10th International Satisfiability Modulo Theories Competition (SMT-COMP 2015): Rules and Procedures

Sylvain Conchon
Paris-Sud University
France
Sylvain.Conchon@lri.fr

David Déharbe
Federal University of Rio Grande do Norte
Brazil
david@dimap.ufrn.br

Tjark Weber
Uppsala University
Sweden
tjark.weber@it.uu.se

This version revised 2015-6-20

Comments on this document should be emailed to the SMT-COMP mailing list (see below) or, if necessary, directly to the organizers.

1 Communication

Interested parties should subscribe to the SMT-COMP mailing list. Important late-breaking news and any necessary clarifications and edits to these rules will be announced there, and it is the primary way that such announcements will be communicated.

- SMT-COMP mailing list: smt-comp@cs.nyu.edu
- Sign-up site for the mailing list: cs.nyu.edu/mailman/listinfo/smt-comp

Additional material will be made available at the competition web site, www.smtcomp.org or smtcomp.sourceforge.net/2015.

2 Important Dates

May 1 Deadline for new benchmark contributions.

June 1 Deadline for first versions of solvers (for all tracks), including information about which tracks and divisions are being entered, and magic numbers for benchmark scrambling.

June 7 Final versions of competition tools (e.g., benchmark scrambler) are made available. Benchmark libraries are frozen.

June 14 Deadline for final versions of solvers, including system descriptions.

June 15 Opening value of NYSE Composite Index used to complete random seed for benchmark scrambling.

July 18/19 SMT Workshop; end of competition, presentation of results.

3 Introduction

The annual Satisfiability Modulo Theories Competition (SMT-COMP) is held to spur advances in SMT solver implementations on benchmark formulas of practical interest. Public competitions are a well-known means of stimulating advancement in software tools. For example, in automated reasoning, the CASC and SAT competitions for first-order and propositional reasoning tools, respectively, have spurred significant innovation in their fields [5, 9]. More information on the history and motivation for SMT-COMP can be found at the competition web site, `www.smtcomp.org`, and in reports on previous competitions ([1, 2, 3, 4, 6, 7, 8]).

SMT-COMP 2015 is affiliated with the SMT Workshop 2015 (`http://smt2015.csl.sri.com/`), which is associated with CAV 2015 (`http://i-cav.org/2015/`). The SMT Workshop and the main CAV conference will include a block of time to present the competitors and results of the competition.

Accordingly, researchers are highly encouraged to submit both new benchmarks and new or improved solvers to raise the level of competition and advance the state-of-the-art in automated SMT problem solving.

SMT-COMP 2015 will have two tracks: the conventional main track and an application (i.e., incremental) track. Within each track there are multiple divisions, where each division uses benchmarks from a specific SMT-LIB logic (or group of logics). We will recognize winners as measured by number of benchmarks solved; we will also recognize solvers based on additional criteria.

The rest of this document, revised from the previous version,[1] describes the rules and competition procedures for SMT-COMP 2015. The principal changes from the previous competition rules are the following:

- All solvers will be run with $n = 4$ cores available. We will recognize both best sequential and best parallel performance in all main track divisions, using different CPU time limits.

[1] Earlier versions of this document include contributions from Clark Barrett, Roberto Bruttomesso, David Cok, David Déharbe, Morgan Deters, Alberto Griggio, Albert Oliveras, Aaron Stump, and Tjark Weber.

- The division scoring will take time spent on unsolved benchmarks into account. Previous competitions only considered time spent on solved benchmarks. However, the number of solved benchmarks still takes precedence over run-time.
- In addition to recognizing the best solver in each division, we will recognize solvers that perform best according to competition-wide criteria, emphasizing breadth of supported logics. These criteria will be a refinement of the criteria used for the FLoC Olympic Games medals in 2014.
- We plan to include new divisions for floating-point arithmetic. These should be considered experimental in 2015, and will not be included in the competition-wide ranking.
- We plan to run solvers on all eligible benchmarks. Previous competitions used a proper subset of eligible benchmarks, causing competition results to be affected by a random selection process.

4 Entrants

Solver submission. An entrant to SMT-COMP is an SMT solver submitted by its authors using the StarExec (`http://www.starexec.org`) service. The execution service enables members of the SMT research community to run solvers on jobs consisting of benchmarks from the SMT-LIB benchmark library. Jobs are run on a shared computer cluster. The execution service is provided free of charge, but it does require registration to create a login account. Registered users may then upload their own solvers to run, or may run public solvers already uploaded to the service.

For participation in SMT-COMP, a solver must be uploaded to StarExec, made publicly available, and the organizers informed of its presence *and the tracks and divisions in which it is participating*. StarExec supports solver configurations; for clarity, each submitted solver must have one configuration only. A submission must also include a 32-bit unsigned integer. These integer numbers, collected from all submissions, are used to seed the benchmark scrambler.

Information about how to configure and upload a solver is contained in the StarExec user guide, `https://wiki.uiowa.edu/display/stardev/User+Guide`.

System description. As part of a submission, SMT-COMP entrants are encouraged to provide a short (1–2 pages) description of the system. This should include a list of all authors of the system, their present institutional affiliations, and any appropriate acknowledgements. The programming language(s) and basic SMT solving approach employed should be described (e.g., lazy integration of a Nelson-Oppen combination with SAT, translation to SAT, etc.). System descriptions are encouraged to include a URL for a web site for the submitted tool. System descriptions may be submitted after the solver deadline, but to be useful should be sent to the organizers before the competition ends. We intend to make system descriptions publicly available.

Multiple versions. The organizers' intent is to promote as wide a comparison among solvers and solver options as possible. However, if the number of solver submissions is too large for the computational resources available to the competition, the organizers reserve the right not to accept multiple versions of solvers from the same solver team.

Other solvers. The organizers reserve the right to include other solvers of interest (such as entrants in previous SMT competitions) in the competition, e.g., for comparison purposes.

Wrapper tools. A *wrapper tool* is defined as any solver which calls one or more SMT solvers not written by the author of the wrapper tool. The other solvers are called the *wrapped* solvers. A wrapper tool must explicitly acknowledge any solvers that it wraps. Its system description should make clear the technical innovations by which the wrapper tool expects to improve on the wrapped solvers.

Attendance. Submitters of an SMT-COMP entrant need not be physically present at the competition or the SMT Workshop to participate or win.

Deadlines

SMT-COMP entrants must be submitted via StarExec (solvers) *and* email to the organizers (accompanying information) until the end of **June 1, 2015** anywhere on earth. After this date no new entrants will be accepted. However, updates to existing entrants on StarExec will be accepted until the end of **June 14, 2015** anywhere on earth.

We strongly encourage participants to use this grace period *only* for the purpose of fixing any bugs that may be discovered, and not for adding new features, as there may be no opportunity to do extensive testing using StarExec after the initial deadline.

The solver versions that are present on StarExec at the conclusion of the grace period will be the ones used for the competition. Versions submitted after this time will not be used. The organizers reserve the right to start the competition itself at any time after the open of the New York Stock Exchange on the day after the final solver deadline.

These deadlines and procedures apply equally to all tracks of the competition.

5 Execution of Solvers

Solvers will be publicly evaluated in all tracks and divisions into which they have been entered. All results of the competition will be made public.

5.1 Logistics

Dates of competition. The bulk of the computation will take place during the weeks leading up to SMT 2015, from about June 15 to July 17. Intermediate results will be regularly posted to the SMT-COMP website as the competition runs.

The organizers reserve the right to prioritize certain competition tracks or divisions to ensure their timely completion, and in exceptional circumstances to complete divisions after the SMT Workshop.

Input and output. In the main track, a participating solver must read a single benchmark script, whose name is presented as the solver's first argument. In the application track, a trace executor will send commands from a benchmark script to the solver's standard input channel.

The benchmark script is in the concrete syntax of the SMT-LIB format, version 2.0, though with a restricted set of commands. A benchmark script is a text file containing a sequence of SMT-LIB v2 commands in which:

1. In the main track, there may be a (single) **set-option :print-success ...** command. Note that `success` outputs are ignored by the post-processor used by the competition.[2]

2. In the application track, the **:print-success** option may not be disabled; the application track executor will send an initial **set-option :print-success true** command to the solver.

3. The (single) **set-logic** command setting the benchmark's logic is the first command after any **set-option** commands.

4. The script next optionally contains (any number of) **set-info** commands, which may be ignored by the solver. Benchmarks will likely have some of the following **set-info** commands: **(set-info :smt-lib-version 2.0)**, **(set-info :category ...)**, **(set-info :source ...)**, and **(set-info :status ...)**.

5. In the main track, there is exactly one **check-sat** command, following possibly several **assert** commands.

6. In the application track, there are one or more **check-sat** commands, each preceded by one or more **assert** commands. There may also be zero or more **push 1** commands, and zero or more **pop 1** commands, consistent with the uses of those commands in the SMT-LIB standard. Each **check-sat** command may be preceded by a **set-info :status** command that indicates the expected result of the **check-sat** command.

7. The script may optionally contain an **exit** command as its last command. In the application track, this command must not be omitted.

8. The formulas in the script belong to the benchmark's logic, with any free symbols declared in the script.

9. All sorts declared with a **declare-sort** command must have zero arity.

10. Application track scripts may have **define-sort** commands.

11. No other commands besides the ones just mentioned may be used.

12. Named formulas are not used in benchmark scripts.

The SMT-LIB format specification is available from the "Documents" section of the SMT-LIB website [10]. Solvers will be given formulas only from the divisions into which they have been entered.

[2] SMT-LIB 2.0 requires that a solver produce a `success` answer after each **set-logic**, **declare-sort**, **declare-fun** and **assert** command (among others), unless the option **:print-success** is set to false. Ignoring the `success` outputs allows for submitting fully SMT-LIB 2.0 compliant solvers without the need for a wrapper script, while still allowing entrants of previous competitions to run without changes.

Time and memory limits. Each SMT-COMP solver will be executed on a dedicated processor of a competition machine, for each given benchmark, up to a fixed wall-clock time limit T. Each processor has 4 cores. Detailed machine specifications are available on the competition web site.

The time limit T is yet to be determined, but it is anticipated to be 40 minutes of wall-clock time per solver/benchmark pair.[3] Solvers that take more than this time limit will be killed. Solvers are allowed to spawn other processes; these will be killed at approximately the same time as the first started process.

The StarExec service also limits the memory consumption of the solver processes. We expect the memory limit per solver/benchmark pair to be on the order of 60 GB. The values of both the time limit and the memory limit are available to a solver process through environment variables. See the StarExec user guide for more information.

5.2 Main track

The main track competition will consist of selected benchmarks in each of the logic divisions. Each benchmark script will be presented to the solver as its first command-line argument. Each SMT-COMP entrant is then expected to attempt to report on its standard output channel whether the formula is satisfiable (`sat`, in lowercase) or unsatisfiable (`unsat`). An entrant may also report `unknown` to indicate that it cannot determine satisfiability of the formula.

The main track competition uses a StarExec post-processor (named "SMT-COMP 2015") to accumulate the results.

Aborts and unparsable output. Any `success` outputs will be ignored. Solvers that exit before the time limit without reporting a result (e.g., due to exhausting memory or crashing) and do not produce output that includes `sat`, `unsat` or `unknown` will be considered to have aborted.

Persistent state. Solvers may create and write to files and directories during the course of an execution, but they must not read such files back during later executions. Each solver is executed with a temporary directory as its current working directory. Any generated files should be produced there (and not, say, in the system's `/tmp` directory). The StarExec system sets a limit on the amount of disk storage permitted—typically 20 GB. See the StarExec user guide for more information. The temporary directory is deleted after the job is complete. Solvers must not attempt to communicate with other machines, e.g., over the network.

5.3 Application track

The application track evaluates SMT solvers when interacting with an external verification framework, e.g., a model checker. This interaction, ideally, happens by means of an online communication between the model checker and the solver: the model checker repeatedly sends queries to the SMT solver, which in turn answers either `sat` or `unsat`. In this interaction an SMT-solver is required to accept queries incrementally via its *standard input channel*.

In order to facilitate the evaluation of the solvers in this track, we will set up a "simulation" of the aforementioned interaction. In particular each benchmark in the application track represents a realistic communication trace, containing multiple **check-sat** commands (possibly with

[3] The time limit may be adjusted once we know the number of competition entrants and eligible benchmarks.

corresponding **push 1/pop 1** commands), which is parsed by a *trace executor*. The trace executor serves the following purposes:

- it simulates the online interaction by sending single queries to the SMT solver (through stdin);
- it prevents "look-ahead" behaviors of SMT solvers;
- it records time and answers for each call;
- it guarantees a fair execution for all solvers by abstracting from any possible crash, misbehavior, etc. that may happen on the model checker side.

The trace executor terminates processing the benchmark script upon receiving an incorrect response from the solver.

The disk space and memory limits for the application track are the same as for the main track (see Section 5.2).

Input and output. Participating solvers will be connected to a trace executor, which will incrementally send commands to the standard input channel of the solver and read responses from the standard output channel of the solver. The commands will be taken from an SMT-LIB 2.0 benchmark script that satisfies the rules for an application script given in Section 5.1.

Solvers must respond to each command sent by the trace executor with the answers defined in the SMT-LIB 2.0 format specification, that is, with a `success` answer for **set-option**, **set-logic**, **declare-sort**, **declare-fun**, **define-sort**, **define-fun**, **assert**, **push 1**, and **pop 1** commands, and with an answer of `sat`, `unsat`, or `unknown` for **check-sat** commands. There will be no **get-unsat-core**, **get-proof**, **get-model**, **get-assignments**, or **get-values** commands in application track benchmarks.

6 Benchmarks and Problem Divisions

Benchmark sources. Benchmarks for each division will be drawn from the SMT-LIB benchmark repository. The main track will use a subset of all *non-incremental* benchmarks; the application track will use a subset of all *incremental* benchmarks.

New benchmarks. The deadline for submission of new benchmarks is **May 1, 2015**. The organizers will be checking and curating these until **June 7, 2015**. The organizers reserve the right to exclude new benchmarks if any prove problematic for some reason. SMT-COMP attempts to give preference to benchmarks that are "real-world," in the sense of coming from or having some intended application outside SMT.

New benchmarks will be made publicly available as soon as possible after the benchmark submission deadline, as they are checked and curated. The set of benchmarks selected for the competition will be published when the competition begins.

Benchmark demographics. In SMT-LIB, benchmarks are organized according to *families*. A benchmark family contains problems that are similar in some significant way. Typically they come

from the same source or application, or are all output by the same tool. Each top-level subdirectory within a division represents a distinct family.

Each benchmark in SMT-LIB also has a *category*. There are four possible categories:

- *check.* These benchmarks are hand-crafted to test whether solvers support specific features of each division.
- *industrial.* These benchmarks come from some real application and are produced by tools such as bounded model checkers, static analyzers, extended static checkers, etc.
- *random.* These benchmarks are randomly generated.
- *crafted.* This category is for all other benchmarks. Usually, benchmarks in this category are designed to be particularly difficult or to test a specific feature of the logic.

Benchmark selection. The benchmark pool is culled as follows:

1. *Retire inappropriate benchmarks.* The competition organizers may remove from the eligibility pool certain benchmarks that are inappropriate[4] or uninteresting for competition, or cut the size of certain benchmark families to avoid their over-representation.
2. *Eliminate benchmarks whose status is unknown.* Any benchmark whose expected output is neither `sat` nor `unsat` is removed. There are a significant number of benchmarks with `unknown` status in the library; these are certainly interesting and are a challenge to solve, but they are not used in the competition.[5]

All remaining benchmarks are used for the competition. There will be no further selection of benchmarks, e.g., based on benchmark difficulty or benchmark category.

Heats. Since the organizers at this point are unsure how long a set of benchmarks may take (which will depend also on the number of solvers submitted), the competition may be run in *heats*. For each track and division, the selected benchmarks may be randomly divided into a number of (possibly unequal-sized) heats. Heats will be run in order. If the organizers determine that there is adequate time, all heats will be used for the competition. Otherwise, incomplete heats will be ignored.

Benchmark scrambling. Benchmarks will be slightly scrambled before the competition, using a simple benchmark scrambler. The benchmark scrambler will be made publicly available before the competition.

Naturally, solvers must not rely on previously determined identifying syntactic characteristics of competition benchmarks in testing satisfiability. Violation of this rule is considered cheating.

Pseudo-random numbers. Pseudo-random numbers used, e.g., for the creation of heats or the scrambling of benchmarks, will be generated using the standard C library function `random()`,

[4] In 2015, this (again) includes all benchmarks that use bit-vector division or remainder, because of an unresolved discussion about the semantics of division by zero.

[5] Incremental benchmarks may contain multiple **check-sat** commands, each with its own status. Incremental benchmarks that contain (some) **check-sat** commands with `unknown` status may still contain a prefix of commands where the status is known for all **check-sat** commands in the prefix. This prefix is eligible for the application track.

seeded (using `srandom()`) with the sum, modulo 2^{30}, of the integer numbers provided in the system descriptions (see Section 4) by all SMT-COMP entrants other than the organizers'. Additionally, the integer part of the opening value of the New York Stock Exchange Composite Index on the first day the exchange is open on or after the date specified in the timeline (Section 2) will be added to the other seeding values. This helps provide transparency, by guaranteeing that the organizers cannot manipulate the seed in favor of or against any particular submitted solver.

7 Scoring

Scores will be computed for all solvers and divisions. However, winners will be declared only for *competitive* divisions. A division in a track is competitive if at least two substantially different solvers (i.e., solvers from two different teams) were submitted. Although the organizers may enter other solvers for comparison purposes, only solvers that are explicitly submitted by their authors determine whether a division is competitive, and are eligible to be designated as winners.

Divisions for floating-point arithmetic are experimental in 2015, and will not be considered competitive.

7.1 Benchmark scoring

A solver's *raw score* for each benchmark is a quadruple $\langle e, n, w, c\rangle$, with $e \in \{0, 1\}$ the number of erroneous results (usually $e = 0$), $0 \leq n \leq N$ the number of correct results,[6] $w \in [0, T]$ the (real-valued) wall-clock time in seconds, and $c \in [0, 4T]$ the (real-valued) CPU time in seconds, measured across all cores and sub-processes, until the solver process terminates.

Main track. More specifically, for the main track, we have

- $e = 0$, $n = 0$ if the solver times out, or the solver aborts without a response, or the result of the **check-sat** command is `unknown`,
- $e = 0$, $n = 1$ if the solver does not time out, and the result of the **check-sat** command is correct,
- $e = 1$, $n = 0$ if the solver does not time out, and the result of the **check-sat** command is incorrect.

Note that for a (correct or incorrect) result to be taken into consideration, the solver process must terminate (normally or abnormally) within the time limit.

Application track. An application benchmark may contain multiple **check-sat** commands. Solvers may partially solve the benchmark before timing out. The benchmark is run by the trace executor, measuring the total time (summed over all individual commands) taken by the solver to respond to commands.[7] Most time will likely be spent in response to **check-sat** commands, but **assert**, **push**

[6] Here, N is the number of **check-sat** commands in the benchmark. Recall that main track benchmarks have just one **check-sat** command; application track benchmarks may have multiple **check-sat** commands.

[7] Times measured by StarExec may include time spent in the trace executor. We expect that this time will likely be insignificant compared to time spent in the solver, and nearly constant across solvers.

or **pop** commands might also entail a reasonable amount of processing. For the application track, we have

- $e = 1$, $n = 0$ if the solver returns an incorrect result for any **check-sat** command within the time limit,
- otherwise, $e = 0$, and n is the number of correct results for **check-sat** commands returned by the solver before the time limit is reached.

7.2 Sequential performance

SMT-COMP has traditionally emphasized sequential performance (i.e., CPU time) over parallel performance (i.e., wall-clock time). StarExec is measuring both, and we intend to recognize both best sequential and best parallel solvers in all competitive main track divisions.

The raw score, as defined in Section 7.1, favors parallel solvers, which may utilize all available processor cores. To evaluate sequential performance, we derive a *sequential score* by imposing a (virtual) CPU time limit equal to the wall-clock time limit T. A solver result is taken into consideration for the sequential score only if the solver process terminates within this CPU time limit. More specifically, for a given raw score $\langle e, n, w, c\rangle$, the corresponding sequential score is defined as $\langle e_S, n_S, c_S\rangle$, where

- $e_S = 0$ and $n_S = 0$ if $c > T$, $e_S = e$ and $n_S = n$ otherwise,
- $c_S = \min\{c, T\}$.[8]

7.3 Division scoring

To compute a solver's score for a division, the solver's individual benchmark scores for all benchmarks in the division are summed component-wise. For main track divisions, we will separately compute the sum of all raw scores (Section 7.1) to assess parallel performance, and the sum of all sequential scores (Section 7.2) to assess sequential performance. For application track divisions, division scores will be based on raw scores.[9]

Division scores are compared lexicographically:

- A sum of raw scores $\langle e, n, w, c\rangle$ is better than $\langle e', n', w', c'\rangle$ iff $e < e'$ or ($e = e'$ and $n > n'$) or ($e = e'$ and $n = n'$ and $w < w'$) or ($e = e'$ and $n = n'$ and $w = w'$ and $c < c'$). That is, fewer errors takes precedence over more correct solutions, which takes precedence over less wall-clock time taken, which takes precedence over less CPU time taken.
- A sum of sequential scores $\langle e_S, n_S, c_S\rangle$ is better than $\langle e'_S, n'_S, c'_S\rangle$ iff $e_S < e'_S$ or ($e_S = e'_S$ and $n_S > n'_S$) or ($e_S = e'_S$ and $n_S = n'_S$ and $c_S < c'_S$). That is, fewer errors takes precedence over more correct solutions, which takes precedence over less CPU time taken.

[8] *Rationale:* Under this score, a solver should not benefit from using multiple processor cores. Conceptually, the sequential score should be (nearly) unchanged if the solver was run on a single-core processor, up to a time limit of T.

[9] Since application track benchmarks may be partially solved, defining a useful sequential score for the application track would require information not provided by the raw score, e.g., detailed timing information for each result.

We will not make any comparisons between raw scores and sequential scores, as these are intended to measure fundamentally different performance characteristics.

7.4 Competition-wide scoring

We define a competition-wide metric for the main track, separately for parallel and sequential performance, as follows. Let N_i be the total number of benchmarks in division i that were used in the competition, and let $\langle e_i, n_i, w_i, c_i \rangle$ be a solver's raw score for this division (Section 7.3). The solver's competition-wide raw score is $\sum_i (e_i == 0 \,?\, (n_i/N_i)^2 \,:\, -e_i) \log N_i$, where the sum is over all competitive divisions into which the solver was entered.[10] The solver's competition-wide sequential score is computed from its sequential division scores (Section 7.3) according to the same formula. We will recognize the best three solvers according to these metrics.

7.5 Other recognitions

The organizers will also recognize the following contributions:

- *Open source.* The top solver that provides its source code as open source will be recognized in each competitive division.
- *Best industrial performance.* There is agreement in the SMT community to emphasize problems that come from real applications. The solver that performs best on industrial benchmarks, as measured by the division scoring restricted to this benchmark category, will be recognized in each competitive division.
- *Best new entrant*. The best performing entrant from a new solver implementation team, as measured by the competition-wide metric.
- *Breadth of logics*. Solvers that cover the most theories and logics.
- *Benchmarks*. Contributors of new benchmarks.

The organizers reserve the right to recognize other outstanding contributions that become apparent in the competition results.

[10] *Rationale:* This metric purposely emphasizes breadth of solver participation—a solver participating in many logics need not be the best in any one of them. The use of the square in the metric limits that somewhat—a solver still needs to do reasonably well compared to winners to be able to catch up by breadth of participation. The non-linear metric also gives added weight to completing close to all benchmarks in a division. The log scaling is a (somewhat arbitrary) means to adjust the scores for the wide variety of numbers of benchmarks in different divisions. It seems a reasonable compromise between linearly combining numbers of benchmarks, which would overweigh large divisions, and simply summing the fraction of benchmarks solved, which would overweigh small divisions. The metric is also quite simple, and the metric for a solver is independent of the performance of other solvers. Time is omitted from the metric because it is only of third importance in the regular competition metric, and is difficult to compare across divisions.

8 Judging

The organizers reserve the right, with careful deliberation, to remove a benchmark from the competition results if it is determined that the benchmark is faulty (e.g., syntactically invalid in a way that affects some solvers but not others); and to clarify ambiguities in these rules that are discovered in the course of the competition. Authors of solver entrants may appeal to the organizers to request such decisions. Organizers that are affiliated with solver entrants will be recused from these decisions. The organizers' decisions are final.

9 Acknowledgments and Disclaimer

SMT-COMP 2015 is organized under the direction of the SMT Steering Committee. The organizing team is

- Sylvain Conchon – Paris-Sud University, France (co-organizer)
- David Déharbe – Federal University of Rio Grande do Norte, Brazil (co-organizer)
- Tjark Weber – Uppsala University, Sweden (chair)

Tjark Weber is responsible for policy and procedure decisions, such as these rules, with input from the co-organizers. He is not associated with any group creating or submitting solvers.

Many others have contributed benchmarks, effort, and feedback. Clark Barrett processed the newly submitted benchmarks. The competition uses the StarExec service, which is hosted at the University of Iowa. Aaron Stump is providing essential StarExec support.

David Déharbe is associated with the solver group producing the veriT solver.

References

[1] Clark Barrett, Leonardo de Moura, and Aaron Stump. Design and Results of the 1st Satisfiability Modulo Theories Competition (SMT-COMP 2005). *Journal of Automated Reasoning*, 35(4):373–390, 2005.

[2] 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*, 31(3):221–239, 2007.

[3] 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.

[4] 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.

[5] Daniel Le Berre and Laurent Simon. The Essentials of the SAT 2003 Competition. In *Sixth International Conference on Theory and Applications of Satisfiability Testing*, volume 2919 of *LNCS*, pages 452–467. Springer, 2003.

[6] David R. Cok, David Déharbe, and Tjark Weber. The 2014 SMT Competition. Submitted.

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

[8] David R. Cok, Aaron Stump, and Tjark Weber. The 2013 Evaluation of SMT-COMP and SMT-LIB. *Journal of Automated Reasoning*, 2015. DOI: 10.1007/s10817-015-9328-2.

[9] F.J. Pelletier, G. Sutcliffe, and C.B. Suttner. The Development of CASC. *AI Communications*, 15(2-3):79–90, 2002.

[10] Silvio Ranise and Cesare Tinelli. The SMT-LIB web site. `http://www.smtlib.org`.