International Journal on Artificial Intelligence Tools

Design and Results of the 3rd Annual Satisfiability Modulo Theories Competition (SMT-COMP 2007)

Clark Barrett
Department of Computer Science, New York University
New York, NY 10012, USA
barrett@cs.nyu.edu

Morgan Deters
Department of Computer Science and Engineering, Washington Univesity in St. Louis
St. Louis, MO 63130, USA
mdeters@cse.wustl.edu

Albert Oliveras*
LSI Department, Technical University of Catalonia
Barcelona, 08034, Spain
oliveras@lsi.upc.edu

Aaron Stump
Department of Computer Science and Engineering, Washington Univesity in St. Louis
St. Louis, MO 63130, USA
stump@cse.wustl.edu

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The Satisfiability Modulo Theories Competition (SMT-COMP) is an annual competition aimed at stimulating the advance of the state-of-the-art techniques and tools developed by the Satisfiability Modulo Theories (SMT) community. As with the first two editions, SMT-COMP 2007 was held as a satellite event of CAV 2007, held July 3-7, 2007. This paper gives an overview of the rules, competition format, benchmarks, participants and results of SMT-COMP 2007.

1. Introduction

Domain-specific procedures or procedures for fragments of certain logics have become an auspicious alternative to traditional generic proof-search methods. Even though most real-world problems cannot be expressed in such a way that they are

*Partially supported by Spanish Min. of Educ. and Science through the LogicTools project (TIN2004-03382)

addressable by a unique domain-specific procedure, they can be decomposed, either manually or automatically, into smaller subproblems for which specific procedures exist. Among these procedures, Satisfiability Modulo Theories (SMT) tools are increasingly being used, e.g. in verification applications[1,4,5,6,7,9], mainly due to both their efficiency and rich input language. Similarly, problems typically solved by ad hoc AI tools, such as scheduling or planning, are currently also addressed by SMT solvers, which have borrowed ideas and techniques from the AI community.

By deciding the satisfiability of a (usually ground) first-order formula modulo a background theory, SMT tools allow one to more naturally encode problems where domain-specific reasoning (e.g., reasoning about numbers, arrays, lists or other data structures) is essential. This is in contrast with SAT solvers, which force one to express facts at a very low level of abstraction, sometimes resulting in loss of important structural information and very large encodings. Similarly, SMT tools also have some advantages with respect to traditional first-order theorem provers: (i) being able to support theories that do not admit a finite first-order axiomatization and (ii) providing efficient decision procedures for *quantifier-free* formulas modulo decidable background theories. Because of these facts, it is increasingly accepted among theorem prover users that SMT tools provide an excellent balance between expressivity and efficiency.

This increase in expressive power greatly complicates the definition of an input language. That makes the evaluation and the comparison of SMT systems a painful task, since translations between formats are rather involved even if one has a precise definition of them, which is not usually the case. In order to avoid the proliferation of independent input formats, the SMT-LIB initiative (see *http://www.smtlib.org*) was created in 2003, establishing a common standard for the specification of benchmarks and of background theories, very much in the flavor of the TPTP library[12]. But it was not until the first annual Satisfiability Modulo Theories Competition (SMT-COMP) in 2005 (Ref. 2) that system implementors started to adopt the SMT-LIB language. As a result, the library has grown from some 1300 benchmarks in 2005, to some 40000 for the 2006 competition[3], and to some 55000 for the 2007 one. Moreover, since 2005, all state-of-the-art SMT systems accept the SMT-LIB language.

Having served the purpose of being the catalyst for the use of a common input language, SMT-COMP is still held annually to achieve its other primary goals: stimulate the advance of SMT techniques, which causes the systems to improve upon last year performances; to become a forum for the exchange of ideas between SMT system implementors, something done in part in a public session held at the SMT workshop (for more information on the workshop see *http://www.lsi.upc.edu/~oliveras/smt07*); and to give publicity to all the research done in the SMT community. For the 2007 edition, two additional concrete goals were achieved, both related to the advance in a concrete type of benchmarks. The first one was to substantially increase the number and quality of bit-vector benchmarks, which have crucial importance for the verification community; the second

concrete goal was to stimulate SMT systems to give some support for quantifiers by making publicly available thousands of quantified industrial verification benchmarks.

With these goals in mind, SMT-COMP 2007 was held July 3-7, 2007, as a satellite event of CAV 2007 in Berlin. The competition was run while CAV 2007 was meeting, in the style of the CADE ATP system competition (CASC)[10,11]. Solvers were run on a cluster of computers at Washington University in St. Louis, where a whole new infrastructure had been created to run the competition and show all sorts of intermediate results on a public screen, thus drawing the attention of CAV attendees. Finally, public results were announced July 7, in a special CAV session, and can be accessed at the SMT-COMP web site (*http://www.smtcomp.org*).

The rest of this paper describes the competition format: rules, problem divisions, and scripts and execution of solvers (Section 2); the benchmarks, with emphasis on the new ones, and their selection for the competition (Section 3); the participants (Section 4) and the final results (Section 5).

2. Competition Format

2.1. *Rules*

This section summarizes the main rules for the competition. For more details, see the full rules on the SMT-COMP web site. Competitors did not need to be physically present at the competition to participate or win. Solvers could be submitted to SMT-COMP 2007 in either source code or binary format. The organizers reserved the right not to accept multiple versions (defined as sharing 50% or more of the source code) of the same solver, and also to submit their own systems. The winners of the 2006 competition were entered to run *hors concours* in the 2007 competition. Special new rules governed the submission of *wrapper tools*, which call a solver not written by the submitter of the wrapper tool. In the end, no wrapper tools were submitted, so these rules were not exercised. Solvers were always called with a single benchmark in SMT-LIB format, version 1.2, presented on their standard input channels. Solvers were expected to report `unsat`, `sat`, or `unknown` to classify the formula. Timeouts and any other behavior were treated as `unknown`.

Each correct answer (within the time limit) was worth 1 point. Incorrect answers were penalized with -8 points. Responses equivalent to `unknown` were awarded 0 points. Four wrong answers in any one division was penalized by disqualification from all divisions of the competition. In the event of a tie for the total number of points in a division, the winner was the tool with the lower CPU time on formulas for which it reported `sat` or `unsat`.

2.2. *Problem Divisions*

The following were the divisions for SMT-COMP 2007. Definitions of the corresponding SMT-LIB logics are available on the SMT-LIB web site. New in 2007 were

two bit-vector divisions: QF_BV and QF_AUFBV. These are described in more detail in the section on benchmarks.

- QF_UF: uninterpreted functions
- QF_RDL: real difference logic
- QF_IDL: integer difference logic
- QF_UFIDL: integer difference logic with uninterpreted functions
- QF_LRA: linear real arithmetic
- QF_LIA: linear integer arithmetic
- QF_UFLIA: linear integer arithmetic with uninterpreted functions
- QF_AUFLIA: linear integer arithmetic with uninterpreted functions and arrays
- QF_BV: Fixed-width bit-vectors (replaces QF_UFBV32 from SMT-COMP 2006)
- QF_AUFBV: Fixed-width bit-vectors with arrays and uninterpreted functions.
- AUFLIA: quantified linear integer arithmetic with uninterpreted functions and arrays
- AUFLIRA: quantified linear mixed integer/real arithmetic with uninterpreted functions and arrays

2.3. *Scripts and Execution*

SMT-COMP ran on a 10-node cluster of identical machines at Washington University in St. Louis each with two 2.4Ghz AMD Opteron 250 processors, 1Mb of cache, and 2Gb of RAM, running GNU/Linux version 2.6.9-55.EL (from CentOS 4.5). One of these machines served as queue manager. The rest were dedicated to executing solvers on SMT-LIB benchmarks; despite the available hardware capabilities of this cluster, each of the execution hosts was configured for single-processor, 32-bit processing to ensure fairness and to match previously published competition specifications. Solvers submitted in source code format were compiled using GCC version 3.4.6.

A *benchmark scrambler* was used to perturb the benchmarks; it obfuscated the name of the benchmark, renamed all predicate and function symbols, removed comments and annotations, and randomly reordered the arguments of associative-commutative operators. The version of the SMT-LIB scrambler used for the competition is available for download on the competition web site.

Sun Grid Engine[a] was used to balance the task load between the nine execution hosts. Each task consisted of all solvers for the division running a single benchmark on a single execution host. This is similar to the approach used in SMT-COMP 2006, and kept the execution hosts from being idle during the competition run.

[a] *http://www.sun.com/software/gridware/*

Each solver's use of resources was monitored by a program called `TreeLimitedRun`, originally developed for the CASC competition. `TreeLimitedRun` was configured to kill the solver if it exceeded 1800 seconds of runtime (30 minutes) or 1.5Gb of memory use. The `ulimit` command was not used to enforce these limits because it does not take into consideration the time and memory consumed by subprocesses. Although the physical amount of memory of each machine is 2Gb, the limit 1.5Gb was used to minimize the number of page faults.

SMT-COMP results were stored in a mysql database.[b] As soon as a solver terminated with a *sat*, *unsat*, or *unknown* answer, or timed out, a record was inserted into this database. The competition web site read directly from this database and thus displayed results as soon as they became available, including newly computed scores. Javascript was employed to poll periodically for new results and highlight them on the results pages during the competition.

3. Benchmarks

As in previous years, one of the main motivations for SMT-COMP 2007 was to collect additional SMT benchmarks. A total of 13263 new benchmarks in 5 divisions were collected, bringing the total number of benchmarks for 2007 to 55397.

3.1. *Organization of Benchmarks*

The benchmarks for the competition were taken from the SMT-LIB library of benchmarks. The benchmarks are organized by division, family, difficulty, category, and status:

- Benchmarks within each division are divided according to *families*. A family is a set of benchmarks that are similar in a significant way and usually come from the same source.
- The *difficulty* of a benchmark is an integer between 0 and 5 inclusive. As in previous years, the difficulty for a particular benchmark was assigned by running as many SMT solvers from the 2006 competition as possible and using the formula:

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

  For new divisions, the difficulty was assigned in a more ad hoc manner using whatever information was available.
- There are four possible categories for a benchmark: check, industrial, random, and crafted. *check* benchmarks are hand-crafted to check complience with basic features of the various divisions. The other categories indicate whether the source of the benchmark is some real application (industrial), hand-crafted (crafted), or randomly generated (random).

[b] *http://www.mysql.com/*

- The status of a benchmark is either *sat*, meaning it is satisfiable, *unsat*, meaning it is unsatisfiable, or *unknown* meaning that its satisfiability is unknown. For those benchmarks for which the status was not included as part of the benchmark, the status was determined by running multiple solvers and checking for agreement. Fortunately, there has never yet been an issue with an incorrect status during a competition, but to be more careful about this, one possible future focus for the competition is to provide *verified* benchmarks: i.e. benchmarks whose status has been determined by a proof-generating SMT solver (e.g. Ref 8) whose proof has been independently checked.

3.2. *New Benchmarks for Existing Divisions*

New verification benchmarks were obtained in both quantified divisions (AUFLIA and AUFLIRA) and in the uninterpreted functions division (QF_UF). In addition, one benchmark was reclassified as being more appropriately in QF_UFIDL than QF_LIA. The lack of new benchmarks in the arithmetic divisions was unfortunate and a focus of SMT-COMP 2008 will be collecting new benchmarks in these divisions. Table 1 lists the number of new benchmarks in each division (if any) as well as the total number of benchmarks in each division.

Table 1. Benchmarks in Existing Divisions

Division	**Benchmark Family**	**Number of Benchmarks**	**Benchmark Category**
AUFLIA	boogie	1254	industrial
AUFLIA	simplify2	2348	industrial
AUFLIA	All 2006 Benchmarks	932	check, industrial, crafted
AUFLIA	Total	4534	
AUFLIRA	why	1325	industrial
AUFLIRA	All 2006 Benchmarks	26511	industrial, crafted
AUFLIRA	Total	27836	
QF_AUFLIA	All 2006 Benchmarks	3729	check, crafted, industrial
QF_IDL	All 2006 Benchmarks	1145	check, industrial, random, crafted
QF_LRA	All 2006 Benchmarks	501	check, industrial
QF_LIA	RTCL	-1	industrial
QF_LIA	All 2006 Benchmarks	204	check, industrial
QF_LIA	Total	203	
QF_RDL	All 2006 Benchmarks	204	check, industrial, crafted
QF_UF	QG-classification	6404	crafted
QF_UF	All 2006 Benchmarks	152	crafted
QF_UF	Total	6556	
QF_UFIDL	RTCL	1	industrial
QF_UFIDL	All 2006 Benchmarks	399	check, industrial
QF_UFIDL	Total	400	
QF_UFLIA	All 2006 Benchmarks	110	check, industrial
All Existing	Total	45218	

3.3. *New Divisions*

Two new benchmark divisions were added for SMT-COMP 2007: QF_AUFBV and QF_BV. This was the result of a major push to bring some challenging and realistic bit-vector benchmarks into the competition. A significant effort went into designing the new QF_BV theory to include a full set of bit-vector operations (including division and modulo operations). The QF_AUFBV division builds on the QF_BV division by adding uninterpreted functions and arrays of bit-vectors. Some of the benchmarks in these divisions come from SMT-COMP 2006's QF_UFBV32 division. This division contained three families: bench_a, crafted, and egt. These were retranslated using the new theories into the families bench_ab, crafted, and egt and placed in the appropriate new divisions. The reason for the name change from bench_a to bench_ab is that the original source (before translation to SMT-LIB format) for these benchmarks included both "a" and "b" sets. The previous bit-vector theory was not expressive enough to accommodate the "b" benchmarks, but the new theories *are* expressive enough, so these have now been included. Table 2 lists the new benchmark families collected for these new divisions together with the number of benchmarks in each family and the category of the benchmark family.

Table 2. New Benchmarks is New Divisions

Division	Benchmark Family	Number of Benchmarks	Benchmark Category
QF_AUFBV	bench_ab	122	industrial
QF_AUFBV	egt	7882	industrial
QF_AUFBV	platania	124	industrial
QF_AUFBV	stp	40	industrial
QF_AUFBV	Total	8168	
QF_BV	bench_ab	288	industrial
QF_BV	crafted	22	crafted
QF_BV	spear	1695	industrial
QF_BV	stp	1	industrial
QF_BV	tacas07	5	industrial
QF_BV	Total	2011	
All New	Total	10179	

3.4. *Selection of Competition Benchmarks*

The benchmark selection algorithm was nearly identical to the one used in 2006, the main differences in the algorithm are: up to 200 benchmarks per division may be selected; and the selection of benchmarks from families tries to maintain a balance of difficulty and status rather than being entirely random. The algorithm is summarized below.

(1) First, each benchmark is categorized as easy-sat, easy-unsat, hard-sat, or hard-unsat as follows: a benchmark is *easy* if it has difficulty 2 or less and *hard* otherwise; a benchmark is *sat* or *unsat* based on its *status* attribute.

(2) All benchmarks in the *check* category are automatically included.
(3) The remaining benchmarks in each division are put into a selection pool as follows: for each family, if the family contains more than 200 benchmarks, then 200 benchmarks are put into the pool. These benchmarks are randomly selected except that a balance of easy-sat, easy-unsat, hard-sat, and hard-unsat is maintained if possible. For families with fewer than 200 benchmarks, all of the benchmarks from the family are put into the pool.
(4) Slots are allocated for 200 benchmarks to be selected from the pool in each division as follows: 85% slots are for industrial benchmarks; 10% are for crafted; and 5% are for random. If there are not enough in one category, then the balance is provided from the other categories.
(5) In order to fill the allocated slots, the pool of benchmarks created in steps 2 and 3 is consulted and partitioned according to category (i.e. industrial, random, crafted). 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. easy-sat, easy-unsat, hard-sat, or hard-unsat). If there are not enough in a sub-category, then its allotment is divided among the other sub-categories.

4. Participants

There were nine entries in SMT-COMP 2007. With respect to SMT-COMP 2006, four new systems were submitted (ArgoLib, Fx7, Spear and Z3) and seven systems participating in 2006 did not enter SMT-COMP 2007 (Ario, CVC, ExtSAT, HTP, JAT, NuSMV and STP). A brief description of each system is given in the following. For more detailed information, including references to papers describing concrete algorithms and techniques, one can access the full system descriptions available at the SMT-COMP 2007 web site. The binaries run during the competition for all solvers are also available there.

ArgoLib v3.5. ArgoLib v3.5 was submitted by Filip Marić and Predrag Janičić from the University of Belgrade, Serbia. ArgoLib v3.5 is a C++ implementation of the DPLL(T) approach, coupling a rational reconstruction of the SAT solver MiniSAT with two rational linear arithmetic solvers, one based on Fourier-Motzkin and another one based on Yices Simplex algorithm. Problem divisions: QF_RDL, QF_LRA.

Barcelogic 1.2. Barcelogic 1.2 was submitted by Miquel Bofill, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell and Albert Rubio from the Technical University of Catalonia, Barcelona. Barcelogic 1.2 is a C++ implementation of the DPLL(T) framework. Problem divisions: QF_UF, QF_IDL, QF_RDL, QF_UFIDL, QF_LRA, QF_LIA and QF_UFLIA.

CVC3 1.2. CVC3 1.2 is a joint project of New York University and the University of Iowa. The project leaders are Clark Barrett (NYU) and Cesare Tinelli (Iowa). Major code contributions have been made by Clark Barrett, Alexander Fuchs (Iowa), Yeting Ge (NYU) and Dejan Jovanovic (NYU). Problem divisions: QF_UF, QF_LRA, QF_LIA, QF_UFLIA, QF_AUFLIA, AUFLIA and AUFLIRA.

Fx7. Fx7 was submitted by Michal Moskal from the University of Wroclaw, Poland, with contributions from Jakub Lopuszański, from the same institution. Fx7 is implemented in the Nemerle language and is designed for software verification queries, which make heavy use of quantifiers. To deal with quantifiers, Fx7 implements two novel matching algorithms. Problem divisions: AUFLIA.

MathSAT 4. MathSAT 4 was submitted by Roberto Bruttomesso, Alessandro Cimatti and Anders Franzén from FBK-IRST, Trento, and Alberto Griggio and Roberto Sebastiani from Università di Trento, Italy. MathSAT 4 is a C++ implementation of the standard "online" lazy integration schema used in many SMT tools. Problem divisions: QF_UF, QF_IDL, QF_RDL, QF_UFIDL, QF_LRA, QF_LIA and QF_UFLIA.

Sateen. Sateen was submitted by Hyondeuk Kim, HoonSang Kin and Fabio Somenzi from the University of Colorado at Boulder. Sateen is a C implementation of the lazy approach to SMT that relies on incremental refinements of a propositional abstraction of the given formula during the enumeration of its solutions. Problem divisions: QF_IDL.

Spear v1.9. Spear v1.9 was submitted by Domagoj Babić from the University of British Columbia. Spear is a theorem prover for bit-vector arithmetic that translates the input formula into a propositional one that is then sent to the core of Spear, a simple lightweight DPLL SAT solver. Problem divisions: QF_BV.

Yices 1.0.10. Yices 1.0.10 was submitted by Bruno Dutertre from SRI International. Yices is a C++ implementation that integrates a modern DPLL-based SAT solver with a core theory solver (handling equalities and uninterpreted functions) and satellite solvers (for arithmetic, arrays, tuples, etc.). Problem divisions: all.

Z3 0.1. Z3 0.1 was submitted by Nikolaj Bjørner and Leonardo de Moura from Microsoft Research. Z3 is a C++ implementation, similar in spirit to Yices, but it also incorporates an E-matching abstract machine to deal with quantifiers and model-based theory combination techniques. Problem divisions: all.

5. Results

The results for each division are summarized in Figures 1 through 26 starting on page 14. More detailed results are available on the SMT-COMP web site,

http://www.smtcomp.org/.

Raw results are reported for each division. Further, each division has two types of associated graphs: a "cactus" graph and a scatter graph. The cactus graph sorts a solver's time on all its correctly-solved benchmarks in the division and plots the solver's cumulative time on the benchmarks. Thus the solver that reaches the furthest right on the graph wins (assuming no wrong answers); for solvers tied by this measure, the lower of all such solvers (least total time) wins the division.

The scatter plot shows a benchmark-by-benchmark comparison between the winner and runner-up in each division. This demonstrates how advanced the winning solver is over its nearest competitor. For divisions that ran last year, a second scatter plot compares last year's winner with this year's winner on this year's competition benchmarks; this demonstrates improvement (or lack thereof) over last year's tools.[c] In the scatter plots, $\triangle$ represents *sat* instances, and $\triangledown$ represents *unsat* instances. For interactive versions of these scatter plots that color-code benchmark families for easy correlation, please view the division results pages at *http://www.smtcomp.org/*.

5.1. *Description of anomalous and surprising results*

In the QF_UFLIA, QF_UFIDL, QF_LRA, QF_LIA, and QF_AUFLIA divisions, the 2006 winner, Yices 1.0, beat the new entries for the 2007 competition. As stated above, 2006 winners ran *hors concours*, and so were not eligible to win officially.

In the bit-vector divisions, a patched version of Z3 was submitted after the submission deadline and included *hors concours*. This version included a fixed version of a third-party arithmetic library that caused the original submission to report an incorrect answer on one benchmark selected for competition.

In QF_BV, an alternate version of Spear v1.9 was submitted after the submission deadline with the same binary file but with different command-line arguments. The second submission was accepted as an *hors concours* participant.

In QF_LIA, CVC3 1.2 reported a wrong answer on one of the scrambled benchmarks selected for competition due to a suspected bug.

In the AUFLIA and AUFLIRA divisions, no *sat* instances were discovered by the solvers (all solvers in the division timed out or reported *unknown* answers for the *sat* instances selected for competition).

5.2. *Description of* unknown *results*

To understand *unknown* results are further broken down:

- QF_RDL: ArgoLib v3.5 reported 43 *unknown* answers. These resulted from a variety of problems in the `skdmxa2` and `scheduling` benchmark families, including segmentation faults, memory exhaustion, and the inability to parse a scrambled benchmark.

[c]There were two tying winners of the SMT-COMP 2006 AUFLIRA division, and therefore two such scatter plots.

- QF_IDL: Sateen gave an *unknown* result (an error message) on the scrambled version of the `sal/lpsat/lpsat-goal-20.smt` benchmark. The authors confirmed that this was due to a corner case bug uncovered by the scrambled benchmark.

  MathSAT 4.0's two *unknown* answers were due to segmentation faults on the benchmarks `queens_bench/toroidal_bench/toroidal_queen97-1.smt` and `queens_bench/toroidal_bench/toroidal_queen100-1.smt`.
- QF_LRA: The submitted revision of CVC3 1.2 had a problem in the logic that dispatches to its QF_LRA solver; this caused CVC3 1.2 to crash on all competition-selected QF_LRA benchmarks (though there were cases out-of-competition that did not crash CVC3 1.2). A patched version of CVC3 1.2, a one-line source change, was included in the results listing *hors concours*. The patched version gave 30 *unknown* answers in the division; these were due to memory exhaustion.
- QF_LIA: CVC3 1.2's 30 *unknown* answers appear to be due to memory exhaustion.

  MathSAT 4.0 aborted (silently) with a `SIGABRT` on benchmark `CIRC/multiplier_prime/MULTIPLIER_PRIME_32.msat.smt`; Barcelogic 1.2 terminated with a segmentation fault on `Averest/parallel_prefix_sum/ParallelPrefixSum_safe_blmc007.smt`.
- QF_AUFLIA: CVC3 1.2's 34 *unknown* answers appear to be due to memory exhaustion.
- QF_BV: The two Spear submissions reported an *unknown* answer on `stp/testcase15.stp.smt` due to memory exhaustion.
- QF_AUFBV:

  The Z3 solver submissions (patched and unpatched) each reported three *unknown* answers due to memory exhaustion.

  Yices reported 13 *unknown* answers, all due to memory exhaustion.
- AUFLIA: Z3 0.1 and Fx7 reported unknowns but did not appear to crash. CVC3 1.2 ran out of memory on 11 benchmarks, reported unknown (without crashing) on 8, gave no output at all on two, and errored silently on three. Yices 1.0.10 ran out of memory on 12 and reported an unknown result (without crashing) on 4. Yices 1.0 reported unknown (without crashing) on 64 and crashed on one.
- AUFLIRA: CVC3 1.2 reported unknown (without crashing) on 7. Z3 0.1 reported unknown (without crashing) on three and ran out of memory on four. Yices 1.0.10 reported unknown (without crashing) on three and ran out of memory on one. Yices 1.0 gave no output on one benchmark, crashed on two, and reported unknown (without crashing) on 7. CVC3 (from SMT-COMP 2006) crashed on one, gave no output on 10, reported unknown (without crashing) on 6, and got a terminal floating-point exception (SIGFPE) on one.

It is important to note in the above analyses that the solver binaries were treated as black boxes; we made no attempt to determine if a solver internally caught errors (such as segfaults or C++ `std::bad_alloc` exceptions) and dutifully reported "unknown" instead of (observably) crashing.

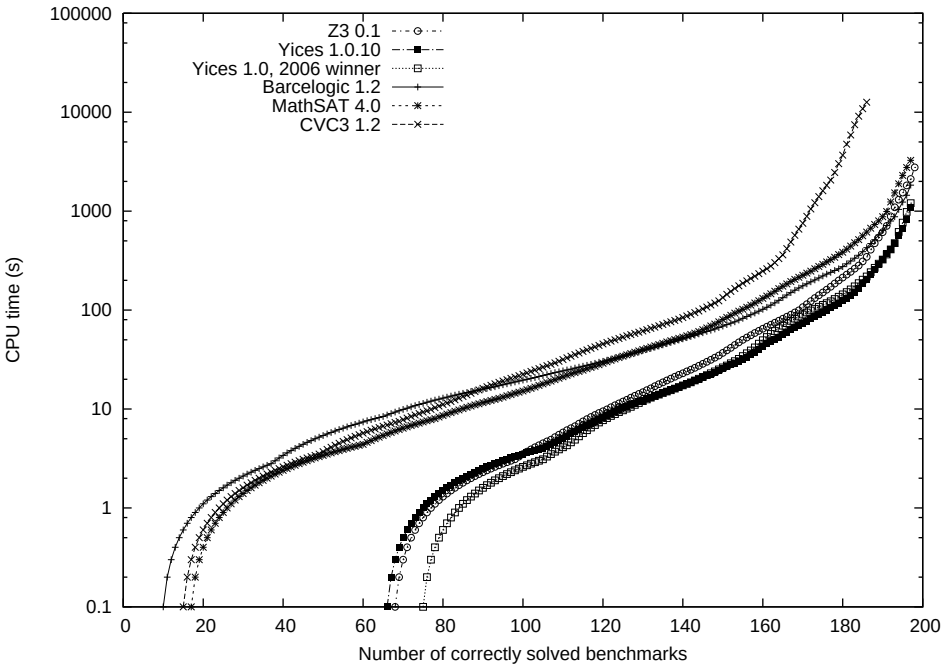
6. Acknowledgements

SMT-COMP would not have been possible without the invaluable support, feedback, and participation of the entire SMT community, with special thanks to Cesare Tinelli and Silvio Ranise, the leaders of the SMT-LIB initiative. The authors at Washington University would like to acknowledge the technical support of Mark Bober of Washington University's Computing Technology Services in ordering and helping operate the cluster. Thanks also to the organizers of CAV 2007 for their support of SMT-COMP 2007 as a satellite event. Finally, the organizers wish to acknowledge the support of the U.S. National Science Foundation, under contract CNS-0551697, for SMT-COMP 2007 and (anticipated) 2008.

References

1. M. Barnett, B.-Y. Chang, R. DeLine, B. Jacobs, and K. Leino. Boogie: A Modular Reusable Verifier for Object-Oriented Programs. In F. de Boer, M. Bonsangue, S. Graf, and W.-P. de Roever, editors, *Fourth International Symposium on Formal Methods for Components and Objects (FMCO'05), Post-Proceedings*, 2006.
2. Clark Barrett, Leonardo de Moura, and Aaron Stump. Design and results of the first satisfiability modulo theories competition (SMT-COMP 2005). *Journal of Automated Reasoning*, 35(4):373–390, November 2005.
3. Clark Barrett, Leonardo de Moura, and Aaron Stump. Design and results of the second satisfiability modulo theories competition (SMT-COMP 2006). *Formal Methods in System Design*, 2007.
4. Clark Barrett, Yi Fang, Ben Goldberg, Ying Hu, Amir Pnueli, and Lenore Zuck. TVOC: A translation validator for optimizing compilers. In Kousha Etessami and Sriram K. Rajamani, editors, *Proceedings of the* 17^{th} *International Conference on Computer Aided Verification (CAV '05)*, volume 3576 of *Lecture Notes in Computer Science*, pages 291–295. Springer-Verlag, July 2005. Edinburgh, Scotland.
5. Satyaki Das and David L. Dill. Counter-example based predicate discovery in predicate abstraction. In M. Aagaard and J. O'Leary, editors, *4th International Conference on Formal Methods in Computer-Aided Design*. Springer-Verlag, 2002.
6. S. Lahiri, R. Nieuwenhuis, and A. Oliveras. SMT Techniques for Fast Predicate Abstraction. In *18th International Conference on Computer-Aided Verification*, pages 424–437. Springer-Verlag, 2006.
7. S. Lerner, T. Millstein, and C. Chambers. Automatically Proving the Correctness of Compiler Optimizations. In R. Gupta, editor, *In ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2003. received best paper award.
8. Sean McLaughlin, Clark Barrett, and Yeting Ge. Cooperating theorem provers: A case study combining HOL-Light and CVC Lite. In Alessandro Armando and Alessandro Cimatti, editors, *Proceedings of the* 3^{rd} *Workshop on Pragmatics of Decision Procedures in Automated Reasoning (PDPAR '05)*, volume 144(2) of *Electronic Notes in Theoretical Computer Science*, pages 43–51. Elsevier, January 2006. Edinburgh, Scotland.

9. S. McPeak and G. Necula. Data Structure Specifications via Local Equality Axioms. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer-Aided Verification*, pages 476–490. Springer-Verlag, 2005.
10. F.J. Pelletier, G. Sutcliffe, and C.B. Suttner. The Development of CASC. *AI Communications*, 15(2-3):79–90, 2002.
11. G. Sutcliffe and C. Suttner. The State of CASC. *AI Communications*, 19(1):35–48, 2006.
12. G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3 0.1	198	2771.3	98	100	0	2	0
Yices 1.0.10	197	1083.3	97	100	0	3	0
Barcelogic 1.2	197	1833.1	97	100	0	3	0
MathSAT 4.0	197	3258.8	97	100	0	3	0
CVC3 1.2	186	12604.2	87	99	0	14	0
Yices 1.0, *2006 winner*	197	1204.5	97	100	0	3	0

Fig. 1. Results in the QF_UF division.

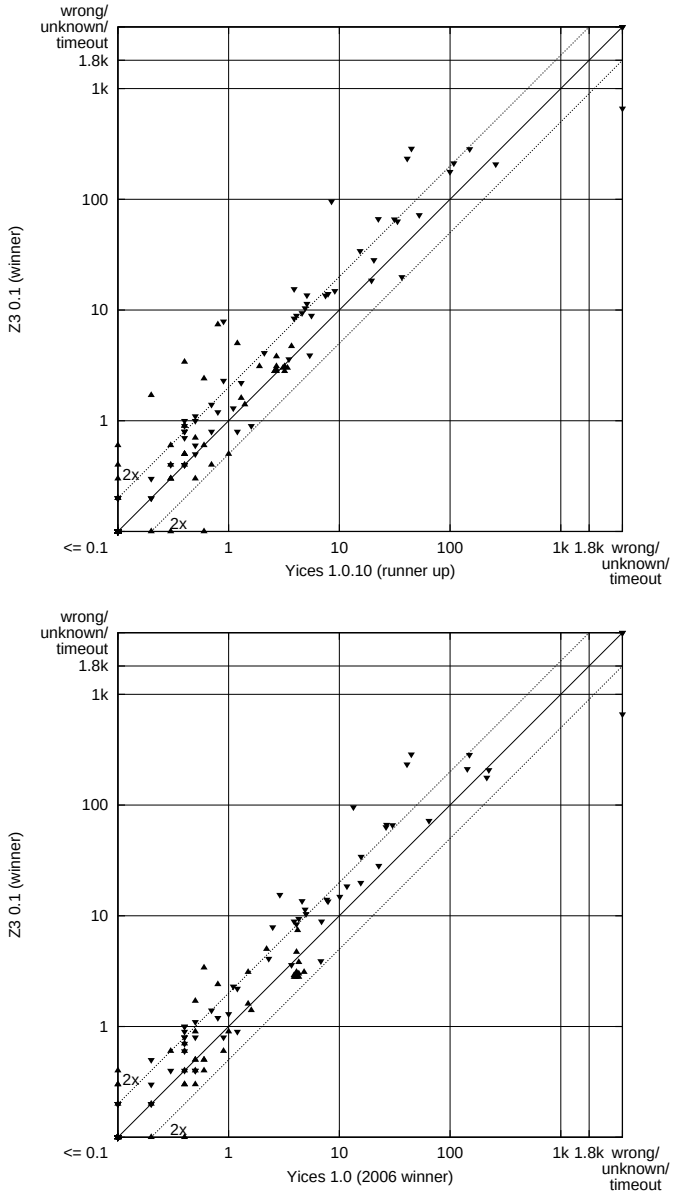
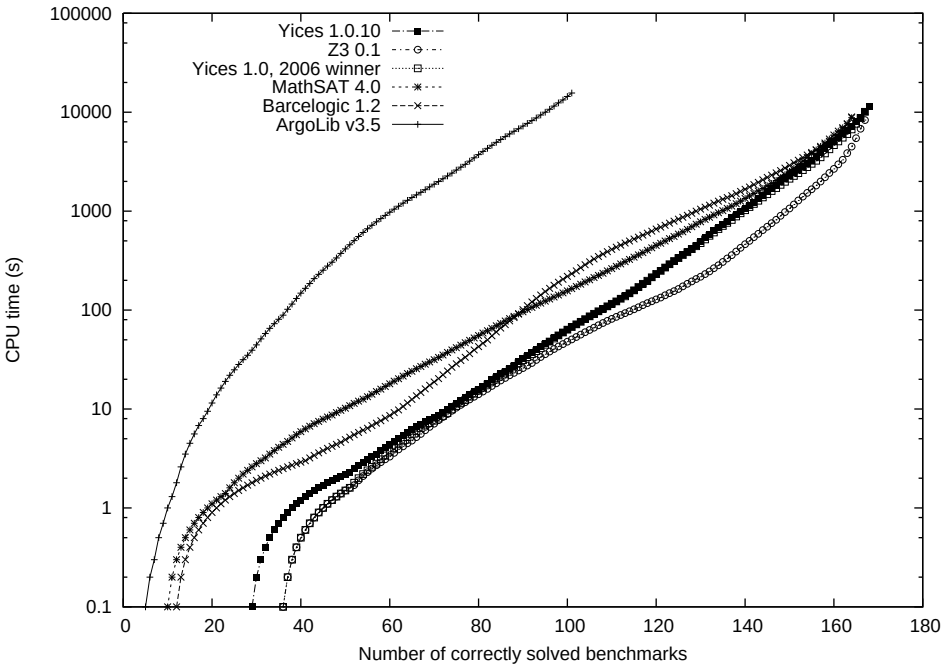


Fig. 2. Benchmark comparisons of (above) the top two contenders in the QF_UF division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Yices 1.0.10	168	11464.6	112	56	0	1	0
Z3 0.1	167	8314.5	111	56	0	2	0
MathSAT 4.0	164	8813.5	110	54	0	5	0
Barcelogic 1.2	164	8981.5	111	53	0	5	0
ArgoLib v3.5	101	15639.4	68	33	43	25	0
Yices 1.0, *2006 winner*	167	10086.8	111	56	0	2	0

Fig. 3. Results in the QF_RDL division.

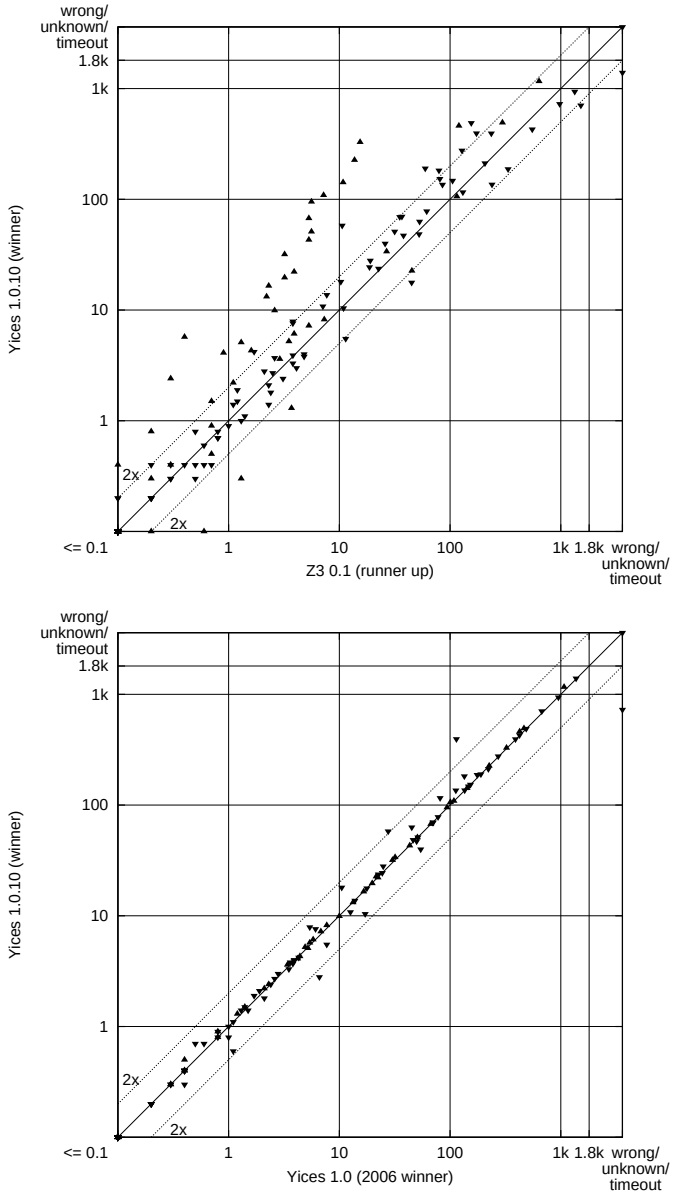
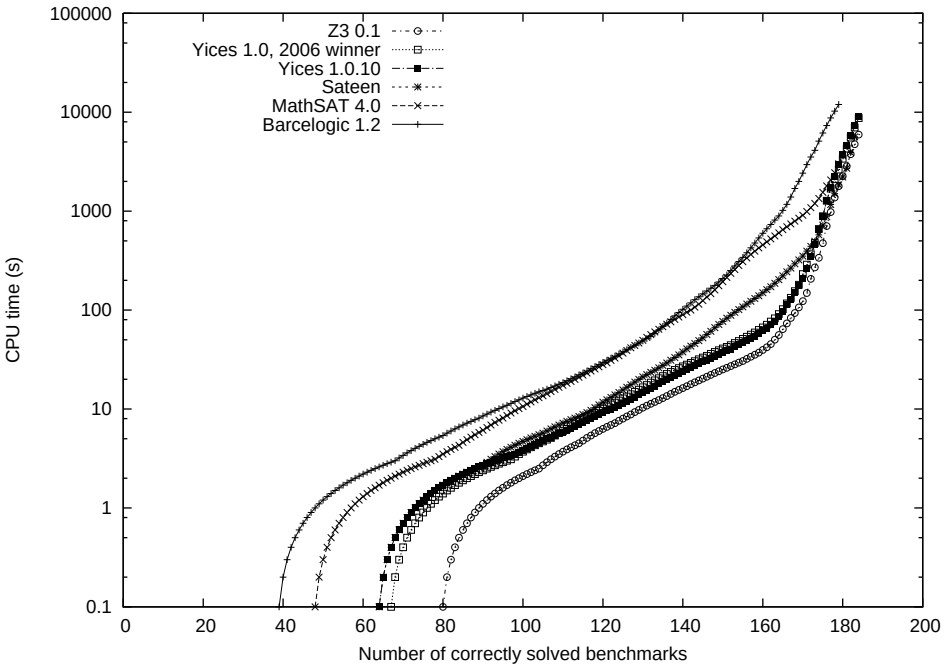


Fig. 4. Benchmark comparisons of (above) the top two contenders in the QF_RDL division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3 0.1	184	5936.5	91	93	0	19	0
Yices 1.0.10	184	9027.9	91	93	0	19	0
Sateen	183	5629.2	91	92	1	19	0
MathSAT 4.0	180	3627.3	87	93	2	21	0
Barcelogic 1.2	179	11974.6	86	93	0	24	0
Yices 1.0, *2006 winner*	184	8725.5	91	93	0	19	0

Fig. 5. Results in the QF_IDL division.

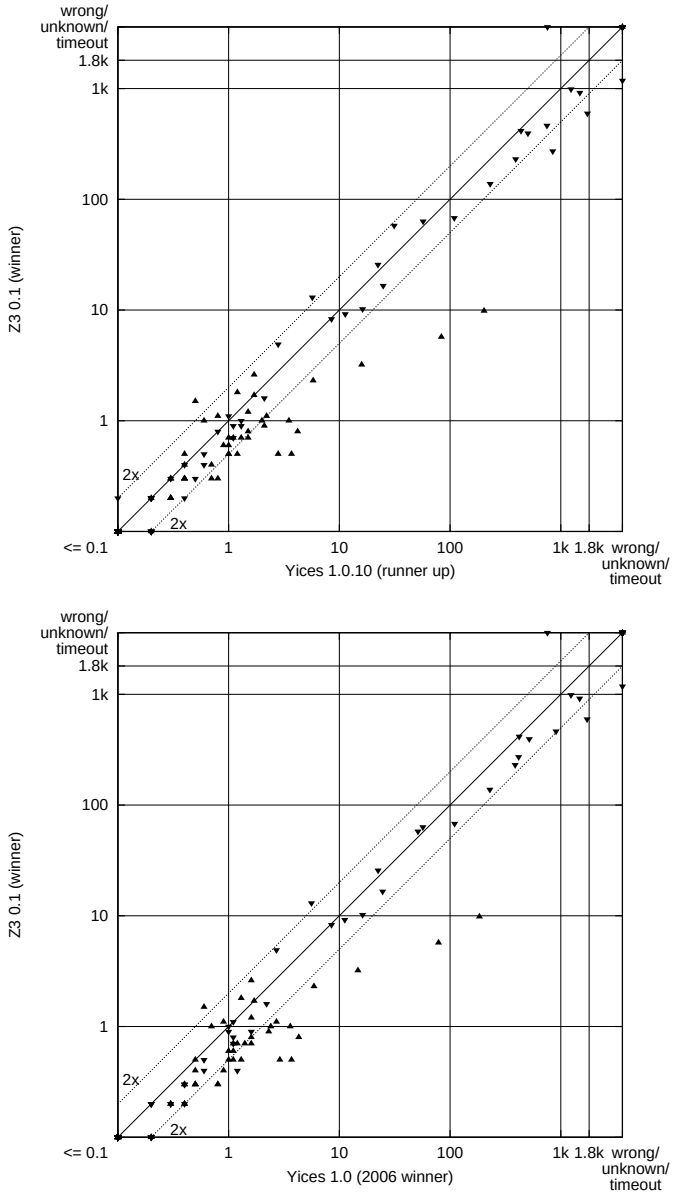
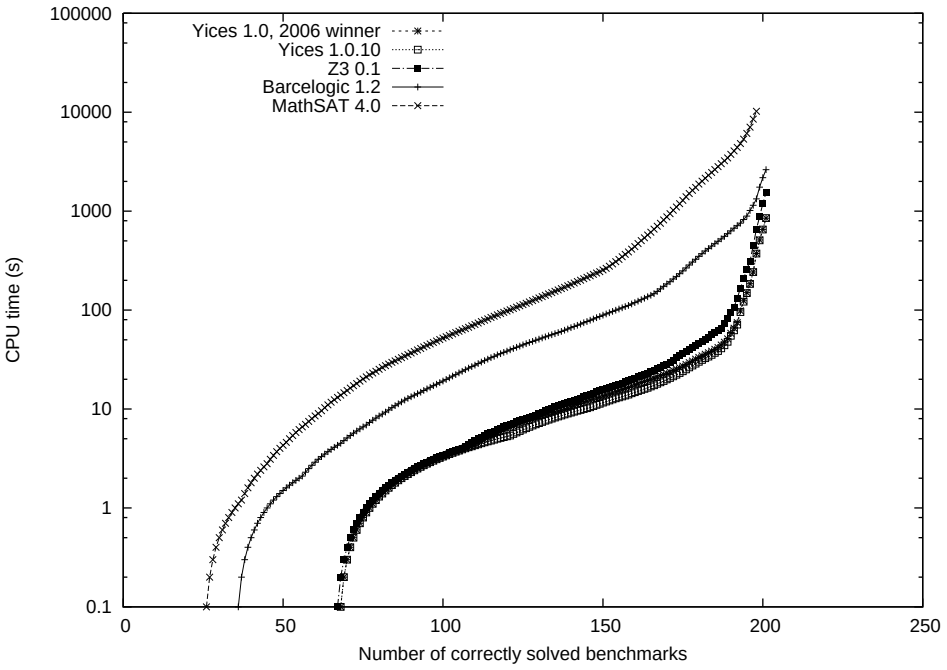


Fig. 6. Benchmark comparisons of (above) the top two contenders in the QF_IDL division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Yices 1.0.10	201	850.3	106	95	0	2	0
Z3 0.1	201	1528.7	106	95	0	2	0
Barcelogic 1.2	201	2625.5	106	95	0	2	0
MathSAT 4.0	198	10211.4	104	94	0	5	0
Yices 1.0, *2006 winner*	201	848.6	106	95	0	2	0

Fig. 7. Results in the QF_UFIDL division.

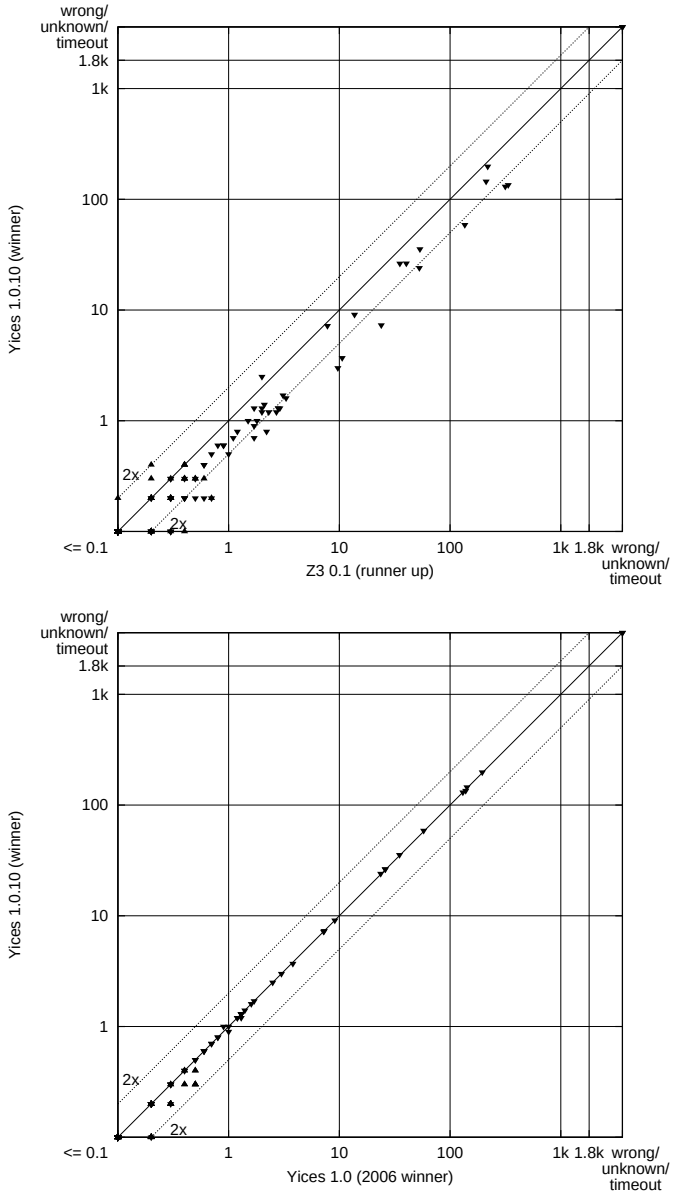
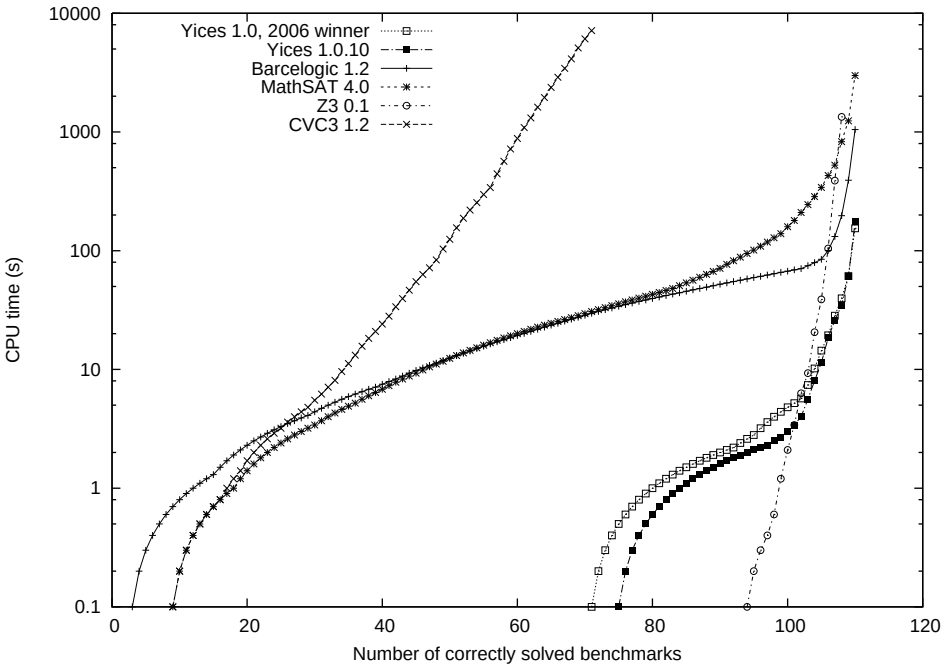


Fig. 8. Benchmark comparisons of (above) the top two contenders in the QF_UFIDL division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Yices 1.0.10	110	174.9	27	83	0	0	0
Barcelogic 1.2	110	1049.3	27	83	0	0	0
MathSAT 4.0	110	2992.6	27	83	0	0	0
Z3 0.1	108	1341.2	25	83	0	2	0
CVC3 1.2	71	7148.3	20	51	0	39	0
Yices 1.0, *2006 winner*	110	154.3	27	83	0	0	0

Fig. 9. Results in the QF_UFLIA division.

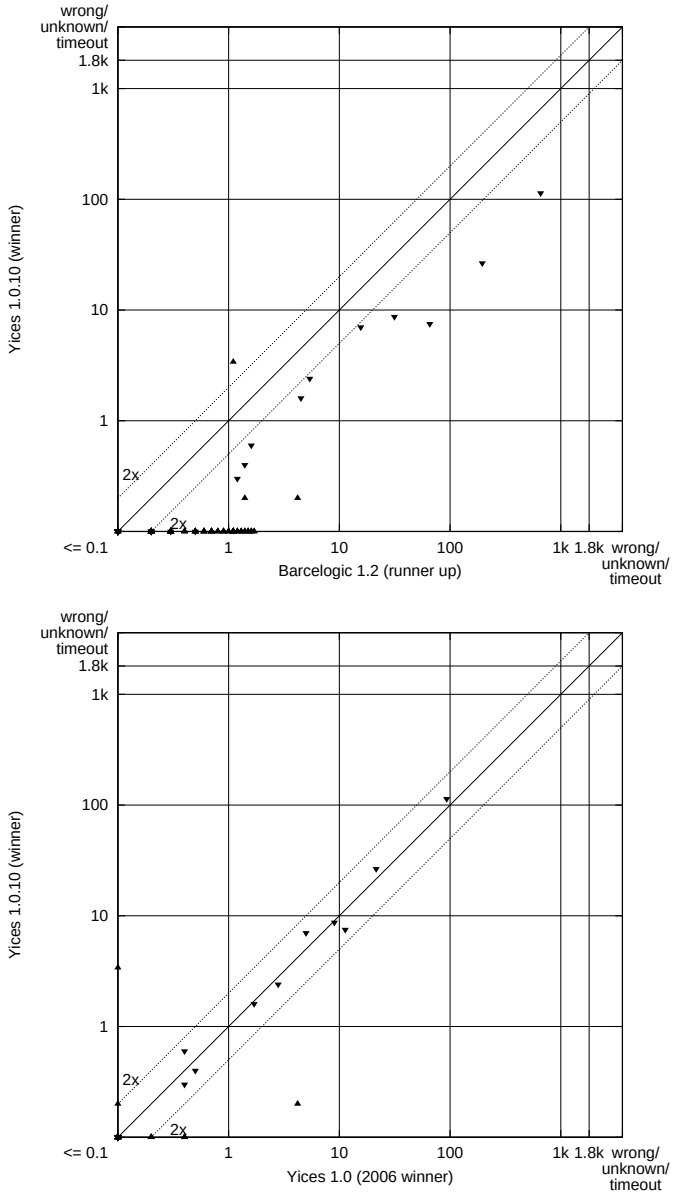
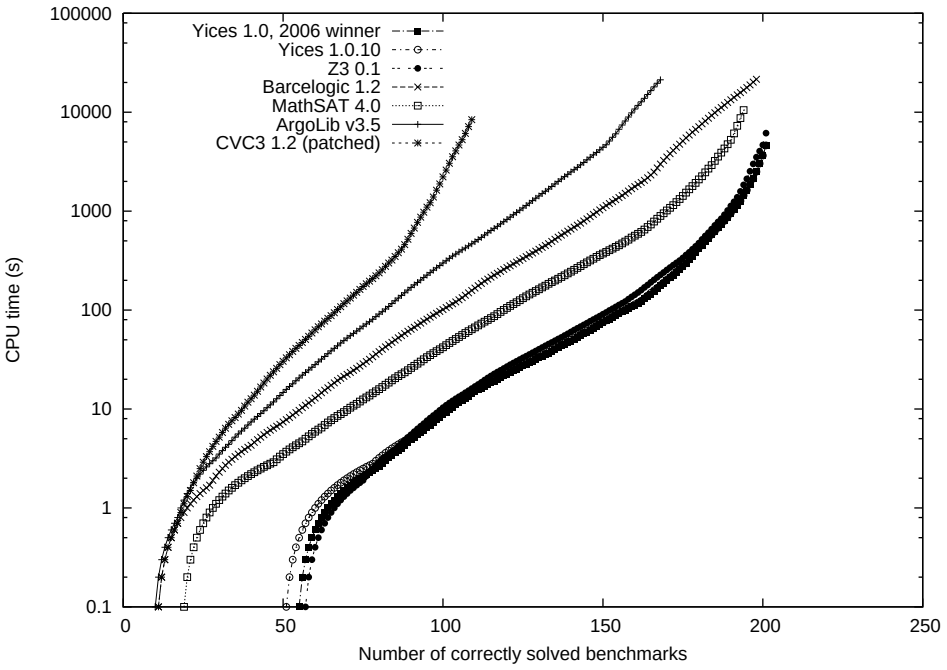


Fig. 10. Benchmark comparisons of (above) the top two contenders in the QF_UFLIA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Yices 1.0.10	201	4612.2	100	101	0	1	0
Z3 0.1	201	6147.3	100	101	0	1	0
Barcelogic 1.2	198	21510.2	98	100	0	4	0
MathSAT 4.0	194	10497.9	95	99	0	8	0
ArgoLib v3.5	168	21272.2	73	95	0	34	0
CVC3 1.2	0	0.0	0	0	202	0	0
Yices 1.0, *2006 winner*	201	4609.3	100	101	0	1	0
CVC3 1.2 (patched)	109	8415.3	49	60	30	63	0

Fig. 11. Results in the QF_LRA division.

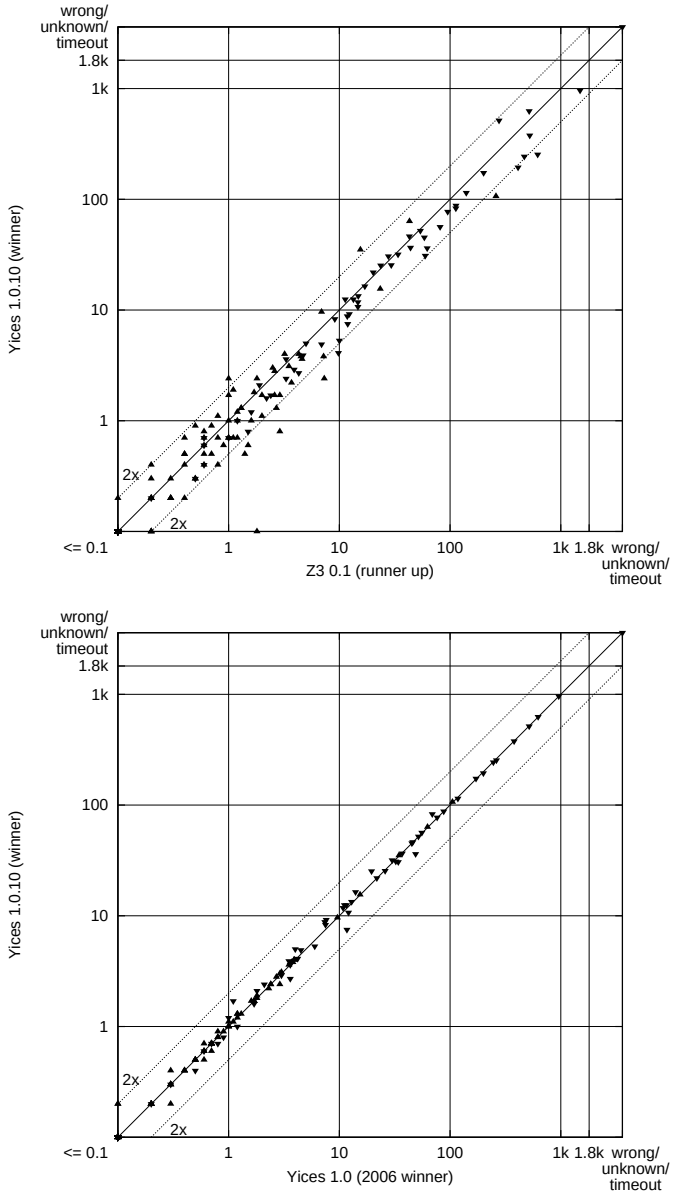
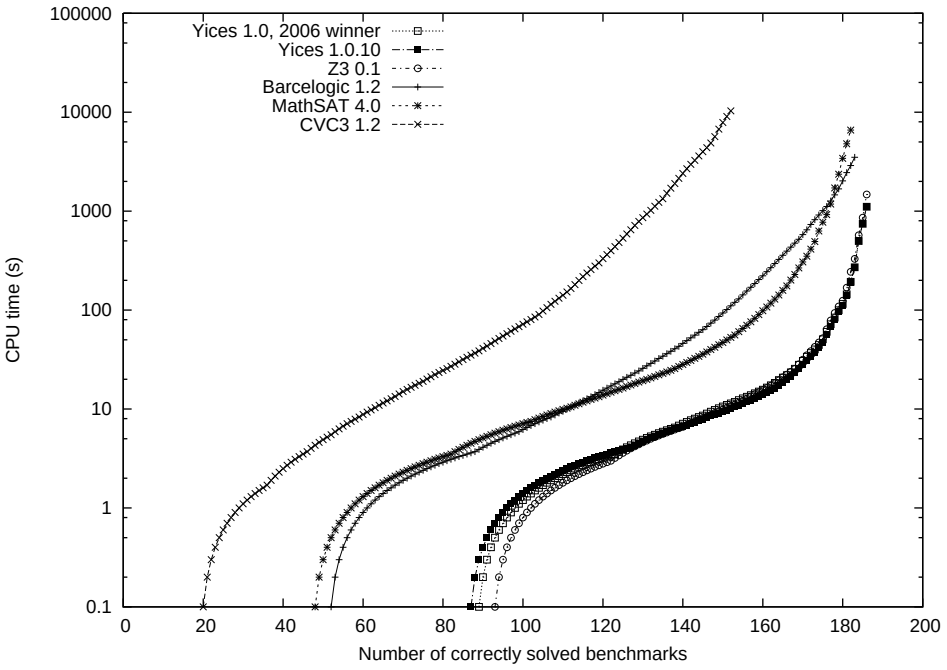


Fig. 12. Benchmark comparisons of (above) the top two contenders in the QF_LRA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Yices 1.0.10	186	1108.7	132	54	0	17	0
Z3 0.1	186	1472.7	132	54	0	17	0
Barcelogic 1.2	183	3501.5	131	52	1	19	0
MathSAT 4.0	182	6588.4	130	52	1	20	0
CVC3 1.2	144	10406.1	114	38	30	20	1
Yices 1.0, *2006 winner*	186	1107.4	132	54	0	17	0

Fig. 13. Results in the QF_LIA division.

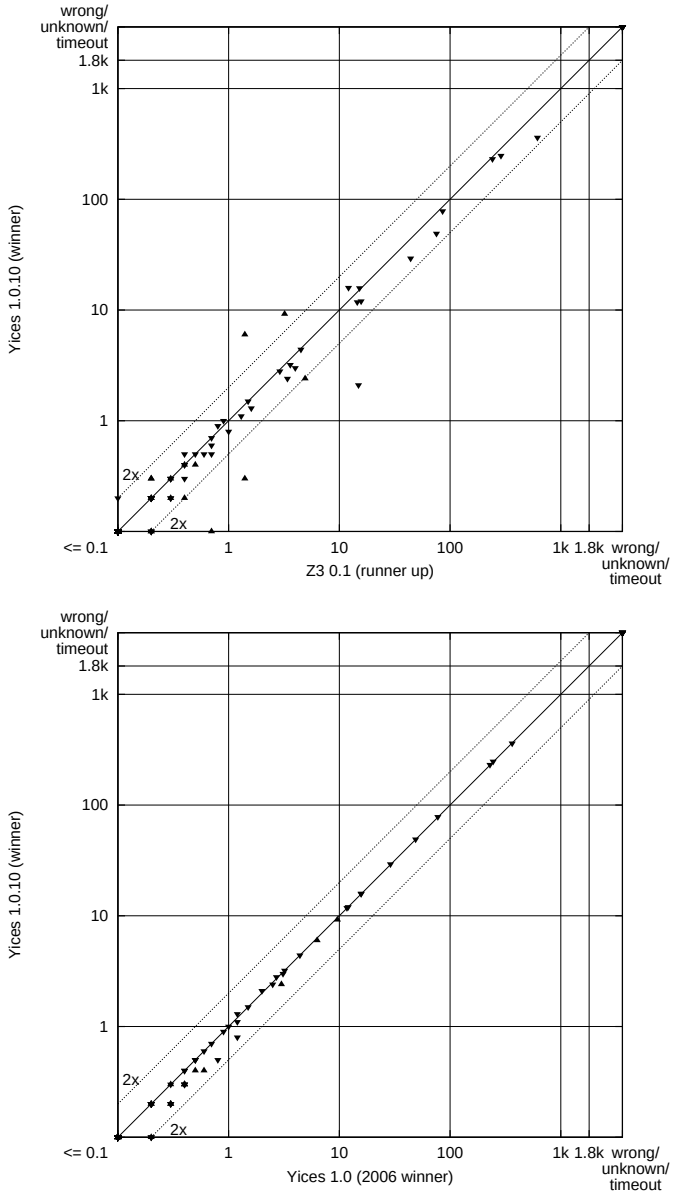
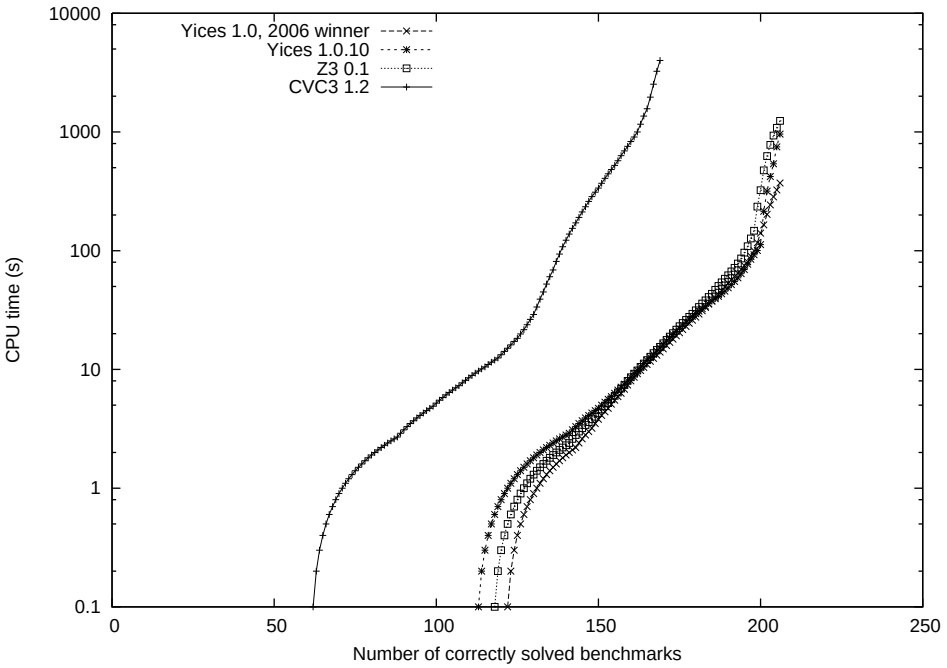


Fig. 14. Benchmark comparisons of (above) the top two contenders in the QF_LIA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Yices 1.0.10	206	960.1	113	93	0	0	0
Z3 0.1	206	1240.0	113	93	0	0	0
CVC3 1.2	169	3999.9	95	74	34	3	0
Yices 1.0, *2006 winner*	206	371.7	113	93	0	0	0

Fig. 15. Results in the QF_AUFLIA division.

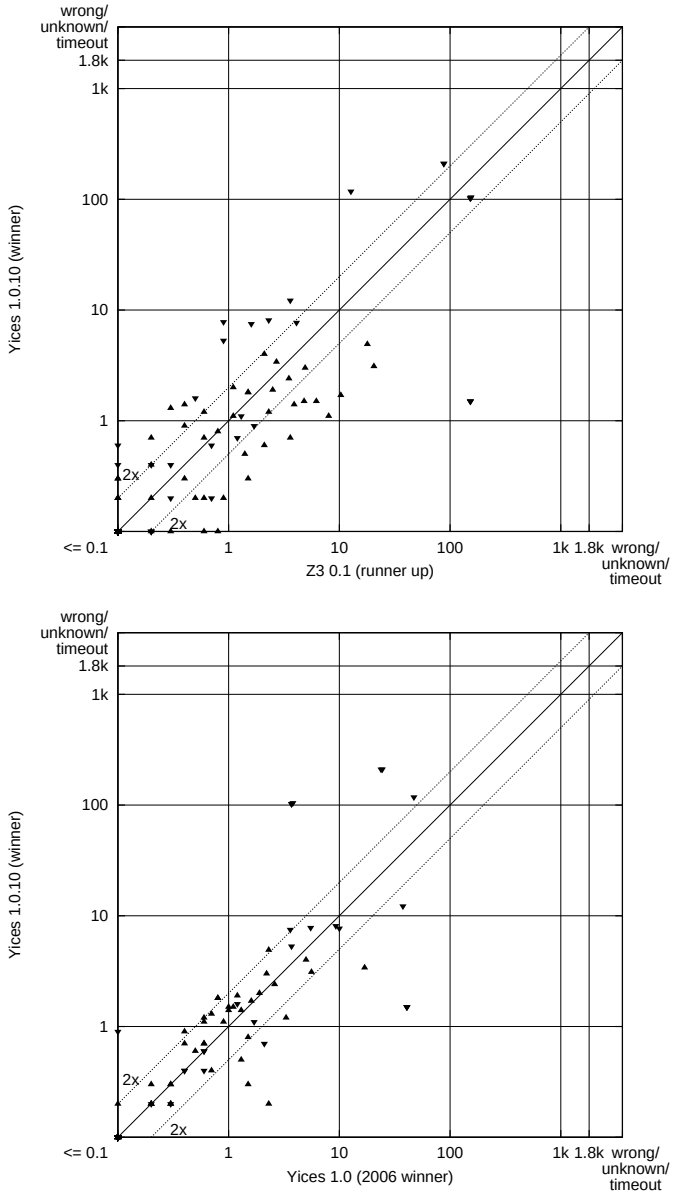
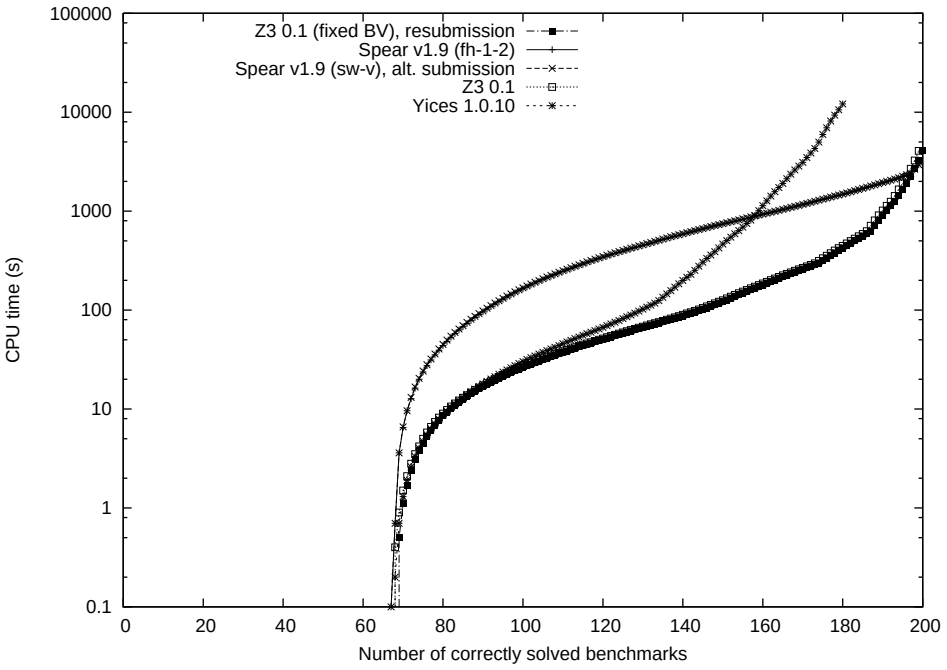


Fig. 16. Benchmark comparisons of (above) the top two contenders in the QF_AUFLIA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Spear v1.9 (fh-1-2)	199	2933.1	38	161	1	0	0
Z3 0.1	191	4069.0	37	162	0	0	1
Yices 1.0.10	180	12113.0	38	142	0	20	0
Z3 0.1 (fixed BV), *resubmission*	200	4069.0	38	162	0	0	0
Spear v1.9 (sw-v), *alt. submission*	199	2948.8	38	161	1	0	0

Fig. 17. Results in the QF_BV division.

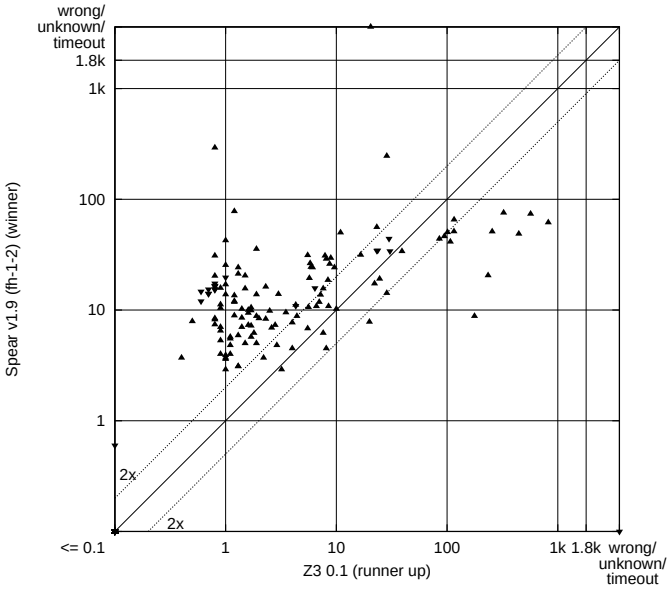
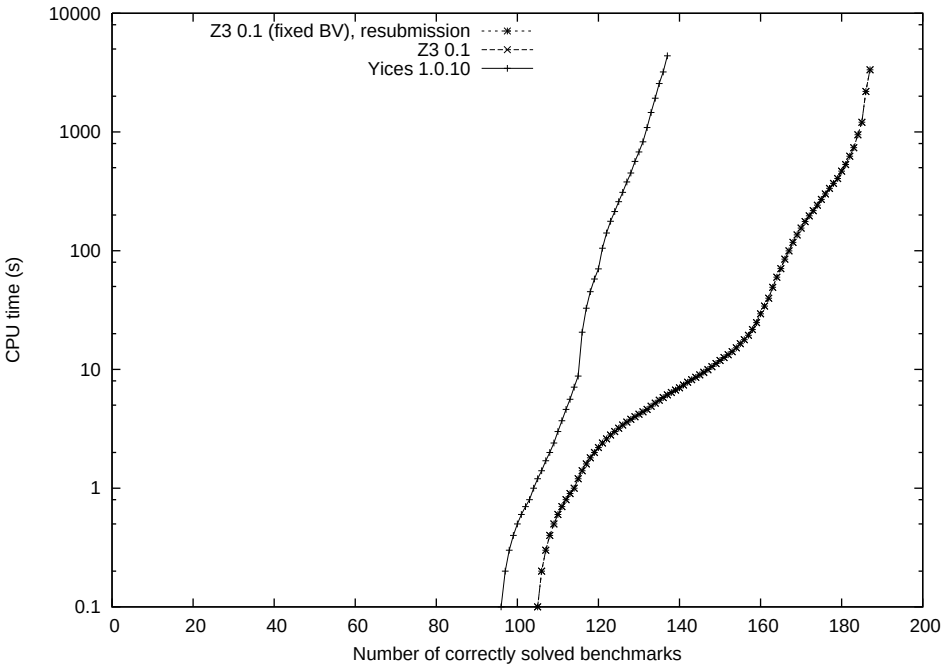


Fig. 18. A benchmark comparison of the top two contenders in the QF_BV division. This division is new in this year's competition.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3 0.1	187	3344.2	74	113	3	10	0
Yices 1.0.10	137	4383.5	58	79	13	50	0
Z3 0.1 (fixed BV), *resubmission*	187	3338.8	74	113	3	10	0

Fig. 19. Results in the QF_AUFBV division.

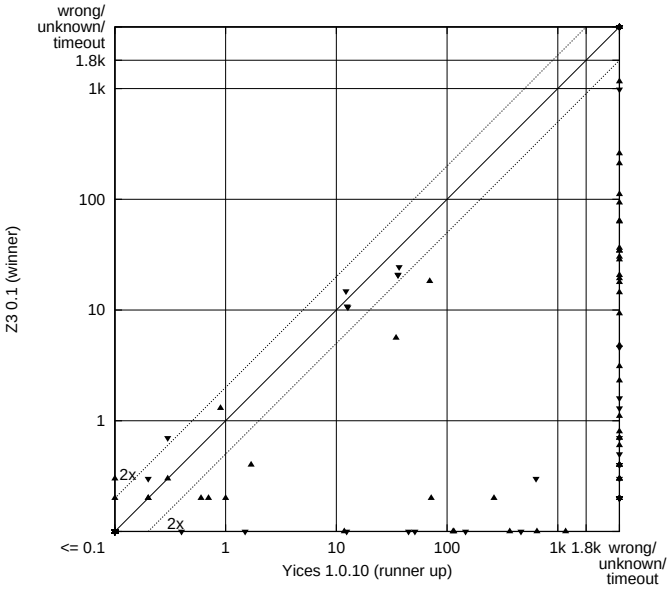
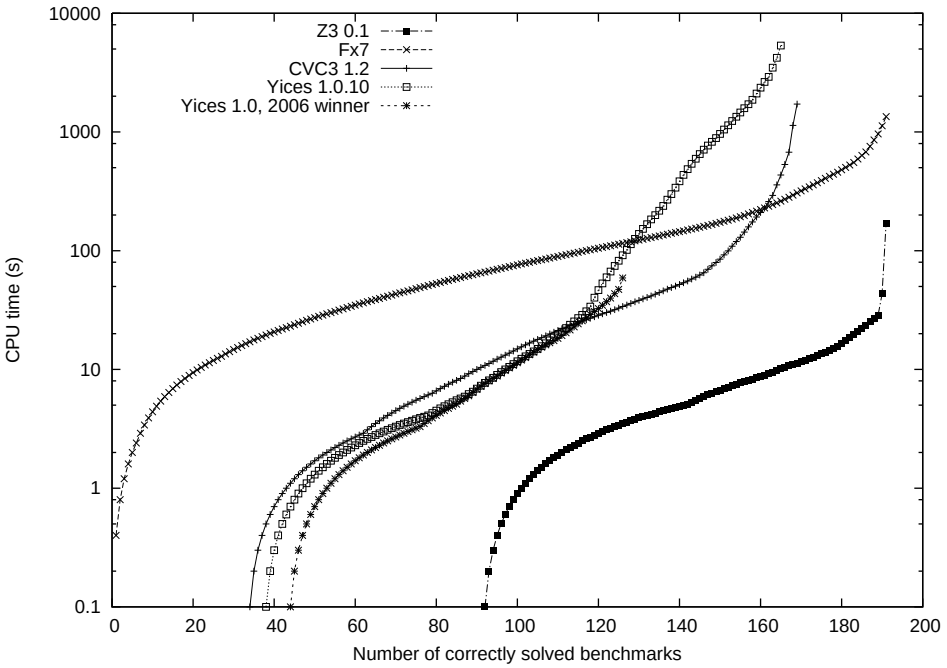


Fig. 20. A benchmark comparison of the top two contenders in the QF_AUFBV division. This division is new in this year's competition.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3 0.1	191	169.2	191	0	9	1	0
Fx7	191	1348.0	191	0	10	0	0
CVC3 1.2	169	1719.7	169	0	24	8	0
Yices 1.0.10	165	5342.0	165	0	16	20	0
Yices 1.0, *2006 winner*	126	59.0	126	0	65	10	0

Fig. 21. Results in the AUFLIA division.

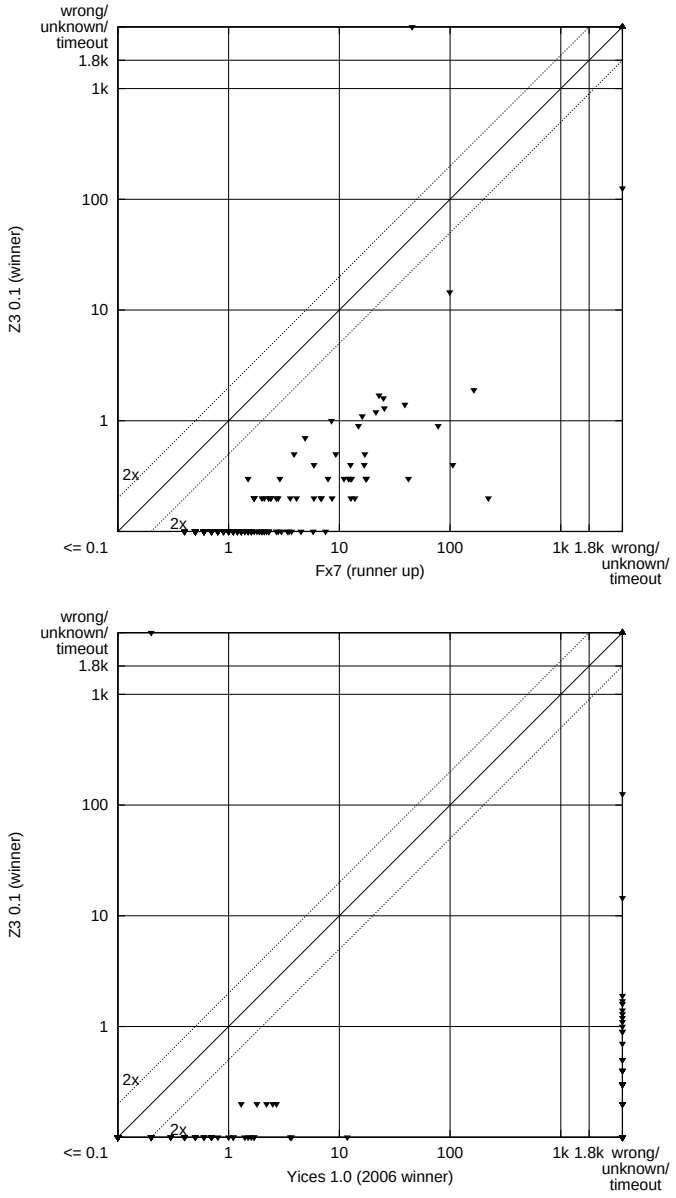
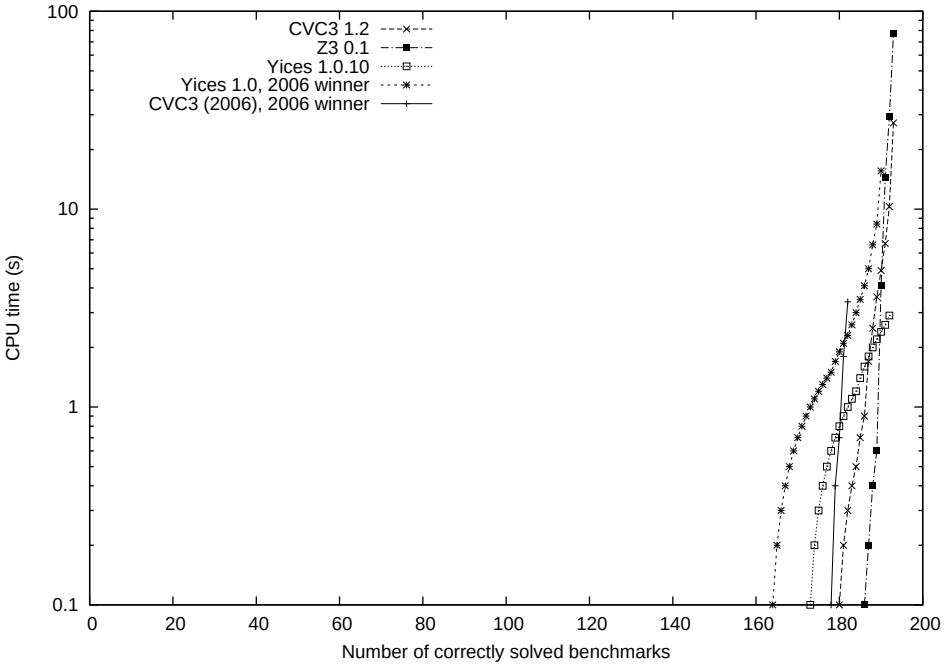


Fig. 22. Benchmark comparisons of (above) the top two contenders in the AUFLIA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
CVC3 1.2	193	27.3	193	0	7	0	0
Z3 0.1	193	77.7	193	0	7	0	0
Yices 1.0.10	192	2.9	192	0	4	4	0
Yices 1.0, *2006 winner*	190	15.6	190	0	10	0	0
CVC3 (2006), *2006 winner*	182	3.4	182	0	18	0	0

Fig. 23. Results in the AUFLIRA division.

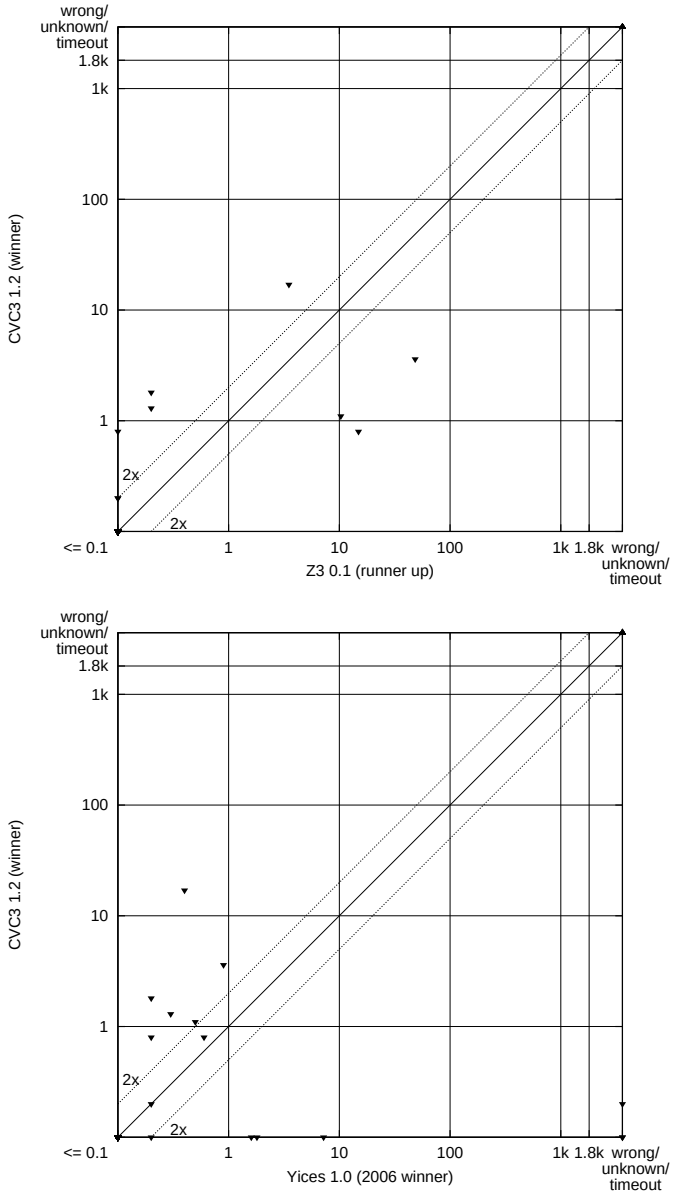


Fig. 24. Benchmark comparisons of (above) the top two contenders in the AUFLIRA division this year, and (below) last year's and this year's winners.

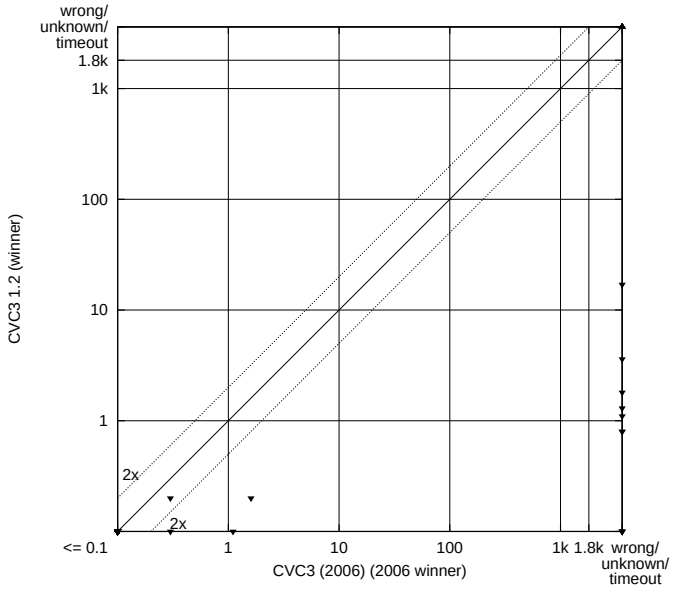


Fig. 25. A benchmark comparison of this year's and last year's winner in the AUFLIRA division.

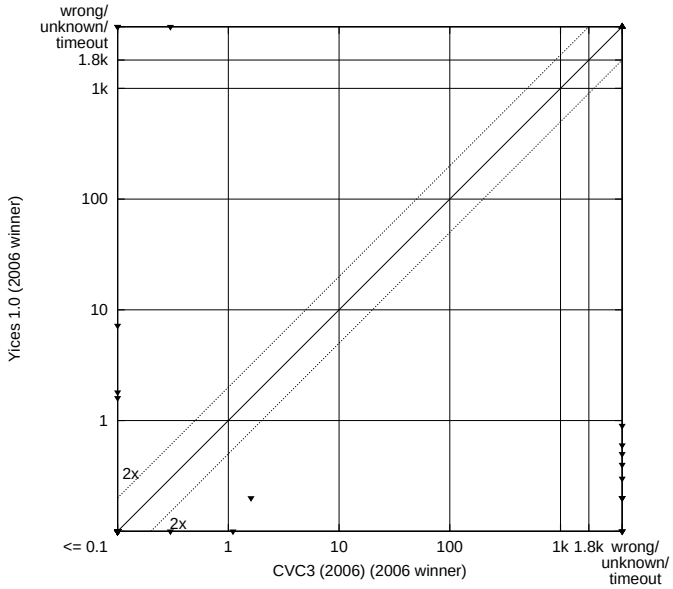


Fig. 26. A benchmark comparison of last year's co-winners on this year's competition benchmarks in the AUFLIRA division.