

# Satisfiability Modulo Theories Competition (SMT-COMP) 2009: Rules and Procedures

Clark Barrett  
Computer Science  
New York University

Morgan Deters  
LSI Department  
Technical University of Catalonia

Albert Oliveras  
LSI Department  
Technical University of Catalonia

Aaron Stump  
Department of Computer Science  
University of Iowa

## 1 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 [6, 5]. More information on the history and motivation for SMT-COMP can be found at the SMT-COMP web site, [www.smtcomp.org](http://www.smtcomp.org), and in reports on previous competitions ([3, 4, 2, 1]). SMT-COMP 2009 is affiliated with the 22nd International Conference on Automated Deduction (CADE-22), held in Montreal, Canada, this year.

The rest of this document, updated from the last year's version, describes the rules and competition procedures for SMT-COMP 2009. As in 2008, solvers shall be entered for competition by way of the SMT-Exec service, and a two-day grace period (after the initial deadline) will be provided to re-upload a solver *provided that some version of it was uploaded by the initial deadline*. The primary changes from last year's version are:

- use of 64-bit architecture in the SMT-Exec service for solver execution.
- small updates to the benchmark selection algorithm.
- elimination of the disqualification rule for solvers giving four or more incorrect answers in a competition division; rather, the scoring system alone will be responsible for punishing solvers that give incorrect answers.

## 2 Entrants

**Solver format.** An entrant to SMT-COMP is an SMT solver submitted using the SMT-Exec service. SMT-Exec 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 10-node computer cluster purchased with funds from a National Science Foundation Computing Research Infrastructure grant. SMT-Exec is provided free of charge, but it does require a minimal registration, which verifies an email address and prevents misuse of the service. Registered users may then upload their own solvers to run, or may run public solvers already uploaded to SMT-Exec. SMT-Exec provides a variety of tabular and graphical displays of results of solver executions, for comparison. For SMT-COMP 2009, SMT-Exec will be configured so that jobs are run on 64-bit architecture and single-processor mode.

For participation in SMT-COMP, a solver must be uploaded as a “competition” solver via the usual SMT-Exec upload mechanism, or, alternatively, a previously-uploaded solver may be marked as a “competition” solver. In either case, *uploads must be marked for competition before the deadline*; uploading a solver to SMT-Exec is not sufficient for competition entry if it’s not marked as being a competition entrant. The md5 checksums of competition submissions will be public immediately after the deadline for competition entry has passed to ensure transparency, and the submissions themselves will be made public after the competition. Source code need not be provided. However, in order to encourage sharing of source code, extra recognition will be given to solvers providing source code distributions including recognition for the top such solver in each division. See SMT-Exec ([www.smtexec.org](http://www.smtexec.org)) for instructions on uploading solvers, as well as machine specifications.

**System description.** As part of their submission via SMT-Exec, SMT-COMP entrants must also include a short (1–2 pages) description of the system. This should include a list of all authors of the system and their present institutional affiliations. The programming language(s) and basic SMT solving approach employed should be described (*e.g.*, lazy integration of a Nelson-Open combination with SAT, translation to SAT, etc.). System descriptions are encouraged to include a URL for a web site for the submitted tool, but this is optional. System descriptions 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.

The SMT-Exec upload system will ask for the system description and the random seed when a solver is marked for competition, so their inclusion in the uploaded archive itself is optional.

**Other systems.** As in previous years, due to limitations on computational resources, the organizers reserve the right not to accept multiple versions of the same solver (defined as sharing 50% or more of its source code). The organizers reserve the right to submit their own systems, or other systems of interest, to the competition.

**Wrapper tools.** A *wrapper tool* is defined as any tool which calls an SMT solver not written by the author of the wrapper tool. The other solver is called the *wrapped tool*. There are several rules governing wrapper tools. For the purposes of these rules, multiple versions of a wrapped tool are considered different tools. The goal of these rules is to require wrapper tools to outperform the

tools they wrap (since otherwise, there is no apparent quantitative way to argue that the wrapper tool improves upon the wrapped tool).

- The name of the wrapper tool must end with “+name”, where name is the name of the wrapped tool (*e.g.* “Flash+CVC3” for a tool wrapping CVC3).
- If the wrapped tool is from last year’s SMT-COMP or earlier, then for each division entered by the wrapper tool, if the wrapper tool does not place ahead, according to the scoring rules below, of last year’s winner in that division, it will be disqualified from that division (but not necessarily from the whole competition).
- If the wrapped tool was released after last year’s SMT-COMP, then the wrapper tool can be entered only if
  - Permission has been given by the author of the wrapped tool
  - The wrapped tool is submitted and entered in every division in which the wrapper tool is entered.

For each division entered by the wrapper tool, if the wrapper tool does not place ahead of the wrapped tool in that division, it will be disqualified from that division.

**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.** To guarantee inclusion in the competition, SMT-COMP entries must be submitted via SMT-Exec by 7pm, Eastern U.S. time, July 30th, 2009. At that time the SMT-Exec service will be closed to the public to prepare for the competition, with the exception that resubmissions of existing entries will be accepted until 7pm, Eastern U.S. time, August 1st, 2009. We strongly encourage participants to use this two day 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 SMT-Exec after the original deadline on July 30th.

The versions that are present on SMT-Exec 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 morning of August 4th. See Section 6 below for a full timeline.

### 3 Execution of Solvers

**Dates of competition.** We anticipate that the bulk of the competition will take place during the course of CADE-22 (August 4–7). Results will be announced in a special session of CADE, on the last day of the conference, as well as on the SMT-COMP web site. Intermediate results will be regularly posted to the SMT-COMP website as the competition runs.

**Input and Output.** Each SMT-COMP entrant, when executed, must read a single input formula presented on its standard input channel. All formulas will be given in the concrete syntax of the SMT-LIB format, version 1.2. The SMT-LIB format specification is publicly available from the “Documents” section of the SMT-LIB website [7]. Solvers will be given formulas just from the Problem Divisions indicated in their system descriptions. Each SMT-COMP entrant is then expected to attempt to report on its standard output channel whether the formula is satisfiable (“sat”, without the quotation marks) or unsatisfiable (“unsat”). An entrant may also report “unknown” to indicate that it cannot determine satisfiability of the formula. For more detailed information on the output format, see the description on the SMT-Exec “Upload a Solver” page.

**Timeouts.** Each SMT-COMP solver will be executed on an unloaded competition machine for each given formula, up to a fixed time limit. The time limit is yet to be determined, but it is anticipated to be around 20 minutes. 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, using the `TreeLimitedRun` script, developed for the CASC competition and available on the SMT-COMP web page. A timeout scores the same as if the output is “unknown”.

**Aborts and unparseable 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. An abort scores the same as if the output is “unknown”.

**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.

## 4 Benchmarks and Problem Divisions

We expect the problem divisions for SMT-COMP 2009 to include the following SMT-LIB *logics*. These logics are specified in SMT-LIB format on the SMT-LIB web page. Note that the “QF\_” prefix means the division’s formulas are quantifier-free. However, the organizers reserve the right to add (remove) divisions if (not) enough benchmarks and solvers exist for a particular division.

- QF\_UF: uninterpreted functions.
- QF\_RDL: real difference logic.
- QF\_IDL: integer difference logic.
- QF\_UFIDL: uninterpreted functions and integer difference logic.
- QF\_UFLIA: uninterpreted functions and linear integer arithmetic.
- QF\_UFLRA: uninterpreted functions and linear real arithmetic.
- QF\_LRA: linear real arithmetic.

- QF\_LIA: linear integer arithmetic.
- QF\_AX: arrays with extensionality.
- QF\_AUFLIA: arrays, uninterpreted functions and linear integer arithmetic.
- QF\_BV: fixed-width bitvectors.
- QF\_AUFBV: arrays, fixed-width bitvectors and uninterpreted functions.
- AUFLIA+ $p$ : (quantified) arrays, uninterpreted functions and linear integer arithmetic, patterns included.
- AUFLIA− $p$ : (quantified) arrays, uninterpreted functions and linear integer arithmetic, patterns not included.
- AUFLIRA: (quantified) arrays, uninterpreted functions and mixed linear integer and real arithmetic.

**Benchmark sources.** Benchmark formulas for these divisions will be drawn from the SMT-LIB library. Any benchmarks added to SMT-LIB by the July 1 release (see the timeline in Section 6) will be considered eligible. 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.** A first release of the competition benchmarks will be made available on June 1st 2009. A second and almost final release will be available on July 1st. 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 release that will be used for the competition will be posted on July 24th. The set of selected benchmarks 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 terminal sub-directory 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.** Before the selection process, each benchmark will be assigned a *difficulty*: an integer between 0 and 5 inclusive. The difficulty for a particular benchmark will be assigned by running SMT solvers from the 2008 competition that finished in good standing and using the formula:

$$\text{difficulty} = 5 \left( 1 - \frac{\text{solved}}{\text{total}} \right) ,$$

where *solved* is the number of SMT solvers that could solve the problem in 10 minutes and *total* is the total number of SMT solvers tried. For new divisions, the difficulty will be computed using whatever solvers are available to the organizers for that purpose. (Note that the corrected version of Alt-Ergo that was submitted after the 2008 deadline, which ran *hors concours* in the competition, will be used instead of its “official” counterpart.)

The following scheme will be used to choose competition benchmarks within each division. Unknown-status benchmarks from SMT-LIB are considered ineligible for competition and are not used. The selection is implemented by our benchmark selection tool, source for which is available at [www.smtcomp.org](http://www.smtcomp.org).

1. **Check benchmarks included.** All benchmarks in category *check* are included.
2. **Retire very easy benchmarks.** The most difficult 300 non-check non-unknown benchmarks in each division are always included, together with all benchmarks on which at least one 2008 solver required more than 5 seconds. *This is intended to have the effect of retiring “very easy” benchmarks that were solved by every 2008 solver (and therefore have difficulty 0), and that each 2008 solver could solve in less than 5 seconds, unless doing so reduces the pool of benchmarks for the division to less than 300.*
3. **Division selection pools created.** For non-*check* benchmarks, selection pools are created. For benchmark families with  $\leq 200$  eligible, non-*check* benchmarks, all are added to this pool; otherwise, 200 such benchmarks are added to the pool with the following distribution:
  - 50 with solution **sat** and difficulty 0, 1, or 2
  - 50 with solution **sat** and difficulty 3, 4, or 5
  - 50 with solution **unsat** and difficulty 0, 1, or 2
  - 50 with solution **unsat** and difficulty 3, 4, or 5

If 50 are not available in one of these subdivisions, all that are available are added, and remaining slots are reallocated to the others. This process is iterated so that it is guaranteed that 200 benchmarks from the benchmark family are in the selection pool, in equal numbers from each subdivision, so far as possible. (In cases where *e.g.* there are only two available slots and they can be allocated to one of three subdivisions, they are allocated randomly but are guaranteed to be allocated to *distinct* subdivisions.)

4. **Category slot allocation.** Next, 200 slots are allocated for the division as follows:

- 170 from category *industrial*
- 20 from category *crafted*
- 10 from category *random*

If there are fewer than 20 (respectively, 10) *crafted* or *random* benchmarks in the division pool, more *industrial* slots are allocated to make 200 total for the division. If there are too few *industrial* benchmarks in the division pool, more *crafted* slots are allocated to make 200 total for the division. (In no division are there not enough of both industrial and crafted benchmarks.)

5. **Category subdivision slot allocation.** For each category, given that it has  $n$  slots allocated to it, the slot allocation is further subdivided as follows:

- $\lfloor \frac{n}{4} \rfloor$  slots for solution **sat** with difficulty 0, 1, or 2
- $\lfloor \frac{n}{4} \rfloor$  slots for solution **sat** with difficulty 3, 4, or 5
- $\lfloor \frac{n}{4} \rfloor$  slots for solution **unsat** with difficulty 0, 1, or 2
- $\lfloor \frac{n}{4} \rfloor$  slots for solution **unsat** with difficulty 3, 4, or 5

Remaining slots are allocated randomly to distinct subdivisions. If there aren't enough benchmarks in the pool meeting one or more of the above subdivision requirements for the category, the subdivision allocation is reduced to the number available in the pool that meet the requirements. To make up the full category allotment, remaining slots are allocated equally to subdivisions with enough benchmarks in the pool meeting their requirements. This process ensures that all slots can be filled with benchmarks from the pool.

6. **Benchmark selection.** Benchmarks from the pool are assigned randomly to slots.

In the end, up to 200 non-*check* benchmarks per division are included in the competition, together with all the *check* benchmarks. Some divisions may have fewer than 200 non-*check* benchmarks, in which case all of them are included using this selection scheme.

The main purpose of the algorithm above is to have a balanced and complete set of benchmarks. The one built-in bias is towards industrial rather than crafted or random 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 2 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 August 4th will be added to the other seeding values. This helps provide transparency, by guaranteeing that the organizers cannot manipulate the seed in favor of their submitted solvers.

<b>Reported</b>	<b>Correct?</b>	<b>Point/penalty</b>
unsat	yes	+1
unsat	no	-8
sat	yes	+1
sat	no	-8
unknown	n.a.	0
<i>timeout</i>	n.a.	0
<i>abort</i>	n.a.	0

Figure 1: Points and Penalties

Benchmarks will also be slightly scrambled before the competition, using the same scrambler as last year, 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 Judging and Scoring

Winners in each Problem Division for which there are at least three entrants from distinct research groups competing will be taken to be those with the highest score, according to the system of points and penalties in Figure 1. In addition to recognizing the overall winner in each division, the top solver providing its source code will also be recognized in each division. As indicated, correct answers are awarded a positive number of points, while incorrect answers are penalized by assigning a negative number of points. Timeouts, aborts, and reports of unknown are awarded zero points. In the event of a tie in total number of points, the solver with the lowest average CPU time on formulas for which it gave a correct answer will be considered the winner. For Problem Divisions with fewer than three entrants, the results will be reported but no winner officially declared.

## 6 Timeline

**June 1** First version of the benchmark library posted for comment.

**July 1** No new benchmarks can be added after this date, but problems with existing benchmarks may be fixed.

**July 24** Benchmark library is frozen.

**July 30** (7pm ET) Final versions of solvers due via SMT-Exec, including system descriptions and magic numbers for pseudo-random benchmark selection and scrambling.

**August 1** (7pm ET) Close of two-day grace period for resubmission of entries; md5 checksums of all entries are posted.

**August 4** Opening value of NYSE Composite Index used to complete random seed.

**August 4–7** Anticipated dates for competition.

## 7 Mailing List

Interested parties should subscribe to the SMT-COMP mailing list, a link to which is found at [www.smtcomp.org](http://www.smtcomp.org). 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.

## References

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.

- [5] 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.
- [6] F.J. Pelletier, G. Sutcliffe, and C.B. Suttner. The Development of CASC. *AI Communications*, 15(2-3):79–90, 2002.
- [7] Silvio Ranise and Cesare Tinelli. The SMT-LIB web site, 2004. <http://combination.cs.uiowa.edu/smtlib>.