

Satisfiability Modulo Theories Competition (SMT-COMP) 2007: Rules and Procedures

Clark Barrett
Computer Science
New York University

Albert Oliveras
LSI Department
Technical University of Catalonia

Aaron Stump
Computer Science and Engineering
Washington University in St. Louis

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 [3, 2]. More information on the history and motivation for SMT-COMP can be found at the SMT-COMP web site, <http://www.smtcomp.org/>, and in reports on previous competitions [1]. SMT-COMP 2007 is affiliated with the Computer-Aided Verification (CAV) conference, held in Berlin, Germany. The rest of this document, updated from the previous year's version, describes the rules and competition procedures for SMT-COMP 2007.

2 Entrants

Solver format. An entrant to SMT-COMP is an SMT solver submitted in either source code or binary format to the organizers. Check the SMT-COMP web page for the exact mechanism for online submission. Binary format submissions must be compatible with the competition hardware and operating system, which will be x86 Linux. For solvers submitted in source code form, the organizers will make reasonable efforts to ensure that the source code is not made public. Binaries for all solvers, however, will be made public after the competition. Parties interested in participating in SMT-COMP must be able to provide, either directly or via source code, a publicly distributable x86 Linux binary.

Solvers provided in source format must include a README file explaining how to compile and install the solver. Solvers must compile using publicly available (or specially provided) compilation tools on the competition machines. Installation and execution of solvers must not require root access. The organizers will make reasonable efforts to install each system, including communication with the submitters of the system in case of difficulties. Nevertheless, the organizers reserve the right to reject an entrant if its compilation or installation process proves overly difficult.

System description. SMT-COMP entrants must come with 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. It should list the Problem Divisions (see Section 4 below) in which the entrant wishes to compete. The programming language(s) and software architecture of the system should also be described, as well as the basic SMT solving approach employed (e.g., lazy integration of a Nelson-Oppen combination with SAT, translation to SAT, etc.). The system description should also include some speculation as to likely performance in SMT-COMP. System descriptions are encouraged to include a URL for a web site for the submitted tool, but this 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 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 for previous SMT-COMPs, 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 by June 25, 2007. See Section 6 below for a full timeline.

3 Execution of Solvers

Dates of competition. Solvers will be executed for SMT-COMP 2007 during the course of CAV 2007 (July 3-7). Results will be announced in a special session of CAV, on CAV's last day, as well as on the SMT-COMP web site. Depending on computational resources and the number of entrants, the competition may begin some days earlier than CAV, to allow sufficient execution time. Intermediate results will be regularly posted to the SMT-COMP website as the competition runs.

Distribution. Each SMT-COMP entrant must be submitted as a (possibly compressed) tarfile, suitable for reading with the unix "tar" command. This tarfile must archive a distribution directory with the same name as the solver. Within this directory must be a file called "README" explaining how to build and configure the solver. The system description should also be included in the distribution. After following the instructions in the "README" file, the distribution directory must contain an executable file called "run". This is the executable that will be used during the competition. This executable will be run, from the distribution directory, as described next. The intention is that "run" should ensure conformance to the input/output requirements below (for example, by setting any necessary command-line flags to the solver). The "run" command will be called during the competition only from the distribution directory.

Input. Each SMT-COMP entrant's "run" command, 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 [4]. Solvers will be given formulas just from the Problem Divisions indicated in their system descriptions. Unlike in previous years, compliance of "run" commands with the simple interface described here will be enforced.

Output. For its given input formula, each SMT-COMP entrant is 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. Whitespace is allowed, but any output other than one of these options may not be correctly parsed by the competition scripts. In such a case, the output will be considered to be "unknown".

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 10 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. Solvers which exit abnormally before the time limit without reporting a result (i.e. due

to exhausting memory or crashing) 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. The restriction on not reading files created in previous runs will not be rigorously enforced, but entrants found to have willfully violated it will be considered to have cheated.

4 Benchmarks and Problem Divisions

The Problem Divisions for SMT-COMP 2007 are 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.

- 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_LRA: linear real arithmetic.
- QF_LIA: linear integer arithmetic.
- QF_AUFLIA: arrays, uninterpreted functions and linear integer arithmetic.
- QF_UFBV: fixed-width bitvectors and uninterpreted functions.
- QF_AUFBV: arrays, fixed-width bitvectors and uninterpreted functions.
- AUFLIA: (quantified) arrays, uninterpreted functions and linear integer arithmetic.
- 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 before June 1 will be considered eligible. SMT-COMP will 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 version of the competition benchmarks will be made available on May 1, 2007. A second version incorporating repairs and possibly some late additions will be made available June 1, 2007. No additional benchmarks will be added after June 1. The

selected benchmarks will be posted when the competition begins. The feedback process and iteration through multiple versions will allow the community to catch any (*a priori* conceivable) errors in the competition benchmarks.

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 2006 competition on it 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.

For each division, the following scheme will be used to choose benchmarks:

1. First, all benchmarks in the *check* category are automatically included.
2. The remaining benchmarks are put into a selection pool as follows: for each family, if the family contains more than 200 benchmarks, then 200 randomly selected benchmarks are put into the pool. Otherwise all of the benchmarks from the family are put into the pool.
3. Slots are allocated for 100 benchmarks to be selected as follows: 85 slots are for industrial benchmarks; 10 are for crafted; and 5 are for random. If there are not enough crafted or random benchmarks, then more industrial slots are allocated. If, on the other hand, there are not enough industrial benchmarks, then more crafted slots are allocated (in no division are there are not enough of both industrial and crafted benchmarks).

Reported	Correct?	Point/penalty
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

- In order to fill the allocated slots, the pool of benchmarks created in step 2 is consulted and partitioned according to category (i.e. industrial, random, crafted). Within each category, the benchmarks are further partitioned into four sub-categories: easy-sat, easy-unsat, hard-sat, and hard-unsat. A benchmark is easy if it has difficulty 0, 1, or 2 and hard if it has difficulty 3, 4, or 5. A benchmark is “sat” or “unsat” based on its *status* attribute. An attempt is made to randomly fill the allocated slots for each category with the same number of benchmarks from each sub-category (i.e. if there are 85 industrial slots, then there should be roughly 21 in each sub-category). If there are not enough in a sub-category, then its allotment is divided among the other sub-categories.

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 (as part of their system descriptions; see Section 2 above) by all SMT-COMP entrants other than the organizers. The seed will be provided as input to a script which runs the above algorithm. The script will be made publicly available before the final submission date. In this way, we hope to ensure that the benchmark selection process is as transparent as possible.

Benchmarks will be lightly scrambled before competition, using a scrambler which will also be made publicly available. 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. 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. As in 2006, SMT-COMP 2007 will allow at most three wrong answers per division. Four wrong answers within a single division will cause the tool to be disqualified from all divisions. In the event of a tie in total number of points, the solver with the lower average CPU time on formulas for which it did not timeout or report unknown 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 (2007)

May 1 First version of the benchmark library posted for comment.

June 1 Revised version of the benchmark library posted.

June 25 Final system descriptions due, with magic numbers for pseudo-random selection of benchmarks.

July 3-7 Anticipated dates for competition.

References

- [1] C. Barrett, L. de Moura, and A. Stump. Design and Results of the 1st Satisfiability Modulo Theories Competition (SMT-COMP 2005). to appear in the Journal of Automated Reasoning.
- [2] 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.
- [3] F.J. Pelletier, G. Sutcliffe, and C.B. Suttner. The Development of CASC. *AI Communications*, 15(2-3):79–90, 2002.
- [4] Silvio Ranise and Cesare Tinelli. The SMT-LIB web site, 2004. <http://combination.cs.uiowa.edu/smtlib>.