The 2013 SMT Evaluation

David R. Cok
GrammaTech, Inc., USA
dcok@grammatech.com

Aaron Stump
University of Iowa, IA, USA
aaron-stump@uiowa.edu

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

June 6, 2014

After 8 years of SMT Competitions, the SMT Steering Committee decided, for 2013, to sponsor an evaluation of the status of SMT benchmarks and solvers, rather than another competition. This report summarizes the results of the evaluation, conducted by the authors. The key observations are that (1) the competition results are quite sensitive to randomness and (2) the most significant need for the future is assessment and improvement of benchmarks in the light of SMT applications. The evaluation also measured competitiveness of solvers, general coverage of solvers, logics, and benchmarks, and degree of repeatability of measurements and competitions.

1 Introduction

1.1 The Competition history and goals

From 2005 through 2012 (and coming again in 2014), the SMT community sponsored an annual competition among SMT solvers (cf. Fig. 1). The purpose of the competition is to encourage advances in SMT solver implementations acting on benchmark formulas of theoretical or practical interest. Public competitions are a well-known means of stimulating advancement in software tools. For example, in automated reasoning, the SAT and CASC competitions for propositional and first-order reasoning tools, respectively, have spurred significant innovation in their fields [5, 6]. Indeed, the SMT competition increased in size each year: more benchmarks were added, new solver teams participated, and more logic divisions were defined.

The particular goals of SMT-COMP include the following*:

Reference?

- enable research on SMT solvers by benchmarking and comparing performance;
- promote a standard format for SMT problems (SMT-LIB v2 [4]);
- collect additional benchmarks;
- identify and develop new theories and logics for SMT, encouraging their inclusion in SMT solvers;
- introduce SMT users and implementors to each other;
- provide a forum for SMT implementors to promote their SMT solvers and for SMT users to assess the comparative performance of solvers; and
- encourage the development of industrial-strength solvers for wide-spread use.

Fig. 1 (and some of the other figs. that look like they come from PowerPoint/ Excel) require more work.

1.2 Concerns prompting an evaluation

In 2013, the SMT Steering Committee decided to take a collective breath and sponsor an evaluation of the current state of the art, without the pressure of a competition. In particular, the implementation teams

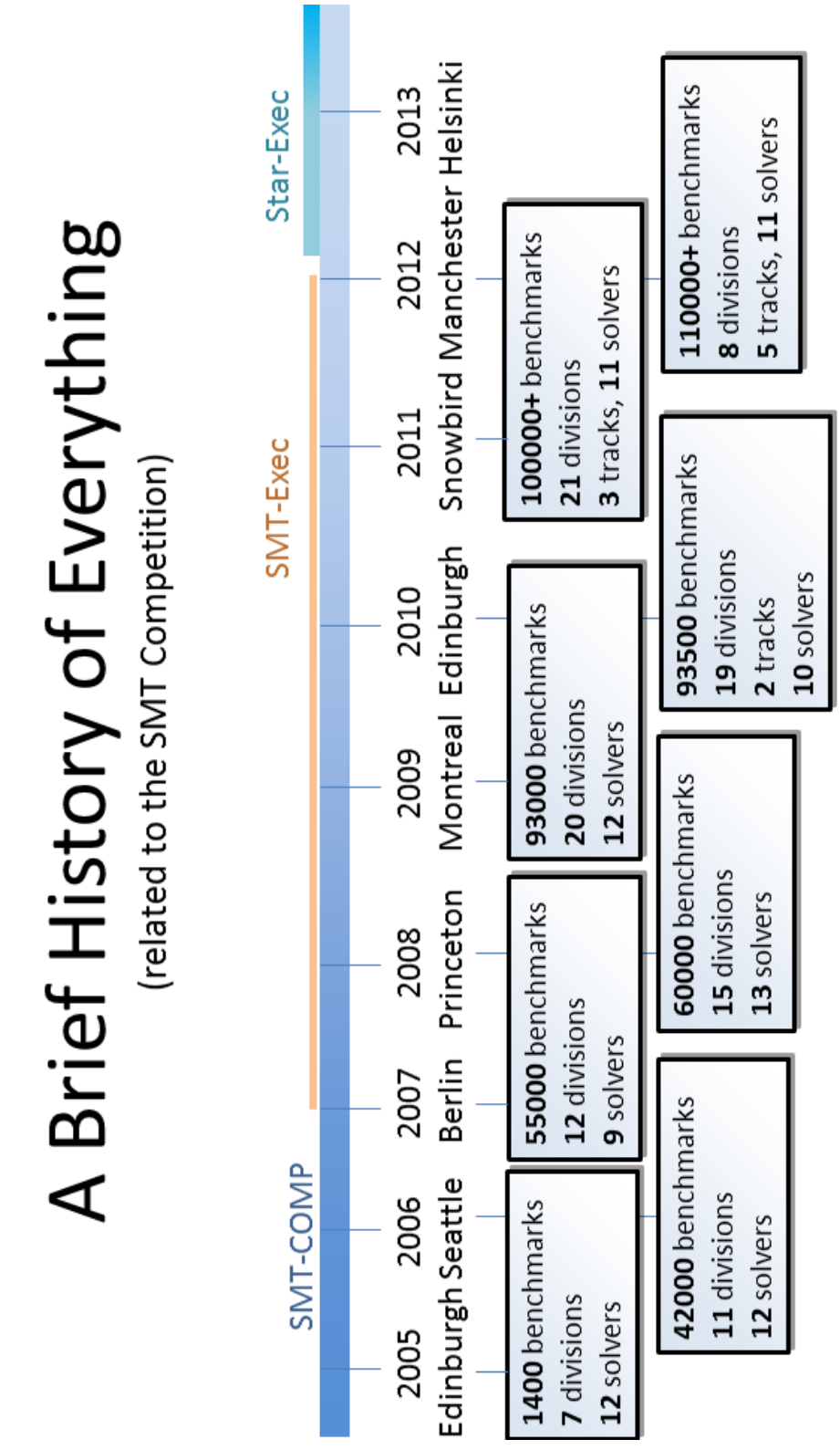


Figure 1: SMT-COMP history

found that preparing for a competition required considerable engineering work that detracted from other goals. On the one hand, the engineering work is necessary for users to use the tools as off-the-shelf applications. But holding the competition every year was causing some otherwise highly involved teams to withdraw. Thus there was a collective desire to pause the competition cycle for a year.

Second, there was a desire to evaluate the state of the SMT community and the competition. The competition had become focused on details of performance on a particular set of benchmarks for particular logics and its results had become somewhat predictable. Perhaps other goals need to be reemphasized. For example, more benchmarks from applications are needed, not just crafted challenge problems. In addition, an evaluation of progress of the SMT community was desired, not just winning narrowly focused competitions. Perhaps a variety of metrics would be useful.

This contradicts our main conclusion (1). Suggestion: insert "there was a feeling that"

And third, for many years the competition had relied on the SMT-Exec computational infrastructure. A new infrastructure, Star-Exec [1] had been funded by the NSF and developed at the University of Iowa, but had not yet been tried out in earnest. In fact, a goal in 2012 had been to use Star-Exec for SMT-COMP 2012; however, Star-Exec was not sufficiently operational and the competition had to revert to SMT-Exec. It was anticipated that an evaluation in 2013, with less deadline pressure, would enable using Star-Exec on a shake-down cruise prior to a competition in 2014 at the Federated Logic Conference (FLoC) Olympic Games [9].

Reference for SMT-Exec?

1.3 Evaluation goals

The section lists *what* we evaluated. To me this is different from our goals, i.e., *why* did we evaluate these things? We could say more about the latter, or simply change the title of this subsection, e.g., to "Evaluation areas."

The SMT Steering Committee appointed a team of evaluators (the authors of this report) to examine areas of interest. The evaluators studied the following topics:

- continuity and turnover in SMT solver participation in past competitions (3.1)
- the performance of all historical and current solvers on the full set of benchmarks, measuring
  - the improvement in performance over time (3.3)
  - repeatability of performance measurements (3.4.1)
  - repeatability of competition results (3.4)
  - competitiveness of solvers (3.5)
- the usefulness of various logics
  - characteristics of existing logics (3.6)
  - which logics are implemented by solvers (3.7)
  - which logics are particularly relevant to application areas (3.9)
- the state of existing benchmarks
  - the range of computational difficulty of the benchmarks (3.8)
  - the degree to which benchmarks discriminate among solvers (3.8)
  - which logics have support in benchmarks (3.8)

2 Evaluation tools

2.1 SMT-LIB benchmarks

One goal of the SMT project since its inception has been collecting benchmarks by which to evaluate SMT solvers and to represent challenge problems in the field. The growth in the number of benchmarks available is shown in the graphics of Fig. 1. *All* of the non-incremental* benchmarks currently present in Star-Exec (95 491* benchmarks) were used for the evaluation. Note that in any previous year, only some of these benchmarks were available.* Furthermore, the competition each year used a random selection of benchmarks (guided by a difficulty distribution). The fact that the set of benchmarks used in competition was different each year muddied any year-to-year comparison of results. By using all of the benchmarks in the evaluation, we intend that any comparative assessments across years or solvers be more accurate. The various kinds of benchmarks and their distribution across logics are discussed below in Section 3.

Aren't all of the SMT benchmarks currently present in Star-Exec non-incremental? I believe we had 95492 benchmarks. (I remember I discovered one benchmark that was in SMT-LIB but missing from Star-Exec, and Aaron or I added this before the evaluation started.)

2.2 Solvers

The SMT competitions required that participating solvers be publicly available Linux applications, and that they be available for any future experimenter to use on new experiments. Thus all historical solvers are still available. However, in 2010, the competition adopted the then new benchmark format, SMT-LIB v2 [4]. Thus solvers prior to 2010 do not run on the current benchmark set. The participating solvers are shown in Fig. 2.

SMT-EVAL used all historical solvers since 2010 (32 total), added 9 versions of previous solvers that were updated in 2013, and included 4 additional experimental solvers,[1] for a total of 45 solvers. All solver implementation teams that we could reach were apprised of the upcoming evaluation and given the opportunity to submit new versions of their solvers. Some teams simply submitted the current version of their solver or advised us to download the current public version from the team's website. Thus the solvers are not necessarily tuned to particular application domains or for competition. Any comparisons for particular applications or kinds of benchmark problems should perform an independent analysis.

This is a bit misleading. Our set of benchmarks does not include some benchmarks used for SMT-COMP 2012 that were not incorporated into SMT-LIB (yet). And according to Fig. 1, there were $> 100,000$ benchmarks already in 2011, while we used less than $100,000$. How to explain this difference? Fig. 2: I don't

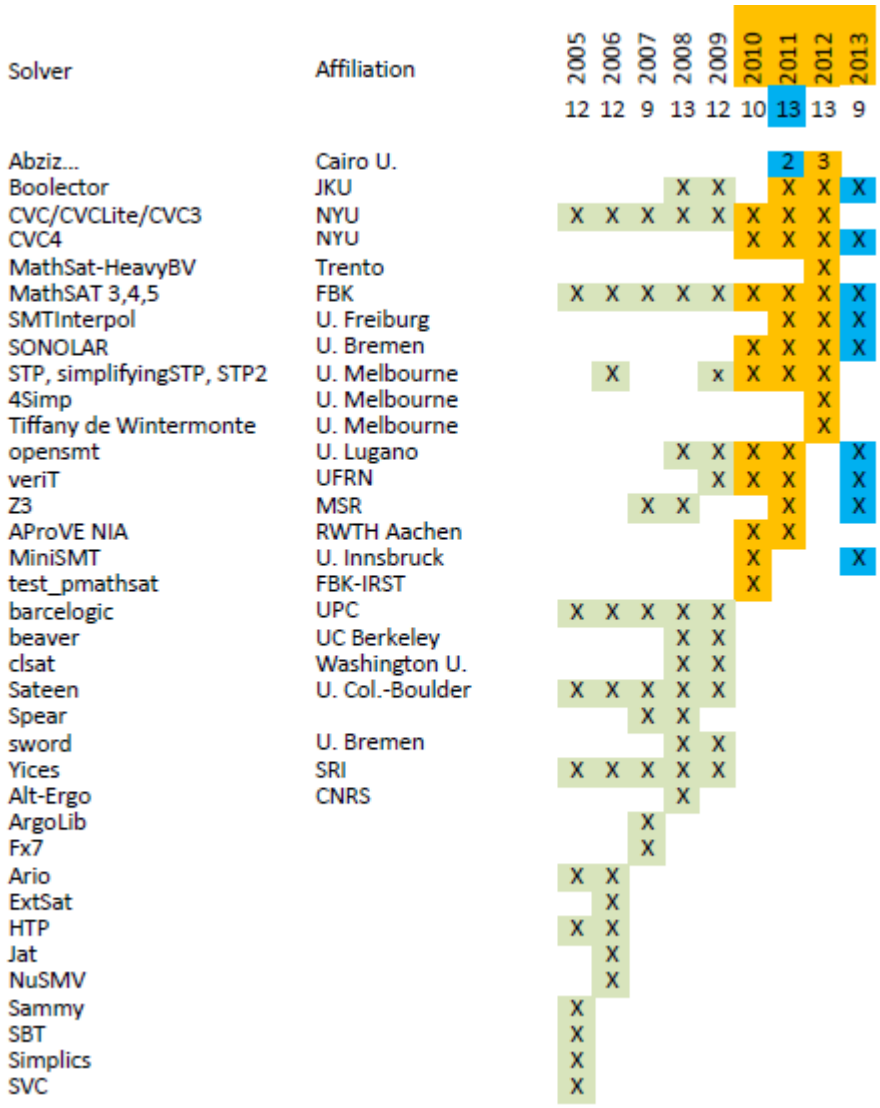
2.3 Star-Exec

SMT-EVAL successfully used the new Star-Exec computational framework [1, 7]. Running SMT-EVAL on Star-Exec did indeed expose a number of bugs and user interface issues; these were corrected in the course of the SMT-EVAL runs. Thus SMT-EVAL served a valuable purpose in preparing Star-Exec for larger scale use.

TBD:: Can Aaron say more about what was learned?? More information about the characteristics of Star-Exec??*

SMT-EVAL's largest computational job was running all 45 of the evaluated solvers on all of the non-incremental benchmarks in the SMT library. The benchmarks belong to different *logics* and solvers are characterized by which logics they support. So for each logic, the evaluation executed the cross product of all benchmarks for that logic and all solvers (in all years) that support problems in that logic,

[1] These were four variations of a portfolio-style solver submitted by Abziz [2, 3].



Solver	Affiliation	2005	2006	2007	2008	2009	2010	2011	2012	2013
		12	12	9	13	12	10	13	13	9
Abziz...	Cairo U.							2	3	
Boolector	JKU				X	X		X	X	X
CVC/CVCLite/CVC3	NYU	X	X	X	X	X	X	X	X	
CVC4	NYU						X	X	X	X
MathSat-HeavyBV	Trento								X	
MathSAT 3,4,5	FBK	X	X	X	X	X	X	X	X	X
SMTInterpol	U. Freiburg							X	X	X
SONOLAR	U. Bremen						X	X	X	X
STP, simplifyingSTP, STP2	U. Melbourne		X			x	X	X	X	
4Simp	U. Melbourne								X	
Tiffany de Wintermonte	U. Melbourne								X	
opensmt	U. Lugano				X	X	X	X		X
veriT	UFRN					X	X	X		X
Z3	MSR			X	X			X		X
AProVE NIA	RWTH Aachen						X	X		
MiniSMT	U. Innsbruck						X			X
test_pmathsat	FBK-IRST						X			
barcelogic	UPC	X	X	X	X	X				
beaver	UC Berkeley				X	X				
clsat	Washington U.				X	X				
Sateen	U. Col.-Boulder	X	X	X	X	X				
Spear				X	X					
sword	U. Bremen				X	X				
Yices	SRI	X	X	X	X	X				
Alt-Ergo	CNRS				X					
ArgoLib				X						
Fx7				X						
Ario		X	X							
ExtSat			X							
HTP		X	X							
Jat			X							
NuSMV			X							
Sammy		X								
SBT		X								
Simplics		X								
SVC		X								

Figure 2: Solvers used in each year of SMT-COMP and SMT-EVAL. Solvers prior to 2010 (with green backgrounds) do not support SMT-LIB v2 and were not used for SMT-EVAL; boxes with blue backgrounds identify solvers that are new for the evaluation; all those since 2010 (orange or blue backgrounds) were used for the evaluation.

year	2005	2006	2007	2008	2009	2010	2011	2012	2013
# of participants	11	11	9	13	12	10	11	11	9
# dropping out		4	6	2	3	7	2	4	
# new participants	(11)	4	4	6	2	5	3	4	

Table 1: Turnover in solver team participation

for a total of 1 663 478 solver-benchmark combinations (called *job-pairs* in Star-Exec). This job took several months of wall time to run. We ran it in quarters, using the result of the first quarter to adjust our procedures and debug some of Star-Exec, before running the rest of the solver-benchmark job pairs. Because of a bug currently being corrected, the results of about 600 job-pairs were not present in the accumulated results. These were identified and rerun as an additional "mop-up" job.

3 Evaluation results

The evaluation team's observations on the questions it considered are presented in the following subsections; our overall conclusions are listed in Section 4. The raw data used as the basis for our observations, collected from Star-Exec, are all archived on the SMT-COMP website, at `http://sourceforge.net/p/smtcomp/code/HEAD/tree/trunk/smtcomp-web/2013/data` as 7z-compressed files. (The largest is 25 MB.)

3.1 Solver participation

Why is there no data for 2013?

The historically participating solvers are shown in Fig. 2. There are a number of observations to be made about solver participation.

- As shown in Table 1, the number of solvers participating each year has been fairly constant, ranging from 9-13, with a median value of 11 participants. The data for 2013 is an anomaly in two respects. The count of 9 only includes those solvers that were explicitly submitted new for 2013. Second, Abziz contributed a portfolio solver that makes an automatic choice, based on machine learning, of which among other existing solvers to apply to a benchmark using observed features of the benchmark. Abziz submitted two instances of his portfolio that used solvers from 2011; those instances of his portfolio solver are not counted in Table 1 since they were not available in 2011, though they are included in Fig. 2. In addition, three variations were contributed in 2012; we count just one participation unit in Table 1 for 2012, though all three (and the two 2011 entries) were used in the evaluations described in this paper.
- Though any team dropping out of future years' competitions is to be regretted, such turnover is to be expected: research projects and Ph.D. students move on to other topics. The year of the most drop-outs is 2010, when the benchmark format changed; not all teams could invest the effort to change their front-ends and to accommodate the new semantic features of SMT-LIB v2.
- Accompanying the drop-outs is a roughly equal number of new participants each year; roughly 1/3 of the participants each year are new. One of the goals of an organized competition is to foster new participation, providing an venue in which it is easy to participate and self-evaluate against

the state-of-the-art. Our observation is that this goal is being accomplished, despite the significant effort required to implement a reasonably competitive solver.

- Despite the turnover, some teams have participated regularly throughout the history of the competition: CVC3/4 and MathSat have participated since the beginning; Boolector, OpenSMT, veriT, and STP also have participated in most of the years since they began being involved.*

As have other solvers, e.g., Z3. Either the list of solvers or the selection criterion should be refined.

3.2 Interpreting solver output

Given the large number of solver-benchmark combinations, it was not feasible to manually inspect the output of each job pair. Thus, we needed to determine mechanically whether a solver reported a benchmark as satisfiable, unsatisfiable or unknown (or did not return a result within the time limit).

On StarExec, this is done by providing a *post-processor*, i.e., an executable that operates on the output of each job pair. Unfortunately, two features of StarExec made it difficult to interpret solver output in a way that is fully SMT-LIB v2 compliant. First, StarExec collects the entire output of a job pair in a single file before passing it to the post-processor. Thus, it is not possible to reliable determine which output a solver has generated specifically in response to a benchmark's `check-sat` command. Second, StarExec conflates standard output and standard error, making it impossible to distinguish between regular solver responses and error messages.

To determine the result of each job pair, we used a post-processor that searched for the words `sat`, `unsat` and `unknown` in the output file, and reported the corresponding result if it found exactly one of them; otherwise, it reported no result. This conservative approach may have caused a small number of (otherwise valid) solver responses to be discarded. It was taken to minimise the number of incorrect results reported by the post-processor.

3.3 Progress in solver performance

We can obtain a measure of the overall year-by-year improvement in solver performance by applying each year's solvers to all the benchmarks (within each logic), observing the best performance on each benchmark. That is, the data shows a *virtual best performer*, obtained by an all-knowing oracle choosing, for each benchmark, the solver that performs best on that benchmark.

Fig. 3 shows the fraction of benchmarks completed (y-axis) within a given time (in seconds, on the x-axis); thus points toward the upper left are better (more completed in less time). The results for all logics are shown together. The four curves are the data for the four years from 2010 to 2013. The lowest (blue) curve is that for 2010. There is clearly significant progress made from 2010 to later years: the number of benchmarks completed within a given time is noticeably higher. The curves for 2011 and 2012 (red and green, respectively) are clearly above 2010, but are not significantly different from each other. The solvers for 2013 are uniformly above those for previous years, but not by a large amount.

By this data, the improvements in raw SMT solver performance on the current set of benchmark problems have slowed down, though they may have improved by other measures.

An alternate measure of solver progress is to count, for a given logic, what fraction of the benchmarks have a better time in a given year than in the previous year, by the best solver for each year. That data

Fig. 3: Shouldn't the 2010-2013 column be $\geq$ the maximum of the other columns? I guess I don't understand what that

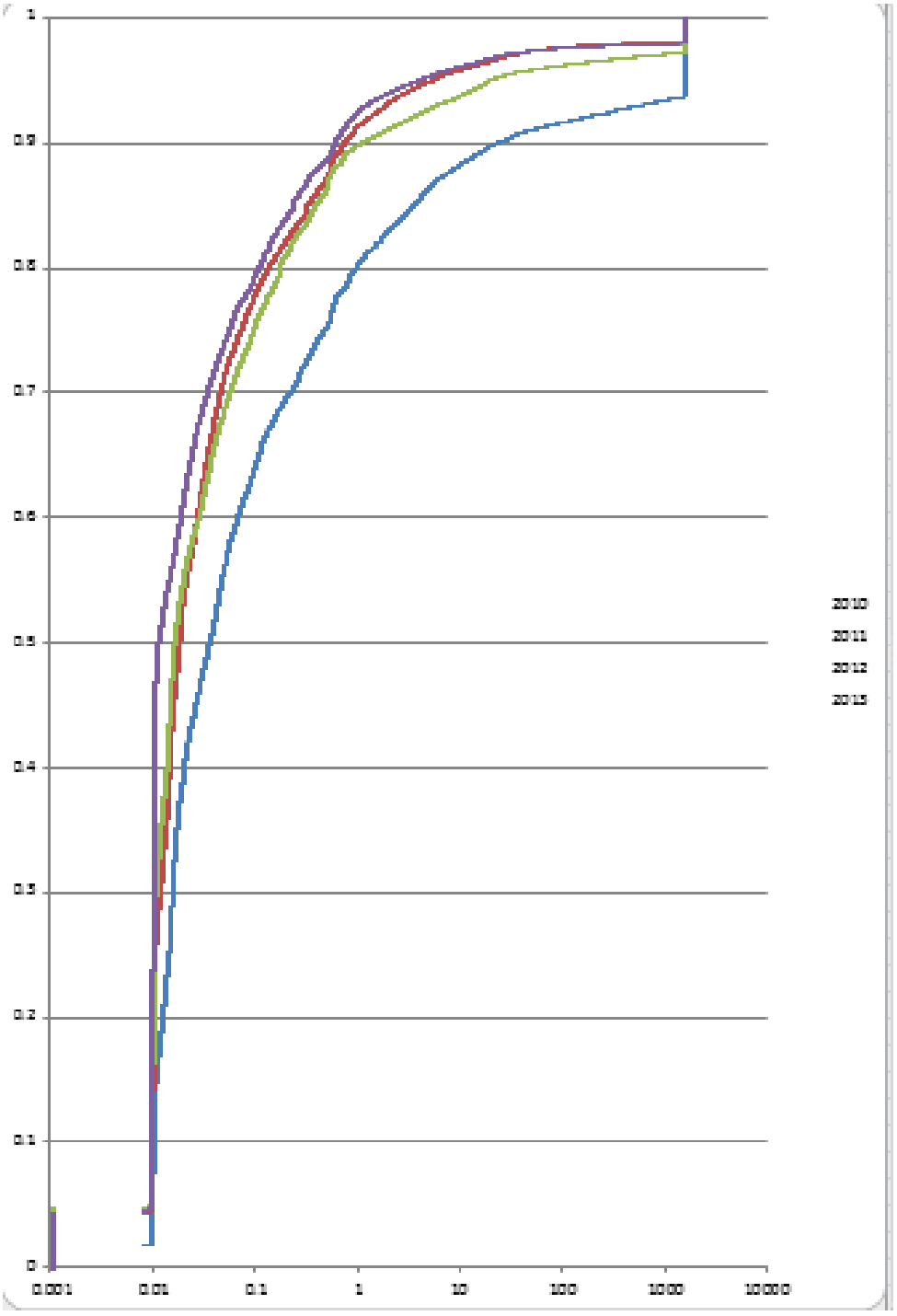


Figure 3: The fraction of benchmarks (y-axis) whose winning time is less than the given number of seconds (x-axis), by year.

Logic	# benchmarks	2010-2011	2011-2012	2012-2013	2010-2013
AUFLIA	6402	90.83%	16.51%	85.16%	88.53%
AUFLIRA	19917	75.66%	64.97%	92.55%	95.37%
AUFNIRA	989	78.67%	61.48%	82.31%	86.65%
LRA	374	80.21%	94.65%	62.57%	91.44%
QF_AUFBV	14335	95.33%	67.39%	50.77%	96.29%
QF_AUFLIA	1140	87.11%	32.19%	91.67%	94.82%
QF_AX	551	88.93%	29.76%	67.70%	84.39%
QF_BV	31747	68.26%	66.87%	33.10%	66.66%
QF_IDL	2170	83.46%	5.71%	85.44%	86.22%
QF_LIA	5882	54.47%	58.98%	91.50%	84.99%
QF_LRA	634	58.83%	40.06%	87.85%	87.22%
QF_NIA	530	43.96%	10.57%	79.43%	45.85%
QF_NRA	166	72.89%	19.28%	97.59%	95.78%
QF_RDL	255	77.25%	4.31%	76.47%	58.43%
QF_UF	6647	85.26%	6.27%	97.23%	93.62%
QF_UFBV	31	100.00%	67.74%	32.26%	100.00%
QF_UFIDL	430	94.88%	1.16%	97.21%	94.19%
QF_UFLIA	564	91.13%	22.16%	92.02%	94.50%
QF_UFLRA	900	51.44%	38.67%	94.33%	88.67%
QF_UFNRA	26	100.00%	15.38%	100.00%	100.00%
UFLRA	5	100.00%	20.00%	20.00%	20.00%
UFNIA	1796	87.58%	11.41%	76.89%	80.85%

Figure 4: The fraction of benchmarks whose best time improved between the given years.

is shown in Fig. 4. Though most data points indicate that most benchmarks improved year to year, the progress is not uniform. The progress from 2010 to 2013 shows nearly all logics having improvement rates above 80%, but there are definitely some laggards.

3.4 Repeatability of competitions

The SMT competition aims to be a repeatable measurement of relative performance of solvers. To that end the details of the competition, such as the rules and selection of benchmarks, are made public. In addition the solvers themselves are required to be made public after the competition.

However, there are aspects of the competition that are not deterministic. We measured two of those in this evaluation:

- *The repeatability of the time a given solver takes on a given benchmark* (Section 3.4.1). This variation is a combination of two factors: any variation in execution and timing by the Star-Exec system itself, and any non-determinism in the execution of the solver.
- *Variation caused by selection of benchmarks* (Section 3.4.2).

There is the additional risk that the set of benchmarks in SMT-LIB is not representative of any particular application area. We do not evaluate that risk here, but expect it to be quite likely to be the case.

3.4.1 Accuracy of performance measurements

To measure the repeatability of a given solver on a given benchmark, we produced a random selection of benchmark-solver pairs (job-pairs), selecting only among job-pairs that did not timeout and had a running time of at least 0.1 seconds. The random selection produced 10 165 job-pairs (out of the total 1 663 478 job-pairs). These job-pairs were run 8 times on 8 different days. One such run took only a few hours. This resulted in 8 repetitions of each of the selected job-pairs. The data is archived with the other evaluation data on the SMT-EVAL website. Star-Exec was lightly loaded on these days, so we did not measure any effect owing to interference with other uses of Star-Exec.

The 8 measured values for a job-pair were sorted and the first, median, and third quartile measured (as the average of sorted values 2 and 3, 4 and 5, and 6 and 7, respectively). We noted the following points:*

Shouldn't we provide the standard deviation?

- Of the measured job-pairs, about 78% had a Q3-Q1 spread less that 10% of the median value. That is, roughly 50% of the time repeated measurements will fall within about 10% of each other.[2]
- 3.5% of job-pairs showed a Q3-Q1 spread more than 50% of the median value.
- Nine cases of the 10165 had a Q3 value more than double the median. These clearly showed multi-modal behavior among the eight data points for each benchmark-solver pair. Such behavior might be attributed to non-deterministic choices within the solver's search algorithms.
- We did not expand our study to determine whether the variation in performance was correlated to the choice of solver or to certain benchmarks.

An approximate conclusion from the above is that variation in timing itself plays a relatively small role in any variation in the competition. It presumably also averages out over many benchmarks and is not correlated with particular solvers. It might have a larger effect on benchmarks with short running times (but we did not quantify this.)

Any variation due to non-determinism within a solver we consider part of that solver's design. Thus it is possible that in some competition a non-deterministic solver might do quite well, while in a repeat of the same competition that solver, making other internal random choices, might do poorly.

3.4.2 Accuracy of competitions

The data we collected allows us to simulate various competition organizations. One such competition design would be, for each logic, to run all solvers for the given year on all benchmarks. The result of such an exercise is shown in Figs. 5, 6 and 7 for the set of 9 solvers for the year 2013. By summing all the cpu times for each benchmark-solver combination, we obtain that such a competition would use about 443 days of cpu time; for this computation we consider all timeout and memory exhaustion results to have taken the full timeout value (1500 sec) of time. It is possible that in some cases a solver may exhaust memory prior to the timeout.

In previous competitions, the winner was determined as the solver that correctly solved the most benchmarks within the timeout limit (solvers producing incorrect results are disqualified). Ties among solvers solving the same number of benchmarks are resolved in favor of the solver taking the least amount of time to produce its solutions. In the tables showing the results, for each logic, the results are reported in the order of a virtual winner.

[2] We did not attempt to measure skew: the degree to which Q3 and Q1 are unequally distant from the median.

# unsolved	secs	solver
AUFLIA 6402 benchmarks , 71.53 days cpu time		
936	1404424.66	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
1278	1957553.10	CVC4-SMT-EVAL-2013
1873	2818137.76	veriT-SMT-EVAL-2013
AUFLIRA 19917 benchmarks , 27.95 days cpu time		
254	389439.43	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
638	976484.46	CVC4-SMT-EVAL-2013
680	1049109.48	veriT-SMT-EVAL-2013
AUFNIRA 989 benchmarks , 0.14 days cpu time		
8	12012.02	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
LRA 374 benchmarks , 3.49 days cpu time		
68	105139.09	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
127	196132.91	CVC4-SMT-EVAL-2013
QF_AUFBV 14335 benchmarks , 64.28 days cpu time		
520	805442.63	Boolector-1.5.118-SMT-EVAL-2013
543	844767.74	SONOLAR-2013-05-15-SMT-EVAL-2013
712	1106772.52	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
864	1332157.54	MathSAT5-5.2.6-SMT-EVAL-2013
948	1465081.78	CVC4-SMT-EVAL-2013
QF_AUFLIA 1140 benchmarks , 3.40 days cpu time		
16	24087.55	veriT-SMT-EVAL-2013
23	34635.67	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
72	113550.39	CVC4-SMT-EVAL-2013
81	121754.16	MathSAT5-5.2.6-SMT-EVAL-2013
QF_AX 551 benchmarks , 1.39 days cpu time		
19	28520.99	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
23	34530.42	MathSAT5-5.2.6-SMT-EVAL-2013
38	57088.31	CVC4-SMT-EVAL-2013
QF_BV 31747 benchmarks , 127.04 days cpu time		
944	1491616.08	Boolector-1.5.118-SMT-EVAL-2013
1102	1750286.98	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
1469	2471860.33	MathSAT5-5.2.6-SMT-EVAL-2013
1590	2498512.90	SONOLAR-2013-05-15-SMT-EVAL-2013
1699	2763975.38	CVC4-SMT-EVAL-2013
QF_IDL 2170 benchmarks , 27.81 days cpu time		
231	407530.76	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
360	689225.35	CVC4-SMT-EVAL-2013
751	1305966.10	veriT-SMT-EVAL-2013

Figure 5: (Part 1) Results of a virtual competition using the 2013 solvers on all benchmarks.

# unsolved	secs	solver
QF_LIA 5882 benchmarks , 86.65 days cpu time		
172	340070.98	MathSAT5-5.2.6-SMT-EVAL-2013
316	614640.90	SMTInterpol-2.0r8402-SMT-EVAL-2013
366	646587.14	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
761	1482588.14	CVC4-SMT-EVAL-2013
2910	4403063.03	veriT-SMT-EVAL-2013
QF_LRA 634 benchmarks , 2.96 days cpu time		
7	15398.96	CVC4-SMT-EVAL-2013
19	33604.65	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
23	43613.79	MathSAT5-5.2.6-SMT-EVAL-2013
36	80377.24	veriT-SMT-EVAL-2013
37	82658.86	SMTInterpol-2.0r8402-SMT-EVAL-2013
QF_NIA 530 benchmarks , 3.26 days cpu time		
17	28080.56	MiniSMT-0.5-SMT-EVAL-2013
160	253992.63	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
QF_NRA 166 benchmarks , 1.64 days cpu time		
0	1.64	veriT-SMT-EVAL-2013
12	22751.00	MiniSMT-0.5-SMT-EVAL-2013
79	118521.63	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
QF_RDL 255 benchmarks , 2.57 days cpu time		
39	64875.77	CVC4-SMT-EVAL-2013
39	65459.76	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
51	91419.17	veriT-SMT-EVAL-2013
QF_UF 6647 benchmarks , 12.56 days cpu time		
30	46692.19	veriT-SMT-EVAL-2013
62	106340.93	CVC4-SMT-EVAL-2013
105	172032.14	MathSAT5-5.2.6-SMT-EVAL-2013
111	171953.63	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
136	237594.15	SMTInterpol-2.0r8402-SMT-EVAL-2013
183	350598.34	OpenSMT-SMT-EVAL-2013
QF_UFBV 31 benchmarks , 0.79 days cpu time		
0	0.32	Boolector-1.5.118-SMT-EVAL-2013
0	0.33	SONOLAR-2013-05-15-SMT-EVAL-2013
7	15962.85	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
12	23942.91	MathSAT5-5.2.6-SMT-EVAL-2013
18	28678.86	CVC4-SMT-EVAL-2013
QF_UFIDL 430 benchmarks , 1.90 days cpu time		
7	11947.79	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
32	71695.10	CVC4-SMT-EVAL-2013
46	80562.09	veriT-SMT-EVAL-2013

Figure 6: (Part 2) Results of a virtual competition using the 2013 solvers on all benchmarks.

# unsolved	secs	solver
QF_UFLIA 564 benchmarks , 2.03 days cpu time		
0	100.67	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
0	355.30	MathSAT5-5.2.6-SMT-EVAL-2013
0	1090.59	CVC4-SMT-EVAL-2013
0	1469.57	SMTInterpol-2.0r8402-SMT-EVAL-2013
68	172440.25	veriT-SMT-EVAL-2013
QF_UFLRA 900 benchmarks , 0.02 days cpu time		
0	40.09	MathSAT5-5.2.6-SMT-EVAL-2013
0	40.86	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
0	57.85	CVC4-SMT-EVAL-2013
0	162.95	veriT-SMT-EVAL-2013
0	1246.65	SMTInterpol-2.0r8402-SMT-EVAL-2013
QF_UFNRA 26 benchmarks , 0.12 days cpu time		
0	0.25	veriT-SMT-EVAL-2013
4	10565.17	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
UFLRA 5 benchmarks , 0.11 days cpu time		
0	39.29	CVC4-SMT-EVAL-2013
2	3103.69	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
4	6300.71	veriT-SMT-EVAL-2013
UFNIA 1796 benchmarks , 1.55 days cpu time		
68	133752.97	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013

Figure 7: (Part 3) Results of a virtual competition using the 2013 solvers on all benchmarks.

However, past competitions have not used all benchmarks, but rather a random subset of the benchmarks. It is worth asking how susceptible the competition results are to the particular subset of benchmarks used for the competition. Typically, the choice is not completely random; rather, the distribution is constrained to have roughly equal representation of various categories of difficulty. For our evaluation here, we will determine the results of a virtual competition by simply randomly selecting different equal-sized subsets (ignoring difficulty measures), determining the competition results for each, and observing whether the competition results vary significantly depending on the subset. Fig. 8 shows the result of an experiment in which a virtual competition was executed on 1000 random subsets; we tallied the fraction of times that the winner changed and that the complete order of the finishers changed.* (We limited our consideration to those logics with at least 100 benchmarks and with more than one solver.)

I don't understand what "changed" means here. (Change is always relative to some existing state.) I think a better measure for Fig. 8 would be the entropy of the winner/order.

The results are quite instructive. In many logics, the competition winners change in only a small fraction of trials. However, in several logics, the fraction of trials in which the winner changes is quite significant, in some cases over 60%. Inspection of the data reveals two contributing causes.

- First, some logics have a relatively large number of benchmarks that are unsolved by the solver set within the timeout limit. Random selection of benchmarks can readily change the subset of unsolved benchmarks included in the virtual competition or change the balance of benchmarks solved by one solver vs. another. Since "winning" the virtual competition is determined primarily by the number of benchmarks solved, rather than the time, any change in the relative number of solved benchmarks may change the order of winners.

  For example, the logic QF_AUFBV has 14335 benchmarks. Of those, 520 are unsolved by Boolector and 543 are unsolved by SONOLAR (the winner and runner up of the virtual competition). Of these only 346 are unsolved by both, leaving 174 unsolved only by Boolector and 197 unsolved only by SONOLAR. It is thus reasonably probable that, in a selection of 10% of the benchmarks, SONOLAR might have fewer unsolved benchmarks selected than Boolector, or vice versa. [TBD - validate with a calculation?]

- Second, even when unsolved benchmarks are not a contributing factor, some pairs of solvers have total times that are close. Variations in time caused by slightly different choices of benchmarks might cause changes in winning order. The experiment described next explores this phenomenon further.

A second virtual competition can be run using only benchmarks that all competitors solved within the timeout period. This would not make a useful real competition because the result could be easily gamed: a solver could win by thoroughly optimizing performance on a few benchmarks and purposely not solving the remainder. However, in this evaluation, no solver has had the chance to do that; performing this evaluation allows comparing running times alone, without the additional factor of unsolved benchmarks.

Figs. 9, 10, and 11 show the result of such a competition, for the 9 2013 solvers; it uses all benchmarks that all the solvers registered for a given logic solve. In addition we performed the same experiment of 1000 virtual competitions each using 10% of these benchmarks. Fig. 12 shows the variation in winning order across these trials. The variation is even more than that shown in Fig. 8. To obtain some insight into this variation, we tabulate, in columns 2 and 3 of Fig. 9ff, the mean and standard deviation[3] of the total solution time for each of the solvers in each logic. The standard deviations are substantial compared to the differences in mean times between competing solvers; even though the distributions of solving times

[3] This is the population standard devation. To obtain the sample standard deviation scale these values by $\sqrt{0.999}$

logic	benchmarks used	# trials	winner changed	order changed	avg. not solved
AUFLIA	640/6402	1000	0.0%	0.0%	0.213
AUFLIRA	1991/19917	1000	0.0%	30.9%	0.026
LRA	37/374	1000	4.6%	4.6%	0.261
QF_AUFBV	1433/14335	1000	33.6%	46.7%	0.050
QF_AUFLIA	114/1140	1000	24.1%	58.4%	0.042
QF_AX	55/551	1000	33.6%	48.0%	0.048
QF_BV	3174/31747	1000	2.4%	26.7%	0.043
QF_IDL	217/2170	1000	0.0%	0.0%	0.206
QF_LIA	588/5882	1000	0.3%	24.4%	0.154
QF_LRA	63/634	1000	13.4%	67.7%	0.038
QF_NIA	53/530	1000	0.0%	0.0%	0.167
QF_NRA	16/166	1000	0.0%	0.0%	0.183
QF_RDL	25/255	1000	67.8%	70.8%	0.169
QF_UF	664/6647	1000	8.0%	71.1%	0.016
QF_UFIDL	43/430	1000	0.0%	18.2%	0.066
QF_UFLIA	56/564	1000	1.5%	24.6%	0.024
QF_UFLRA	90/900	1000	63.2%	68.8%	0.000

Figure 8: Results of virtual competitions using the 2013 solvers on random subsets of 10% of the benchmarks.

are not necessarily Gaussian, the data indicate a substantial probability that the order would change based solely on the random choice of benchmark subset.

I don't understand the "avg. not solved" column.

3.5 Competitiveness of solvers

To assess competitiveness of the competitions and the set of solvers, we measured four quantities:

- *The ratio of the first to second place times on each benchmark in each logic.* The closer this quantity is to 1.0, the closer the race and the more the runner-up is challenging the leader. The ratio is averaged over all the benchmarks for each logic and year. The results are shown in Figs. 13 and 14: for each logic and year that had more than one competitor, the mean, and 1st, median, and 3rd quartiles of the distribution of runner-up ratios are shown.

  A few logics, LRA and UFLRA, are very uncompetitive - the median case has the winner more than 10 times better than the runner-up. In a few others, such as QF_IDL, QF_NIA, QF_NRA the winner is often less than half the time of the runner-up. But in most cases, the ratio is in the 50%-100% range. That is, the races are not neck-and-neck but the winners have only a modest lead over the next finisher. Note that the particular solver that is the winner varies from benchmark to benchmark.

- *The degree to which the leader for a given benchmark changes from year to year.* Since all solvers are new each year, we placed solvers into families by the research group that produced them (e.g., the CVC family includes the CVC3 and CVC4 series of solvers). We counted it a *turnover* if the family of the winning solver changed from the previous year; we measured the fraction of

total secs	mean for 10%	s.dev. for 10%	solver
AUFLIA 4174 benchmarks , 0.22 days cpu time			
134.84	13.57	1.47	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
8572.19	881.87	725.38	veriT-SMT-EVAL-2013
10524.23	1017.33	644.67	CVC4-SMT-EVAL-2013
AUFLIRA 18699 benchmarks , 0.35 days cpu time			
239.75	23.99	1.14	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
1190.59	113.12	245.28	CVC4-SMT-EVAL-2013
29062.40	2906.81	1592.05	veriT-SMT-EVAL-2013
LRA 209 benchmarks , 0.10 days cpu time			
3104.20	308.70	505.21	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
5619.18	531.80	581.02	CVC4-SMT-EVAL-2013
QF_AUFBV 12796 benchmarks , 1.09 days cpu time			
9132.25	927.34	667.35	Boolector-1.5.118-SMT-EVAL-2013
15551.14	1571.56	789.27	MathSAT5-5.2.6-SMT-EVAL-2013
18228.79	1858.63	915.22	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
19760.57	1997.13	1010.26	SONOLAR-2013-05-15-SMT-EVAL-2013
31608.01	3146.44	1223.43	CVC4-SMT-EVAL-2013
QF_AUFLIA 1015 benchmarks , 0.07 days cpu time			
33.34	3.32	1.52	veriT-SMT-EVAL-2013
93.14	9.25	11.90	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
206.75	20.42	11.43	MathSAT5-5.2.6-SMT-EVAL-2013
5529.68	542.85	517.03	CVC4-SMT-EVAL-2013
QF_AX 491 benchmarks , 0.00 days cpu time			
19.87	1.97	1.48	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
27.64	2.76	1.28	MathSAT5-5.2.6-SMT-EVAL-2013
86.80	8.57	8.89	CVC4-SMT-EVAL-2013
QF_BV 28981 benchmarks , 5.95 days cpu time			
24741.62	2466.39	574.41	Boolector-1.5.118-SMT-EVAL-2013
33090.65	3315.48	924.08	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
58252.81	5780.94	1699.68	SONOLAR-2013-05-15-SMT-EVAL-2013
180688.61	18097.26	2555.59	CVC4-SMT-EVAL-2013
217464.30	21676.36	2996.98	MathSAT5-5.2.6-SMT-EVAL-2013
QF_IDL 1407 benchmarks , 2.45 days cpu time			
10837.79	1075.06	400.21	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
28005.20	2794.68	804.94	CVC4-SMT-EVAL-2013
173260.34	17004.68	2911.07	veriT-SMT-EVAL-2013

Figure 9: (Part 1) A virtual competition run only on benchmarks that all solvers solve.

total secs	mean for 10%	s.dev. for 10%	solver
QF_LIA 2085 benchmarks , 1.18 days cpu time			
5108.99	517.56	288.02	MathSAT5-5.2.6-SMT-EVAL-2013
9989.45	1026.38	847.97	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
23068.05	2333.97	1019.42	CVC4-SMT-EVAL-2013
30542.46	3107.09	1214.87	veriT-SMT-EVAL-2013
33029.34	3264.05	1243.38	SMTInterpol-2.0r8402-SMT-EVAL-2013
QF_LRA 583 benchmarks , 0.60 days cpu time			
1410.69	138.20	57.97	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
2108.31	206.48	99.11	CVC4-SMT-EVAL-2013
3892.58	394.05	395.39	MathSAT5-5.2.6-SMT-EVAL-2013
21918.95	2141.70	1153.96	SMTInterpol-2.0r8402-SMT-EVAL-2013
22838.44	2236.65	1019.30	veriT-SMT-EVAL-2013
QF_NIA 357 benchmarks , 0.19 days cpu time			
2474.66	233.56	284.13	MiniSMT-0.5-SMT-EVAL-2013
13683.23	1333.87	938.66	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
QF_NRA 87 benchmarks , 0.01 days cpu time			
0.84			veriT-SMT-EVAL-2013
21.63			Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
548.45			MiniSMT-0.5-SMT-EVAL-2013
QF_RDL 203 benchmarks , 0.24 days cpu time			
2759.19	255.65	229.84	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
3296.74	319.74	161.95	CVC4-SMT-EVAL-2013
14350.83	1404.44	870.90	veriT-SMT-EVAL-2013
QF_UF 6332 benchmarks , 1.27 days cpu time			
878.33	87.54	10.37	veriT-SMT-EVAL-2013
2740.19	266.55	285.91	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
5374.81	525.67	428.07	MathSAT5-5.2.6-SMT-EVAL-2013
5420.78	551.93	483.45	CVC4-SMT-EVAL-2013
20575.64	2039.15	507.66	SMTInterpol-2.0r8402-SMT-EVAL-2013
74537.24	7344.50	1841.05	OpenSMT-SMT-EVAL-2013
QF_UFBV 12 benchmarks , 0.03 days cpu time			
0.11			Boolector-1.5.118-SMT-EVAL-2013
0.13			SONOLAR-2013-05-15-SMT-EVAL-2013
312.28			Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
786.01			MathSAT5-5.2.6-SMT-EVAL-2013
1625.99			CVC4-SMT-EVAL-2013
QF_UFIDL 382 benchmarks , 0.26 days cpu time			
385.76	40.19	32.75	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
10350.75	1047.52	766.90	veriT-SMT-EVAL-2013
11878.04	1213.77	586.08	CVC4-SMT-EVAL-2013

Figure 10: (Part 2) A virtual competition run only on benchmarks that all solvers solve.

total secs	mean for 10%	s.dev. for 10%	solver
QF_UFLIA 496 benchmarks , 0.83 days cpu time			
66.61	7.36	14.24	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
191.79	19.21	13.67	MathSAT5-5.2.6-SMT-EVAL-2013
628.60	64.03	58.11	CVC4-SMT-EVAL-2013
706.64	71.69	46.38	SMTInterpol-2.0r8402-SMT-EVAL-2013
70440.25	6930.04	1853.95	veriT-SMT-EVAL-2013
QF_UFLRA 900 benchmarks , 0.02 days cpu time			
40.09	3.99	0.27	MathSAT5-5.2.6-SMT-EVAL-2013
40.86	4.08	0.92	Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
57.85	5.75	0.63	CVC4-SMT-EVAL-2013
162.95	16.24	12.40	veriT-SMT-EVAL-2013
1246.65	124.03	8.82	SMTInterpol-2.0r8402-SMT-EVAL-2013
QF_UFNRA 22 benchmarks , 0.05 days cpu time			
0.21			veriT-SMT-EVAL-2013
4565.17			Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
UFLRA 1 benchmarks , 0.00 days cpu time			
0.87			CVC4-SMT-EVAL-2013
39.07			Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013
300.71			veriT-SMT-EVAL-2013

Figure 11: (Part 3) A virtual competition run only on benchmarks that all solvers solve.

logic	benchmarks used	# trials	winner changed	order changed
AUFLIA	417/4174	1000	0.0%	41.1%
AUFLIRA	1869/18699	1000	2.0%	3.2%
LRA	20/209	1000	37.4%	37.4%
QF_AUFBV	1279/12796	1000	31.3%	89.9%
QF_AUFLIA	101/1015	1000	23.3%	32.5%
QF_AX	49/491	1000	12.1%	35.2%
QF_BV	2898/28981	1000	16.3%	34.7%
QF_IDL	140/1407	1000	0.0%	0.0%
QF_LIA	208/2085	1000	35.1%	86.8%
QF_LRA	58/583	1000	25.9%	81.0%
QF_NIA	35/357	1000	12.2%	12.2%
QF_RDL	20/203	1000	19.0%	19.5%
QF_UF	633/6332	1000	0.2%	58.6%
QF_UFIDL	38/382	1000	0.0%	37.6%
QF_UFLIA	49/496	1000	7.9%	32.5%
QF_UFLRA	90/900	1000	60.9%	67.1%

Figure 12: Stability of competition results when run only on benchmarks solved by all solvers.

benchmarks for a given logic and year that saw a turnover. The results are shown in Fig. 15. There are many cases in which there is complete turnover from one year to the next; often this is because the previous solver family no longer participated or a new solver joined the competition and dominated the results. However, overall most divisions see a more than 50% turnover from year to year. We see this as indicative of reasonably robust competition.

- *The distribution of winning solvers across the benchmarks within a logic.* A highly competitive environment would have each competing solver win a roughly equal fraction of the benchmarks; a non-competitive environment would have a single solver winning nearly all of the benchmarks. Our measure of competitiveness is a scaled entropy measure: if f_i is the fraction of benchmarks won by solver i, and there are N competing solvers, then the competitiveness measure is $2^{(-\sum_i f_i \log_2 f_i)}/N$. This quantity is p/N if the wins are equally distributed among p of the N solvers ($f_i = 1/p$ for p of the solvers and 0 for the other $N-p$). The results are shown in Figs. 16 and 17; this data only includes benchmarks that were solved by a winner. The first entropy column gives the value of $2^{(-\sum_i f_i \log_2 f_i)}$, whose value is roughly the number of solvers over which the wins are distributed. The second entropy column is $2^{(-\sum_i f_i \log_2 f_i)}/N$, which scales the first value between $1/N$ and 1.

  There is a trivial case of one solver, with all wins distributed equally over the N=1 solvers and an unscaled 'competitiveness' measure of 1.0; in all the combinations of logic and year, there are 18 such cases. Of the other year-logic combinations, note that in all but 11 (of the 70), despite any dominance by one solver, all competitors won at least one benchmark. The 'competitiveness' metric itself shows that in most cases there are approximately 2 solvers sharing the bulk of the wins, increasing to about 3 in a few cases that have many participants. The case of the most participants – 11 solvers for QF_BV in 2012 – had a distribution among about 3.5 winning solvers. Thus, although nearly all solvers contribute something, performance is dominated by a few in nearly all competitive logics.

- *SOTAC.* As a final measurement in this subcategory, we measured the *state of the art contribution* (SOTAC), as proposed in [8]. This measures the uniqueness of the contribution of each solver. It does not consider the time taken to solve a benchmark, but just whether a solver solves a benchmark (within the timeout period). The contribution of a benchmark to a solver's SOTAC is 0 if the solver does not solver the benchmarks and 1/(the number of solvers that solve that benchmark) if it does. Thus the maximum contribution is obtained when a solver is the only one to solve a benchmark. Table 2 shows the computation for each of the solvers for the year 2013. The first column shows the sum of the SOTAC over all benchmarks in which a solver participated; the second column is the average over all the benchmarks that that solver attempted (including ones that timed out), but not benchmarks in logics in which the solver did not participate; the third column[4] is the average over all benchmarks for which the solver was successful (did not time out).

  The CVC4 and Z3 solvers have a high total SOTAC because they contribute to a broad range of logics. MiniSMT has a high average SOTAC because it does well at just a few logics. VeriT suffers on an overall average, but has a relatively high SOTAC averaged over those benchmarks that it solves.

[4]The third column corresponds to Sutcliffe's definition of SOTAC.

Year	Solver	#benchmarks	Mean	1st quartile	Median	3rd quartile
2011	AUFLIA	6390	0.37	0.04	0.29	0.64
2012	AUFLIA	6392	0.57	0.27	0.66	0.88
2013	AUFLIA	5858	0.52	0.17	0.52	0.87
2011	AUFLIRA	19874	0.69	0.61	0.71	0.83
2012	AUFLIRA	19869	0.73	0.66	0.78	0.88
2013	AUFLIRA	12433	0.75	0.68	0.79	0.91
2011	AUFNIRA	988	0.65	0.60	0.65	0.81
2011	LRA	374	0.26	0.01	0.09	0.31
2012	LRA	374	0.09	0.00	0.01	0.05
2013	LRA	372	0.27	0.00	0.02	0.58
2011	QF_AUFBV	10110	0.61	0.43	0.64	0.85
2012	QF_AUFBV	7360	0.73	0.66	0.81	0.92
2013	QF_AUFBV	6145	0.69	0.54	0.77	0.93
2011	QF_AUFLIA	1127	0.49	0.28	0.49	0.67
2012	QF_AUFLIA	1111	0.54	0.33	0.57	0.81
2013	QF_AUFLIA	771	0.71	0.62	0.80	0.92
2011	QF_AX	500	0.66	0.49	0.73	0.87
2012	QF_AX	528	0.58	0.37	0.63	0.84
2013	QF_AX	545	0.66	0.54	0.73	0.83
2010	QF_BV	25209	0.40	0.12	0.29	0.72
2011	QF_BV	21667	0.73	0.61	0.79	0.93
2012	QF_BV	20189	0.72	0.50	0.88	0.98
2013	QF_BV	20561	0.66	0.48	0.75	0.89
2010	QF_IDL	2142	0.48	0.06	0.45	0.87
2011	QF_IDL	2142	0.37	0.09	0.26	0.57
2012	QF_IDL	2165	0.37	0.03	0.23	0.63
2013	QF_IDL	2125	0.39	0.05	0.32	0.69
2010	QF_LIA	5871	0.65	0.60	0.70	0.77
2011	QF_LIA	5882	0.46	0.22	0.39	0.73
2012	QF_LIA	5875	0.29	0.08	0.16	0.43
2013	QF_LIA	5831	0.47	0.23	0.45	0.73
2010	QF_LRA	616	0.78	0.70	0.81	0.91
2011	QF_LRA	619	0.72	0.58	0.77	0.90
2012	QF_LRA	634	0.66	0.52	0.70	0.84
2013	QF_LRA	613	0.71	0.60	0.77	0.90
2010	QF_NIA	530	0.39	0.05	0.35	0.68
2011	QF_NIA	530	0.34	0.04	0.25	0.59
2013	QF_NIA	529	0.22	0.00	0.01	0.46
2010	QF_NRA	165	0.40	0.12	0.37	0.62
2011	QF_NRA	166	0.07	0.00	0.00	0.02
2013	QF_NRA	146	0.26	0.00	0.02	0.56

Figure 13: (Part 1) Runner-up: Ratio of winning time to runner-up time.

Year	Solver	#benchmarks	Mean	1st quartile	Median	3rd quartile
2010	QF_RDL	253	0.44	0.09	0.43	0.77
2011	QF_RDL	253	0.46	0.23	0.37	0.69
2012	QF_RDL	253	0.27	0.01	0.05	0.44
2013	QF_RDL	253	0.58	0.32	0.60	0.88
2010	QF_UF	6630	0.85	0.80	0.89	0.95
2011	QF_UF	6620	0.77	0.69	0.80	0.90
2012	QF_UF	6642	0.62	0.55	0.62	0.70
2013	QF_UF	6630	0.69	0.61	0.71	0.82
2011	QF_UFBV	15	0.85	0.68	0.94	0.98
2012	QF_UFBV	8	0.94	0.91	0.95	0.99
2013	QF_UFBV	18	0.81	0.76	0.88	0.94
2010	QF_UFIDL	428	0.53	0.37	0.55	0.75
2011	QF_UFIDL	426	0.39	0.13	0.34	0.58
2012	QF_UFIDL	428	0.45	0.11	0.43	0.77
2013	QF_UFIDL	430	0.34	0.08	0.28	0.55
2010	QF_UFLIA	563	0.60	0.34	0.70	0.86
2011	QF_UFLIA	564	0.48	0.31	0.47	0.64
2012	QF_UFLIA	562	0.66	0.53	0.72	0.85
2013	QF_UFLIA	560	0.50	0.30	0.48	0.72
2010	QF_UFLRA	900	0.76	0.67	0.77	0.85
2011	QF_UFLRA	900	0.73	0.65	0.75	0.85
2012	QF_UFLRA	900	0.72	0.65	0.74	0.82
2013	QF_UFLRA	900	0.80	0.72	0.82	0.91
2011	QF_UFNRA	26	0.23	0.05	0.08	0.39
2013	QF_UFNRA	26	0.01	0.00	0.00	0.00
2011	UFLRA	5	0.04	0.00	0.00	0.10
2012	UFLRA	5	0.19	0.00	0.00	0.00
2013	UFLRA	5	0.09	0.02	0.02	0.13
2011	UFNIA	1796	0.31	0.01	0.30	0.49

Figure 14: (Part 2) Runner-up: Ratio of winning time to runner-up time.

Logic	2011	2012	2013		Logic	2011	2012	2013
AUFLIA	0.79	0.79	0.87		QF_NIA	0.74	0.35	1.00
AUFLIRA	0.15	0.15	0.97		QF_NRA	0.30	0.00	1.00
AUFNIRA	0.14	0.14	1.00		QF_RDL	0.82	0.91	0.77
LRA	0.55	0.55	0.66		QF_UF	0.80	0.89	0.98
QF_AUFBV	0.97	0.63	0.64		QF_UFBV	1.00	0.42	0.52
QF_AUFLIA	0.75	0.75	0.94		QF_UFIDL	0.86	0.98	0.98
QF_AX	0.57	0.57	0.55		QF_UFLIA	0.90	0.90	0.95
QF_BV	0.94	0.69	0.72		QF_UFLRA	0.79	0.82	0.91
QF_IDL	0.86	0.92	0.87		QF_UFNRA	0.62	0.62	1.00
QF_LIA	0.41	0.40	0.59		UFLRA	0.20	0.20	0.00
QF_LRA	0.71	0.69	0.76		UFNIA	0.78	0.78	1.00

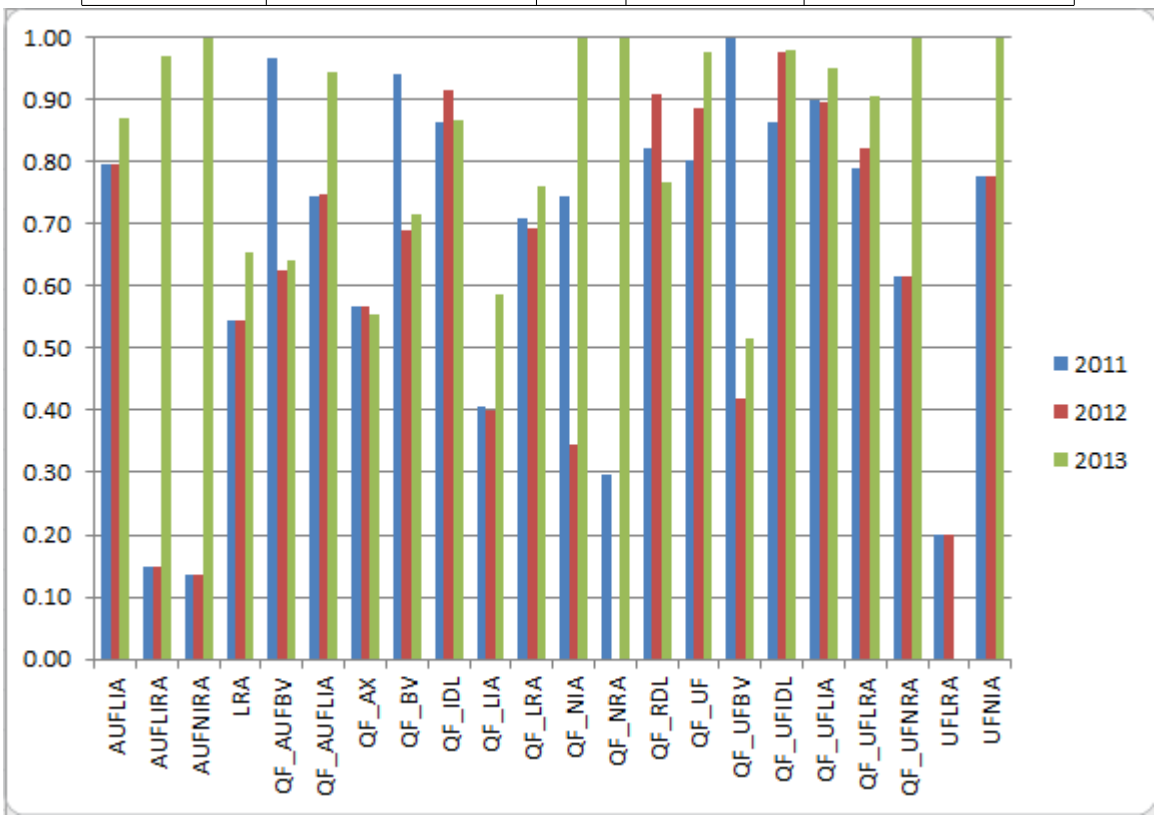


Figure 15: Turnover: Fraction of benchmarks for the given year and logic for which the winning solver is from a different solver family than the prior year.

Year	Logic	# solvers	# winners	winning dist.,	scaled	# benchmarks
2011	AUFLIA	2	2	1.36	0.68	6056
2012	AUFLIA	2	2	1.61	0.81	5685
2013	AUFLIA	3	3	1.69	0.56	5771
2011	AUFLIRA	2	2	1.34	0.67	19791
2012	AUFLIRA	2	2	1.42	0.71	19753
2013	AUFLIRA	3	3	1.73	0.58	19845
2011	AUFNIRA	2	2	1.32	0.66	987
2011	LRA	2	2	1.58	0.79	322
2012	LRA	2	2	1.02	0.51	358
2013	LRA	2	2	1.52	0.76	344
2011	QF_AUFBV	5	5	2.16	0.43	14027
2012	QF_AUFBV	6	6	2.44	0.41	14124
2013	QF_AUFBV	5	5	2.55	0.51	14128
2011	QF_AUFLIA	3	3	1.60	0.53	1126
2012	QF_AUFLIA	3	3	2.01	0.67	1118
2013	QF_AUFLIA	4	4	1.84	0.46	1131
2011	QF_AX	4	4	2.27	0.57	547
2012	QF_AX	2	2	1.56	0.78	541
2013	QF_AX	3	3	1.87	0.62	548
2010	QF_BV	3	3	1.71	0.57	31094
2011	QF_BV	8	8	3.02	0.38	31320
2012	QF_BV	11	11	3.46	0.31	31369
2013	QF_BV	5	5	2.36	0.47	31230
2010	QF_IDL	3	3	1.87	0.62	1756
2011	QF_IDL	4	4	1.36	0.34	1954
2012	QF_IDL	2	2	1.15	0.57	1830
2013	QF_IDL	3	3	1.41	0.47	1956
2010	QF_LIA	3	3	1.45	0.48	5833
2011	QF_LIA	4	4	1.83	0.46	5829
2012	QF_LIA	4	4	1.44	0.36	5789
2013	QF_LIA	5	5	2.38	0.48	5859
2010	QF_LRA	5	5	2.10	0.42	618
2011	QF_LRA	6	6	2.42	0.40	620
2012	QF_LRA	4	4	1.63	0.41	619
2013	QF_LRA	5	5	2.42	0.48	627

Figure 16: (Part 1) Winner distribution: The degree to which winning is distributed among solvers vs. dominated by one solver.

Year	Logic	# solvers	# winners	winning dist.,	scaled	# benchmarks
2010	QF_NIA	3	3	1.64	0.55	530
2011	QF_NIA	3	3	1.78	0.59	530
2013	QF_NIA	2	2	1.58	0.79	526
2010	QF_NRA	2	2	1.52	0.76	166
2011	QF_NRA	2	1	1.00	0.50	166
2013	QF_NRA	3	3	1.17	0.39	166
2010	QF_RDL	3	3	1.61	0.54	220
2011	QF_RDL	4	3	1.32	0.33	222
2012	QF_RDL	2	2	1.04	0.52	213
2013	QF_RDL	3	3	1.80	0.60	218
2010	QF_UF	5	5	2.15	0.43	6643
2011	QF_UF	7	7	2.29	0.33	6647
2012	QF_UF	4	4	1.57	0.39	6643
2013	QF_UF	6	6	1.71	0.28	6647
2011	QF_UFBV	4	2	1.59	0.40	31
2012	QF_UFBV	5	3	2.03	0.41	31
2013	QF_UFBV	5	3	1.75	0.35	31
2010	QF_UFIDL	3	3	1.81	0.60	422
2011	QF_UFIDL	4	4	1.43	0.36	423
2012	QF_UFIDL	2	2	1.41	0.70	404
2013	QF_UFIDL	3	3	1.32	0.44	423
2010	QF_UFLIA	3	3	2.08	0.69	564
2011	QF_UFLIA	4	4	1.41	0.35	564
2012	QF_UFLIA	4	4	1.88	0.47	564
2013	QF_UFLIA	5	4	1.70	0.34	564
2010	QF_UFLRA	3	3	1.38	0.46	900
2011	QF_UFLRA	4	2	1.43	0.36	900
2012	QF_UFLRA	4	4	1.29	0.32	900
2013	QF_UFLRA	5	4	1.95	0.39	900
2011	QF_UFNRA	2	2	1.59	0.79	26
2013	QF_UFNRA	2	1	1.00	0.50	26
2011	UFLRA	2	2	1.41	0.71	5
2012	UFLRA	2	1	1.00	0.50	5
2013	UFLRA	3	1	1.00	0.33	5
2011	UFNIA	2	2	1.41	0.71	1732

Figure 17: (Part 2) Winner distribution: The degree to which winning is distributed among solvers vs. dominated by one solver.

Solver	total SOTAC	mean over all attempted	mean over all solved
Boolector-1.5.118-SMT-EVAL-2013	9352.38	6.39	0.21
CVC4-SMT-EVAL-2013	21562.27	3.55	0.25
MathSAT5-5.2.6-SMT-EVAL-2013	12285.92	4.47	0.21
MiniSMT-0.5-SMT-EVAL-2013	397.00	13.69	0.60
OpenSMT-SMT-EVAL-2013	1083.43	5.92	0.17
SMTInterpol-2.0r8402-SMT-EVAL-2013	2829.63	5.79	0.20
SONOLAR-2013-05-15-SMT-EVAL-2013	9031.47	4.23	0.21
Z3-4.3.2.a054b099c1d6-x64-debian-6.0.6-SMT-EVAL-2013	25664.33	6.09	0.28
veriT-SMT-EVAL-2013	11446.57	1.77	0.30

Table 2: State of the art contribution from each of the 2013 solvers.

3.6 Lattice of logics

Wouldn't it make sense to have this subsection much earlier in Section 3 (since we refer to these logics in most tables etc.))?

SMT-LIB defines a number of theories and logics. The logics are a combination of theories with additional functions or logic symbols or restrictions on the vocabulary of the theories. For example, the `QF_IDL` logic uses the underlying `Ints` theory, but restricts terms to equalities and inequalities between constants and simple differences of variables. The combination of the `Ints` theory and the `Reals` theory adds functions that convert between integers and reals (among other things). Currently, no logic combines bit-vectors with integer or real arithmetic; if one did, it would be useful to provide conversions between numeric values and their corresponding (2's-complement or IEEE floating point) bit-vectors. Note that the `QF_ABV` logic is defined, but is often subsumed in `QF_AUFBV` and has no benchmarks of its own; it is not usually listed separately in other tables in this report.

Leaving aside difference logics as special cases of integer and real logics, there are essentially these mostly orthogonal characteristics underlying the theories and logics:

- IA: integers
- RA: reals
- N: non-linear arithmetic
- A or AX: arrays
- UF: uninterpreted functions
- QF_: quantification
- BV: bit-vectors

This list of characteristics partially corresponds to the set of SMT-LIB theories. The exceptions are quantification and non-linearity, both of which designate lifting restrictions on the kinds of terms allowed in the resulting logic. (Since they do not correspond exactly to underlying SMT theories, the term *characteristics* is used in the discussion below.)

It is possible to define a logic with any powerset of these characteristics in an almost purely combinatorial sense. The only restrictions are

- the interaction between bitvectors and integers or reals described above,
- the interaction between integers and reals described above, and

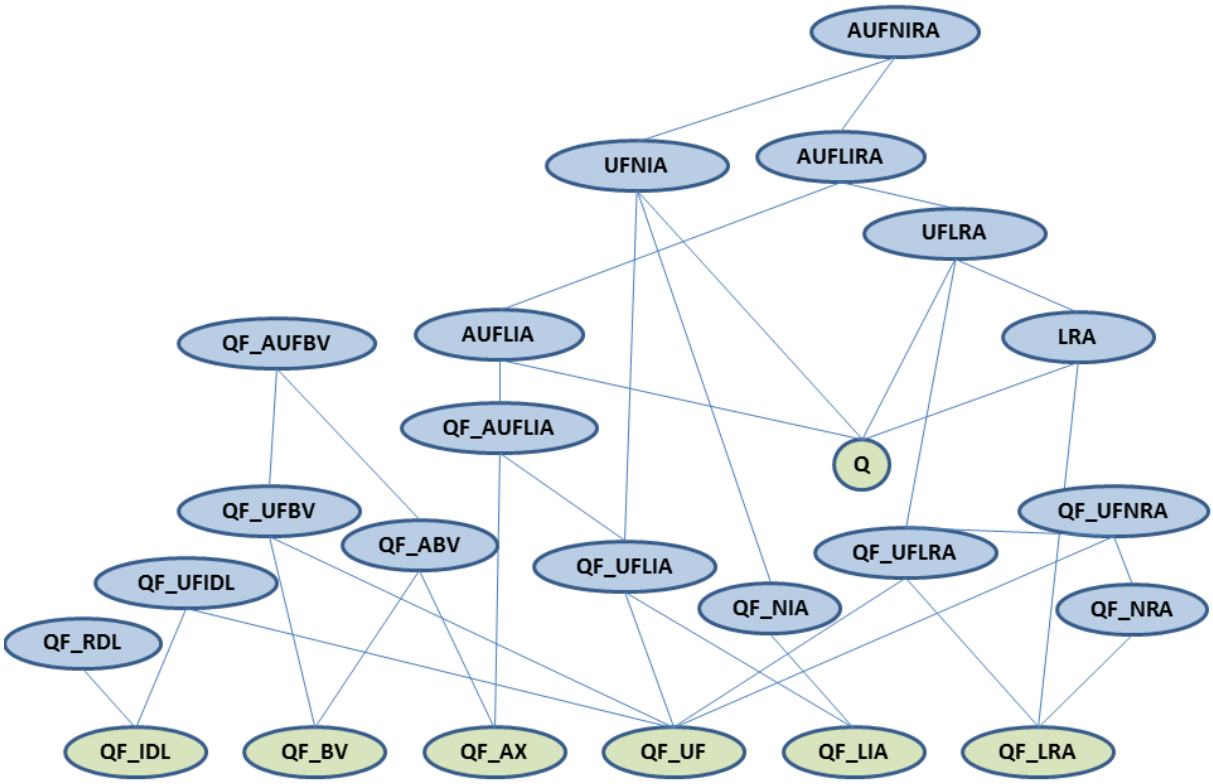


Figure 18: Lattice of SMT logics defined in SMT-LIB. `QF_ABV` is defined, but is often subsumed in `QF_AUFBV`, and is not usually listed separately in other tables in this report. `Q` is not a logic by itself but is a characteristic indicating that quantified expressions are permitted in the logic.

- one cannot have non-linearity without having either integers or reals,

Barring non-linearity without arithmetic and the useless empty set, there are then 111 combinations of characteristics (instead of the full 128). SMT-LIB has definitions and benchmarks for 20 of these logics (excluding QF_RDL, QF_IDL, and QF_UFIDL). There are natural containment relationships among these, as shown in Fig. 18. There are no combinations of bit-vectors with arithmetic.

The naming convention for logics is almost but not quite simply a combination of the letters indicated in the list above, corresponding to the characteristics of the logic, with the letters listed in order by convention: `[QF_][A|AX][UF][BV][N|L][IA|RA|IRA]`. The non-uniformities are these:

- quantification is indicated by *removing* a `QF_` prefix;
- the absence of non-linearity (`N`), i.e., linearity, is indicated by an `L`, instead of by no designator;
- integer and real arithmetic are indicated by two letters (`IA` and `RA`) and their combination by `IRA`;
- the `A` used in the arithmetic designators could be ambiguous with the designator for arrays, except for position;
- a logic with just arrays is named `QF_AX`, whereas in other combinations including arrays, just an `A` instead of `AX` is used.

For example, `UFNIA` includes quantification (no `QF_` prefix), uninterpreted functions (UF), non-linearity (N), and integer arithmetic (IA), leaving out arrays, bit-vectors, and reals.

The ability to mix and match underlying characteristics makes it easy to define a logic with an (almost) arbitrary set of underlying characteristics. It would be useful to standardize the naming convention along the lines of current use, perhaps with slight adjustments (e.g., renaming `QF_AX` to `QF_A` and removing the excess `A` from `IA` and `RA`), in order to define precisely the logic implied by any combination of designators. This would also make it easier to accurately characterize new benchmarks and to summarize the logics supported by a given solver.

The absence of many combinations appears to be simply a matter of ad hoc lack of research or non-existence of benchmarks. For example, all five meaningful combinations of `UF`, `N` and `RA` are present, but only four of the combinations of `UF`, `N` and `IA`. Obviously there is no point to filling out all combinations, just for completeness. However, benchmarks could come in any form, and it would be useful to be able to categorize them precisely. The collection of benchmarks should be driven by technical interest and application need. With the exception of some of the needed conversions (e.g., between bitvectors and numbers), there seems little technical impediment to combining characteristics once the corresponding underlying theories and decision procedures are implemented. That leaves application need as a driver; this is discussed further in section 3.9. It is also conceivable that some logics should be deemphasized, either because solvers have progressed to such an extent that the logic no longer poses interesting technical questions or because applications have little need for the logic.

3.7 Coverage of logics by solvers

Each solver supports some subset of the SMT logics and the corresponding theories and characteristics described in the previous section. The set of logics supported by solvers (as stated by the solver teams) is shown in Fig. 19. CVC and Z3 support all or nearly all logics, with CVC4 just missing full support for non-linear arithmetic (it does have partial support); veriT and MathSat support a significant number of logics.

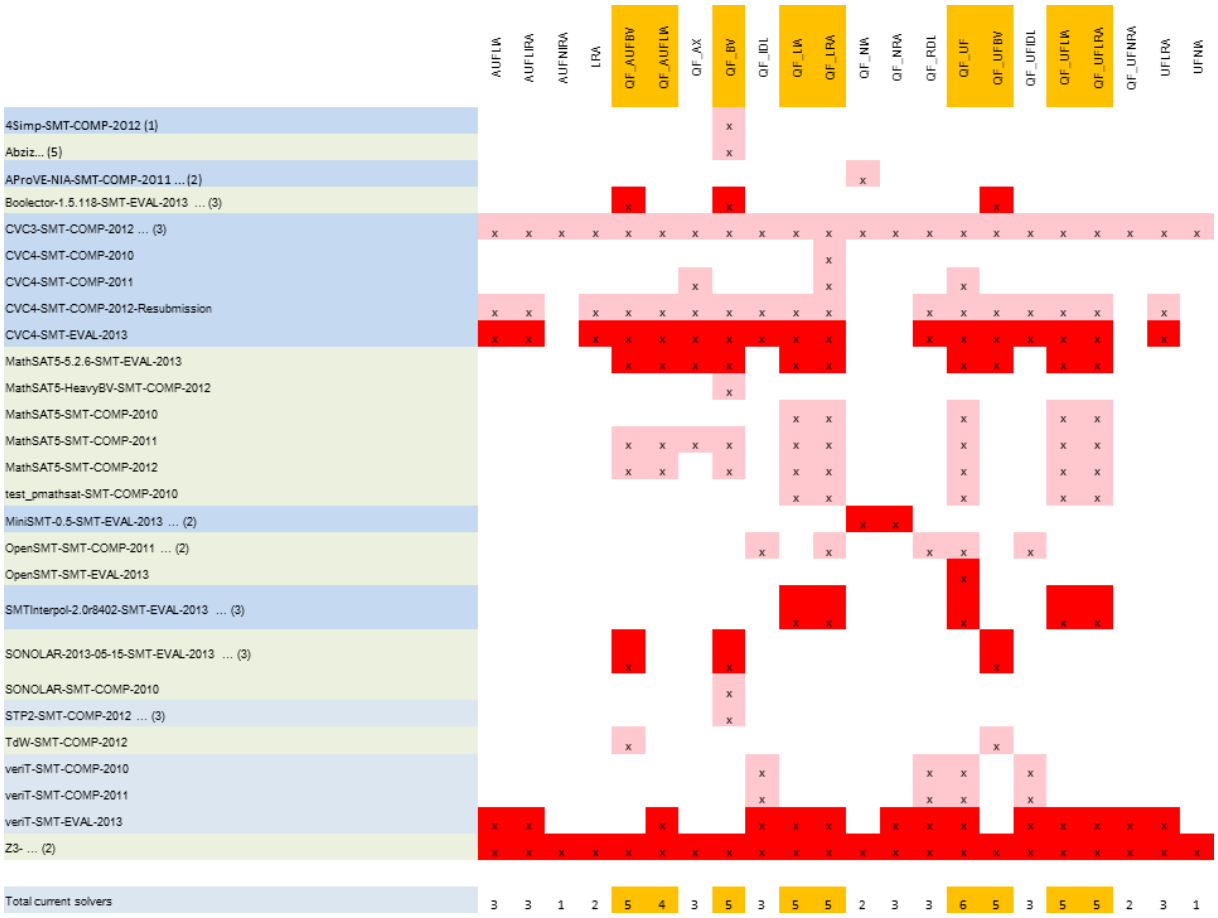


Figure 19: Support for each logic as stated by solver implementors. Darker highlighting identifies the 2013 versions of solvers.

The row of numbers along the bottom of the figure state the number of current (2013) solvers that support a given logic. The logics with support from 4 or more solvers are highlighted. The characteristics most lacking are quantifiers and nonlinearity. Those with the most support are bitvectors, uninterpreted functions and arithmetic.

Note that the support indicated is that stated by the solver supplier. In some cases, a solver supports one logic but is not listed as supporting a subset of that logic. The general reason is that the subset, with restricted characteristics, allows for specific optimizations that are not implemented. The more general solver could act on problems of the more restricted benchmarks but is not deemed competitive and thus was not formally entered into the competition for the restricted logic.

3.8 Benchmarks

The number of benchmarks has successfully grown since SMT-LIB was established; there are now more than 100,000 in all of the logics combined. However, the logics differ significantly in the number of available benchmarks and in their overall difficulty.

Fig. 21 has a column showing the number of benchmarks for each logic. The numbers vary considerably. Some logics, such as UFLRA, have just a very few benchmarks, while others have tens of thousands. SMT-LIB distinguishes four kinds of benchmarks:

- *check* benchmarks are simple tasks that are designed to ensure that a solver has the basic functionality required for a division;
- *industrial* benchmarks are generated from some application; ideally these are substantial examples showing real-world variation, but they be from toy applications running on a toy examples;
- *crafted* benchmarks are hand-crafted to exercise a particular bit of functionality or technical challenge
- *random* benchmarks are randomly generated from some distribution.

[TBD - need data and examples of these] In addition, some benchmarks are *incremental*; that is, they contain more than one `check-sat` command in a command script. This is relevant to interactive applications, but is not the main focus of the competition. In 2012, there were demonstration divisions on generating unsat cores and proof generation. These did not require special benchmarks, but do require different evaluation.

I mentioned this before, but I don't think that we had incremental benchmarks. In fact, I verified (using grep) that every benchmark had exactly one occurrence of `check-sat`.

TBD- more on this aspect of competition design

SMT-COMPs in previous years used benchmark scrambling [**?**]. We did not scramble benchmarks for SMT-EVAL, mainly because support for this feature was not yet available in StarExec. In principle, this means that solvers could have cheated by matching benchmark contents or filenames against a database of known SMT-LIB benchmarks, or even by simply extracting the `:status` information present in most benchmarks. Ruling out cheating with certainty would require careful inspection of solver sources, which are not available for all solvers. However, based on the evaluation data, and taking into account the lack of strong incentives for cheating in SMT-EVAL, we have no reason to believe that cheating has occurred.

Several solvers reported syntax errors on some benchmarks. We validated all benchmarks with two independently developed SMT-LIB v2 parsers [**?**, **?**] to ensure that they are syntactically conforming.

Logic	# benchmarks	2010 all	2011 all	2012 all	2013 all	2010 any	2011 any	2012 any	2013 any
AUFLIA	6402	0.80	0.75	0.73	0.65	0.80	0.95	0.89	0.90
AUFLIRA	19917	0.98	0.97	0.95	0.94	0.98	0.99	0.99	1.00
AUFNIRA	989	0.98	0.98	0.99	0.99	0.98	1.00	0.99	0.99
LRA	374	0.78	0.57	0.73	0.56	0.78	0.86	0.96	0.92
QF_AUFBV	14335	0.85	0.79	0.80	0.89	0.85	0.98	0.99	0.99
QF_AUFLIA	1140	0.93	0.86	0.85	0.89	0.93	0.99	0.98	0.99
QF_AX	551	0.94	0.86	0.90	0.89	0.94	0.99	0.98	0.99
QF_BV	31747	0.87	0.84	0.82	0.91	0.98	0.99	0.99	0.98
QF_IDL	2170	0.44	0.47	0.50	0.65	0.81	0.90	0.84	0.90
QF_LIA	5882	0.80	0.24	0.33	0.35	0.99	0.99	0.98	1.00
QF_LRA	634	0.74	0.73	0.73	0.92	0.97	0.98	0.98	0.99
QF_NIA	530	0.78	0.56	0.98	0.67	1.00	1.00	0.98	0.99
QF_NRA	166	0.93	0.46	1.00	0.52	1.00	1.00	1.00	1.00
QF_RDL	255	0.52	0.51	0.52	0.80	0.86	0.87	0.84	0.85
QF_UF	6647	0.96	0.95	0.96	0.95	1.00	1.00	1.00	1.00
QF_UFBV	31	1.00	0.26	0.42	0.39	1.00	1.00	1.00	1.00
QF_UFIDL	430	0.73	0.73	0.72	0.89	0.98	0.98	0.94	0.98
QF_UFLIA	564	0.91	0.91	0.91	0.88	1.00	1.00	1.00	1.00
QF_UFLRA	900	0.89	0.61	0.61	1.00	1.00	1.00	1.00	1.00
QF_UFNRA	26	1.00	0.96	1.00	0.85	1.00	1.00	1.00	1.00
UFLRA	5	1.00	0.60	0.40	0.20	1.00	1.00	1.00	1.00
UFNIA	1796	0.63	0.74	0.74	0.96	0.63	0.96	0.74	0.96

Figure 20: The fraction of benchmarks completed by the solvers for the given year and logic.

Fig. 20 shows the fraction of benchmarks (for each logic and year) that are solved within the 25 minute timeout period. For each combination of solver and year we report the fraction of benchmarks that are completed by all solvers and the fraction that are completed by at least one solver. In 2013, in about half of the logics, all benchmarks are completed by at least one solver within the timeout period (though not necessarily the same solver); in all but one logic (QF_RDL), at least 95% of the benchmarks are completed by at least one solver. The statistics for fraction of benchmarks completed by all solvers are not as high, since some of the solvers may be initial experimental versions and not tuned for competition. Even so, in more than half of the logics, at least 85% of benchmarks were completed by all solvers.

Fig. 21 is another view of benchmark difficulty. Here, for each logic, the distributions of winning times among the 2013 solvers are shown. In particular, selected percentiles of each distribution are tabulated (the value for the *n*th percentile is the number of seconds for which that fraction of the benchmarks are completed by the winning solver for that benchmark). In all but three logics more than 80% of the benchmarks for that logic take less than just a few seconds, if not less than a second. Only four of the logics have more than 5% of their benchmarks that take more than the timeout period.

These results indicate that there is substantial room for more difficult benchmarks in almost all of the logics.

Logic	benchmarks	median	80th percentile	90th percentile	95th percentile
AUFLIA	6402	0.02	0.07	7.26	1600.00
AUFLIRA	19917	0.01	0.01	0.01	0.02
AUFNIRA	989	0.01	0.01	0.02	0.02
LRA	374	0.02	0.07	2.81	1600.00
QF_AUFBV	14335	0.01	0.02	0.05	0.16
QF_AUFLIA	1140	0.01	0.01	0.02	0.04
QF_AX	551	0.02	0.03	0.05	0.07
QF_BV	31747	0.01	0.15	0.58	1.06
QF_IDL	2170	0.74	28.61	1355.54	1600.00
QF_LIA	5882	0.30	3.91	11.97	19.02
QF_LRA	634	0.09	1.12	8.89	25.38
QF_NIA	530	0.02	0.15	0.69	1.61
QF_NRA	166	0.01	0.01	0.01	0.01
QF_RDL	255	1.76	41.93	1600.00	1600.00
QF_UF	6647	0.03	0.12	0.28	0.48
QF_UFBV	31	0.01	0.01	0.01	0.01
QF_UFIDL	430	0.13	0.67	2.28	15.11
QF_UFLIA	564	0.02	0.03	0.05	0.10
QF_UFLRA	900	0.03	0.04	0.05	0.06
QF_UFNRA	26	0.01	0.01	0.01	0.01
UFLRA	5	3.30	28.19	28.19	28.19
UFNIA	1796	0.22	2.40	59.47	383.20

Figure 21: The distribution of winning benchmark times by logic (for 2013 solvers).

3.9 Application needs

As part of its evaluation, the SMT-EVAL team solicited input on applications that use SMT-LIB. The response was not broad enough to be representative. In addition, the authors have encountered, by happenstance, enough users of SMT-LIB who are not active in the user community to indicate that there is likely a wide variety of uses that are not well-organized or well-represented in benchmarks on specific logics. A few application domains are fairly well-known, including software verification, scheduling optimization, and function and predicate synthesis. Software verification applications in particular will benefit from broader support for combinations of theories and better heuristics for quantification.

In general, a better understanding of the variety of application needs is needed to target future development of SMT solvers and the SMT benchmark library.

4 Conclusions and recommendations

4.1 Observations from the evaluation data

The analyses described point to three principal conclusions and a number of observations.

- First, unsurprisingly, there is still a need for more and better benchmarks, despite the successful growth and current large quantity ($>$ 100,000) of benchmarks in SMT-LIB. Some logics have few benchmarks. In most logics, only a small fraction of benchmarks are significantly challenging (measured by time required to solve them).
- Related to the above, a better sense of the application areas of SMT is needed. Such an analysis will drive solver research in application-oriented directions, provide focus on application-oriented logics, and guide application-relevant benchmark acquisition.
- Finally, we discovered that using a random subset of benchmarks as the basis for the annual competition, together with inherent non-determinism in running solvers, significantly lessens the ability of a competition to determine 'best' solvers at a given point in time. Simply rerunning a competition is quite likely to result in a different ordering of results. The best mitigation is to run as large a benchmark set as possible in a competition (and to work toward application-relevant benchmark sets).

Other observations are these:

- *Participation*. The number of participants is relatively stable (9-13) with an average turnover of 35% (low of 16%, high of 50%) each year. There is also a core of continuing participants (about 1/3 of the total).
- *Progress in solver performance*. Solver performance increased significantly from 2010 to 2013, but most of that improvement occurred from 2010 to 2011. There is improvement across nearly all logics, but some logics are lagging.
- *Repeatability*. The repeatability of benchmarks is reduced by solver nondeterminism. Most importantly, however, competition repeatability is compromised by significant differences in performance of solvers on particular benchmarks, such that different random subsets of benchmarks will

produce different assessments of solvers. The relevance of competitions is also threatened by the relevance of the total benchmark set.

- *Competitiveness of solvers*. Within a logic nearly all solvers contribute something (e.g., a solution that no other solver produces). However, generally the winning times across benchmarks within a logic are obtained by 2-3 solvers that dominate that division. Comparing first-place times to second-place times shows that (a) in many logics the runner-up is quite close to the winner, but (b) in some logics, presumably with state-of-the-art tuning and technical algorithms, the winner far outstrips the other competitors. It is also relevant that most benchmarks change winner from one year to the next. Overall, we judge that there is a moderate degree of competitiveness among solvers - a few tend to dominate in any logic, but other solvers do also make their contribution.
- The naming convention for logics could be more consistent.

4.2 Recommendations for competitions and for SMT-LIB

The experience of past competitions and this evaluation point to a number of ideas for future competitions. Most past policies have worked well and should be continued: openness, transparency, reproducibility, public submission of solvers. Other aspects could use improvement:

- To reduce potential variation caused by choice of benchmarks in a competition, a competition should include as many benchmarks as possible.
- Invigorate the collection of benchmarks and promote a round of curating the existing benchmarks. This might include discontinuing or combining some logics.
- Encourage the participation of new entrants through tool support and recognition. Some ideas are recognizing the best new entrant and promoting reference infrastructure for elements like parsing input files or standard reporting mechanisms.
- Encourage broad participation by measuring performance in all logics, even when some logics are deemed more relevant than others.
- Encourage broadening measures of performance beyond solution of single sat/unsat problems, by including, for example, determining unsat cores, measuring performance on incremental benchmarks, computing interpolants, or solvers that use multi-processing implementations.
- Standardize the naming convention for logics.*
- There is a relative paucity of difficult benchmarks (those that are not solved within the timeout period). Perhaps this is because of progress in solver performance and in hardware speed. In either case, additional difficult benchmarks are needed.

I am hesitant about this. While the current naming convention is obviously not as systematic as it could be, it never caused me (personally) any headache. I am not sure the benefits of adopting a more systematic convention outweigh

4.3 Future work

The most important aspect of future work is to increase and improve benchmark quality, as described above. Thus we would encourage

- a broad survey of SMT application domains and their needs in solver support, both logics and functionality beyond determining satisfiability. This would also drive collection of relevant benchmarks.

There are also a number of questions that we posed but did not have the time or data to answer.

- Are there interesting differences in performance between satisfiable and unsatisfiable benchmarks?
- Do solvers support all of SMT-LIB v2? Should the community encourage expansion to new capabilities (e.g., unsat cores, incremental, interpolants, model generation, proof generation)? Should the community define a core language that is used for the competition, with other aspects being optional?
- Which logics should be retired or deemphasized, based on solver progress or application need?
- Competitions primarily measure ability to solve benchmarks, with the time to do so a secondary criterion. However, in some applications it is important to discharge easy problems quickly. We did not assess this characteristic of the available solvers, but it would be worth doing so.
- What variations in performance measurements occur because of interference from concurrent uses of Star-Exec?

5 TODO

TBD:

- get information on kinds of benchmarks - crafted, application, etc.

- details on Star-Exec cluster; what was learned from initial use of Star-Exec

- text in png figures not very readable

- fix justification in sotac table

- how many other benchmarks

- this question in the outline is not really addressed: the degree to which benchmarks discriminate among solvers

- format problems: - what document class to use (where is this to be submitted); - better legibility of figures - unusual spacing on p.2/3 - fix bad boxes

- fix footer

Acknowledgments

The cost of executing the SMT competition is underwritten by the SMT Workshop. The evaluation used the Star-Exec cluster at the University of Iowa. This computation cluster replaces the SMT-Exec cluster used for all previous competitions. The cluster is supported by the U.S. National Science Foundation under grant CNS-0551697. Cok's contribution is supported by GrammaTech, Inc. and in part by the

National Science Foundation under grant ACI-1314674. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] StarExec web site. `http://www.starexec.org`.

[2] Mohammad A. Abdulaziz. A novel portfolio solver for satisfiability modulo theory problems, 2013.

[3] Mohammad Abdul Aziz, Amr Wassal, and Nevine Darwish. A machine learning technique for hardness estimation of qfbv smt problems (work in progress). In *SMT Workshop 2012 10th International Workshop on Satisfiability Modulo Theories SMT-COMP 2012*, page 56.

[4] Clark Barrett, Aaron Stump, and Cesare Tinelli. The SMT-LIB Standard: Version 2.0. In A. Gupta and D. Kroening, editors, *Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (Edinburgh, UK)*, 2010.

[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] Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. Starexec: a cross-community infrastructure for logic solving. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Proceedings of the 7th International Joint Conference on Automated Reasoning*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2014.

[8] Geoff Sutcliffe. The CADE ATP System Competition Design and Organization. `http://www.cs.miami.edu/~tptp/CASC/24/Design.html#Evaluation`.

[9] Federated Logic Conference (FLoC) Olympic Games. `http://vsl2014.at/olympics/`.