Satisfiability Modulo Theories Competition (SMT-COMP) 2014: Rules and Procedures

David R. Cok
GrammaTech, Inc.
Ithaca, NY (USA)
cok@frontiernet.net

David Deharbe
Federal University of Rio Grande do Norte
Brazil
David.Deharbe@loria.fr

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

This version revised 2014-6-11

Comments on this document should be emailed to the smtcomp mailing list or, if necessary, directly to the organizers (cf. email addresses in the following section and in the authors' affiliation information).

1 Important dates and information

- May 15 - deadline for new benchmarks
- June 1 - deadline for solver entrants
- June 15 - deadline for final versions of solvers (but no new entrants after May 25).
- July 17 - SMT workshop; completion of competition.
- SMT-COMP mailing list: smt-comp@cs.nyu.edu
- signup site for mailing list: `cs.nyu.edu/mailman/listinfo/smt-comp`
- competition web site: `www.smtcomp.org` or `smtcomp.sourceforge.net/2014`

2 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 [8, 7]. More information on the history and motivation for SMT-COMP can be found at the SMT-COMP web site, `www.smtcomp.org`, and in reports on previous competitions ([5, 6, 4, 3, 1, 2]). SMT-COMP 2014 is affiliated with the SMT workshop (`http://smt2014.it.uu.se/`) at the co-located meetings of CAV, IJCAR, and SAT at FLoC 2014 (`http://vsl2014.at/`); the competition is part of the FLoC Olympic Games (`http://vsl2014.at/olympics/`).

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 2014 will have two tracks: the conventional main track and a repeat of 2012's application track. Within a track there are multiple divisions, where each division uses benchmarks from a specific SMT-LIB logic (or group of logics).

The rest of this document, revised from the previous version,[1] describes the rules and competition procedures for SMT-COMP 2014. The principal changes from the previous competition rules are the following:

- The competition will be run with StarExec (`www.starexec.org`). Accordingly we are simplifying the competition somewhat, since some of the infrastructure needs to be recreated.

- We will hold just a main and application track. [The demonstration unsat core, and proof generation tracks will be deferred until benchmarks and result analysis tools are reorganized.] We encourage submission of solvers and benchmarks for both tracks.

- In the main track, we will measure comparative performance within all divisions to which at least two solvers are submitted. We will use the data to determine both sequential and parallel winners.

- We will recognize winners as measured by number of benchmarks solved; we will also recognize best open source contributions and best new entrants, as described below. Also, importantly, the FLoC Olympic Games is sponsoring award ceremonies (with actual medals). A selection of the SMT-COMP winners (as described below) will be awarded FLoC Olympic Games medals.

All results of the competition will be made public. However, winners will be declared only for divisions in which at least two solvers are submitted. Although the organizers may add other solvers for comparison purposes, only solvers that are explicitly submitted by their authors are eligible to be designated as winners.

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

3 Entrants

Solver format. 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 logic divisions in which it is participating*. For clarity, the solver should have just one configuration, or at least, that the default configuration is the one to be used for competition. A submission must also include a 32-bit unsigned integer. These numbers, collected from all submissions, are used to seed the pseudo-random benchmark selection algorithm, as well as the benchmark scrambler.

Information about how to configure and upload a solver is contained in the StarExec user guide on its wiki: `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 (July 16).

Other systems. The organizers' intent is to promote as wide a comparison among 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. The organizers also reserve the right to include other systems of interest (such as entrants in previous competitions) in the competition.

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

Attendance. As with previous SMT-COMPs, submitters of an SMT-COMP entrant need not be physically present at the competition to participate or win.

Deadlines

Main competition track. SMT-COMP entries must be submitted via StarExec and email to the organizers by the end of June 1, 2014 (anywhere on earth). At that time new entrants will be closed to the public to prepare for the competition, with the exception that resubmissions of

existing entries will be accepted until the end of June 15, 2014 (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 will be no opportunity to do extensive testing using StarExec after the initial deadline.

Competition solvers should be submitted with just one StarExec configuration so that it is clear what configuration should be used for the competition. Separately named StarExec solvers can wrap the same underlying tool with a different configuration, if needed, for the application track.

The versions that are present on the execution service at the conclusion of the grace period will be the ones used for the competition, and 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. See Section 7 below for a full timeline.

Application track. The same deadlines and procedure for submitting to the main track will be used also for the application track. Submissions to both the application track and the main competition are independent: participants must submit explicitly to both events to participate in both, and they may submit different (or differently configured) solvers to each. Benchmarks will be scrambled also for this division, using the same scrambler and seed as the main track. Entrants should still include a system description, as for the regular competition.

The infrastructure for the application track is in the process of being recreated for StarExec. If it is not ready in time, the organizers may choose to drop this track or only run it as a demonstration track.

4 Execution of Solvers

Solvers will be publicly evaluated in the main and application tracks, each in several logic divisions. All divisions with at least two solver entrants are competition divisions, i.e., a winner will be announced. The other divisions are exhibition divisions, i.e., solvers are evaluated and results posted, but no awards are announced.

4.1 Logistics

Dates of competition. We anticipate that the bulk of the main track of the competition will take place during the weeks leading up to SMT 2014, from about June 16 to July 17. Results will be announced at SMT 2014 and in an awards ceremony scheduled by the FLoC Olympic Games coordinator. Intermediate results will be regularly posted to the SMT-COMP website as the competition runs.

The organizers will prioritize the running of the competition tracks, and may shift the time period or order of the competition or exhibition tracks in order to complete SMT-COMP prior to the SMT workshop and awards ceremony.

Input and Output. A participating solver must read a single benchmark script presented as its first argument. Note that previous competitions presented benchmarks to a solver on its standard

input channel, and that the application track will send command inputs to the solver's standard input. Accordingly, solvers are advised to support both input mechanisms. The organizers have easily adapted solvers that use only the standard input to StarExec with a simple wrapper script.

The 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 just a text file containing a sequence of SMTLIBv2 commands in which:

1. The (single) **set-logic** command setting the benchmark's logic is the first command after any **set-option** commands described below.
2. 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; in the application track, the :print-success option may not be disabled.
3. The application track executor will send an initial **set-option :print-success true** command to the solver.
4. The script next optionally contains (any number of) **set-info** commands, which may be ignored by the solver. Benchmarks are expected to have the following **set-info** commands: **(set-info :smt-lib-version 2.0)**, **(set-info :category ...)**, **(set-info :source ...)** (indicating the authorship or tool that produced the benchmark), and **(set-info :status ...)**.
5. The **exit** command is the last command.
6. For tracks other than the application track, there is exactly one **check-sat** command, following possibly several **assert** commands.
7. For 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.
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** and **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 publicly available from the "Documents" section of the SMT-LIB website [9]. Solvers will be given formulas just from the Problem Divisions indicated during their submission to SMT-COMP.

4.2 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, without the quotation marks) 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 postprocessor (named "SMT2 results conserv") to accumulate the results.

Timeouts. Each SMT-COMP solver will be executed on an unloaded competition machine for each given benchmark, up to a fixed time limit. The time limit is yet to be determined, but it is anticipated to be 1500 seconds (25 minutes) of CPU time and the same limit on wall clock time.[2] 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 limit per solver-benchmark job-pair to be quite high, on the order of 100GB. The values of both the timeout limit and the memory limit are available to a solver process through environment variables defined and set by the StarExec service (see the StarExec user guide for more information).

Aborts and unparsable output. Solvers which exit before the time limit without reporting a result (*i.e.* due to exhausting memory, crashing, or producing output other than `sat`, `unsat`, or `unknown`) will be considered to have aborted. Also, as a further measure to prevent misjudgments of solvers, any "`success`" outputs will be ignored.[3]

Persistent state. Solvers are allowed to create and write to files and directories during the course of an execution, but they are not allowed to read such files back during later executions. Any files written should be put in the directory in which the tool is started, or in a subdirectory.

The solver is executed with a temporary directory as the current working directory. Any generated files should be produced there, not, for example, in the system tmp area. The StarExec system sets a limit on the amount of disk storage permitted - typically 20GB (cf. the User Guide). The temporary directory and its contents are deleted after the job is complete.

4.3 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*.

[2] The time limit may be adjusted once we understand the resources and capabilities of the StarExec service and the number of competition solvers.

[3] Note that SMT-LIBv2 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 therefore allows for submitting fully-compliant solvers without the need of a wrapper script, while still allowing entrants of previous competitions to run without changes.

In order to facilitate the evaluation of the solvers in this track, we will set up a "simulation" of the aforementioned interaction, as was done in 2012. In particular each benchmark in the application track represents a realistic communication trace, containing multiple **check-sat** commands (possibly with 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, possibly aborting the execution in case of a wrong answer;
- it guarantees a fair execution for all solvers by abstracting from any possible crash, misbehavior, etc. that may happen on the model checker side.

Note that the executor terminates processing the script upon receiving an incorrect response.

The disk space and memory limits on the application track are the same as for the main track.

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 **incremental benchmark script**, which is an SMT-LIBv2 script which satisfies the rules for an application script given above. Note also that the trace executor will send a single **set-option :print-success true** command to the solver before sending commands from the incremental benchmark script.

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, with a `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 these benchmarks.

Timeouts. A time limit is set for the whole execution of each application benchmark (consisting of multiple queries). We anticipate the timeout to be around 25 minutes (as it has been in the past).[4]

5 Benchmarks and Problem Divisions

In 2012 the competition ran only some logics as competitive divisions. In 2014 we will revert to previous practice and run each logic as a competitive division, if there are two or more solvers submitted.

[4]The time limit may be adjusted once we understand the resources and capabilities of the StarExec service and the number of competition solvers.

5.1 Main track

Benchmark sources. Benchmark formulas for each division of the main track will be drawn from the SMT-LIB library. The deadline for new benchmarks is May 15. The organizers will be checking and curating these until June 1; the organizers reserve the option 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.

Benchmark availability. The deadline for new benchmarks is May 15, 2014. The benchmarks will be made available as soon as possible after that, as they are checked and curated. The goal for completing the set of benchmarks is June 1. 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. In particular, there are checks for integer completeness (*i.e.* benchmarks that are satisfiable under the reals but not under the integers) and big number support (*i.e.* benchmarks that are likely to fail if integers cannot be represented beyond some maximum value, such as $(2^{31} - 1)$).

- *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 benchmarks are assigned a difficulty. For 2014, the difficulty measure is the CPU time taken by the best performing solver in the 2013 SMT-EVAL. For new benchmarks, we will determine an approximate difficulty using trial runs. The difficulty values will be posted as soon after May 15 as possible. The difficulty values are best estimates by the organizers and may not always completely correlate with actual difficulty.

The selection of benchmarks for the competition will be biased towards more difficult benchmarks as follows. First, the benchmark pool is culled as follows:

1. **Retire very easy benchmarks.** Eliminate benchmarks for which all of the current solvers in SMT-EVAL-2013 solved the benchmark in under 5 seconds, *unless* doing so reduces the pool of benchmarks for the division to less than 300.

2. **Retire inappropriate benchmarks.** The competition organizers will remove from the eligibility pool certain SMT-LIB benchmarks that are inappropriate or uninteresting for competition, or cut the size of certain benchmark families to avoid their over-representation.

3. **Eliminate benchmarks whose status is unknown.** The selection process ignores any benchmark whose expected output is neither sat nor unsat. There is a significant number of 'unknown' benchmarks in the library; these are certainly interesting and are a challenge to solve, but they are not used in the competition.

Then, for each division, the benchmarks are divided into quintiles according to the difficulty measure. Next a random selection of N competition benchmarks is made as follows:

- N*40% are randomly selected from the top quintile, if sufficient benchmarks are available
- additional benchmarks are randomly selected from the second quintile, to produce N*60% benchmarks, if sufficient benchmarks are available
- additional benchmarks are randomly selected from the third quintile, to produce N*75% benchmarks, if sufficient benchmarks are available
- additional benchmarks are randomly selected from the fourth quintile, to produce N*90% benchmarks, if sufficient benchmarks are available
- any remaining needed benchmarks are randomly selected from the bottom quintile.

Within each quintile selection,

- if the industrial benchmarks comprise less than 85% of the benchmarks in that quintile, 85% will be chosen from the industrial benchmarks (if a sufficient number are available), and the others from the population of random and crafted benchmarks;
- if the industrial benchmarks comprise at least 85% of the benchmarks in that quintile, the benchmarks are chosen at random from the population of industrial, random and crafted benchmarks in that quintile.

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 will be run in *heats*. For each division, some number N of benchmarks will be selected as described and randomly divided into a number of (possibly unequal-sized) heats. The heats will be run in order. If the competition ends before all heats have completed, the remaining heats will be ignored. All benchmarks will be chosen at the beginning (that is, without reference to the results of any given heat).

The main purpose of the algorithm above is to have a balanced and complete set of benchmarks. The built-in bias is towards industrial rather than crafted or random benchmarks and towards more difficult benchmarks. This reflects a desire by the organizers and agreed upon by the SMT community to emphasize problems that come from real applications.

Pseudo-random numbers will be generated using the standard C library function `random()`, seeded (using `srandom()`) with the sum, modulo 2^{30}, of the numbers provided in the system

descriptions (see Section 3 above) 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 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. Benchmarks will also be slightly scrambled before the competition, using a simple benchmark scrambler seeded with the same seed as the benchmark selector. Both the scrambler and benchmark selector will be 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 is considered cheating).

5.2 Application track

Benchmark sources. Benchmarks for the application track will be collected by the SMT-COMP organizers. Any benchmark available to the organizers by May 15 will be considered eligible.

Benchmark availability. A first release of the application track benchmarks will be made available as soon after the benchmark deadline as the organizers can arrange. No additional benchmarks will be added after this date, but benchmarks can be modified or removed to fix possible bugs or other issues. The final selection that will be used for the competition will be posted when the competition begins.

Benchmark demographics and selection. A random selection of all the available benchmarks will be used for the competition. As was the case in 2012, no difficulty will be assigned to the benchmarks for the application track. Benchmarks will be slightly scrambled before the competition, using the same scrambler and random seed as the main track. If the organizers determine that there is adequate time to complete the competition, all of the benchmarks will be used. Otherwise an unbiased random subset will be used to ensure timely completion of the competition.

6 Judging and Scoring

6.1 Scoring

The score for each benchmark is a triple $\langle e, n, m\rangle$, with e a non-negative number of erroneous results (usually 0, possibly 1 for main track benchmarks, and possibly more than one for application benchmarks), $n \in [0, N]$ an integral number of points scored for the benchmark, where N is the number of **check-sat** commands in the benchmark, and $m \in [0, T]$ is the (real-valued) CPU time in seconds, where T is the timeout. Recall that main track benchmarks will have just one **check-sat** command; application track benchmarks may have multiple **check-sat** commands.

Main track - sequential mode. For the main track, the score on a given benchmark is the single $\langle e, n, m\rangle$ triple, with m being the CPU time it takes to process the benchmark, that is, until the solver process terminates, and

- $e = 0, n = 1$ if the result of the check-sat command is correct,

- $e = 1, n = 0$ if the result is incorrect.
- $e = 0, n = 0$ if the result is `unknown` or the solver times-out or otherwise aborts without a response, and

Main track - parallel mode. The StarExec infrastructure reports both wall-clock time and CPU time for a given benchmark-solver job-pair. Each StarExec node has multiple (currently 4) cores; a multi-threaded solver can in principle make use of these cores to reduce its wall-clock time. Since the competition has not to date emphasized parallel solvers, we will use total CPU time (across all cores and subprocesses) as the measure of performance in the sequential mode of the main track; the imposed time out will also be on the CPU time.

However, we can also measure the performance using wall-clock time, the key performance criterion for a parallel solver. We will use the same competition data to determine a parallel-solver winner as follows. The competition will be run with a timeout T for both CPU time and wall-clock time. Since the computation nodes have C=4 cores, we can simulate a competition that had a time out of T CPU seconds and T/C wall-clock seconds by simply declaring that any solver taking more than T/C wall-clock time has timed out for the parallel competition. We then recompute the scoring as above but using wall-clock time and a timeout value of T/C.

Parallel track scores are not directly comparable to the sequential track scores. However, we will use them to determine a parallel-track winner in each division.

Application track. An application benchmark may have multiple check-sat commands and may partially solve the benchmark before timing out. The benchmark is run by the benchmark executor, measuring the total time (summed over all individual commands) taken by the solver to respond to commands. It is expected that nearly all of this time will be in response to a check-sat command, but assert, push or pop commands might also entail a reasonable amount of processing.

- If the solver completes the benchmark without timing out, the solver receives a score of $\langle e, n, m\rangle$, where e is the number of erroneous responses to commands, n is the number of correct responses to check-sat commands, and m is the CPU time taken.
- If the solver times out while processing the benchmark, the solver receives a score of $\langle e, n, m\rangle$, where e is the number of erroneous responses to commands, n is the number of correct responses to check-sat commands before timing out, and m is equal to the timeout value.

As queries are only presented in order, this scoring system may mean that relatively "easier" queries are hidden behind more difficult ones located at the middle of the query sequence. We will not perform a parallel performance analysis of the application track.

Competition scoring. Benchmarks' scores are summed component-wise to form a solver's total score for that division of the competition. Total scores are compared lexicographically—a score $\langle e, n, m\rangle$ is better than $\langle e', n', m'\rangle$ iff $e < e'$ or ($e = e'$ and $n > n'$) or ($e = e'$ and $n = n'$ and $m < m'$). That is, fewer errors takes precedence over more correct solutions, which takes precedence over less time taken.

6.2 Judging

It may happen that faulty benchmarks are discovered in the course of the competition. The judges 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). Organizers on solver teams will be recused from such decisions. Solver teams may appeal to the organizers if they suspect that a benchmark is faulty or otherwise not within the rules.

6.3 Other recognitions

The organizers will also recognize the following contributions:

- *Open source*. In addition to recognizing the overall winner in each division, the top solver that provides its source code as open source will also be recognized in each division.
- *Best new entrant*. The best performing entrant from a new solver implementation team, as measured by the Olympic Games metric defined below.
- *Breadth of logics*. Tools 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.

6.4 FLoC Olympic Games medals

The FLoC Olympics games medals will be awarded based on competition-wide critera.

We define a competition-wide metric M as follows. Each solver competing in a division receives a number of points for its performance in that division: $\langle e, (n/N)^2 \rangle$, where e is the number of wrong answers to benchmarks, n is the number of benchmarks solved correctly, and N is the number of benchmarks used in that division. A solver's score for the whole competition is a weighted sum of these tuples over all the divisions in which the solver participates. The best value of M is the one with the least total value of e, and secondly, the largest total value for the second component of the tuple. The weight for each division is the log_{10} of the number of benchmarks used in the competition for that division. For the purpose of determining the winner of a medal, a division is competitive if there are at least two *teams* competing in that division.

The Olympic Games medals will be awarded as follows.

- The gold medal we will be awarded to the team entering the solver with the best competition-wide value of M.
- The silver medal will be awarded to the team entering the solver with the second-best competition-wide value of M.
- The bronze medal will be awarded to the solver that wins the (sequential) QF_BV division according to the usual $\langle e, n, m \rangle$ metric of subsection 6.1.

- The team winning the gold medal is excluded from consideration for the silver medal; the teams winning the gold and silver medals are excluded from consideration for the bronze medal.

Rationale. The metric M 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 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 both to simplify the metric and because it is only of third importance in the regular competition metric and is difficult to compare across divisions. The divisions are weighted with a weak weight (log) because many divisions have a small number of benchmarks. The restriction requiring at least two teams to enter a division for the division to be counted toward the medal avoids a situation where a team enters multiple solvers (albeit with different source code) just to add more competitive divisions.

The gold and silver medals emphasize breadth of logic support; the organizers chose to award the bronze medal to excellence in a single logic. Rather than try to compare performance across logics, we selected one logic, QF_BV, for this award. This logic is a commonly supported logic and many solvers concentrate on QF_BV problems alone.

7 Timeline

May 15 Deadline for benchmark contributions.

June 1 First versions of solver competitors are due to the organizers, including divisions being entered. Versions of the benchmark scrambler, benchmark selector and trace executor made available.

June 8 Final version of the benchmark scrambler, benchmark selector and trace executor made available.

June 8 Benchmark libraries are frozen.

June 15 Final versions of solvers due via StarExec (for all tracks), including divisions being entered, system descriptions and magic numbers for benchmark scrambling.

June 16 Opening value of NYSE Composite Index used to complete random seed.

July 17 SMT 2014; end of competition.

8 Mailing List

Interested parties should subscribe to the SMT-COMP mailing list (at `cs.nyu.edu/mailman/listinfo/smt-comp`). 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.

9 Acknowledgments and Disclaimer

- David Cok, David Deharbe and Tjark Weber are the organizers of SMT-COMP 2014. David Cok is the lead and 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. He has used solvers in industrial settings and is keenly interested to know which are the best for various applications.
- Many others have contributed benchmarks, effort, and feedback. In particular, Clark Barrett and Morgan Deters processed the many benchmark submissions and completed the work of transitioning benchmarks from SMTExec to StarExec. Aaron Stump fielded many questions and diagnosed and managed the fixing of bugs in Star Exec.
- The StarExec cluster is supported by NSF grants #1058748 and #1058925.
- David Deharbe is associated with the solver group producing the veriT solver.

References

[1] SMT-COMP 20112 competition report. `http:smtcomp.sourceforge.net/2012/reports/SMTCOMP2012.pdf`.

[2] SMT-EVAL 2013 evaluation report. In preparation. It will be available at `http://smtcomp.sourceforge.net/2013/reports/SMTEVAL2013.pdf`.

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

[4] C. Barrett, L. de Moura, and A. 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.

[5] Clark Barrett, Morgan Deters, Albert Oliveras, and Aaron Stump. Design and results of the 4th annual satisfiability modulo theories competition (SMT-COMP 2008). In preparation.

[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] D. Le Berre and L. 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-Verlag, 2003.

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

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