part of the FLoC Olympic Games, which comprised 14 competitions in various areas of computational logic. We report on the design and selected results of the SMT Competition during the last FLoC Olympiad, from 2015 to 2018. These competitions set several new records regarding the number of participants, number of benchmarks used, and amount of computation performed.

Keywords: *SMT solver, SMT-COMP, SMT-LIB, Satisfiability Modulo Theories, competitions*

*Submitted November 2018; revised August 2019; published September 2019*

## 1. Introduction

Satisfiability Modulo Theories (SMT) is a generalization of Boolean satisfiability (SAT), the satisfiability decision problem for propositional logic. In place of Boolean variables, SMT formulas may contain terms that are built from function and predicate symbols drawn from a number of background theories. Background theories, which are motivated by application domains, include the theory of arrays, integer and real arithmetic, bit-vectors, and floating-point numbers, among others [8]. For instance, the following is an SMT formula over the combination of integer arithmetic and uninterpreted functions:

$$x \leq y \wedge y \leq x \wedge P(f(x) - f(y)) \wedge \neg P(0).$$

This formula asserts that $x$ is less than or equal to $y$, that $y$ is less than or equal to $x$, that $P$ holds for the difference of $f(x)$ and $f(y)$, and that $P$ does not hold for 0. (Here, $x$ and $y$ are integers, $f$ is a function from integers to integers, and $P$ is a predicate over integers.) Software tools to determine the satisfiability of such formulas are called SMT solvers. With its rich input language, SMT has applications in software engineering, optimization, and many other areas [23].

Internally, many SMT solvers employ SAT solving techniques to deal with the propositional structure of the formula, and combine these with (semi-)decision procedures for the background theories that the solver supports [3, 13, 24, 38]. Historically, SMT solvers have focused on quantifier-free formulas and decidable combinations of background theories. Increasingly, however, SMT solvers also support quantified formulas. Hence there is overlap between SMT solving and automated theorem proving for quantified Boolean formulas (QBF), first-order logic, and even higher-order logic [2].

The International Satisfiability Modulo Theories Competition (SMT-COMP) [10, 4, 5, 6, 7, 17, 18] is an annual competition between SMT solvers. It was instituted in 2005, and is affiliated with the International Workshop on Satisfiability Modulo Theories. Solvers are submitted to the competition by their developers, and pitted against each other in a number of tracks and divisions.

The SMT Competition is one of a growing number of tool competitions in the broad area of formal methods, which are summarised in a recent review of the area [11]. The two closest competitions in terms of goals and design are the SAT competition (http://www.satcompetition.org/), which evaluates SAT solvers, and CASC (http://www.tptp.org/CASC/), which evaluates automated theorem provers (ATPs). In recent years, we have seen a blurring of lines between SMT-COMP and CASC with SMT solvers competing in CASC and ATPs competing in SMT-COMP. The main (informal) distinction remains that SMT-COMP focuses on problems that are heavy in theories, whereas CASC focuses on problems heavy in quantifiers. Another closely related competition is SV-COMP (https://sv-comp.sosy-lab.org) where many tools rely on advances in SMT solving technology.

The SMT Competition was part of the Federated Logic Conference (FLoC) Olympic Games, which comprised 14 competitions in various areas of computational logic, in Vienna in 2014, and again in Oxford (UK) in 2018. It was last described in a 2014 competition report [17]. In this paper, we report on the design and selected results of the competition during the last FLoC Olympiad, from 2015 to 2018.

Complete data about past competitions since 2015 (and partly for earlier competitions) is available from the competition website, http://www.smtcomp.org. This includes detailed results for each solver/benchmark combination, totaling nearly 6 million data records (about 1.1 GB in CSV format). Competition solvers and benchmarks may be downloaded from StarExec, https://www.starexec.org.

The rest of this paper is structured as follows. In Section 2, we present the goals and organization of the competition. Section 3 introduces the SMT-LIB language and library, and Section 4 discusses competition divisions and benchmarks. The competition is being run on StarExec, which is described in Section 5. Section 6 gives an overview of participating solvers, and Section 7 discusses the scoring rules. Selected competition results are presented in Section 8, and analyzed in more detail in Section 9. We conclude with suggestions for future SMT Competitions in Section 10.

## 2. The Competition Goals and Organization

The original goals of the SMT Competition were to spur adoption of the community-designed SMT-LIB format (see Section 3), and to spark further advances in SMT, especially for verification [10]. These, together with providing a useful yardstick of performance for users and developers of SMT solvers, are still its main goals to date. The competition has been successful in establishing SMT-LIB as the de facto standard language for SMT solvers. In recent years, the focus in this regard has shifted towards promoting some of the newer extensions of the SMT-LIB language [8], such as floating-point numbers and algebraic datatypes.

The competition is organized under the direction of the SMT Steering Committee, which appoints the competition chair [1]. The competition chair then assembles a team of organizers, and oversees the work for the next edition of the competition. The SMT Workshop includes a block of time to present the competitors and results of the competition. The workshop is affiliated with a major conference in the field of automated reasoning each year: with CAV in 2015, with IJCAR in 2016, again with CAV in 2017, and with IJCAR (as part of FLoC) in 2018. Consequently, its date varies. During 2015–2018, the workshop always took place in July.

Important competition deadlines are determined by calculating backwards from the date of the next SMT Workshop. The competition's computational workload should be completed approximately two weeks before the workshop date. This gives participants and other parties a chance to scrutinize the data for errors, allows partial re-runs of competition jobs if necessary, and gives the organizers time to prepare the results presentation. While making all data publicly available well in advance of the official results presentation may impact the suspense, it has on several occasions helped to uncover serious problems with the competition tools or the StarExec framework (see Section 5) that could have led to invalid results otherwise.

It typically takes up to three weeks to run all competition jobs on the StarExec cluster, so the final submission deadline for solvers needs to be about five weeks before the workshop date. The competition imposes a separate deadline for first versions of solvers about two weeks before the final submission deadline. This earlier deadline has proved useful for the organizers to obtain an accurate estimate on the number of competing solvers, and to run preliminary tests with some of the submitted solvers to identify potential issues. Participants can still make changes to their solver until the final deadline but are encouraged to use this period for bug-fixing only.

Around the initial solver deadline, the organizers also aim to publish the latest version of the competition tools, and—in collaboration with the SMT-LIB maintainers—to release a new version of the SMT Library (see Section 3). SMT-LIB releases were made on June 1 in 2015, on May 23 in 2016, on June 5 in 2017, and on May 20 in 2018. Anyone may submit new benchmarks to the SMT Library, and competitors are allowed to tune their solvers accordingly. Therefore all benchmarks eligible for the competition must be released at least

```
(set-info :smt-lib-version 2.6)
(set-logic QF_UFLIA)
(set-info :status unsat)
(declare-fun x () Int)
(declare-fun y () Int)
(declare-fun f (Int) Int)
(declare-fun P (Int) Bool)
(assert (and (<= x y) (<= y x) (P (- (f x) (f y))) (not (P 0))))
(check-sat)
(exit)
```

**Figure 1:** A benchmark problem in SMT-LIB syntax.

some time before the final solver submission deadline. To give the SMT-LIB maintainers sufficient time for curation, the deadline for new benchmark contributions has usually been about four to six weeks before the initial solver deadline, typically in April or early May.

The competition rules are revised each year, and a (near final) draft version of the rules is made public on the competition website around mid-April. The competition chair has ultimate responsibility for the rules, but changes are often preceded by discussions among the organizers or within the SMT community. Such discussions are initiated by a Call for Comments that is sent out to the SMT-COMP mailing list [39] in February or early March. Publicly, this call is the first harbinger of a new edition of the SMT Competition.

## 3. The SMT-LIB Language and Library

All problems used in the SMT Competition come from the SMT Library (SMT-LIB). This is a large collection of benchmark problems from various sources. The SMT Library is available from a public repository [40], and each new release is mirrored on StarExec. Users of SMT solvers are encouraged to submit new and interesting benchmarks to the library, which has grown from 1352 benchmarks in 2005 [10] to 347 011 benchmarks in 2018.

Benchmarks in the SMT Library are written in the SMT-LIB language [8], a text-based format that defines the syntax and semantics of solver input and output. This includes a language for terms and formulas, and a command language for interacting with SMT solvers. For instance, Figure 1 shows the example formula from Section 1 as a benchmark in SMT-LIB syntax, with commands to the SMT solver typeset in bold.

In 2016, benchmarks were updated from version 2.0 to version 2.5 of the SMT-LIB language. Since 2017, benchmarks are written in version 2.6. Fortunately, versions 2.5 and 2.6 are largely backwards-compatible to 2.0, so that older solvers are still able to compete with at most minor modifications.

The SMT Library classifies benchmarks according to their *logic*, that is, according to the specific combination of background theories that the benchmark uses. Logic names are composed of letter groups that refer to these background theories: A or AX for the theory of arrays, BV for the theory of bit-vectors, DT for algebraic datatypes, FP for the floating-point theory, IA for integer arithmetic, RA for real arithmetic, IRA for mixed in-

**Table 1:** Number of benchmarks and logics in the SMT Library

| | | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|---|
| Non-incremental | benchmarks | 196375 | 196114 | 258079 | 336844 |
| | logics | 40 | 40 | 49 | 51 |
| Incremental | benchmarks | 10019 | 10024 | 6262 | 10167 |
| | logics | 15 | 15 | 22 | 26 |

teger and real arithmetic, S for the theory of strings, UF for uninterpreted functions (i.e., free symbols). Additionally, logics may impose further syntactic restrictions, such as the absence of quantifiers (QF_), the restriction to difference logic over the integers (IDL) or reals (RDL), or to the (non-)linear fragment of arithmetic (N or L, respectively, before IA, RA or IRA). For instance, the logic QF_UFLIA contains quantifier-free formulas with uninterpreted functions over linear integer arithmetic. These restrictions are typically motivated by the existence of efficient decision procedures for certain syntactic fragments.

Additionally, the SMT Library distinguishes between *incremental* and *non-incremental* benchmarks. Non-incremental benchmarks contain a single `check-sat` command, which instructs the SMT solver to determine the satisfiability of the benchmark. (Valid solver responses are `sat`, `unsat`, and `unknown`.) Incremental benchmarks exercise additional solver features, such as the ability to retract assertions, and typically contain multiple `check-sat` commands. These benchmarks originate from applications such as bounded model checking [30], where SMT solvers interact with other tools in a feedback loop.

The 2018 release of the SMT Library contained 336 844 non-incremental benchmarks in 51 logics, as well as 10 167 incremental benchmarks (with a total of nearly 32 million `check-sat` commands) in 26 logics. Table 1 presents further data on the size of the SMT Library since 2015. While the library usually grows from one year to the other because of the inclusion of new benchmarks and logics, continuous curation efforts may also cause benchmarks to be removed or reclassified on occasion, thereby causing individual logics or even the entire library to shrink. From 2015 to 2018, the total number of benchmarks has increased by 68 %. The largest benchmark in the 2018 release had a size of 1041 MB; the total size of the release was 91 GB.

Benchmarks in the SMT Library are annotated with additional information, such as the source of the benchmark and its status (i.e., whether the benchmark is satisfiable or unsatisfiable) if this is known. (For incremental benchmarks, each `check-sat` command has a separate status annotation.) To prevent solvers from simply looking up the benchmark's status in the SMT Library, benchmarks in the competition are lightly scrambled before they are presented to solvers [43].

## 4. Competition Divisions and Benchmarks

The SMT Competition consists of several tracks. Each track is further split into (logic) *divisions*, where each division corresponds to a logic from the SMT Library. As indicated in Tables 1–2, there are slightly fewer divisions (Table 2) than logics in the SMT Library (Table 1) because not every logic contains benchmarks eligible for the competition.

**Table 2:** Number of divisions in each competition track

| | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| Main Track | 40 | 40 | 49 | 50 |
| Application Track | 14 | 14 | 16 | 21 |
| Unsat-Core Track | – | 40 | 41 | 44 |

**Tracks.** The *Main Track*, which has been a staple of the competition since its inception in 2005, features non-incremental benchmarks. The *Application Track*, which has been part of the competition since 2011 [5], features incremental benchmarks. These benchmarks are passed to each solver one command at a time by a trace executor, which waits for the solver's response before passing the next benchmark command (to prevent look-ahead behavior). The *Unsat-Core Track* uses Main Track benchmarks that are known to be unsatisfiable, and solvers are evaluated on their ability to find small unsatisfiable cores (i.e., a small but still unsatisfiable subset of the benchmark's assertions). The Unsat-Core Track had first been part of the competition in 2012 [18], but was then discontinued (partly because of the competition's move to StarExec and inadequate tool support). It was re-introduced as an experimental track in 2016, and has been a regular competition track again since 2017.

**Divisions.** Table 2 shows the number of divisions in each track since 2015. Participants can choose which tracks and divisions they want to enter, and competition results are reported separately for each track and division. Thus, competing solvers do not need to support all logics, and may or may not support features such as incremental solving or unsat-core generation.

**Eligible benchmarks.** Some benchmarks typically have to be excluded for syntactic reasons: in 2015 and 2016, 11 966 benchmarks were excluded because they contained partial or underspecified operations (e.g., bit-vector division), whose precise semantics were being debated at the time. Additionally in 2016, one benchmark was excluded because it contained incorrect status information. In 2017, 1104 benchmarks were excluded because their logic had been classified incorrectly, and 8 benchmarks were excluded because of incorrect or missing status information. Moreover, divisions containing datatypes (DT)—which were new in 2017—were excluded from the Unsat-Core track because of inadequate tool support. In 2018, 3603 benchmarks were excluded because they contained named terms (a feature of the SMT-LIB language that is not permitted in competition benchmarks), and 5 benchmarks were excluded because of missing status information.

**Benchmark selection.** SMT Competitions until 2014 employed a pseudo-random selection process to further cull the number of benchmarks used in the competition. However, a 2013 evaluation of SMT-COMP and SMT-LIB found that this "significantly lessens the ability of a competition to determine 'best' solvers" [19]. Together with the increased computational resources available to the competition since its move to StarExec, this finding prompted a paradigm shift. Since 2015, the competition is evaluating all solvers on all eligible benchmarks.

**Benchmarks with unknown status.** Another paradigm shift concerns the treatment of benchmarks with unknown status. Because of the difficulty to judge solver responses for such benchmarks, SMT Competitions until 2015 only used benchmarks whose status was known (i.e., that had been solved previously by existing SMT solvers). In 2015, 30 717 non-incremental benchmarks from the SMT Library were not used for the competition because their status was unknown. However, this approach was criticized for rewarding imitation of existing solvers rather than true innovation. In 2016, the SMT Library contained 29 724 non-incremental benchmarks with unknown status; solver performance on these was evaluated but reported separately. Since 2017, all non-incremental benchmarks are eligible for the competition's Main Track regardless of status, and they directly affect the competition results.

Solver responses for such benchmarks are assumed to be correct, unless there is a disagreement between two or more solvers that are otherwise (i.e., on benchmarks with known status) sound. Benchmarks with disagreements are removed from the competition results. This conservative approach was chosen over a voting system as the latter is unreliable (especially in divisions with few solvers), could lead to a solver being punished wrongly, and could be gamed by entering multiple copies of a solver into the competition. Moreover, the size and time frame of the competition demand an automated resolution. Disagreements are reported to the solver developers involved, who typically perform a manual investigation at their leisure, and in all cases have led to bugs in solvers being identified and fixed. In 2016, there were disagreements on 79 benchmarks; in 2017, only one benchmark was removed; while in 2018, there were no disagreements between otherwise sound solvers.

For incremental benchmarks, the situation is different: an incremental benchmark is ineligible for the Application Track if its first `check-sat` command has unknown status. Otherwise, the prefix of the benchmark that only contains `check-sat` commands with known status is eligible. The main reason for this difference is that the competition tools for the Application Track (in particular, the trace executor) had not been adapted yet to support `check-sat` commands with unknown status. Even though the trace executor has now been modified, we cannot check what impact this may have had on previous results without rerunning the Application Track as the trace executor drives the solver.

Tables 3–5 show the number of benchmarks that were used in each competition track and division since 2015. A dash indicates that no eligible benchmarks were available, and hence the corresponding SMT-LIB logic was not actually a competition division in that track and year. The size of divisions varies widely, from just one benchmark in ABVFP to 72 705 benchmarks in QF_SLIA.

## 5. Infrastructure

Since 2014 (and previously for the evaluation in 2013 [19]), the computational workload of the competition is being run on StarExec, a "cross-community infrastructure for logic solving" [41] developed at the University of Iowa and supported by the National Science Foundation. Each machine in the StarExec cluster is equipped with two E5-2609 Intel Xeon processors (10 MiB Cache, 2.4 GHz). This hardware configuration has been unchanged since 2013. Up to two solvers are run simultaneously on each machine: one solver on each processor, with four cores available. Memory usage is limited to 60 GiB per solver.

**Table 3:** Number of benchmarks used in the Main Track (by division). Numbers in parentheses indicate the (rounded) percentage of benchmarks with different statuses, which are (sat/unsat) in 2015 and 2016, and (sat/unsat/unknown) in 2017 and 2018.

| Division | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| ABVFP | – | – | – | 1 (0/0/100) |
| ALIA | 42 (2/98) | 42 (2/98) | 42 (2/98/0) | 42 (2/98/0) |
| AUFBVDTLIA | – | – | 1709 (0/1/98) | 1709 (0/1/98) |
| AUFDTLIA | – | – | 728 (0/0/100) | 728 (0/0/100) |
| AUFLIA | 4 (25/75) | 4 (25/75) | 4 (25/75/0) | 4 (25/75/0) |
| AUFLIRA | 19849 (1/99) | 19849 (1/99) | 20011 (0/99/1) | 20011 (0/99/1) |
| AUFNIRA | 1050 (0/100) | 1050 (0/100) | 1480 (0/71/29) | 1480 (0/71/29) |
| BV | 85 (34/66) | 85 (34/66) | 5151 (2/2/97) | 5751 (10/86/4) |
| BVFP | – | – | – | 24 (0/0/100) |
| FP | – | – | – | 61 (0/0/100) |
| LIA | 201 (5/95) | 201 (5/95) | 388 (38/60/2) | 388 (38/60/2) |
| LRA | 339 (6/94) | 339 (6/94) | 2419 (17/46/38) | 2419 (29/64/8) |
| NIA | 9 (67/33) | 9 (67/33) | 14 (43/29/29) | 14 (43/29/29) |
| NRA | 3788 (0/100) | 3788 (0/100) | 3811 (0/100/0) | 3813 (0/100/0) |
| QF_ABV | 14720 (68/32) | 14720 (68/32) | 15061 (69/31/0) | 15066 (69/31/0) |
| QF_ABVFP | – | – | – | 18129 (78/22/1) |
| QF_ALIA | 134 (40/60) | 139 (42/58) | 139 (42/58/0) | 139 (42/58/0) |
| QF_ANIA | 6 (0/100) | 8 (0/100) | 8 (0/100/0) | 8 (0/100/0) |
| QF_AUFBV | 37 (16/84) | 37 (16/84) | 31 (19/81/0) | 31 (19/81/0) |
| QF_AUFLIA | 1009 (49/51) | 1009 (49/51) | 1009 (49/51/0) | 1009 (49/51/0) |
| QF_AUFNIA | 21 (29/71) | 21 (29/71) | 17 (29/71/0) | 17 (29/71/0) |
| QF_AX | 551 (49/51) | 551 (49/51) | 551 (49/51/0) | 551 (49/51/0) |
| QF_BV | 26414 (35/65) | 26414 (35/65) | 40043 (32/59/8) | 40102 (36/64/0) |
| QF_BVFP | 7 (14/86) | 7 (14/86) | 17215 (81/18/0) | 17215 (81/18/0) |
| QF_DT | – | – | 8000 (0/0/100) | 8000 (45/55/0) |
| QF_FP | 34413 (50/50) | 34413 (50/50) | 40302 (50/50/1) | 40300 (50/50/1) |
| QF_IDL | 2094 (61/39) | 2094 (61/39) | 2193 (58/37/5) | 2193 (58/37/4) |
| QF_LIA | 5839 (51/49) | 5839 (51/49) | 6141 (52/46/2) | 6947 (55/43/1) |
| QF_LIRA | 6 (17/83) | 6 (17/83) | 7 (14/71/14) | 7 (14/71/14) |
| QF_LRA | 1626 (61/39) | 1626 (61/39) | 1649 (57/41/2) | 1649 (57/41/1) |
| QF_NIA | 8475 (98/2) | 8593 (96/4) | 23876 (59/13/28) | 23876 (59/20/20) |
| QF_NIRA | 2 (0/100) | 2 (0/100) | 3 (0/67/33) | 3 (0/67/33) |
| QF_NRA | 10184 (52/48) | 10245 (52/48) | 11354 (44/47/10) | 11489 (44/47/9) |
| QF_RDL | 220 (49/51) | 220 (49/51) | 255 (43/44/13) | 255 (43/44/13) |
| QF_SLIA | – | – | – | 72705 (0/0/100) |
| QF_UF | 6649 (38/62) | 6649 (38/62) | 6650 (38/62/0) | 7423 (42/58/0) |
| QF_UFBV | 31 (0/100) | 31 (0/100) | 31 (0/100/0) | 1224 (53/47/0) |
| QF_UFIDL | 441 (24/76) | 441 (24/76) | 428 (25/75/0) | 428 (25/75/0) |
| QF_UFLIA | 598 (67/33) | 598 (67/33) | 583 (69/31/0) | 583 (69/31/0) |
| QF_UFLRA | 1627 (48/52) | 1627 (48/52) | 1284 (60/40/0) | 1284 (60/40/0) |
| QF_UFNIA | 7 (0/100) | 7 (0/100) | 7 (0/100/0) | 7 (0/100/0) |
| QF_UFNRA | 34 (47/53) | 34 (47/53) | 36 (58/31/11) | 36 (58/31/11) |
| UF | 2839 (28/72) | 2839 (28/72) | 7562 (10/44/46) | 7562 (10/46/44) |
| UFBV | 71 (25/75) | 71 (25/75) | 200 (34/48/17) | 200 (34/48/17) |
| UFDT | – | – | 4535 (0/9/90) | 4527 (1/41/58) |
| UFDTLIA | – | – | 303 (0/0/100) | 303 (0/0/100) |
| UFIDL | 68 (9/91) | 68 (9/91) | 68 (4/84/12) | 68 (4/84/12) |
| UFLIA | 8404 (0/100) | 8404 (0/100) | 10137 (0/76/24) | 10137 (0/76/23) |
| UFLRA | 25 (20/80) | 25 (20/80) | 15 (33/67/0) | 15 (33/67/0) |
| UFNIA | 2319 (0/100) | 2319 (0/100) | 3308 (0/74/26) | 3308 (0/74/26) |
| Total | 154238 (40/60) | 154424 (40/60) | 256973 (37/46/17) | 333241 (33/39/28) |

**Table 4:** Number of benchmarks used in the Application Track (by division)

| Division | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| ALIA | 24 | 24 | 24 | 24 |
| ANIA | 3 | 3 | 3 | 3 |
| AUFNIRA | – | – | – | 117 |
| BV | – | – | – | 17 |
| LIA | 6 | 6 | 6 | 6 |
| QF_ABV | – | – | – | 15 |
| QF_ALIA | 44 | 44 | 44 | 44 |
| QF_ANIA | 5 | 5 | 5 | 5 |
| QF_AUFBV | – | – | – | 10 |
| QF_AUFLIA | 72 | 72 | 72 | 72 |
| QF_BV | 18 | 18 | 18 | 815 |
| QF_BVFP | – | – | 2 | 2 |
| QF_FP | – | – | 2 | 2 |
| QF_LIA | 65 | 69 | 68 | 69 |
| QF_LRA | 10 | 10 | 10 | 10 |
| QF_NIA | 10 | 10 | 10 | 10 |
| QF_UFBV | – | – | – | 2327 |
| QF_UFLIA | 905 | 905 | 780 | 780 |
| QF_UFLRA | 3331 | 3331 | 3056 | 3058 |
| QF_UFNIA | 1 | 1 | 1 | 1 |
| UFLRA | 5358 | 5358 | 1870 | 1870 |
| Total | 9852 | 9856 | 5971 | 9257 |

On the software side, the cluster is running Red Hat Enterprise Linux Workstation. The 2015 competition was running on release 6.3 (Santiago, kernel 2.6.32-431). This was updated to 7.2 (Maipo, kernel 3.10.0-327) before the 2016 competition, and the kernel was further updated to 3.10.0-514 before the 2017 competition. The platform emphasizes stability over bleeding edge technology. This has on occasion caused problems with solver binaries that had been compiled on other Linux distributions and linked against library versions not (yet) available on the cluster. Usually, such problems are easily solved by building the solver on StarExec instead. The StarExec developers have made a virtual machine image available to facilitate this and similar tasks.

Interaction with StarExec is mostly via its web interface, `https://www.starexec.org`. The interface allows registered users to upload and share benchmarks and solvers, to create and manage jobs that apply selected solvers to a selection of benchmarks, and to view and download job results. Other features, such as pre-processing of benchmarks (e.g., for scrambling) and post-processing of solver output, are also supported.

Jobs consist of individual *job pairs.* Each job pair corresponds to a specific solver/benchmark combination. Figure 2 shows the total number of job pairs for the SMT Competition from 2015 to 2018. Over this period, the number has grown by 72 %, from 1 028 615 job

**Table 5:** Number of benchmarks used in the Unsat-Core Track (by division)

| Division | 2016 | 2017 | 2018 |
|---|---:|---:|---:|
| ALIA | 41 | 41 | 41 |
| AUFBVDTLIA | – | – | 25 |
| AUFLIA | 3 | 3 | 3 |
| AUFLIRA | 19749 | 19771 | 19771 |
| AUFNIRA | 1046 | 1050 | 1053 |
| BV | 56 | 94 | 4937 |
| LIA | 191 | 233 | 233 |
| LRA | 319 | 1106 | 1539 |
| NIA | 3 | 4 | 4 |
| NRA | 3788 | 3801 | 3801 |
| QF_ABV | 4644 | 4673 | 4677 |
| QF_ABVFP | – | 3934 | 3934 |
| QF_ALIA | 80 | 80 | 80 |
| QF_ANIA | 8 | 8 | 8 |
| QF_AUFBV | 31 | 25 | 25 |
| QF_AUFLIA | 516 | 516 | 516 |
| QF_AUFNIA | 15 | 12 | 12 |
| QF_AX | 279 | 279 | 279 |
| QF_BV | 17172 | 23732 | 25700 |
| QF_BVFP | 6 | 3174 | 3174 |
| QF_DT | – | – | 4422 |
| QF_FP | 17213 | 20028 | 20026 |
| QF_IDL | 816 | 816 | 816 |
| QF_LIA | 2840 | 2844 | 3019 |
| QF_LIRA | 5 | 5 | 5 |
| QF_LRA | 633 | 671 | 683 |
| QF_NIA | 316 | 3130 | 4842 |
| QF_NIRA | 2 | 2 | 2 |
| QF_NRA | 4948 | 5296 | 5357 |
| QF_RDL | 113 | 113 | 113 |
| QF_UF | 4100 | 4101 | 4330 |
| QF_UFBV | 31 | 31 | 575 |
| QF_UFIDL | 335 | 322 | 322 |
| QF_UFLIA | 195 | 183 | 183 |
| QF_UFLRA | 853 | 511 | 511 |
| QF_UFNIA | 7 | 7 | 7 |
| QF_UFNRA | 18 | 11 | 11 |
| UF | 2039 | 3316 | 3442 |
| UFBV | 53 | 97 | 97 |
| UFDT | – | – | 1863 |
| UFIDL | 62 | 57 | 57 |
| UFLIA | 8377 | 7714 | 7743 |
| UFLRA | 20 | 10 | 10 |
| UFNIA | 2318 | 2432 | 2457 |
| Total | 93241 | 114233 | 130705 |

**Table 6:** Wall-clock time limits (in minutes)

| | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| Main Track | 40 | 40 | 20[1.] | 20 |
| Application Track | 40 | 40 | 40 | 40 |
| Unsat-Core Track | – | 40 | 40 | 40 |

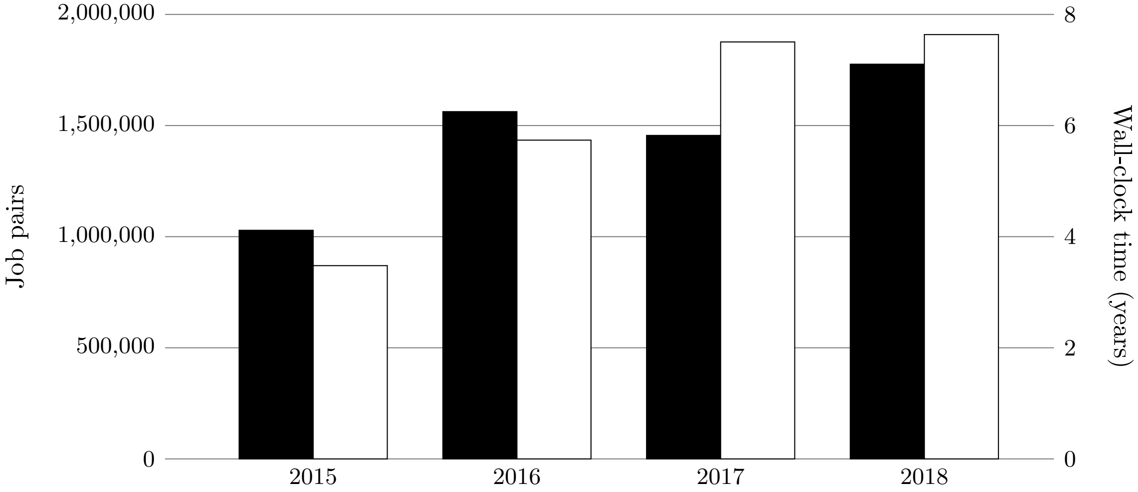


**Figure 2:** Number of job pairs (■) and total wall-clock time (□).

pairs in 2015 to 1 776 062 job pairs in 2018. (This is the number of unique solver/benchmark combinations that were part of the competition. The number of job pairs that were actually run is even higher: test jobs and job pairs that had to be re-run, e.g., because of issues with the StarExec infrastructure, are not included.) For comparison, the 2014 competition only ran 339 714 job pairs [17].

Also shown in Figure 2 is the total wall-clock time (summed over all job pairs) for the competition. This has more than doubled, from 3.5 years in 2015 to 7.6 years in 2018. These are solver run-times only; time that the cluster spent on pre- and post-processing of benchmarks and solver output or on other auxiliary tasks is not included. The increase in 2016 is largely due to the inclusion of (non-incremental) benchmarks with unknown status; these were evaluated with a wall-clock time limit of 10 minutes. In 2017, when these benchmarks had become eligible for the Main Track, Main Track job pairs were first run with a time limit of 10 minutes. When it became apparent to the organizers that there were sufficient resources to increase this limit, job pairs that had timed out were re-run with a time limit of 20 minutes. (In this way, about 2.9 years of wall-clock time in 2017 were spent on job pairs that were then re-run. This time is not included in Figure 2.) Table 6 shows the wall-clock time limits that were imposed on job pairs for each track and year.

StarExec typically reserves between 80 to 150 machines for the SMT Competition, allowing its workload to be completed within two to three weeks.

[1.] These job pairs were first run with a time limit of 10 minutes. See the text for details.

The use of StarExec ensures the replicability of the competition. The publicly accessible space https://www.starexec.org/starexec/secure/explore/spaces.jsp?id=2641 stores all of the information required to rerun the competition, e.g., all benchmarks and copies of all tools. In addition, a GitHub organization (https://github.com/SMT-COMP/smt-comp) has been set up to archive previous results and store the tools used by organizers to run the competition.

## 6. Participants

Participants must upload their solvers to StarExec to enter them into the competition, and additionally submit information about each competing solver—such as the tracks and divisions into which the solver is being entered—to the competition organizers. Until 2016, this information was collected by email. Since 2017, a web form is being used for this purpose. This has greatly reduced the number of incomplete or ambiguous submissions.

Participants are required to be authors of the tool they submit. In 2015–2018, the competition rules did not further define authorship, but it is understood that an author contributed code to the tool. As a special case of this rule, participants are required to identify if their tool is a wrapper tool, i.e., any solver that calls (or is in some other way based on) one or more SMT solvers of which the author of the tool is not an author.

Tables 7–9 list the solvers that participated between 2015 and 2018, together with contributing team members (in alphabetical order) and their institutional affiliations (if any; these may have changed over time) as reported by the development teams.

Further, to promote a wide comparison between tools, there was no restriction on the number of solver versions participants may submit, e.g., for different competition tracks or with differently tuned internal components. Organizers reserved the right to not accept multiple versions of a solver in case the number of solver submissions is too large for the computational resources, but in the past four years, exercising this right was not necessary. Table 10 shows in more detail how many versions of each solver were submitted in each year.

Participants were not required to provide source code with their submission. There were no licensing requirements/restrictions for the submitted binaries of the participating solvers. Participants were encouraged to submit a short system description of their solver, including a list of its authors and an explanation of the SMT solving approach used. System descriptions are published on the competition website. They were submitted for about 60 % of all solver versions since 2015.

In addition to the solvers that were submitted to the competition by their respective developers, the organizers included the most recent stable versions of MathSAT and Z3 for comparison purposes. Both solvers are strong tools, but—except for two experimental versions of Z3 in 2015—their development teams chose not to prepare competition versions. These solvers were therefore participating hors concours. Moreover, in 2015 a bug was discovered in the Application Track version of Boolector and a fixed version was submitted after the deadline; in 2016, the CVC4 team did not enter their solver into the Application Track; and in 2018, an experimental version of CVC4 as well as the OpenSMT2 solver were submitted after the deadline. These solvers were also participating hors concours. In result tables, they are listed with their name in square brackets (e.g., [MathSAT]).

**Table 7:** Participants (2015–2018) Part I

| Solver | Team members (affiliations) |
|---|---|
| ABC [14] | Valeriy Balabanov (Mentor Graphics)<br>Robert Brayton (UC Berkeley)<br>Alan Mischenko (UC Berkeley) |
| Alt-Ergo [20] | Sylvain Conchon (CNRS, University of Paris Sud)<br>Albin Coquereau (ENSTA, University of Paris Sud)<br>Mohamed Iguernlala (OCamlPro SAS)<br>Alain Mebsout (OCamlPro SAS) |
| AProVE [29] | Cornelius Aschermann (RWTH Aachen University)<br>Karsten Behrmann (RWTH Aachen University)<br>Marc Brockschmidt (Microsoft Research)<br>Andrej Dyck (RWTH Aachen University)<br>Fabian Emmes (RWTH Aachen University)<br>Florian Frohn (RWTH Aachen University)<br>Carsten Fuhs (University College London; Birkbeck, University of London)<br>Jürgen Giesl (RWTH Aachen University)<br>Jera Hensel (RWTH Aachen University)<br>Patrick Kabasci (RWTH Aachen University)<br>Carsten Otto<br>Martin Plücker (RWTH Aachen University)<br>Peter Schneider-Kamp (University of Southern Denmark)<br>Thomas Ströder (RWTH Aachen University)<br>Stephanie Swiderski<br>René Thiemann (University of Innsbruck) |
| Boolector [37] | Armin Biere (Johannes Kepler University)<br>Aina Niemetz (Johannes Kepler University, Stanford University)<br>Mathias Preiner (Johannes Kepler University, Stanford University) |
| COLIBRI [36] | Benjamin Blanc (CEA, List)<br>François Bobot (CEA, List)<br>Zakaria Chihani (CEA, List)<br>Bruno Marre (CEA, List)<br>Patricia Mouy (CEA, List)<br>Franck Vedrine (CEA, List) |
| Ctrl-Ergo [12] | Mohamed Iguernlala (OCamlPro SAS) |
| CVC3 [9] | Kshitij Bansal (New York University)<br>Clark Barrett (New York University)<br>Morgan Deters (New York University) |

**Table 8:** Participants (2015–2018) Part II

| Solver | Team members (affiliations) |
|---|---|
| CVC4 [3] | Kshitij Bansal (New York University)<br>Haniel Barbosa (University of Iowa)<br>Clark Barrett (New York University, Stanford University)<br>François Bobot (CEA)<br>Martin Brain (Oxford University)<br>Christopher Conway (Google)<br>Morgan Deters (New York University)<br>Liana Hadarean (Oxford University)<br>Duligur Ibeling (Stanford University)<br>Dejan Jovanović (SRI International)<br>Timothy King (Verimag)<br>Tianyi Liang (University of Iowa)<br>Paul Meng (University of Iowa)<br>Aina Niemetz (Stanford University)<br>Andres Nötzli (Stanford University)<br>Mathias Preiner (Stanford University)<br>Andrew Reynolds (EPFL, University of Iowa)<br>Cesare Tinelli (University of Iowa) |
| MapleSTP | Vijay Ganesh (University of Waterloo)<br>Jimmy Liang (University of Waterloo) |
| MinkeyRink | Trevor Hansen (University of Melbourne) |
| OpenSMT2 [33] | Matteo Marescotti (University of Lugano)<br>Antti Hyvärinen (University of Lugano) |
| ProB [35] | Sebastian Krings (University of Düsseldorf)<br>Michael Leuschel (University of Düsseldorf) |
| Q3B | Martin Jonáš (Masaryk University) |
| raSAT [42] | Mizuhito Ogawa (Japan Advanced Institute of Technology)<br>To Van Khanh (Vietnam National University, Hanoi)<br>Vu Xuan Tung (Japan Advanced Institute of Technology) |
| Redlog [25] | Haniel Barbosa (Inria)<br>Marek Kosta (Slovak Academy of Sciences, Bratislava)<br>Thomas Sturm (CNRS, MPI Saarbrücken) |
| SMT-RAT [22] | Erika Ábrahám (RWTH Aachen University)<br>Florian Corzilius (RWTH Aachen University)<br>Rebecca Haehn (RWTH Aachen University)<br>Sebastian Junges (RWTH Aachen University)<br>Gereon Kremer (RWTH Aachen University)<br>Stefan Schupp (RWTH Aachen University) |

**Table 9:** Participants (2015–2018) Part III

| Solver | Team members (affiliations) |
|---|---|
| SMTInterpol [16] | Jürgen Christ (University of Freiburg)<br>Jochen Hoenicke (University of Freiburg)<br>Tanja Schindler (University of Freiburg) |
| SPASS-SATT [15] | Martin Bromberger (Max Planck Institute for Informatics)<br>Christoph Weidenbach (Max Planck Institute for Informatics) |
| STP [28] | Vijay Ganesh (University of Waterloo)<br>Ryan Govostes (Woods Hole Oceanographic Institute)<br>Trevor Hansen (University of Melbourne)<br>Dan Liew (Imperial College London)<br>Norbert Manthey (CoNP Solutions)<br>Khoo Yit Phang (University of Maryland)<br>Mate Soos (National University of Singapore) |
| toysmt | Masahiro Sakai (Toshiba Corporation) |
| Vampire [34] | Evgeny Kotelnikov (Chalmers University)<br>Laura Kovács (TU Wien, Chalmers University)<br>Giles Reger (University of Manchester)<br>Simon Robillard (Chalmers University)<br>Martin Suda (University of Manchester, TU Wien)<br>Andrei Voronkov (University of Manchester, TU Wien) |
| veriT [13] | David Déharbe (Federal University of Rio Grande do Norte)<br>Haniel Barbosa (Inria)<br>Pablo Federico Dobal (University of Lorraine)<br>Pascal Fontaine (University of Lorraine)<br>Daniel El Ouraoui (Inria)<br>Hans-Jörg Schurr (Inria) |
| XSat [27] | Zhoulai Fu (UC Davis)<br>Zhendong Su (UC Davis)<br>Martin Velez (UC Davis) |
| Yices [26] | Bruno Dutertre (SRI International)<br>Dejan Jovanović (SRI International)<br>Ian Mason (SRI International) |
| Z3 [24] | Christoph Wintersteiger (Microsoft Research)<br>Aleksandar Zeljić (Microsoft Research, Uppsala University) |

**Table 10:** Number of versions submitted for each solver (by year). Numbers in square brackets indicate solver versions that were participating hors concours.

| Solver | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| ABC | | 2 | | |
| Alt-Ergo | | | | 1 |
| AProVE | 1 | 1 | 1 | 1 |
| Boolector | 3 [+1] | 2 | 2 | 2 |
| COLIBRI | | | 1 | 1 |
| Ctrl-Ergo | | | | 1 |
| CVC3 | 2 | | | |
| CVC4 | 4 | 1 [+1] | 3 | 3 [+1] |
| MapleSTP | | 4 | | |
| [MathSAT] | [+2] | [+3] | [+3] | [+3] |
| MinkeyRink | | 1 | 1 | 2 |
| OpenSMT2 | 2 | 1 | 1 | [+1] |
| ProB | | 1 | | |
| Q3B | | 1 | 1 | 1 |
| raSAT | 1 | 2 | | |
| Redlog | | | 1 | |
| SMT-RAT | 2 | 1 | 1 | 2 |
| SMTInterpol | 1 | 1 | 1 | 1 |
| SPASS-SATT | | | | 1 |
| STP | 4 | 8 | 2 | 3 |
| toysmt | | 1 | | |
| Vampire | | 2 | 1 | 1 |
| veriT | 1 | 1 | 3 | 2 |
| XSat | | | 1 | |
| Yices | 3 | 2 | 2 | 3 |
| Z3 | 2 [+1] | [+1] | [+1] | [+1] |
| Total | 26 [+4] | 32 [+5] | 22 [+4] | 25 [+6] |

Table 11 summarizes these numbers and shows how many solver versions were submitted to each track of the competition. Note that solver versions may be entered into multiple tracks. Therefore, the total number of solver versions for each year is typically less than the sum over all tracks.

It can be easy for developers to create multiple versions of their solver, which may differ only in configuration settings or other minor details. We caution against over-interpreting these numbers, which to some extent depend on how adventurous development teams were in any given year.

By number of solver versions submitted, the competitions in 2015–2018 were the four largest in the history of SMT-COMP. On average, 26 solver versions were submitted each

**Table 11:** Participation by track and year. Numbers in square brackets indicate solver versions that were participating hors concours.

| | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| Main Track | 21 [+2] | 25 [+2] | 19 [+2] | 20 [+4] |
| Application Track | 10 [+3] | 8 [+3] | 4 [+2] | 4 [+2] |
| Unsat-Core Track | — | 1 [+4] | 2 [+2] | 3 [+2] |
| Total | 26 [+4] | 32 [+5] | 22 [+4] | 25 [+6] |

year since 2015. In contrast, the competitions from 2005–2014 only received an average of 12 submissions [17].

## 7. Scoring

A detailed description of the scoring can be found in the competition rules [31]. We summarize the most important aspects in this section.

### 7.1 Division Rankings

Solvers in each division (of each track) are ranked according to a metric that is based, first, on the number of erroneous responses (which is usually 0, but solvers that did give erroneous responses are ranked below solvers that did not); second, on the number of correct responses; third and fourth, on the wall-clock and CPU time, respectively, that the solver process consumed. The *raw benchmark score* of a solver is thus a quadruple $\langle e, n, w, c\rangle$, with

- $e \geq 0$ number of erroneous responses (usually $e = 0$),
- $0 \leq n \leq N$ number of correct responses (resp. *reduction* for the Unsat-Core Track),
- $w \in [0, T]$ wall-clock time in seconds (real-valued),
- $c \in [0, 4T]$ CPU time in seconds (real-valued).

For the Main Track and Application Track, $e$ is the number of returned statuses that disagree with the given expected status; $e \in \{0, 1\}$ for the Main Track. For the Unsat-Core Track, $e$ includes, in addition, the number of returned unsat cores that are ill-formed or are not, in fact, unsatisfiable (as validated by a selection of solvers selected by the organizers).

For the Main Track and Application Track, $N$ is defined as the number of `check-sat` commands ($N = 1$ for the Main Track), and $n$ is defined as the number of correct responses. For the Unsat-Core Track (which uses unsatisfiable benchmarks only), $N$ is defined as the number of named top-level assertions, and $n$ is defined as the *reduction*, i.e., the difference between $N$ and the size of the unsat core returned by the solver.

As discussed in Section 4, Main Track benchmarks with unknown status are removed from the competition results if two (or more) solvers that are sound on benchmarks with known status disagree on their result. Otherwise, solver responses for benchmarks with unknown status are assumed to be correct.

A division is *competitive* if at least two substantially different solvers (i.e., solvers from two different teams) were submitted. (Experimental divisions are never competitive.) Official winners were declared only for competitive divisions.

**Parallel vs. sequential performance.** All solvers are run with four cores available, and the time limit $T$ that is imposed on each job pair (see Table 6) is a wall-clock limit. Thus, solvers that take advantage of parallelism might use up to $4T$ CPU time. For the Main Track and the Unsat-Core Track, the competition reports parallel and sequential performance separately. The raw benchmark score corresponds to the *raw parallel score*—all values are determined for solver performance within the wall-clock time limit. We derive a *raw sequential score* by imposing a (virtual) CPU time limit equal to the wall-clock time limit $T$. A solver response is taken into consideration for the sequential score only if the solver process terminates within this CPU time limit. For the Application Track, only parallel performance is reported (because response times to individual `check-sat` commands are not recorded by StarExec).

**Benchmark weights.** Until 2015, the competition used these raw scores for all tracks to determine the division score of a solver as the sum of the scores of all benchmarks in that division. However, there are vast differences in size between benchmark *families* (i.e., groups of benchmarks with similar characteristics) in the SMT Library. The pseudo-random selection process of benchmarks that was used until 2014 (see Section 4) ensured that large families were not overemphasized in the competition. Since 2016, the competition uses a scoring method that assigns a weight to each benchmark in the Main Track and Unsat-Core Track to achieve the same goal [21, 32, 31].

The *weight* of a benchmark $b$ in the Main Track is computed as $\alpha_b = (1 + \ln F_b)/F_b$, where $F_b \geq 1$ is the size of the benchmark's family. We then define its *normalized weight* as $\alpha'_b = \alpha_b / (\sum_{b'} \alpha_{b'})$, where the sum is over all benchmarks in the division. The logarithmic scaling seems a reasonable compromise between linearly combining numbers of benchmarks, which would overemphasize large families, and simply averaging the fraction of benchmarks solved, which would overemphasize small families.

The *(weighted) score* of a solver in a (Main Track or Unsat-Core Track) division $i$ is then defined as the weighted sum of all raw scores

$$\sum_{b \in i} \alpha'_b \cdot \langle e_b \cdot N_i, n_b \cdot N_i, w_b, c_b \rangle$$

where the sum is over all benchmarks in the division, and $N_i$ is the number of benchmarks in $i$. This is computed separately for parallel and sequential scores.

For Application Track divisions, division scores were still based on raw scores only, since Application Track benchmarks may be partially solved.

### 7.2 Competition-Wide Ranking (Main Track)

In 2014, the SMT Competition was part of the FLoC Olympic Games, which sponsored three Kurt Gödel medals. These were awarded to the winner of the Main Track's QF_BV division (which had historically been the logic with the largest number of solver submissions and job pairs), as well as to the two best Main Track solvers according to a competition-wide ranking that had been instituted especially for this purpose [17]. This ranking combines each solver's performance across all competitive Main Track divisions into a single score, giving more weight to larger divisions, and to solvers that solve a large fraction of the benchmarks [31]. It emphasizes breadth of solver participation—a solver participating in many logics does not need to be the best in any one of them to rank in high position.

The competition-wide ranking for the Main Track was retained and refined after 2014. Sequential and parallel performance are reported separately. Let $N_i$ be the total number of benchmarks in division $i$ that were used in the competition (and not removed because of disagreements), and let $\langle e_i, n_i, w_i, c_i \rangle$ be a solver's raw score for this division. In 2015 and 2016, the *competition-wide raw score* for a solver was defined as

$$\sum_i (e_i == 0\,?\,(n_i/N_i)^2 : -e_i) \ln N_i$$

where the sum is over all competitive divisions into which the solver was entered. In 2017, the (per-division) penalty for erroneous results was changed from the number of errors to a fixed constant, since the division weight $\ln N_i$ already takes the number of benchmarks in the division into account, and multiplying it by the number of errors overemphasizes large divisions. In 2017 and 2018, the *competition-wide raw score* for a solver was thus defined as

$$\sum_i (e_i == 0\,?\,(n_i/N_i)^2 : -4) \ln N_i$$

The value $-4$ was chosen to balance the strong interest in correct solvers against the risk of stifling innovation. In the division rankings (Section 7.1), an incorrect response causes a solver to be ranked below all correct solvers for that division. But the competition's Main Track consists of many independent divisions; a solver bug that affects one division need not manifest itself in other divisions. With this value, entering a (possibly buggy) solver that can solve all benchmarks into a division has a positive expected score if the probability that a soundness bug in the solver will be triggered is below 20 %.

## 8. Results

### 8.1 Division Rankings

Tables 12–14 show the highest-ranked solvers for all (competitive and non-competitive) divisions from 2015–2018. We include solvers that were running hors concours (and hence non-competing and ineligible to be recognized as official competition winners). Non-competitive divisions use gray font. Entries $A[B]$ indicate that solver $A$ was the highest competing solver, but non-competing solver $B$ actually ranked best if non-competing solvers were taken into consideration. Entries $A|B$ indicate that solver $A$ ranked first considering sequential performance, while $B$ ranked first considering parallel performance. When the same solver ranked first in both, which is most often the case, its name appears just once. Table cell colors (other than white) indicate the best competing solver performance; no cell color indicates that the best solver performance was by a non-competing solver and no competing solvers participated in that division. Full division rankings are available from the competition website.

It is interesting to note that the non-competing solvers MathSAT and Z3 often appear as the best performing solver. We view this as confirmation of the industrial significance of the competition results. Both solvers are industrial-strength tools, actively maintained and tuned for applications. The reasons for non-entry into the competition are orthogonal to their suitability for the competition. Indeed, Z3 has been entered into the competition competitively in the past, and one can assume that it was trained on the current benchmarks. The fact that industrial-strength solvers perform well in the competition supports the notion that the benchmarks in the competition are industrially relevant.

### 8.2 Competition-Wide Ranking

Table 15 shows the best three Main Track solvers according to the competition-wide ranking for 2015–2018. Also shown is Z3, which ranked in high position each year but was running hors concours. Sequential and parallel performance are reported separately. Note that scores are not directly comparable between different years: the set of benchmarks changes from one year to the other, and also the formula to compute the score changed (see Section 7.2). Moreover, readers should keep in mind that this ranking—like any method that maps the performance of solvers for a broad range of logics to a one-dimensional value—is inherently limited. We refer to the individual division rankings (Section 8.1) for more detailed information about each solver.

The 2018 edition of the competition was part of the FLoC Olympic Games, which sponsored seven medals. The competition organizers awarded the following solvers:

- The best three Main Track solvers according to the competition-wide ranking (CVC4, Yices, and SMTInterpol; see Table 15),
- the solvers that won the QF_BV division in the Main Track (Boolector and Minkey-Rink; see Table 12),
- the solver that won the most competitive divisions in the Application Track (Yices; see Table 13),
- the solver that won the most competitive divisions in the Unsat-Core Track (Yices; see Table 14).

In comparison to the competition-wide ranking for the Main Track, the metric used for the Application Track and Unsat-Core Track—simply counting the number of competitive divisions that a solver has won—was rather unsophisticated. Future editions of the competition should perhaps refine this to use a ranking scheme that is more similar to the Main Track ranking, by weighing divisions according to their relative importance, and also by rewarding good performance from solvers other than division winners. A difficulty in adapting the Main Track ranking for the other tracks is that each Main Track benchmark is worth one point (before benchmark weights are applied) if solved correctly, while benchmarks in the Application Track and Unsat-Core Track are worth widely varying numbers of points. Moreover, in the Unsat-Core Track, the size of the smallest unsatisfiable core—and hence the number of points attainable on any given benchmark—may not be known precisely.

**Table 12:** Best Main Track solvers (by division)

| Division | 2015 | 2016 | 2017 | 2018 |
|---:|:---:|:---:|:---:|:---:|
| ABVFP | | | | CVC4 |
| ALIA | CVC4 [Z3] | CVC4 [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| AUFBVDTLIA | | | CVC4 | CVC4 |
| AUFDTLIA | | | CVC4 | CVC4 |
| AUFLIA | CVC4 | CVC4 | CVC4 | CVC4 |
| AUFLIRA | CVC4 [Z3] | Vampire [Z3] | Vampire [Z3] | CVC4 [Z3] |
| AUFNIRA | CVC4 | Vampire | Vampire | CVC4 |
| BV | CVC4 [Z3] | Q3B | Q3B [Z3] | CVC4 |
| BVFP | | | | CVC4 |
| FP | | | | CVC4 |
| LIA | CVC4 | CVC4 | CVC4 [Z3] | CVC4 [Z3] |
| LRA | CVC4 | CVC4 | CVC4 [Z3] | CVC4 [Z3] |
| NIA | CVC4 [Z3] | ProB [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| NRA | CVC4 | Vampire | Redlog | Vampire [Z3] │ Vampire |
| QF_ABV | Boolector | Boolector | Boolector | Boolector |
| QF_ABVFP | | | — | CVC4 |
| QF_ALIA | Yices | Yices | Yices | Yices |
| QF_ANIA | CVC4 [Z3] | CVC4 | CVC4 | CVC4 [Z3] |
| QF_AUFBV | CVC4 [MathSAT] | CVC4 [MathSAT] | Yices [MathSAT] | CVC4 |
| QF_AUFLIA | Yices | Yices | Yices | Yices |
| QF_AUFNIA | CVC4 | CVC4 | CVC4 [Z3] | CVC4 [Z3] |
| QF_AX | Yices | Yices | Yices | Yices |
| QF_BV | Boolector | Boolector | Boolector │ MinkeyRink | Boolector │ MinkeyRink |
| QF_BVFP | Z3 | [Z3] | COLIBRI [Z3] | CVC4 |
| QF_DT | | | CVC4 | CVC4 |
| QF_FP | Z3 | [MathSAT] | COLIBRI [Z3] | COLIBRI |
| QF_IDL | Yices [Z3] | Yices [Z3] | Yices | Yices |
| QF_LIA | CVC4 [MathSAT] | CVC4 [MathSAT] | CVC4 [MathSAT] | SPASS-SATT |
| QF_LIRA | Yices | Yices | Yices [Z3] | Yices [Z3] |
| QF_LRA | CVC4 | CVC4 | CVC4 | CVC4 |
| QF_NIA | AProVE [Z3] | Yices [Z3] | CVC4 | CVC4 |
| QF_NIRA | CVC4 | CVC4 | SMT-RAT | SMT-RAT |
| QF_NRA | Yices [Z3] | Yices [Z3] | Yices | Yices [Z3] |
| QF_RDL | Yices | Yices | Yices | Yices |
| QF_SLIA | | | | CVC4 |
| QF_UF | Yices | Yices | Yices | Yices |
| QF_UFBV | Boolector | Boolector | Boolector | Boolector |
| QF_UFIDL | Yices | Yices | Yices | Yices |
| QF_UFLIA | Yices [Z3] | Yices [Z3] | Yices | Yices |
| QF_UFLRA | Yices | Yices | Yices | Yices |
| QF_UFNIA | CVC4 | Yices │ CVC4 | Yices | Yices |
| QF_UFNRA | CVC3 [Z3] | Yices | Yices [Z3] | Yices |
| UF | CVC4 | CVC4 | Vampire | CVC4 │ Vampire |
| UFBV | CVC4 [Z3] | CVC4 [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| UFDT | | | CVC4 | CVC4 |
| UFDTLIA | | | Vampire | CVC4 |
| UFIDL | CVC4 [Z3] | CVC4 | CVC4 | CVC4 [Z3] |
| UFLIA | CVC4 | CVC4 | CVC4 | CVC4 |
| UFLRA | CVC3 | Vampire [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| UFNIA | CVC4 | Vampire | Vampire | Vampire [Z3] │ Vampire |

**Table 13:** Best Application Track solvers (by division)

| Division | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| ALIA | CVC4 [Z3] | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| ANIA | CVC4 [Z3] | [Z3] | CVC4 | CVC4 |
| AUFNIRA | | | | CVC4 |
| BV | | | | CVC4 [Z3] |
| LIA | CVC4 [Z3] | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_ABV | | | | Boolector |
| QF_ALIA | Yices [Z3] | SMTInterpol [Z3] | SMTInterpol [Z3] | SMTInterpol [Z3] |
| QF_ANIA | CVC4 [Z3] | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_AUFBV | | | | Yices |
| QF_AUFLIA | Yices | Yices | Yices | Yices |
| QF_BV | Yices [MathSAT] | Yices [MathSAT] | Yices [MathSAT] | Yices [MathSAT] |
| QF_BVFP | | | [Z3] | CVC4 |
| QF_FP | | | [Z3] | CVC4 [Z3] |
| QF_LIA | Yices | Yices | Yices | Yices |
| QF_LRA | Yices [MathSAT] | SMTInterpol [MathSAT] | SMTInterpol [MathSAT] | Yices [MathSAT] |
| QF_NIA | CVC4 [Z3] | [ CVC4] | CVC4 | CVC4 |
| QF_UFBV | | | | Boolector |
| QF_UFLIA | CVC4 [Z3] | Yices [Z3] | CVC4 [Z3] | SMTInterpol [Z3] |
| QF_UFLRA | Yices [Z3] | Yices [Z3] | Yices | Yices [Z3] |
| QF_UFNIA | CVC4 [Z3] | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| UFLRA | CVC4 [Z3] | [Z3] | CVC4 [Z3] | CVC4 [Z3] |

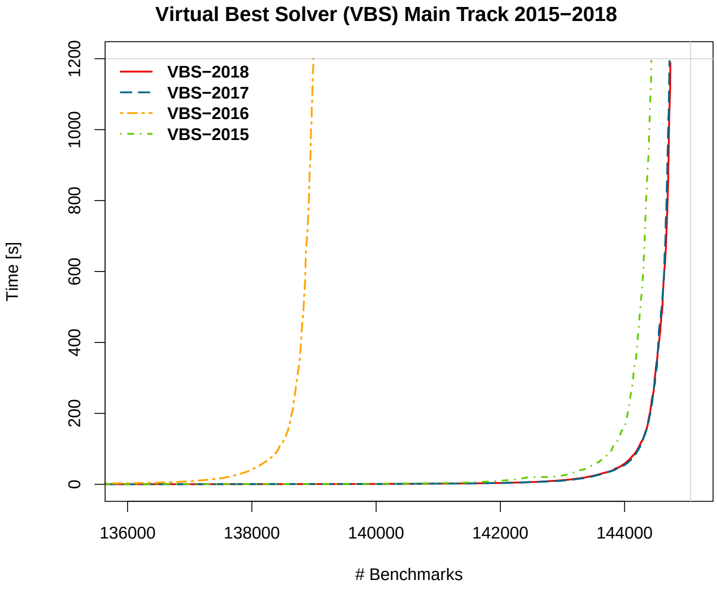


**Figure 3:** VBS on common benchmarks 2015–2018 of all divisions in the Main Track with a time limit of 1200 s. Number of common benchmarks is indicated with a gray vertical line.

**Table 14:** Best Unsat-Core Track solvers (by division)

| Division | 2016 | 2017 | 2018 |
|---:|:---:|:---:|:---:|
| ALIA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| AUFBVDTLIA | | | CVC4 |
| AUFLIA | [Z3] | CVC4 [Z3] | CVC4 |
| AUFLIRA | [Z3] | CVC4 [Z3] | CVC4 |
| AUFNIRA | [Z3] | CVC4 | CVC4 |
| BV | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| LIA | [veriT] | CVC4 | CVC4 [Z3] |
| LRA | [veriT] | CVC4 | CVC4 |
| NIA | [Z3] | CVC4 \| CVC4 [Z3] | CVC4 |
| NRA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_ABV | [Z3] | CVC4 [Z3] | Yices |
| QF_ABVFP | | — | CVC4 |
| QF_ALIA | SMTInterpol [Z3] | SMTInterpol [Z3] | Yices [Z3] |
| QF_ANIA | [Z3] | CVC4 | CVC4 |
| QF_AUFBV | [Z3] | CVC4 [MathSAT] | Yices [MathSAT] |
| QF_AUFLIA | SMTInterpol [Z3] | CVC4 | CVC4 |
| QF_AUFNIA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_AX | SMTInterpol [Z3] | SMTInterpol [Z3] | Yices |
| QF_BV | [MathSAT] | CVC4 [MathSAT] | Yices |
| QF_BVFP | [Z3] | [Z3] | CVC4 [Z3] |
| QF_DT | | | CVC4 |
| QF_FP | [MathSAT] | [Z3] | CVC4 |
| QF_IDL | SMTInterpol [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_LIA | SMTInterpol [Z3] | SMTInterpol \| SMTInterpol [Z3] | SMTInterpol |
| QF_LIRA | SMTInterpol [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_LRA | SMTInterpol | SMTInterpol | SMTInterpol |
| QF_NIA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_NIRA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_NRA | [Z3] | CVC4 | CVC4 |
| QF_RDL | SMTInterpol [veriT] | CVC4 [Z3] | CVC4 [Z3] |
| QF_UF | SMTInterpol [MathSAT] | CVC4 | CVC4 |
| QF_UFBV | [MathSAT] | CVC4 [MathSAT] | Yices [Z3] |
| QF_UFIDL | SMTInterpol [Z3] | SMTInterpol [Z3] | Yices |
| QF_UFLIA | SMTInterpol [MathSAT] | SMTInterpol [Z3] | SMTInterpol [Z3] |
| QF_UFLRA | SMTInterpol [Z3] | SMTInterpol [MathSAT] | Yices [MathSAT] |
| QF_UFNIA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| QF_UFNRA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| UF | [Z3] | CVC4 | CVC4 |
| UFBV | [Z3] | CVC4 [Z3] | CVC4 |
| UFDT | | | CVC4 |
| UFIDL | [Z3] | CVC4 | CVC4 |
| UFLIA | [Z3] | CVC4 | CVC4 |
| UFLRA | [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| UFNIA | [Z3] | CVC4 | CVC4 |

**Table 15:** Competition-wide ranking for the Main Track: best solvers (by year)

2015

| Rank | Solver | Score (sequential) | Score (parallel) |
|---|---|---|---|
| | [Z3] | 159.36 | 159.36 |
| 1. | CVC4 | 144.67 | 144.74 |
| 2. | CVC4(exp)[2.] | 140.47 | 140.51 |
| 3. | Yices | 101.91 | 101.91 |

2016

| Rank | Solver | Score (sequential) | Score (parallel) |
|---|---|---|---|
| | [Z3] | 185.09 | 185.09 |
| 1. | CVC4 | 180.95 | 181.19 |
| 2. | Yices | 119.29 | 119.29 |
| 3. | veriT | 75.11 | 75.11 |

2017

| Rank | Solver | Score (sequential) | Score (parallel) |
|---|---|---|---|
| | [Z3] | 171.99 | 171.99 |
| 1. | CVC4 | 161.38 | 161.76 |
| 2. | Yices | 110.63 | 110.63 |
| 3. | SMTInterpol | 65.96 | 66.00 |

2018

| Rank | Solver | Score (sequential) | Score (parallel) |
|---|---|---|---|
| 1. | CVC4 | 211.99 | 211.99 |
| | [Z3] | 186.19 | 186.19 |
| 2. | Yices | 115.26 | 115.26 |
| 3. | SMTInterpol | 65.32 | 65.38 |

## 9. Further Analysis

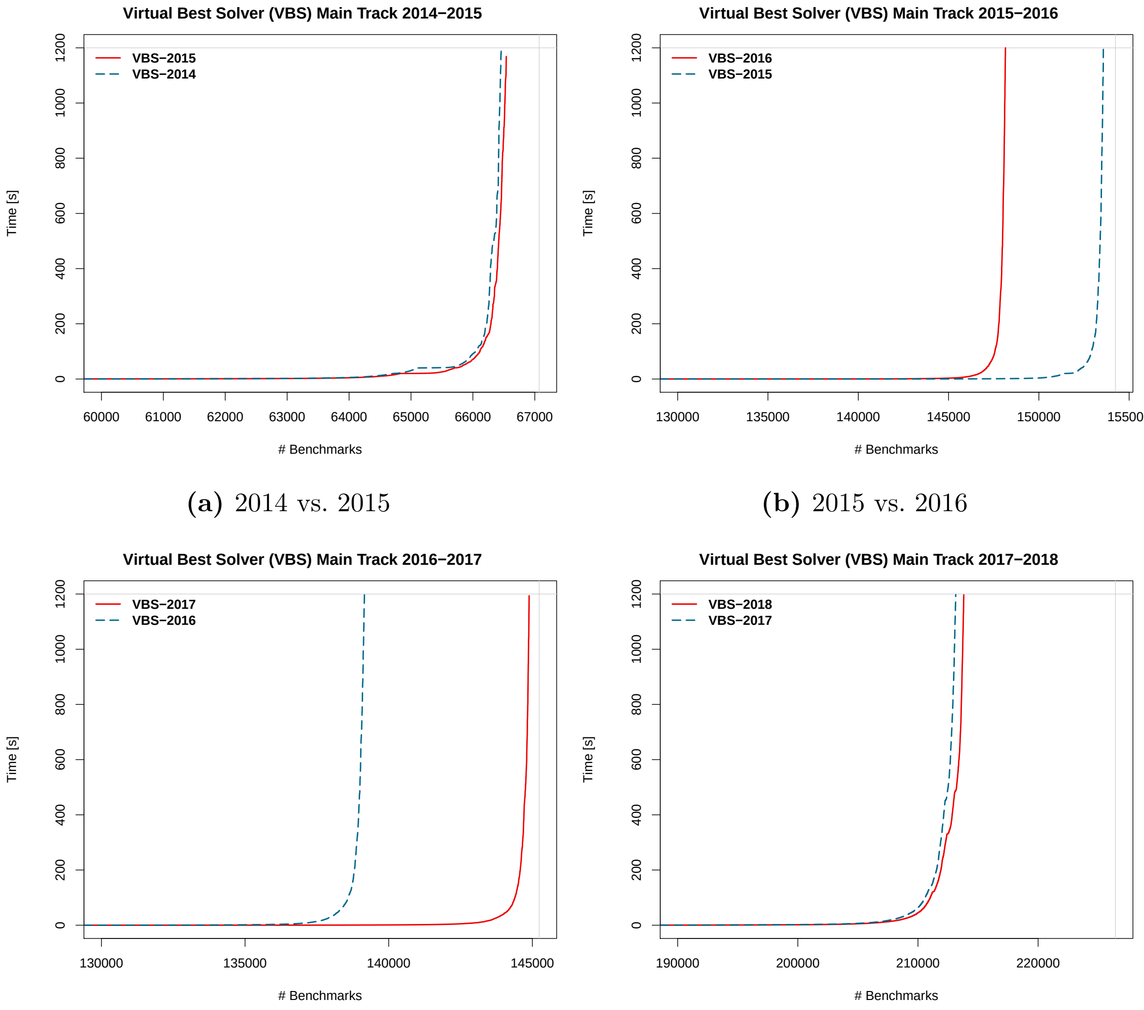
Beyond the competition rankings, the competition data provides ample opportunity for further analysis. Here, we report on several additional aspects: progress in solver performance, number of uniquely solved benchmarks, use of parallelism in SMT solvers, performance on satisfiable versus unsatisfiable benchmarks, performance for a low time limit, and the impact of benchmark families and of benchmarks with unknown status.

### 9.1 Progress in Solver Performance

The SMT Competition consists of several tracks with a multitude of divisions per track. Not all participating solvers enter all tracks, and the majority of solvers participates only in a handful of divisions. To give a measure of overall progress in solver performance we use, for each of the three tracks, the notion of a *virtual best solver* (VBS) over all divisions.

For the Main Track, we determine the VBS as the best wall-clock performance per (correctly solved) instance with the time limit of 1200 s used in the competitions 2017 and 2018. Figure 3 visualizes VBS performance on common benchmarks from 2015–2018 as cactus plot. Figure 4 shows VBS performance on common benchmarks over pairs of consecutive years from 2014 to 2018. Both plots show the wall-clock solving time per instance over all instances, sorted by solving time. Between 2014 (67 426 benchmarks) and 2015 (154 238 benchmarks), the number of common benchmarks is 67 070; between 2015 and 2016 (154 424 benchmarks) it is 154 238; between 2016 and 2017 (238 758 benchmarks) it is 145 236; and between 2017 and 2018 (333 241 benchmarks) it is 226 429. The number of solved instances on these sets of common benchmarks increased from 2014 to 2015 by 0.1 %, from 2016 to 2017 by 3.9 %, and from 2017 to 2018 by 0.3 %. Interestingly, it decreased from 2015 to 2016 by 3.5 %. This is mainly due to the fact that, in 2015, more than 5600

[2.] This was a distinct version of CVC4 that was designated as experimental.



**(a)** 2014 vs. 2015

**(b)** 2015 vs. 2016

**(c)** 2016 vs. 2017

**(d)** 2017 vs. 2018

**Figure 4:** VBS on common benchmarks for pairs of years of all Main Track divisions with a time limit of 1200 s. Number of common benchmarks is indicated with a gray vertical line.

benchmarks of division QF_FP were solved by one of the two participating configurations of Z3 but by no other solver, and in 2016 Z3 did not participate in this division.

For the Application Track, we determine the VBS as the best performance in terms of number of correct `check-sat` answers within the time limit of 2400 s (the time limit of the Application Track in the competitions 2015–2018). The total number of eligible `check-sat` queries (i.e., all queries with status sat or unsat up to the first query with status unknown) on common benchmarks from 2015–2018 is 30 370 517. Figure 5 visualizes VBS performance on these benchmarks as a bar plot. In 2015, already 99.95 % of the eligible queries were answered correctly, which naturally does not leave a lot of room for improvement. In 2016, performance decreased by 185 625 correct answers (0.61 %), which can be entirely attributed to a performance regression of Z3 on two benchmarks in the QF_NIA division.

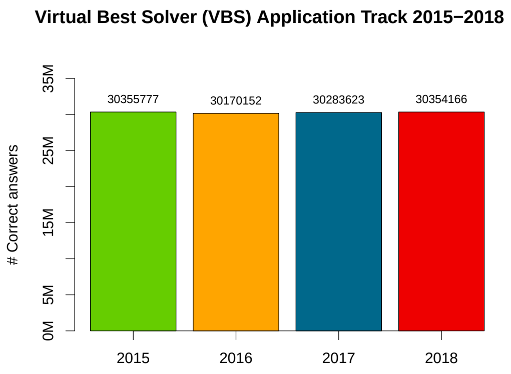


**Figure 5:** VBS in terms of correct `check-sat` answers (out of 30 370 517 total) within a time limit of 2400 s on common benchmarks 2015–2018 of all divisions in the Application Track.

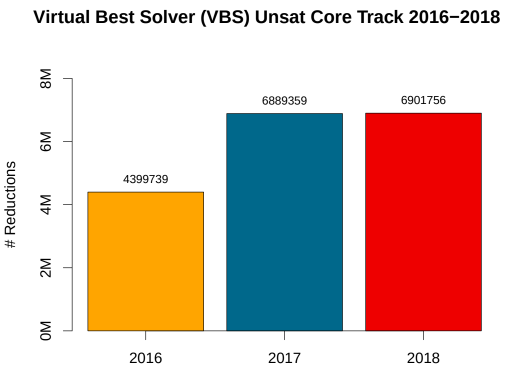


**Figure 6:** VBS in terms of number of unsat core reductions within a time limit of 2400 s on common benchmarks 2016–2018 of all divisions in the Unsat-Core Track.

For the Unsat-Core Track, we determine the VBS as the best performance in terms of reduction, i.e., the difference in the number of assertions of the input formula and the unsat core, within the time limit of 2400 s (the time limit of the Unsat-Core Track in the competitions 2016–2018). Figure 6 visualizes VBS performance on the common benchmarks (91 455 total) from 2016–2018 as a bar plot. For 32 377 of these benchmarks, extracting an unsat core is trivial since they contain only one single `assert` command. They are thus not considered in the bar plot, since their maximum possible reduction is trivially 0.

### 9.2 Unique Solutions

For each SMT solver, the number of uniquely solved benchmarks—that is, the number of benchmarks that were solved by this solver only—is a measure of how much the solver contributes to the state-of-the-art, and the extent to which it is complementary to other solvers. In particular, if a solver that uniquely solves $n$ benchmarks had not participated in the competition, the virtual best solver would have been able to solve $n$ fewer benchmarks. Table 16 shows the number of uniquely solved benchmarks for each Main Track solver from 2015–2018. (Multiple versions of a solver are consolidated in this table, and answers that were given by unsound solver versions are excluded.)

Solvers that rank in high position in the competition-wide ranking (Section 8.2) often also solve many benchmarks uniquely, but there are exceptions. For instance, SMTInterpol has few unique solutions, despite achieving third place in the competition-wide ranking in 2017 and 2018. However, almost every (sound) solver in the competition solves at least some benchmarks uniquely, and hence contributes to the state-of-the-art.

The largest numbers in Table 16 are due to divisions in which there was only one sound solver: in 2016, MathSAT was the only solver to support QF_FP; in 2017, Z3 was the only sound solver in QF_FP; and in 2018, CVC4 was the only solver to support QF_SLIA, as well

**Table 16:** Number of Main Track benchmarks uniquely solved by each solver (by year)

| Solver | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| ABC | | 7 | | |
| Alt-Ergo | | | | 1 |
| AProVE | 5 | 0 | 413 | 553 |
| Boolector | 37 | 25 | 97 | 62 |
| COLIBRI | | | 3 | 22 |
| Ctrl-Ergo | | | | 12 |
| CVC3 | 70 | | | |
| CVC4 | 4863 | 113 | 10258 | 100679 |
| MapleSTP | | 0 | | |
| [MathSAT] | 21 | 28745 | 11 | 5 |
| MinkeyRink | | 13 | 27 | 16 |
| OpenSMT2 | 0 | 0 | 0 | 0 |
| ProB | | 2 | | |
| Q3B | | 0 | 89 | 0 |
| raSAT | 0 | 14 | | |
| Redlog | | | 4 | |
| SMT-RAT | 12 | 0 | 85 | 2 |
| SMTInterpol | 1 | 1 | 5 | 4 |
| SPASS-SATT | | | | 34 |
| STP | 4 | 1 | 4 | 5 |
| toysmt | | 0 | | |
| Vampire | | 130 | 753 | 652 |
| veriT | 7 | 19 | 55 | 57 |
| XSat | | | 0 | |
| Yices | 86 | 128 | 2379 | 1535 |
| Z3 | 6118 | 244 | 40992 | 888 |
| Total | 11224 | 29442 | 55175 | 104527 |

as the only sound solver in QF_ABVFP. Note that divisions with fewer than two competing solvers are non-competitive and do not affect the competition-wide ranking.

### 9.3 Use of Parallelism

It is evident both from the division rankings (Section 8.1) and the competition-wide ranking (Section 8.2) that there is very little difference between sequential and parallel performance. Even though parallel solvers may use up to four times as much CPU time, sequential and parallel division winners are most often identical, as are sequential and parallel scores for the competition-wide ranking.

In Table 17, we report the quotient of CPU time over wall-clock time (summed up over all benchmarks) for each Main Track solver, as well as for all solvers combined. When there were multiple versions of a solver (such as a sequential and a parallel version), only

**Table 17:** Quotient of CPU time over wall-clock time for each Main Track solver (by year). Values below 1.1 are shown in gray.

| Solver | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| ABC | | 1.0 | | |
| Alt-Ergo | | | | 3.0 |
| AProVE | 1.0 | 1.0 | 0.8 | 0.8 |
| Boolector | 1.0 | 1.0 | 1.1 | 1.3 |
| COLIBRI | | | 1.0 | 1.0 |
| Ctrl-Ergo | | | | 4.0 |
| CVC3 | 1.0 | | | |
| CVC4 | 1.2 | 1.1 | 1.1 | 1.0 |
| MapleSTP | | 3.8 | | |
| [MathSAT] | 1.0 | 1.0 | 1.0 | 1.0 |
| MinkeyRink | | 3.6 | 2.0 | 3.9 |
| OpenSMT2 | 3.7 | 1.0 | 1.0 | 1.0 |
| ProB | | 1.0 | | |
| Q3B | | 2.9 | 2.9 | 3.0 |
| raSAT | 1.0 | 2.0 | | |
| Redlog | | | 1.0 | |
| SMT-RAT | 1.3 | 1.0 | 1.0 | 1.0 |
| SMTInterpol | 1.4 | 1.1 | 1.1 | 1.1 |
| SPASS-SATT | | | | 1.0 |
| STP | 1.0 | 3.6 | 3.8 | 3.8 |
| toysmt | | 1.0 | | |
| Vampire | | 3.4 | 3.9 | 4.0 |
| veriT | 1.0 | 1.0 | 1.0 | 1.0 |
| XSat | | | 1.8 | |
| Yices | 1.0 | 1.0 | 1.0 | 1.0 |
| Z3 | 1.0 | 1.0 | 1.0 | 1.0 |
| All solvers | 1.0 | 1.3 | 1.3 | 1.3 |

the highest value is shown. We note that this quotient is close to 1 for most solvers: only about 4–8 Main Track solvers each year were participating with a version that made use of parallelism.

### 9.4 Focusing on Satisfiability or Unsatisfiability

In many cases, an SMT application produces either mainly satisfiable or mainly unsatisfiable queries to the SMT solver. A typical example is bounded model checking, where a query is only satisfiable if a property of the model does not hold. As a consequence, some solvers may implement specialized techniques and optimizations for the more frequent (the expected) case. Therefore, it is interesting to ask whether the winners of divisions of a track might have changed if benchmarks in that division had been restricted to one of these results.

**Table 18:** Alternative best solvers in the Main Track if only *satisfiable* benchmarks had been considered. These results are in reference to those given in Table 12.

| Division | 2015 | 2016 | 2017 | 2018 |
|---:|:---:|:---:|:---:|:---:|
| ALIA | | | | [Z3] |
| AUFLIRA | | [Z3] | [Z3] | [Z3] |
| AUFNIRA | CVC4 [Z3] | CVC4 [Z3] | CVC4 [Z3] | CVC4 [Z3] |
| BV | | Q3B [Z3] | Boolector [Z3] | Boolector |
| NIA | [Z3] | | | |
| NRA | | | | [Z3] |
| QF_ABV | | Yices | Yices | |
| QF_AUFBV | Yices | Yices | Yices | Yices |
| QF_AUFNIA | | | CVC4 | CVC4 |
| QF_BV | | Boolector [Z3] \| MinkeyRink | MinkeyRink \| | |
| QF_FP | | | | CVC4 |
| QF_IDL | Yices | Yices | | |
| QF_LIRA | | | Yices | Yices |
| QF_LRA | | | | Yices |
| QF_NIA | AProVE | SMT-RAT [Z3] | | |
| QF_NRA | | | Yices [Z3] | |
| QF_UFBV | | | | Yices |
| QF_UFIDL | \| Yices [Z3] | Yices [Z3] | \| Yices [Z3] | veriT [Z3] |
| QF_UFLIA | Yices | Yices | | |
| UF | | Vampire | | Vampire \| |
| UFBV | | Boolector [Z3] | | |
| UFLIA | CVC4 [Z3] | | | |
| UFLRA | | [Z3] | | |
| UFNIA | | | CVC4 | CVC4 |

Tables 18–19 show the changes in division winners if only satisfiable or unsatisfiable results are considered. These tables only lists the result for a division if it has changed with respect to Table 12. A non-competing solver is listed by itself if there was no competing solver with a positive division score under the given constraints. It is interesting to note that we see some new solvers winning divisions that did not win any division in the main results. For example, veriT solves the most satisfiable benchmarks in QF_UFIDL in 2018 and the most unsatisfiable benchmarks in all four years in UFLRA, and SMTInterpol performs very well on unsatisfiable benchmarks in QF_LIA.

We only consider the Main Track here, which contains a mixture of benchmarks that are satisfiable and unsatisfiable. A percentage of satisfiable, unsatisfiable and unknown status benchmarks per division and year is given in Table 3. For the Application Track, this analysis is less meaningful since a benchmark usually contains a multitude of both satisfiable and unsatisfiable incremental queries (which may depend on each other). For the Unsat-Core Track, only known unsatisfiable benchmarks from the Main Track are used, which renders this analysis useless for this track.

### 9.5 Very Quick Solutions

Different application domains of SMT typically impose a wide range of time limits (from hours to seconds). In domains with very low time limits, these limits are, in the incremental case, usually even defined per incremental query. With 20 and 40 minutes, the time limits used in the competition for all tracks in all four years were relatively long and the same

**Table 19:** Alternative best solvers in the Main Track if only *unsatisfiable* benchmarks had been considered. These results are in reference to those given in Table 12.

| Division | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| AUFLIRA | | | Vampire | |
| BV | | CVC4 | Q3B | |
| LIA | | | CVC4 | CVC4 |
| NIA | CVC4 | CVC4 | Vampire | Vampire [Z3] |
| QF_AUFBV | | | CVC4 [MathSAT] | |
| QF_AUFNIA | CVC4 [Z3] | CVC4 [Z3] | | |
| QF_BV | | | │ Boolector | |
| QF_FP | | | COLIBRI | |
| QF_IDL | CVC4 [Z3] | | CVC4 [Z3] | CVC4 [Z3] |
| QF_LIA | SMTInterpol | SMTInterpol | SMTInterpol | SMTInterpol |
| QF_NIA | CVC3 [Z3] | | Yices | |
| QF_NRA | | Yices | | CVC4 |
| QF_UFLIA | | | Yices [Z3] | |
| QF_UFNRA | | CVC4 [Z3] | │ CVC4 [Z3] | CVC4 [Z3] |
| UF | | | CVC4 | │ CVC4 |
| UFBV | | CVC4 | | |
| UFIDL | | Vampire [Z3] | CVC4 [Z3] | |
| UFLRA | veriT | veriT | veriT | veriT |

for all benchmarks, thus agnostic to the application domain of a benchmark. Hence, it is interesting to investigate how the results change if we consider a much shorter time limit. Table 20 shows the results of virtually imposing a time limit of 24 seconds[3] in the Main Track. As in the previous analysis, this highlights the different winners under this new scheme only. Here we see Alt-Ergo (AUFLIRA in 2018) and STP (QF_BV in 2016) winning a division for the first time in our analysis, and Yices performing well.

We only consider the Main Track here, since for the other two tracks, the results are less interesting. In the Application Track, for such low time limits we are rather interested in a time limit per query, which we are unable to impose on the existing data since it is not possible to extract wallclock run time information per incremental query from the data of the competitions (as mentioned above, StarExec does not record this information). In the Unsat-Core Track, the main focus is on producing small unsat-cores and solving times are of secondary interest. Additionally, benchmarks with known unsatisfiable status from the Main Track are used, and thus these results are already available from the Main Track analysis in Table 20 (disregarding the fact that enabling unsat core production may slow down a solver, depending on the technique).

### 9.6 Do Families Matter?

In 2016 the Main Track competition scoring was changed to weight the correctness score for benchmarks by the size of the family the benchmark belongs to. Here we ask what the winners would have been if this change had not been made. The answer to this question is given in Table 21. Relatively few changes would have occurred. Unsurprisingly, the differences are in the larger and more structured divisions.

[3] This time limit is used for the new 24 seconds score introduced in SMT-COMP 2019 and was inspired by a use case at Amazon.

**Table 20:** Alternative best solvers in the Main Track if the time limit had been 24 seconds

| Division | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|
| AUFLIA | | Vampire [Z3] | veriT | |
| AUFLIRA | CVC3 [Z3] | | | │ Alt-Ergo [Z3] |
| AUFNIRA | CVC3 [Z3] | | | CVC4 [Z3] |
| BV | | | | CVC4 [Z3] │ Boolector [Z3] |
| NRA | | | | CVC4 [Z3] |
| QF_ABV | | Yices | | |
| QF_ANIA | | CVC4 [Z3] | CVC4 [Z3] | |
| QF_BV | │ CVC4 | STP [Z3] | | │ Boolector |
| QF_IDL | Yices | Yices | | |
| QF_LIA | Yices | Yices [MathSAT] | Yices [MathSAT] | |
| QF_LIRA | | | Yices | Yices |
| QF_LRA | Yices | Yices | Yices | Yices |
| QF_NIA | | | Yices | Yices |
| QF_NIRA | | | Yices [Z3] | Yices [Z3] |
| QF_NRA | | | Yices [Z3] | |
| QF_UF | veriT | veriT | | |
| QF_UFLIA | │ Yices | Yices | | |
| QF_UFNRA | CVC4 [Z3] | CVC4 [Z3] | CVC4 [Z3] | Yices [Z3] |
| UF | | Vampire | | Vampire │ |
| UFBV | | Boolector [Z3] | | |
| UFIDL | | CVC4 [Z3] | CVC4 [Z3] | |
| UFLIA | | | | CVC4 [Z3] |
| UFLRA | | Vampire | veriT [Z3] | veriT |
| UFNIA | CVC4 [Z3] | | | CVC4 [Z3] │ Vampire [Z3] |

Whilst producing these results we noticed that the description of what constitutes a family had been incorrectly interpreted by the scoring scripts in 2016–2018. The scoring scripts interpreted the bottom most directory containing benchmarks in a logic as family, while the rules stated that the top most subdirectory in a logic is to be interpreted as family. After fixing this misinterpretation we note that only five division winners change—in 2018, division AUFLIRA should have been won by Vampire instead of CVC4; in 2016, AUFNIRA should have been won by CVC4 instead of Vampire; in 2016 and 2017, QF_LIA should have been won by Yices instead of CVC4; and in 2018, QF_NRA should have been won by CVC4 instead of Yices. Table 22 shows the results in the Main Track when the top most subdirectory in a logic is interpreted as family in reference to the competition results in Table 12, where the bottom most directory was interpreted as family.

The combination of the small changes in winners and the lack of difference a misinterpretation of the notion of family made suggests that families do not have a significant impact on the competition.

### 9.7 What Impact Did Adding Unknowns in the Main Track Have?

Until 2016, only benchmarks with known status were used in the Main Track. In 2016, benchmarks with unknown status were evaluated separately, but not considered in the overall results of the track. Since 2017, non-incremental benchmarks with unknown status were eligible for the Main Track. Table 23 shows alternative winners for 2017 and 2018 if benchmarks with unknown status had not been included.

**Table 21:** Alternative best solvers in the Main Track if families were not considered in computing scores (i.e., if the 2015 scoring system were used)

| Division | 2016 | 2017 | 2018 |
|---|---|---|---|
| AUFNIRA | CVC4 | | |
| BV | | Boolector [Z3] | |
| QF_BV | | MinkeyRink | Boolector |
| QF_FP | | | CVC4 |
| QF_IDL | | Yices [Z3] | |
| QF_LRA | | | SPASS-SATT |
| QF_NIA | | Yices | Yices |
| QF_NRA | | | Yices |
| UFIDL | CVC4 [Z3] | | |

**Table 22:** Alternative best solvers in the Main Track if the notion of families had been implemented as stated in the rules, i.e., top most directory instead of bottom most

| Division | 2016 | 2017 | 2018 |
|---|---|---|---|
| AUFLIRA | | | Vampire [Z3] |
| AUFNIRA | CVC4 | | |
| BV | | Q3B | |
| QF_IDL | | Yices [Z3] | Yices [Z3] |
| QF_LIA | Yices [MathSAT] | Yices [MathSAT] | |
| QF_NRA | | | CVC4 [Z3] |
| UFBV | CVC4 | | |
| UFIDL | CVC4 [Z3] | | |
| UFLIA | | | CVC4 [Z3] |
| UFNIA | | | Vampire [Z3] |

**Table 23:** Alternative best solvers in the Main Track if benchmarks with unknown status had not been included in 2017 onwards

| Division | 2017 | 2018 |
|---|---|---|
| AUFLIRA | CVC4 [Z3] | |
| AUFNIRA | CVC4 | |
| BV | Q3B | Boolector |
| LRA | CVC4 | |
| NRA | | Vampire [Z3] |
| QF_FP | | CVC4 |
| QF_NRA | Yices [Z3] | |
| UF | CVC4 | CVC4 |
| UFNIA | | CVC4 [Z3] |

## 10. Conclusions

The SMT Competition was instituted in 2005. Thirteen years later, it is a well-established event that garners wide interest from developers and users of SMT solvers. The competition continues to contribute to its original goals, including adoption of the SMT-LIB format and sparking further advances in SMT, and provides a yardstick of performance for SMT solvers.

The last FLoC Olympiad was a period of significant growth for the SMT Competition. The average number of solver submissions increased from 12 (for 2005–2014) to 26 (for 2015–2018). The number of SMT-LIB benchmarks used in the competition increased from 77 352 in 2014 [17] to 342 498 in 2018. This is partly because new benchmarks were added to the SMT Library, but also because benchmarks with unknown status are now eligible for the competition, and because the competition now uses all eligible benchmarks (rather than a randomly selected subset).

Since the re-introduction of the Unsat-Core Track in 2016, the competition again has three separate tracks. This, together with new logics in the SMT Library, has brought the total number of competition divisions to 115 in 2018, up from just 40 divisions in 2014 [17]. The number of job pairs increased from 339 714 in 2014 to 1 776 062 in 2018, and the total wall-clock time in 2018 exceeded 7.6 years. By these measures, the SMT Competition is one of the largest competitions in automated reasoning.

The StarExec infrastructure has been vitally important in allowing the competition to grow to its current size. The competition organizers have given the StarExec developers comprehensive feedback, and filed dozens of feature requests and bug reports since 2013. The infrastructure has become increasingly stable; serious issues are now rare. When they did surface during ongoing competitions, they received a swift response by the StarExec team (under the lead of Aaron Stump). This has been critical to the success of the competition on more than one occasion.

Despite the substantial amount of computational resources available to the competition through StarExec, the competition has reached a size where future organizers will have to carefully consider the number of participants and benchmarks each year, to determine time limits that will allow the computational workload to complete before the SMT Workshop.

**The Next Competition** At the time of writing, the 2019 iteration of the competition has just come to a close. It was too late to be included in this report but certain changes were made, partly influenced by the writing of this report. The three main changes were:

- *Additional Scores.* This report made it clear that simply looking at the number of problems solved was a simplified view. In 2019 the competition introduced new *additional scores* for satisfiable/unsatisfiable benchmarks and benchmarks solved within 24 seconds in the Main Track. Organizing results by satisfiability is not new. The SAT competition previously split satisfiable and unsatisfiable benchmarks into separate tracks, but merged them completely from 2016. The CASC competition separates *theorems* and *non-theorems* for most tracks. However, other competitions do not tend to look at short time limits (CASC had the SLH division in 2017).

- *Benchmarks.* Partly to combat the growing size of SMT-LIB, and partly to inject more competition-like variability and focus into the results, in 2019 the competition removed *non-competitive divisions* and then filtered benchmarks to exclude those seen as *trivial* (taken as those solved by all competitors in the previous competition in less than 1 second). This brings it closer to the CASC competition where a small subset of problems are chosen based on difficulty, although there was no inclusion of a difficulty measure in benchmark selection beyond the initial filtering. The SAT competition takes a different approach to benchmark selection with a *bring your own benchmark* approach.

- *Competition-Wide Scoring.* Motivated by the need to avoid competition-wide scoring becoming overly focused on the *breadth* of a solver, in 2019 the competition replaced the notion of *competition-wide scoring* with awards that do not attempt to compare divisions. Two awards were introduced—the *Largest Contribution* award looked at the contribution a solver made to the virtual best solver, and the *Biggest Lead* award measured the distance between the winner and second place. No other competition (that we are aware of) attempts to rank solvers across multiple areas in this way. However, the CASC competition does have a notion of *State of the Art Contribution*, similar to the largest contribution measure.

Other changes included

- the inclusion of `check-sat` commands with unknown status in the Application Track,
- the introduction of a new experimental *model validation track* with a single QF_BV division where the models produced by solvers are checked,
- the introduction of a new *industrial challenge track* consisting of unsolved benchmarks submitted specifically for the track.

**The Future** Looking to the future of the competition, it is clear that further development should be undertaken with care, keeping in mind that major changes require not just the competition organizers, but also participants to adapt.

For the inclusion of new tracks or divisions, a two-phase approach has proven successful. A new division is first declared *experimental*, meaning that results will be reported but no official winner will be announced. This allows competition organizers and participants to gain valuable experience, while reducing the impact of mistakes. If successful, the division may then become regular in the following year.

Organizing the competition is a non-trivial amount of work. For instance, the competition tools (such as the benchmark scrambler and trace executor) often require updating when new divisions are added, or when changes are made to StarExec. One could imagine adding further tracks to the competition (e.g., for proof-producing solvers, or for variants such as max-SMT), but the effort required to support such tracks will need to be balanced against their added value.

To ensure that the competition results are practically relevant, it is important that the SMT Library contains benchmarks that cover a wide range of applications. To this end, we again encourage users of SMT solvers to submit new and interesting benchmarks to the library, especially for underrepresented logics. This is not just a valuable contribution to the community, but also—since these benchmarks will then be used in the competition—a straightforward way to incentivize solver developers to tune their solvers accordingly.

**Acknowledgments** We would like to thank Aaron Stump for allowing the SMT Competition to run on StarExec; the SMT-LIB maintainers, namely Clark Barrett, Pascal Fontaine, Aina Niemetz and Mathias Preiner, for preparing the annual SMT-LIB release; all contributors of new benchmarks to the SMT Library; and, last but not least, all solver developers who entered their solver into the competition.

## References

[1] Alessandro Armando, Clark Barrett, Alessandro Cimatti, Byron Cook, Leonardo de Moura, Bruno Dutertre, Sava Krstic, Albert Oliveras, Silvio Ranise, Roberto Sebastiani, Ofer Strichman, and Cesare Tinelli. SMT Workshop bylaws, May 2009. Available at `http://smt-workshop.cs.uiowa.edu/bylaws.shtml`.

[2] Haniel Barbosa, Jasmin Christian Blanchette, Simon Cruanes, Daniel El Ouraoui, and Pascal Fontaine. Language and proofs for higher-order SMT (work in progress). In Catherine Dubois and Bruno Woltzenlogel Paleo, editors, *Fifth Workshop on Proof eXchange for Theorem Proving, PxTP 2017, Brasília, Brazil, 23-24 September 2017. Proceedings*, **262** of *EPTCS*, pages 15–22, 2017.

[3] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, **6806** of *Lecture Notes in Computer Science*, pages 171–177. Springer, 2011.

[4] Clark Barrett, Leonardo de Moura, and Aaron Stump. Design and results of the 2nd Annual Satisfiability Modulo Theories Competition (SMT-COMP 2006). *Formal Methods in System Design*, 2007.

[5] Clark Barrett, Morgan Deters, Leonardo de Moura, Albert Oliveras, and Aaron Stump. 6 years of SMT-COMP. *Journal of Automated Reasoning*, **50**(3):243–277, 2013.

[6] Clark Barrett, Morgan Deters, Albert Oliveras, and Aaron Stump. Design and results of the 3rd Annual Satisfiability Modulo Theories Competition (SMT-COMP 2007). *International Journal on Artificial Intelligence Tools*, **17**(4):569–606, 2008.

[7] Clark Barrett, Morgan Deters, Albert Oliveras, and Aaron Stump. Design and results of the 4th Annual Satisfiability Modulo Theories Competition (SMT-COMP 2008). Technical Report TR2010-931, New York University, 2010.

[8] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2017. Available at `www.SMT-LIB.org`.

[9] Clark Barrett and Cesare Tinelli. CVC3. In Werner Damm and Holger Hermanns, editors, *Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007. Proceedings*, **4590** of *Lecture Notes in Computer Science*, pages 298–302. Springer, 2007.

[10] Clark W. Barrett, Leonardo Mendonça de Moura, and Aaron Stump. Design and results of the first satisfiability modulo theories competition (SMT-COMP 2005). *Journal of Automated Reasoning*, **35**(4):373–390, 2005.

[11] Ezio Bartocci, Dirk Beyer, Paul E. Black, Grigory Fedyukovich, Hubert Garavel, Arnd Hartmanns, Marieke Huisman, Fabrice Kordon, Julian Nagele, Mihaela Sighireanu, Bernhard Steffen, Martin Suda, Geoff Sutcliffe, Tjark Weber, and Akihisa Yamada. TOOLympics 2019: An overview of competitions in formal methods. In Dirk Beyer, Marieke Huisman, Fabrice Kordon, and Bernhard Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part III*, **11429** of *Lecture Notes in Computer Science*, pages 3–24. Springer, 2019.

[12] François Bobot, Sylvain Conchon, Evelyne Contejean, Mohamed Iguernelala, Assia Mahboubi, Alain Mebsout, and Guillaume Melquiond. A simplex-based extension of Fourier-Motzkin for solving linear integer arithmetic. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, **7364** of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2012.

[13] Thomas Bouton, Diego Caminha Barbosa De Oliveira, David Déharbe, and Pascal Fontaine. veriT: An open, trustable and efficient smt-solver. In Renate A. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, **5663** of *Lecture Notes in Computer Science*, pages 151–156. Springer, 2009.

[14] Robert K. Brayton and Alan Mishchenko. ABC: an academic industrial-strength verification tool. In Tayssir Touili, Byron Cook, and Paul B. Jackson, editors, *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, **6174** of *Lecture Notes in Computer Science*, pages 24–40. Springer, 2010.

[15] Martin Bromberger, Mathias Fleury, Simon Schwarz, and Christoph Weidenbach. SPASS-SATT a CDCL(LA) solver. In *CADE-27*, Lecture Notes in Computer Science. Springer, 2019. (Forthcoming).

[16] Jürgen Christ, Jochen Hoenicke, and Alexander Nutz. SMTInterpol: An interpolating SMT solver. In Alastair F. Donaldson and David Parker, editors, *Model Checking Software - 19th International Workshop, SPIN 2012, Oxford, UK, July 23-24, 2012. Proceedings*, **7385** of *Lecture Notes in Computer Science*, pages 248–254. Springer, 2012.

[17] David R. Cok, David Déharbe, and Tjark Weber. The 2014 SMT competition. *Journal on Satisfiability, Boolean Modeling and Computation*, **9**:207–242, 2014.

[18] David R. Cok, Alberto Griggio, Roberto Bruttomesso, and Morgan Deters. The 2012 SMT competition, 2012. Available online at http://smtcomp.sourceforge.net/2012/reports/SMTCOMP2012.pdf.

[19] David R. Cok, Aaron Stump, and Tjark Weber. The 2013 evaluation of SMT-COMP and SMT-LIB. *Journal of Automated Reasoning*, **55**(1):61–90, 2015.

[20] Sylvain Conchon, Albin Coquereau, Mohamed Iguernlala, and Alain Mebsout. Alt-Ergo 2.2. In *SMT Workshop: International Workshop on Satisfiability Modulo Theories*, Oxford, United Kingdom, July 2018.

[21] Sylvain Conchon, David Deharbé, Matthias Heizmann, and Tjark Weber. 11th International Satisfiability Modulo Theories Competition (SMT-COMP 2016): Rules and procedures, 2016. Available online at http://smtcomp.sourceforge.net/2016/rules16.pdf.

[22] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp, and Erika Ábrahám. SMT-RAT: an open source C++ toolbox for strategic and parallel SMT solving. In Marijn Heule and Sean Weaver, editors, *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015. Proceedings*, **9340** of *Lecture Notes in Computer Science*, pages 360–368. Springer, 2015.

[23] Leonardo De Moura and Nikolaj Bjørner. Satisfiability Modulo Theories: Introduction and applications. *Communications of the ACM*, **54**(9):69–77, September 2011.

[24] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, **4963** of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[25] Andreas Dolzmann and Thomas Sturm. REDLOG: computer algebra meets computer logic. *ACM SIGSAM Bulletin*, **31**(2):2–9, 1997.

[26] Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Computer-Aided Verification (CAV'2014)*, **8559** of *Lecture Notes in Computer Science*, pages 737–744. Springer, July 2014.

[27] Zhoulai Fu and Zhendong Su. XSat: A fast floating-point satisfiability solver. In Swarat Chaudhuri and Azadeh Farzan, editors, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016. Proceedings, Part II*, **9780** of *Lecture Notes in Computer Science*, pages 187–209. Springer, 2016.

[28] Vijay Ganesh. *Decision Procedures for Bit-Vectors, Arrays and Integers*. PhD thesis, Computer Science Department, Stanford University, CA, United States, 2007.

[29] Jürgen Giesl, Cornelius Aschermann, Marc Brockschmidt, Fabian Emmes, Florian Frohn, Carsten Fuhs, Jera Hensel, Carsten Otto, Martin Plücker, Peter Schneider-Kamp, Thomas Ströder, Stephanie Swiderski, and René Thiemann. Analyzing program termination and complexity automatically with AProVE. *Journal of Automated Reasoning*, **58**(1):3–31, 2017.

[30] Henning Günther and Georg Weissenbacher. Incremental bounded software model checking. In *2014 International SPIN Symposium on Model Checking of Software. Proceedings*, SPIN 2014, pages 40–47, New York, NY, USA, 2014. ACM.

[31] Matthias Heizmann, Aina Niemetz, Giles Reger, and Tjark Weber. 13th International Satisfiability Modulo Theories Competition (SMT-COMP 2018): Rules and procedures, 2018. Available online at http://smtcomp.sourceforge.net/2018/rules18.pdf.

[32] Matthias Heizmann, Giles Reger, and Tjark Weber. 12th International Satisfiability Modulo Theories Competition (SMT-COMP 2017): Rules and procedures, 2017. Available online at http://smtcomp.sourceforge.net/2017/rules17.pdf.

[33] Antti E. J. Hyvärinen, Matteo Marescotti, Leonardo Alt, and Natasha Sharygina. OpenSMT2: An SMT solver for multi-core and cloud computing. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016. Proceedings*, **9710** of *Lecture Notes in Computer Science*, pages 547–553. Springer, 2016.

[34] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, **8044** of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.

[35] Michael Leuschel, Jens Bendisposto, Ivo Dobrikov, Sebastian Krings, and Daniel Plagge. *From Animation to Data Validation: The ProB Constraint Solver 10 Years On*, chapter 14, pages 427–446. John Wiley & Sons, Ltd, 2014.

[36] Bruno Marre, Zakaria Chihani, Benjamin Blanc, Franck Vedrine, Patricia Mouy, and François Bobot. COLIBRI, 2018. System description. Available online at http://smtcomp.sourceforge.net/2018/systemDescriptions/COLIBRI.pdf.

[37] Aina Niemetz, Mathias Preiner, Clifford Wolf, and Armin Biere. Btor2, BtorMC and Boolector 3.0. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018. Proceedings, Part I*, **10981** of *Lecture Notes in Computer Science*, pages 587–595. Springer, 2018.

[38] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, **53**(6):937–977, November 2006.

[39] SMT-COMP – Discussion list for the Satisfiability Modulo Theories Competition. Available at https://cs.nyu.edu/mailman/listinfo/smt-comp.

[40] SMT-LIB benchmark library. Available at https://clc-gitlab.cs.uiowa.edu:2443/explore/groups.

[41] Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. StarExec: A cross-community infrastructure for logic solving. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, **8562** of *Lecture Notes in Computer Science*, pages 367–373. Springer, 2014.

[42] Vu Xuan Tung, To Van Khanh, and Mizuhito Ogawa. raSAT 0.3 and raSAT 0.4 (exp) for SMT-COMP 2016, 2016. System description. Available online at http://smtcomp.sourceforge.net/2016/systemDescriptions/raSAT.pdf.

[43] Tjark Weber. Scrambling and descrambling SMT-LIB benchmarks. In Tim King and Ruzica Piskac, editors, *14th International Workshop on Satisfiability Modulo Theories affiliated with the International Joint Conference on Automated Reasoning, SMT@IJCAR 2016, Coimbra, Portugal, July 1-2, 2016. Proceedings*, **1617** of *CEUR Workshop Proceedings*, pages 31–40. CEUR-WS.org, 2016.