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.Deharbe@loria.fr

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

This version revised 2015-2-17

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 [7, 8]. 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, 5, 6]).

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). [Additional tracks, e.g., for unsat-core extraction or proof generation, are currently being considered.] 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:

[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.

- All solvers will be run with $n = 4$ cores available, and the CPU timeout will be n times the wall-clock timeout. We will recognize both best sequential and best parallel performance in all divisions.
- In addition to recognizing the best solver in each division, we will recognize the three 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.
- [Pending availability of suitable benchmarks and solvers, we may include a new division for floating-point arithmetic. This should be considered experimental in 2015, and will not be included in the competition-wide ranking.]
- [Benchmark selection and relative weight of benchmark families are under review.]

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 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, `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.

Other systems. 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. 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. Submitters of an SMT-COMP entrant need not be physically present at the competition 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, 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.

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. 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** 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 only from the divisions into which they have been entered.

Timeouts. Each SMT-COMP solver will be executed on a dedicated processor with 4 cores of a competition machine for each given benchmark, up to a fixed time limit. Detailed machine specifications are available on the competition web site. The time limit is yet to be determined, but it is anticipated to be 40 minutes of wall-clock time (and 160 minutes of CPU 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.

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

The StarExec service also limits the memory consumption of the solver processes. We expect the limit per solver-benchmark job-pair to be on the order of 60 GB. 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).

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 postprocessor (named "SMT2 results conserv") to accumulate the results.

Aborts and unparsable output. Solvers which exit before the time limit without reporting a result (i.e., due to exhausting memory or crashing) and do not produce output that includes `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 (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 User Guide). The temporary directory and its contents are deleted after the job is complete.

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

[3] Note that SMT-LIB v2 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.

- 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, and 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.

6 Benchmarks and Problem Divisions

6.1 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 1, 2015**. The organizers will be checking and curating these until **June 7, 2015**; 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. New benchmarks will be made 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. 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. [This section is under review.] Each benchmark is assigned a difficulty. For 2015, the difficulty measure is the CPU time taken by the best performing solver in the 2014 SMT-COMP. For new benchmarks, we will determine an approximate difficulty using trial runs. The difficulty values will be posted as soon after May 1 as possible. The difficulty values are best estimates by the organizers and may not always reflect 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 [or perhaps some] of the current solvers in SMT-COMP 2014 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 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.

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 4 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).

6.2 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 1, 2015** 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.

7 Scoring

Scores will be computed for all solvers and divisions. However, winners will be declared only for *competitive* divisions. A division 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.

7.1 Benchmark scoring

A solver's score for each benchmark is a quadruple $\langle e, n, w, c\rangle$, with $e \in \{0, 1\}$ the number of erroneous results (usually $e = 0$), $n \in [0, N]$ an integral number of points scored for the benchmark, where N is the number of **check-sat** commands in the benchmark, $c \in [0, 4T]$ is the (real-valued) CPU time in seconds, and $w \in [0, T]$ is the (real-valued) wall-clock 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. [Clarify how $\langle e, n, c, w\rangle$ depend on solver behavior.]

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 **check-sat** commands, but **assert**, **push** or **pop** commands might also entail a reasonable amount of processing.

7.2 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. Total scores are compared lexicographically: a score $\langle e, n, w, c\rangle$ 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.

7.3 Competition-wide scoring (main track)

We define a competition-wide metric for the main track as follows. Let B_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

score for this division. The solver's competition-wide 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. [Rationale.]

7.4 Sequential performance

SMT-COMP has traditionally emphasized sequential performance (i.e., CPU time) over parallel performance (i.e., wall-clock time). Accordingly, we will recognize the best *sequential* solver, both in each division and according to the competition-wide metric (for the main track). A solver is eligible for this recognition if, for each benchmark in the division, the solver's CPU time does not exceed its wall-clock time.

7.5 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.

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. [Specific individuals will be named later.] 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] 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`.