Design and Results of the 4th Annual Satisfiability Modulo Theories Competition (SMT-COMP 2008)

Clark Barrett
Department of Computer Science
New York University

Morgan Deters
LSI Department
Technical University of Catalonia

Albert Oliveras
LSI Department
Technical University of Catalonia

Aaron Stump
Department of Computer Science
University of Iowa

New York University Technical Report: TR2010–931

Abstract

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 three editions, SMT-COMP 2008 was held as a satellite event of CAV 2008, held July 7–14, 2008. This report gives an overview of the rules, competition format, benchmarks, participants and results of SMT-COMP 2008.

1 Introduction

In a wide variety of applications, domain-specific reasoning turns out to be crucial for the success of automated reasoning tools. A paradigmatic example is the one of arithmetic reasoning: in planning applications, for example, it comes in handy when forcing that the number of resources consumed at a certain time does not exceed a given limit, and, similarly, in placement problems it might be helpful when imposing some given distances between certain objects. In some other applications, like software or hardware verification, what matters is modular arithmetic due to the finiteness of the numbers representable in a computer, which may cause overflows undetectable using plain arithmetic. But arithmetic reasoning alone does not always suffice, as many applications also require to reason about sets, lists, queues or even reachability between nodes in a given graph, and, moreover, all this domain-specific information is typically surrounded by a big amount of boolean connectives.

A successful way to tackle these type of problems is to cast them as Satisfiability Modulo Theories (SMT) formulas, where the goal is to decide the satisfiability of a given formula modulo a background theory, like, e.g. linear arithmetic, modular arithmetic, sets or lists. SMT allows one to encode these problems in a very natural and compact way, preventing the formula from blowing up or losing important structural information. After the encoding is done, one may decide, depending on the problem, to solve it via a reduction to SAT, a translation into a simpler background theory, combining a SAT solver with several dedicated theory solvers, or even axiomatizing (part of) the theory and using first-order reasoning methods.

In order to evaluate and facilitate the proliferation of different SMT approaches, algorithms and implementations, 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, heavily inspired by the TPTP library [6]. Also influenced by the well-known success of the SAT competition for SAT solvers (see `http://www.satcompetition.org`) and the CASC competition for first-order theorem provers (see `http://www.cs.miami.edu/~tptp/CASC/`),

in 2005 the first Satisfiability Modulo Theories Competition (SMT-COMP) was held. As a first immediate consequence, the SMT-LIB format was supported by all state-of-the-art SMT solvers, hence simplifying the task of comparing different solvers and avoiding the annoying and complicated task of converting benchmarks between different formats. Partly due to this common format, the number of collected benchmarks has grown from some 1300 benchmarks in 2005, to some 40000 for the 2006 competition [1], to some 55000 in 2007 [2], and to over 60000 for the 2008 one.[1]

But increasing the number of benchmarks in the SMT-LIB library is not the ultimate goal of SMT-COMP. Other more important goals are to facilitate newcomers to enter the area by creating divisions for which developing an SMT solver is not very complicated, but also to get closer to the needs of potential users by creating divisions with higher expressive power or simulating frequent situations that arise in the real user of SMT solvers. In this latter direction, SMT-COMP 2008 included the division AUFLIA+p, in which benchmarks contained some hints added by users arising from extra knowledge they have about that particular problem, as it may happen in practice. This variety of different goals has produced an important increase in the number of divisions in the competition, which started with 7 divisions in 2005 and had 15 divisions in the 2008 edition.

With these and several other goals in mind, SMT-COMP 2008 was held July 7–14, 2008, as a satellite event of CAV 2008 in Princeton. The competition was run while CAV 2008 was meeting, in the style of the CADE ATP system competition (CASC) [4, 5]. 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 in 2007 and show intermediate results on a public screen, drawing the attention of CAV attendees. Finally, public results were announced July 13, in a special CAV session, and can be accessed at the SMT-COMP web site (`http://www.smtcomp.org`).

After a brief discussion of what was new in 2008 (Section 2), the rest of this report describes the competition format: rules, problem divisions, and scripts and execution of solvers (Section 3); the benchmarks, with emphasis on the new ones, and their selection for the competition (Section 4); the participants (Section 5) and the final results (Section 6).

2 Novelties in 2008

This section briefly indicates what was new in 2008 from the 2007 competition. Results for 2008 are not described in this section, but rather in Section 6.

2.1 Rules & format

Binary submissions have always been welcome at SMT-COMP. 2008 marked the first year that SMT-COMP specifically recognized *open-source* solvers during its awards ceremony. While there was no official SMT-COMP track restricted only to open-source entrants, the best open-source solver in each division (whether it was the overall winner or not) was mentioned during the awards ceremony.

It was decided for 2008 to retire the *easiest* of the SMT-LIB benchmarks. As such, those benchmarks that *all* 2007 solvers could solve correctly in less than 30 seconds were generally not considered for selection in the 2008 competition. Full details of benchmark selection for 2008 are discussed in Section 4.4.

During the CAV meeting, the organizers of SMT-COMP determined early that the competition was requiring too long and would not finish by the end of the conference week as desired. As such, competition divisions were re-ordered so that competition divisions more "interesting" to the community executed first (new divisions, divisions containing new entrants, and divisions in which the winner was most uncertain). Further, timeouts on some competition divisions were lowered to finish the competition more quickly. The primary cause of this increase in runtime over previous years was the retirement of the very easy benchmarks as described above.

For additional details about the format and rules of SMT-COMP, see Section 3.

[1] Specifically, these counts include only competition divisions of SMT-LIB, excluding the AUFNIRA combined logic containing non-linear arithmetic, nor the retired QF_UFBV32 bit-vector logic.

2.2 Benchmarks

Two new divisions were introduced for the 2008 competition, QF_AX (extensional theory of arrays) and QF_UFLRA (the combined theory of uninterpreted functions and linear real arithmetic). QF_AX contained only recategorized benchmarks from (see Section 4), but was notable this year as some entrants had performed significant work since 2007 on their array solvers. QF_UFLRA contained 900 new, randomly-generated benchmarks.

The AUFLIA division for the 2008 competition was split into AUFLIA$+p$ and AUFLIA$-p$.[2] These divisions pull from the same pool of SMT-LIB benchmarks, the AUFLIA collection. However, in the AUFLIA$-p$ division, SMT-LIB *:pat* annotations were stripped from the benchmarks (at the time of benchmark scrambling), and in AUFLIA$+p$ they were left in for the solvers to exploit. To support this, the benchmark scrambler in use for the 2008 competition (see Section 3.3) was extended to permit the scrambling of these annotations (where previously it always stripped them).

Initially, the organizers intended to treat AUFLIRA similarly to AUFLIA$[+-]p$; however, at the time of the competition there were no *:pat* annotations in AUFLIRA, so such a segregation into two divisions would have served no purpose.

Many new benchmarks since 2007 (6147 in total) were part of SMT-LIB 2008 competition divisions; see Section 4 for details.

2.3 Participating solvers

For full details on SMT-COMP 2008 participants, see Section 5.

There were thirteen entries in SMT-COMP 2008, compared to nine in 2007. There were six new systems submitted (Alt-Ergo, Beaver, Boolector, CL-Sat, OpenSMT, and SWORD). Two systems that entered SMT-COMP 2007 did not enter the 2008 competition (ArgoLib and Fx7). The seven common systems between the two years were Barcelogic, CVC3, MathSat, Sateen, Spear, Yices, and Z3, though many of these had been considerably updated, or rewritten entirely, since the time of the 2007 competition.

Of these tools, Alt-Ergo, Beaver, CL-Sat, CVC3, and OpenSMT are open-source solvers.

3 Competition format

This section describes the rules, divisions, and execution infrastructure of the competition.

3.1 Rules

Here we summarize 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 were submitted to SMT-COMP 2008 by way of the SMT-Exec solver execution service in 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 each division of the 2007 competition were entered to run *hors concours* in their respective divisions of the 2008 competition. Special 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` answers.

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 were 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] In this report, we refer to the combined AUFLIA divisions as AUFLIA$[+-]p$.

3.2 Problem divisions

The following were the divisions for SMT-COMP 2008. Definitions of the corresponding SMT-LIB logics are available on the SMT-LIB web site. New in 2008 were an array division, QF_AX, and the new combined division QF_UFLRA. Also this year AUFLIA was split into AUFLIA$+p$ (leaving the *:pat* annotations in benchmarks) and AUFLIA$-p$ (stripping them).[3] 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_BV: Fixed-width bit-vectors
- QF_AUFBV: Fixed-width bit-vectors with arrays and uninterpreted functions.
- QF_UFIDL: integer difference logic with uninterpreted functions
- QF_AX: Arrays with extensionality
- AUFLIA$+p$: quantified linear integer arithmetic with uninterpreted functions and arrays (with *:pat* annotations)
- AUFLIA$-p$: quantified linear integer arithmetic with uninterpreted functions and arrays (no *:pat* annotations)
- AUFLIRA: quantified linear mixed integer/real arithmetic with uninterpreted functions and arrays
- QF_AUFLIA: linear integer arithmetic with uninterpreted functions and arrays
- QF_UFLRA: linear real arithmetic with uninterpreted functions
- QF_UFLIA: linear integer arithmetic with uninterpreted functions
- QF_LRA: linear real arithmetic
- QF_LIA: linear integer arithmetic

The first five divisions ran with a timeout of 30 minutes, the final ten with a timeout of 20 minutes.

3.3 Scripts and execution

SMT-COMP ran on the *SMT-Exec* execution service, a ten-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). SMT-Exec serves as a year-round execution service and experiment platform for SMT solvers; immediately before and during the annual competition, the public service is taken down to devote the cluster to running the competition. The competition uses the same hardware and software infrastructure, in essence running the competition as a public "experiment" consisting of all the competing solvers. 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.

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 (except for *:pat* annotation in the AUFLIA$+p$ division), 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.

[3] It was originally intended to split the AUFLIRA division similarly, as specified in the official rules for the 2008 competition, but that division currently contains no such annotations.

Sun Grid Engine[4] 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 the past for SMT-COMP, and kept the execution hosts from being idle during the competition run.

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 the timeout[5] 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, a limit of 1.5Gb was utilized (and published prior to competition).

SMT-COMP results were stored in a mysql database.[6] As soon as a solver terminated with a *sat*, *unsat*, or *unknown* answer, or timed out, a result 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. Asynchronous Javascript (AJAX) was employed to poll periodically for new results and highlight them on the results pages during the competition.

4 Benchmarks

As in previous years, one of the main motivations for SMT-COMP 2008 was to collect additional SMT benchmarks. A total of 6147 new benchmarks over 11 SMT-LIB logics were collected, bringing the total number of benchmarks for 2008 to 61544.[7]

4.1 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 SMT solvers from the 2007 competition with a 10-minute timeout and using the formula:

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

  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 compliance 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*).
- 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,

[4] `http://www.sun.com/software/gridware/`

[5] The timeout for SMT-COMP 2008 was 30 minutes for divisions QF_UF, QF_RDL, QF_IDL, QF_BV, and QF_AUFBV, and 20 minutes for divisions QF_UFIDL, QF_AX, AUFLIA$+p$, AUFLIA$-p$, AUFLIRA, QF_AUFLIA, QF_UFLRA, QF_UFLIA, QF_LRA, and QF_LIA.

[6] `http://www.mysql.com/`

[7] These numbers count AUFLIA$[+-]p$ benchmarks only once, although internally this same pool of benchmarks is used for both AUFLIA divisions.

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.* [3]) whose proof has been independently checked.

4.2 New benchmarks for existing divisions

New benchmarks were obtained in almost every division, the only exceptions being QF_RDL and QF_AUFLIA. The benchmarks came from a wide variety of research groups. Unlike in previous years, translation into SMT-LIB format was done by the submitters, not by the competition organizers, indicating that the SMT-LIB standard can be successfully used by users as well as developers of SMT solvers.

New benchmarks spanned all three categories (random, crafted, and industrial). Industrial benchmarks came from a number of applications: software verification (uclid/catchconv, sexpr, mathsat/Wisa, nec-smt), hardware verification (brummayerbiere2), hybrid systems verification (uclid/tcas), and optimization (miplib). Table 1 lists the new benchmark families in each division (if any) along with their size (number of benchmarks) and category.

4.3 New divisions

Two new benchmark divisions were added for SMT-COMP 2008: QF_AX and QF_UFLRA. QF_AX was created by reclassifying those benchmarks from QF_AUFLIA that only make use of the theory of arrays. There were 1485 such benchmarks and this constituted the entirety of the QF_AX division. As shown in Table 1, QF_UFLRA was comprised of new random benchmarks.

4.4 Selection of competition benchmarks

The benchmark selection algorithm was close to the one used in 2007. It was updated only to "retire" some particularly easy benchmarks. 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. Of course, *unknown*-status benchmarks are never eligible for inclusion.

2. All benchmarks in the *check* category are automatically included.

3. *New in 2008:* The most difficult 300 non-check non-unknown benchmarks in each division are always included, together with all benchmarks on which at least one 2007 solver required more than 30 seconds. *This had the effect of retiring "very easy" benchmarks that were solved by every 2007 solver (and therefore have difficulty 0) and that all 2007 solvers could solve in less than 30 seconds,* unless *doing so reduced the pool of benchmarks for the division to less than 300.*

4. 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.

5. 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.

6. 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.

Division	Benchmark family	# new benchmarks	Benchmark category
QF_UF	eq_diamond	100	crafted
	total	100	–
QF_IDL	parity	248	crafted
	schedulingIDL	280	crafted
	total	528	–
QF_BV	brummayerbiere	13	crafted
	brummayerbiere2	65	industrial
	uclid/catchconv	414	industrial
	uclid/tcas	2	industrial
	total	494	–
QF_AUFBV	brummayerbiere	293	crafted
	total	293	–
QF_UFIDL	bcnscheduling	13	crafted
	mathsat/EufLaArithmetic/vhard	19	crafted
	total	32	–
AUFLIA$[+-]p$	sexpr	32	industrial
	total	32	–
AUFLIRA	peter	198	crafted
	total	198	–
QF_UFLRA	mathsat/RandomCoupled	400	random
	mathsat/RandomDecoupled	500	random
	total	900	–
QF_UFLIA	mathsat/EufLaArithmetic/hard	17	crafted
	mathsat/EufLaArithmetic/medium	16	crafted
	mathsat/Hash	198	crafted
	mathsat/Wisa	223	industrial
	total	454	–
QF_LRA	miplib	42	industrial
	total	42	–
QF_LIA	nec-smt/large/bftpd_login	361	industrial
	nec-smt/large/checkpass	242	industrial
	nec-smt/large/checkpass_pwd	642	industrial
	nec-smt/large/getoption	20	industrial
	nec-smt/large/getoption_directories	65	industrial
	nec-smt/large/getoption_group	356	industrial
	nec-smt/large/getoption_user	259	industrial
	nec-smt/large/handler_sigchld	130	industrial
	nec-smt/large/int_from_list	173	industrial
	nec-smt/large/mygetpwnam	8	industrial
	nec-smt/large/user_is_in_group	125	industrial
	nec-smt/med/checkpass_pwd	215	industrial
	nec-smt/med/config_read_line	28	industrial
	nec-smt/med/getoption_group	2	industrial
	nec-smt/med/int_from_list	111	industrial
	nec-smt/med/mygetpwnam	2	industrial
	nec-smt/med/print_file	6	industrial
	nec-smt/small/checkpass_pwd	14	industrial
	nec-smt/small/config_read_line	2	industrial
	nec-smt/small/int_from_list	15	industrial
	nec-smt/small/print_file	4	industrial
	rings	294	crafted
	total	3074	–
all new benchmarks		6147	–

Table 1: New Benchmarks

5 Participants

There were thirteen entries in SMT-COMP 2008. With respect to SMT-COMP 2007, six new systems were submitted (Alt-Ergo, Beaver, Boolector, CL-Sat, OpenSMT, and SWORD) and two systems participating in 2007 did not enter SMT-COMP 2008 (ArgoLib and Fx7). A brief description of each system follows. 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 2008 web site. The binaries run during the competition for all solvers are also available there.

Alt-Ergo. Submitted by Sylvain Conchon (LRI, Université Paris-Sud and INRIA SaclayIle-de-France) and Evelyne Contejean (LRI, CNRS, and INRIA SaclayIle-de-France), Alt-Ergo is an Ocaml implementation of a generic congruence closure algorithm CC(X) that provides for a method of theory combination similar to Shostak's approach. Alt-Ergo includes its own SAT engine.
Problem divisions: AUFLIA$+p$, AUFLIA$-p$, AUFLIRA.

Barcelogic 1.3. Submitted by Miquel Bofill (Universitat de Girona) and Morgan Deters, Germain Faure, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio (Technical University of Catalonia in Barcelona), Barcelogic is a DPLL(T) solver supporting arithmetic and arrays.
Problem divisions: QF_UF, QF_RDL, QF_IDL, QF_UFIDL, QF_AX, QF_AUFLIA, QF_UFLRA, QF_UFLIA, QF_LRA, QF_LIA.

Beaver 1.0. Submitted by Susmit Jha, Rhishikesh Limaye, and Sanjit A. Seshia (UC Berkeley), Beaver is an SMT solver for the theory of quantifier-free finite-precision bit-vector arithmetic. Beaver operates by performing a series of rewrites and simplifications that transform the starting bit-vector arithmetic formula into a Boolean circuit and then into a Boolean satisfiability (SAT) problem in CNF.
Problem divisions: QF_BV.

Boolector 0.4. Submitted by Robert Brummayer and Armin Biere (Johannes Kepler University, Linz, Austria), Boolector is a decision procedure for the quantifier-free theory of Bit-vectors, and the quantifier-free extensional theory of arrays with bit-vectors and uninterpreted functions. Additionally, Boolector can be used as a model checker for word-level safety properties [4]. Boolector is implemented in pure C; Picosat is used as the SAT solver.
Problem divisions: QF_BV, QF_AUFBV.

clsat 0.17f. Submitted by Duckki Oe, Timothy Simpson, Aaron Stump, and Terry Tidwell (Washington University in St. Louis), the clsat solver integrates a clauselearning SAT solver with a standard graphbased IDL solver. Started as a class project, it incorporates modern SAT techniques, including watched literals, failuredriven assertions, and conflict clause simplification.
Problem divisions: QF_IDL.

CVC3 1.5. CVC3 is a joint project of New York University and the University of Iowa. The project leaders are Clark Barrett (NYU) and Cesare Tinelli (University of Iowa). Code contributions since last year have been made by Clark Barrett, Yeting Ge (NYU), Dejan Jovanovic (NYU), Alexander Fuchs (Iowa), Lorenzo Platania (University of Genoa), and Darren Kelley (NYU).
Problem divisions: *all*.

MathSAT 4.2. MathSAT was submitted by Alessandro Cimatti and Anders Franzén from FBK-IRST, Trento, and Alberto Griggio and Roberto Sebastiani from University of Trento, Italy. MathSAT is a C++ implementation of the standard "online" lazy integration schema used in many SMT tools. New this year are support for bit-vectors, an improved integer linear arithmetic solver, and support for delayed theory combination.
Problem divisions: QF_UF, QF_RDL, QF_IDL, QF_BV, QF_UFIDL, QF_UFLRA, QF_UFLIA, QF_LRA, QF_LIA.

OpenSMT 0.1. Submitted by Roberto Bruttomesso and Natasha Sharygina (Università della Svizzera Italiana, Lugano, Switzerland), OpenSMT is a small and open-source SMT solver, written in C++, which provides a basic infrastructure for helping non-experts to develop theory-solvers without having to start from scratch. OpenSMT includes a parser for SMT-LIB language, a state- of-the-art SAT-Solver, and a core solver for QF UF logic. An empty, template theory-solver is provided, to facilitate the development of solvers for other logics.
Problem divisions: QF_UF.

Sateen 2.1.1. Sateen was submitted by Hyondeuk Kim (University of Colorado at Boulder), Hoonsang Jin (Cadence Design Systems), and Fabio Somenzi (Colorado at Boulder). It is a satisfiability solver that combines a propositional reasoning engine with theory-specific procedures. It uses the lazy approach that relies on incremental refinements of a propositional abstraction of the given formula during the enumeration of its solutions.
Problem divisions: QF_RDL, QF_IDL.

Spear. Submitted by Domagoj Babić, Spear is a theorem prover for bit-vector arithmetic. It was designed chiefly for software verification, but also for other industrial problems, like bounded hardware modelchecking.
Problem divisions: QF_BV.

SWORD v0.2. Submitted by Robert Wille, André Sülflow, and Rolf Drechsler (University of Bremen), SWORD is a SAT-like solver that facilitates word level information. The main idea behind SWORD is based on the following observation: Current SAT solvers perform very well on instances with a large number of logic operations. But when more complex functions like arithmetic units are considered, the performance degrades with increasing data-path width. In contrast, pure word level approaches handle *e.g.* arithmetic operations very fast but suffer from complexity problems when irregularities in the word level structure (*e.g.* bit slicing) occur. SWORD uses MiniSat as a SAT engine.
Problem divisions: QF_BV.

Yices2 (proto c). Submitted by Bruno Dutertre (SRI International, Menlo Park, California), this version of Yices 2 is a preliminary prototype of the successor to the Yices 1 SMT solver. This new tool will address several limitations of Yices 1, including typechecking issues and limited functionality of the Yices API. The new solver supports a simpler specification language that can be statically typechecked and it provides a full API to access all functions of the solver. It is intended to offer similar or better performance than Yices 1 on most benchmarks, while being more modular, extensible, and maintain- able.
Problem divisions: QF_UF, QF_RDL, QF_IDL, QF_LRA.

Z3.2$^{\alpha}$. Submitted by Leonardo de Moura and Nikolaj Bjørner (Microsoft Research, Redmond), Z3.2$^{\alpha}$ is a new version of the Z3 solver supporting linear real and integer arithmetic, fixed-size bit-vectors, extensional arrays, uninterpreted functions, and quantifiers. It can read problems in SMT-LIB and Simplify formats.
Problem divisions: *all*.

6 Results

The results for each division are summarized in Figures 1 through 30 starting on page 12. 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. 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/`.

6.1 Description of anomalous and surprising results

This year, there was significant improvement over last year's winner in the QF_UF, QF_BV, QF_AUFBV, AUFLIRA, and QF_LIA divisions. There was not so much improvement in QF_RDL, QF_IDL, and QF_LRA. There was not much improvement (but also not much *room* for improvement) in QF_UFIDL, AUFLIA$+p$, AUFLIA$-p$, QF_AUFLIA, QF_UFLIA. QF_AX and QF_UFLRA were new in 2008; there was thus no "winner" from the 2007 competition to which to compare the results.

Competing in the quantified competition divisions AUFLIA$[+-]p$ and AUFLIRA, Alt-Ergo demonstrated incompleteness, incorrectly reporting as *satisfiable* 17 unsatisfiable instances of AUFLIA+p, 14 unsatisfiable instances of AUFLIA-p, and 52 unsatisfiable instances of AUFLIRA. It was therefore disqualified from the competition as per the rules. (It is listed separately, in the *hors concours* section of the results, for that reason). For purposes of comparison, an artificial "revision" of Alt-Ergo (appearing as "Alt-Ergo (revised)" corrects this incompleteness error.[8] In this way it achieves a positive result, but as this is an after-deadline entry it ran *hors concours* as well.

6.2 Description of *unknown* results

Unknown results from solvers arise for different reasons. A solver may report *unknown* explicitly, or fail to give a proper *sat* or *unsat* response. In some cases, it is possible to tell the cause of the *unknown* response—the output wasn't *sat* or *unsat* but rather an assertion failure, or a C++ bad_alloc exception escaping the program's top-level. In other situations, we cannot discern the cause—an explicit response of *unknown* may have resulted from a bug or out-of-memory situation which is caught internally, thereby leaving no trace of the cause). In this section we try to detail the causes, when possible, of *unknown* results, based on the detailed solver output logs collected during the competition.

- **QF_IDL:** The clsat solver failed to parse the two *bignum* benchmarks, categorized *check*, citing integer overflow.
- **QF_BV:** Beaver failed on 20 benchmarks in the *brummayerbiere2* family, trying to open a nonexistent file (presumably which existed on a development machine). Z3 and MathSAT failed on a few of the *brummayerbiere/countbits* benchmarks (presumably running out of memory). SWORD failed with a segmentation fault on *stp/testcase15.stp.smt* and *brummayerbiere/countbits1024.smt*. This year's Spear failed on lots of the *brummayerbiere* benchmarks as well as some others (45 parse errors, "Resources exceeded" on *stp/testcase15.stp.smt*, and 8 explicit "unknown" responses were observed); last year's Spear (the 2007 winner in this division) failed on only a subset of these (the same 45 parse errors, out of memory on *stp/testcase15.stp.smt*); CVC3 fails on various benchmarks (2 due to assertion failures, 12 out of memory).
- **QF_AUFBV:** Z3 runs out of memory on a few (the old Z3 failed on 4, and the new Z3.2 is slightly better, running out of memory on only 3); CVC3 failed on many (segmentation fault on 4, perhaps out-of-memory related, and the rest clear memory exhaustion).
- **AUFLIA$+p$:** Z3.2 gave no response on *misc/set14.smt* or *misc/set9.smt*, and ran out of memory on *misc/set19.smt*. Alt-Ergo ran out of memory on *misc/set2.smt*. All other "unknown" results in this division were due to an explicit "unknown" reported by the solvers.
- **AUFLIA$-p$:** The explanation here is exactly the same as for AUFLIA$+p$. The benchmarks selected for AUFLIA$-p$ were the same for small benchmark families due to the way benchmark selection is performed

[8] It was "artificial" in that it wasn't a submission that actually ran in competition, but rather an adjustment of the data collected from the disqualified entrant by changing all *satisfiable* responses by the solver to *unknown* responses; it thus simulates how a revised Alt-Ergo may have placed in competition if it had a wrapper script that changed *sat* to *unknown*. The time differences reported in the results are due to a difference in how times are scored: *unknown* and *timeout* results are not counted against a solver's time, but incorrect answers are.

for SMT-COMP. However, there were some differences in the benchmark make-up for these two similar divisions, and that explains the discrepancies. In particular, Z3 reported one additional (explicit) "unknown" on *piVC/piVC_849b63.smt*, a benchmark not present in AUFLIA+p.

- **AUFLIRA:** All *unknown* reponses in AUFLIRA were explicitly-reported unknowns; there were no crashes or memory-outs in this division. Z3.2 gave 1 explicit "unknown" response on *misc/set9.smt;* CVC3 1.5 gave 46 explicit "unknown" responses in several of the *nasa* benchmark families, and also in the *misc* and *peter* families; CVC3 1.2 (the 2007 winner) reported fewer "unknowns" (34), but did so across the same benchmark families. Alt-Ergo gave no "unknown" responses in AUFLIRA. However, it reported 83 as *satisfiable*, 52 of which were in fact *unsatisfiable* (causing disqualification of the solver from the competition). In the revised Alt-Ergo numbers, which change all satisfiable responses of the solver to explicit "unknowns", leading to the figure of 83 in that row.

- ***various*:** Besides those above, CVC3 gave 82 unknown answers in QF_AX, QF_AUFLIA, QF_UFLRA, QF_LRA, and QF_LIA. These were due to memory exhaustion (in 42 cases), segmentation faults (4), and assertion failures (36).

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.

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 2008 for their support of SMT-COMP 2008 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 2008.

References

[1] 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.

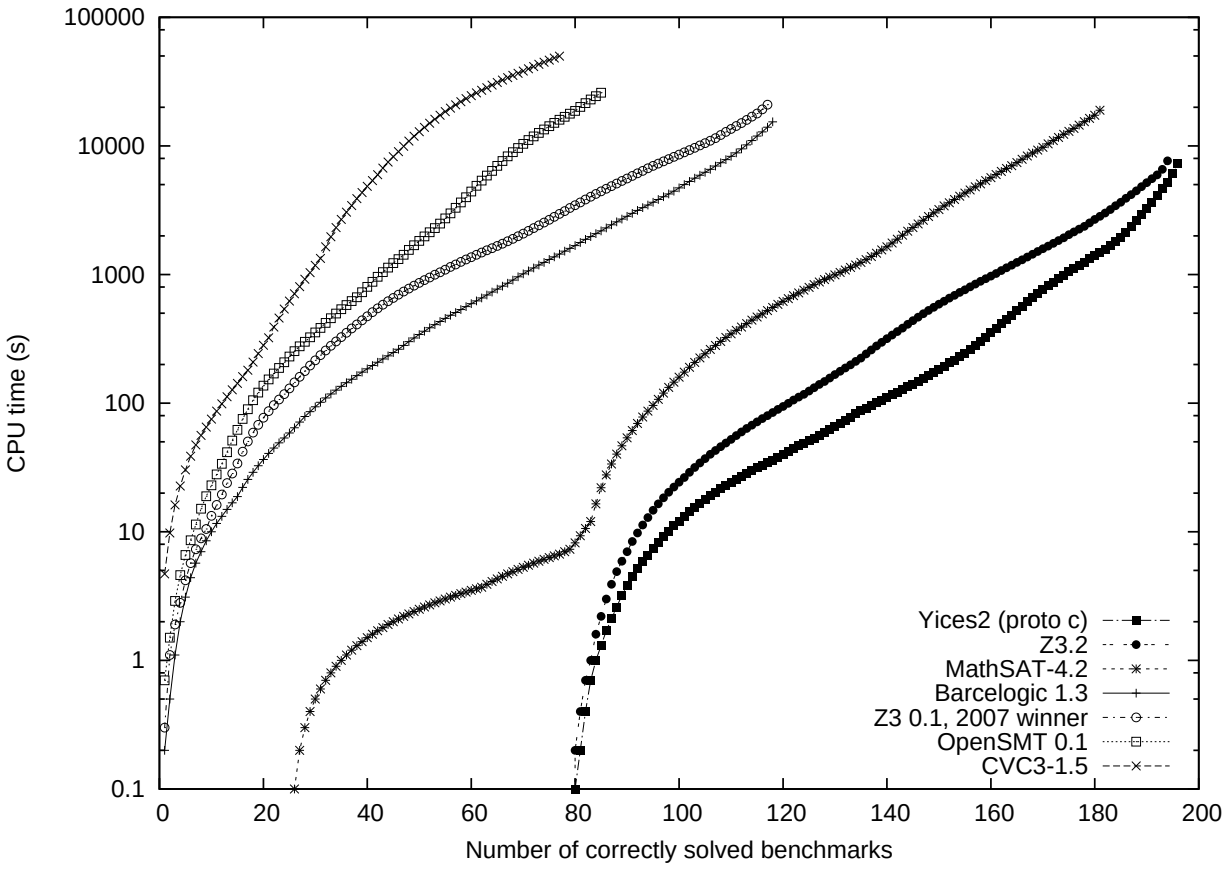
[2] Clark Barrett, Morgan Deters, Albert Oliveras, and Aaron Stump. Design and results of the 3rd annual satisfiability modulo theories competition (SMT-COMP 2007). *International Journal on Artificial Intelligence Tools*, 17(4):569–606, 2008.

[3] 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.

[4] F.J. Pelletier, G. Sutcliffe, and C.B. Suttner. The Development of CASC. *AI Communications*, 15(2-3):79–90, 2002.

[5] G. Sutcliffe and C. Suttner. The State of CASC. *AI Communications*, 19(1):35–48, 2006.

[6] 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
Yices2 (proto c)	196	7299.0	187	9	0	4	0
Z3.2	194	7653.5	185	9	0	6	0
MathSAT-4.2	181	18927.1	172	9	0	19	0
Barcelogic 1.3	118	15384.2	109	9	0	82	0
OpenSMT 0.1	85	25941.8	76	9	0	115	0
CVC3-1.5	77	49914.8	69	8	0	123	0
Z3 0.1, 2007 winner	117	20936.8	108	9	0	83	0

Figure 1: Results in the QF_UF division.

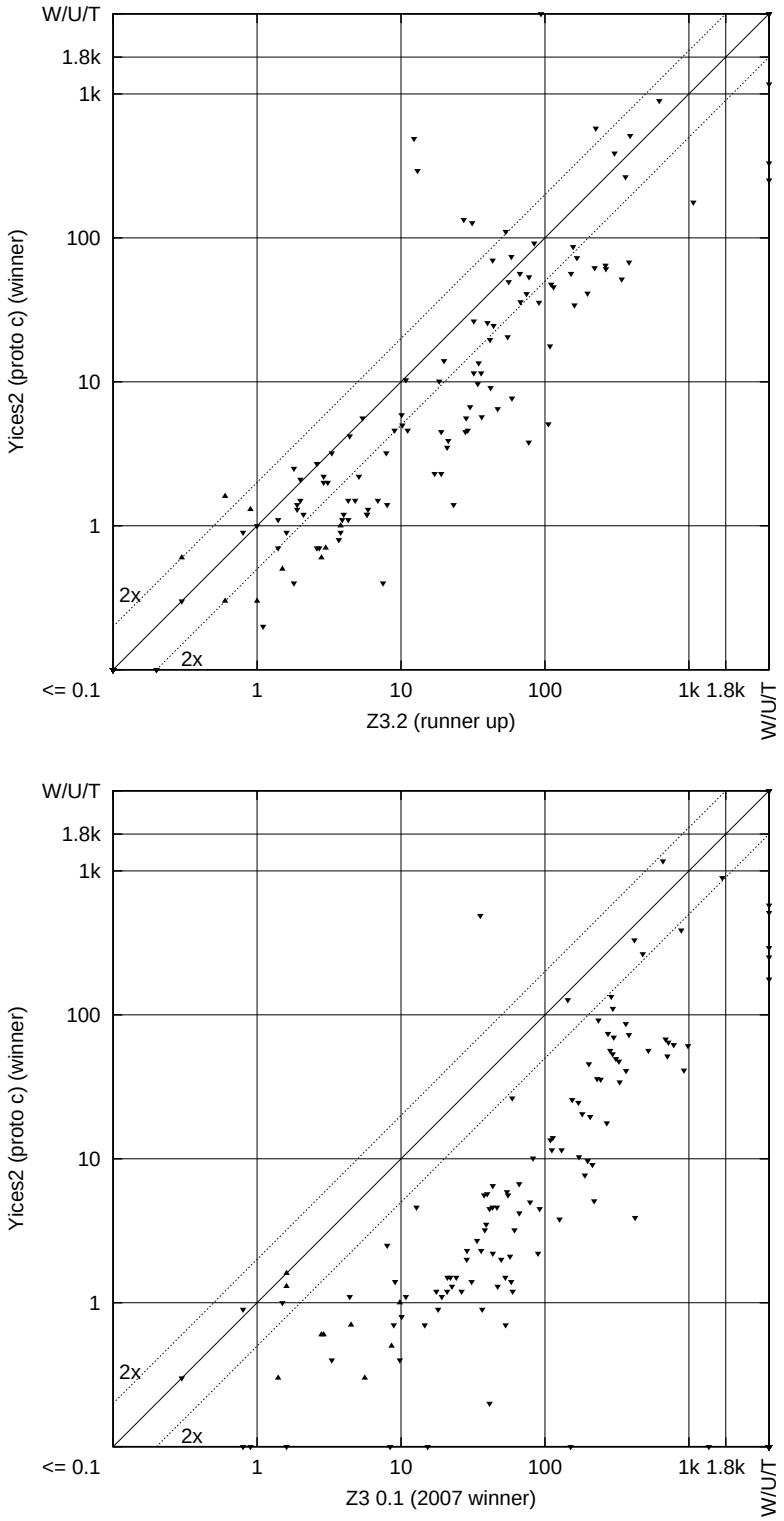
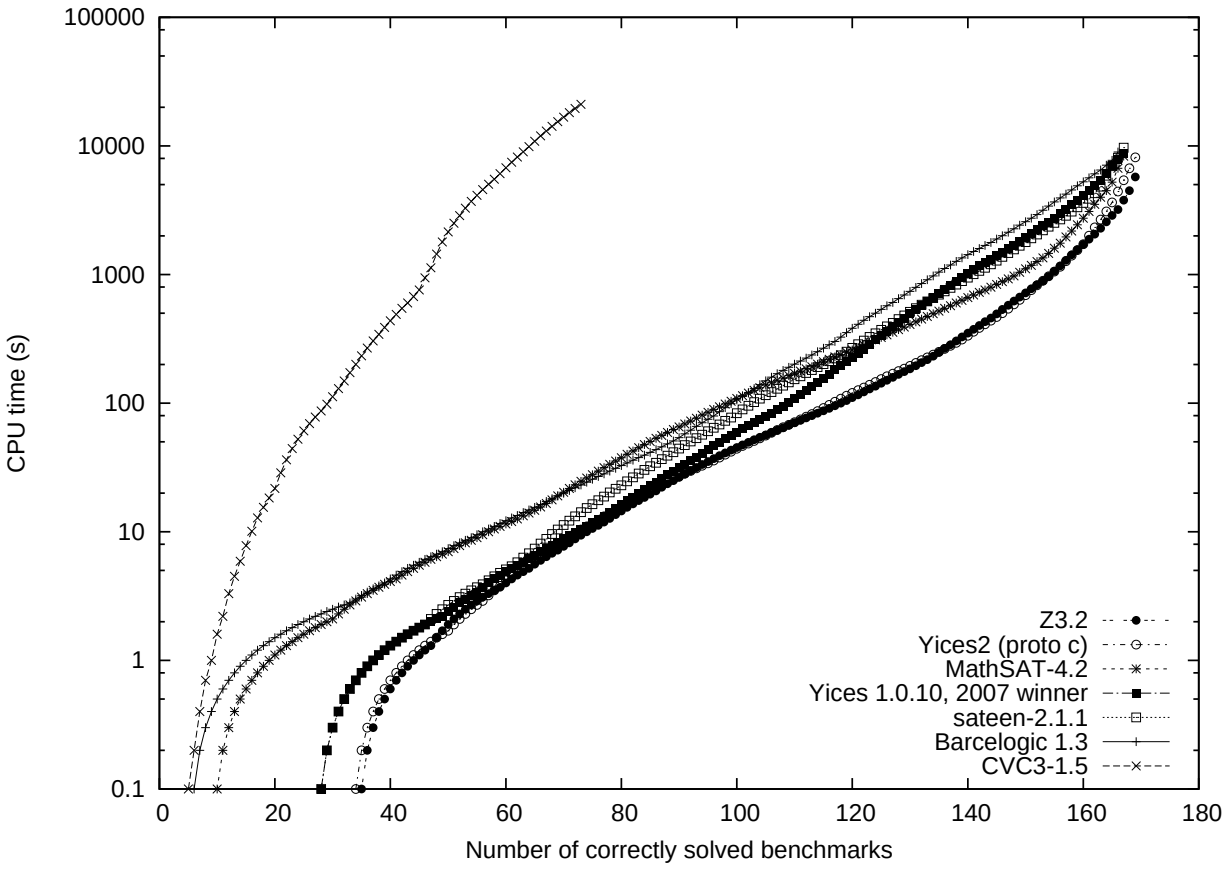


Figure 2: Benchmark comparisons of (above) the top two contenders in the QF_UF division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	169	5740.3	112	57	0	1	0
Yices2 (proto c)	169	8141.5	112	57	0	1	0
MathSAT-4.2	167	8324.9	112	55	0	3	0
sateen-2.1.1	167	9705.3	111	56	0	3	0
Barcelogic 1.3	166	8967.8	111	55	0	4	0
CVC3-1.5	73	21045.4	57	16	0	97	0
Yices 1.0.10, 2007 winner	167	8760.6	111	56	0	3	0

Figure 3: Results in the QF_RDL division.

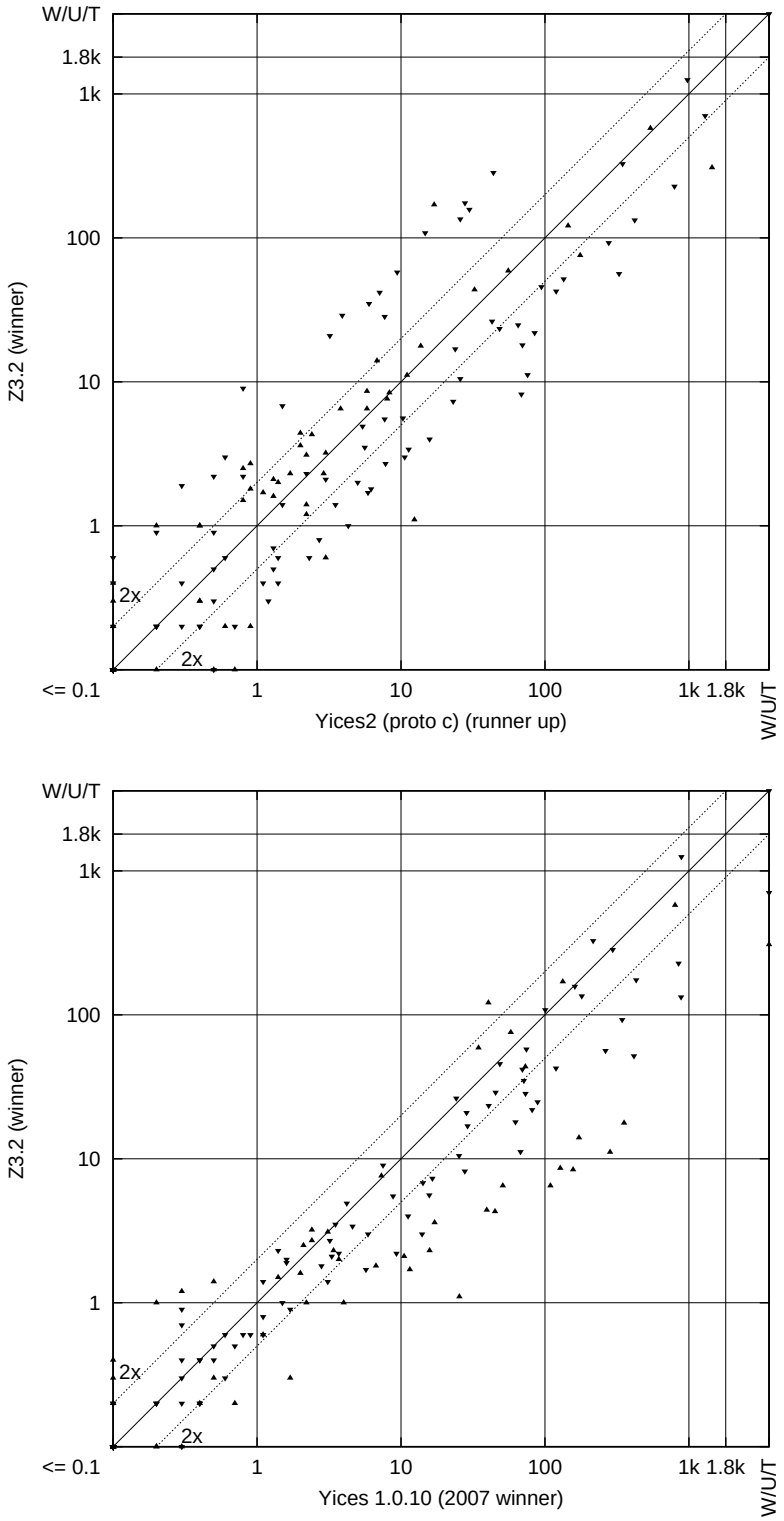
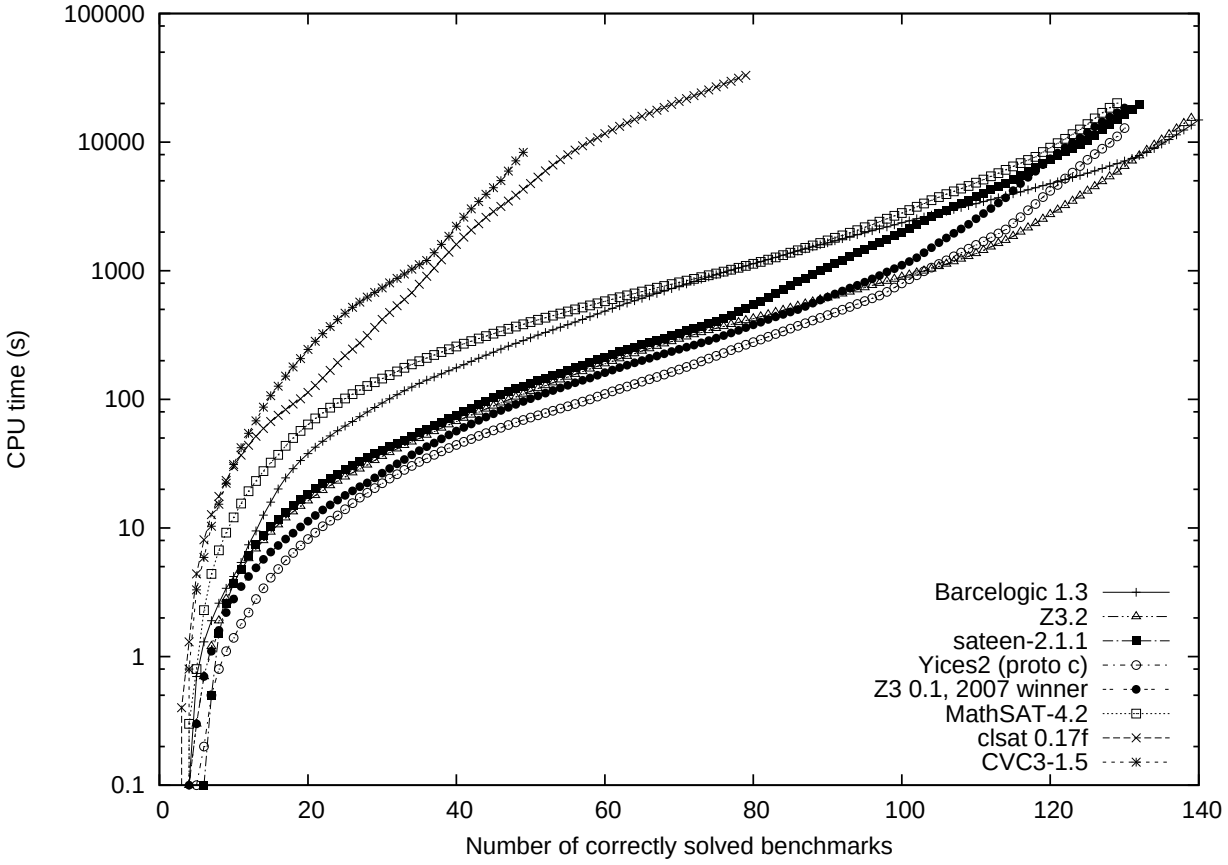


Figure 4: Benchmark comparisons of (above) the top two contenders in the QF_RDL division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Barcelogic 1.3	140	14895.1	78	62	0	63	0
Z3.2	139	15359.4	76	63	0	64	0
sateen-2.1.1	132	19487.0	72	60	0	71	0
Yices2 (proto c)	130	12854.4	67	63	0	73	0
MathSAT-4.2	129	20095.2	68	61	0	74	0
clsat 0.17f	79	33050.8	47	32	2	122	0
CVC3-1.5	49	8339.2	34	15	0	154	0
Z3 0.1, 2007 winner	130	18436.8	70	60	0	73	0

Figure 5: Results in the QF_IDL division.

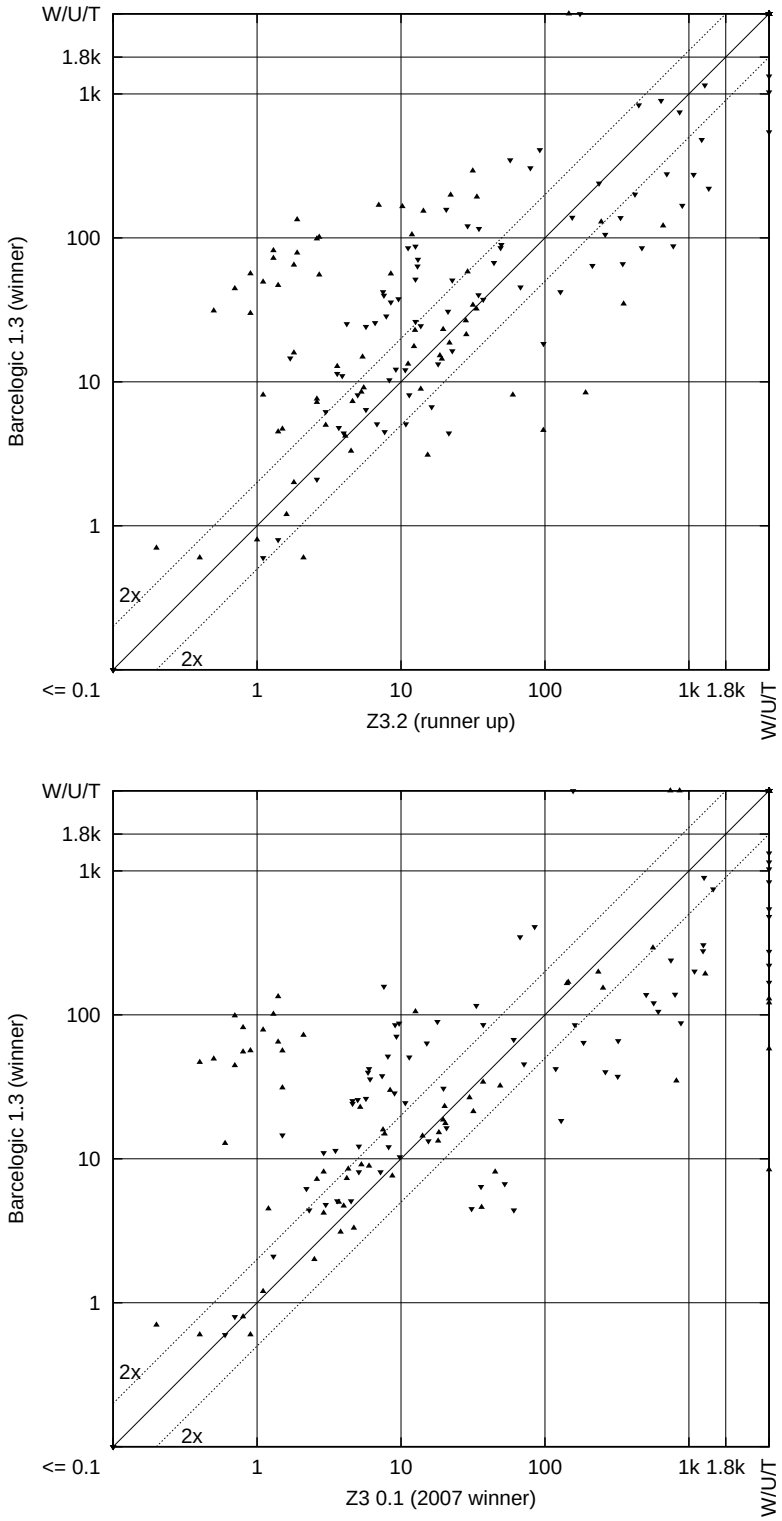
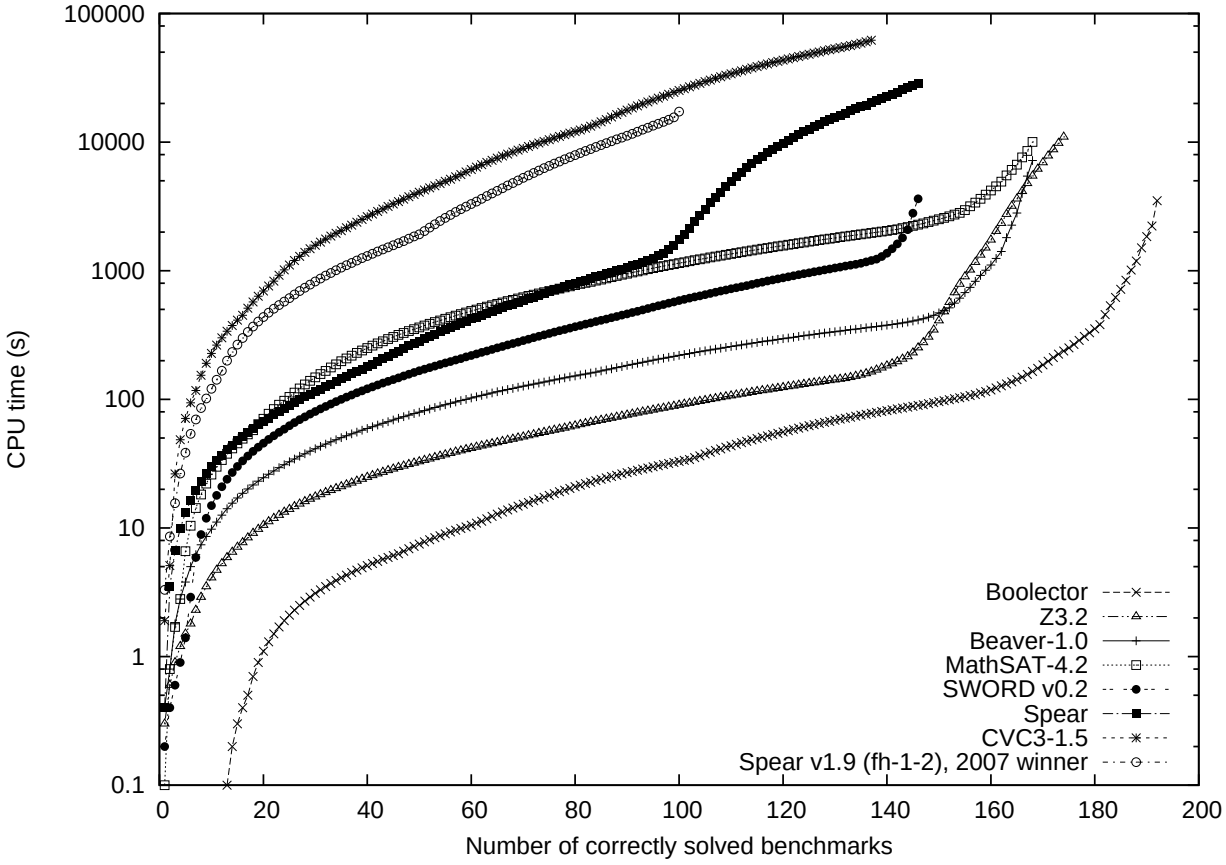


Figure 6: Benchmark comparisons of (above) the top two contenders in the QF_IDL division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Boolector	192	3502.5	85	107	0	8	0
Z3.2	174	10971.2	72	102	4	22	0
Beaver-1.0	168	7186.4	67	101	20	12	0
MathSAT-4.2	168	10034.9	62	106	2	30	0
SWORD v0.2	146	3630.3	47	99	2	52	0
Spear	146	28801.1	49	97	54	0	0
CVC3-1.5	137	61943.0	41	96	14	49	0
Spear v1.9 (fh-1-2), 2007 winner	100	17289.3	47	53	46	54	0

Figure 7: Results in the QF_BV division.

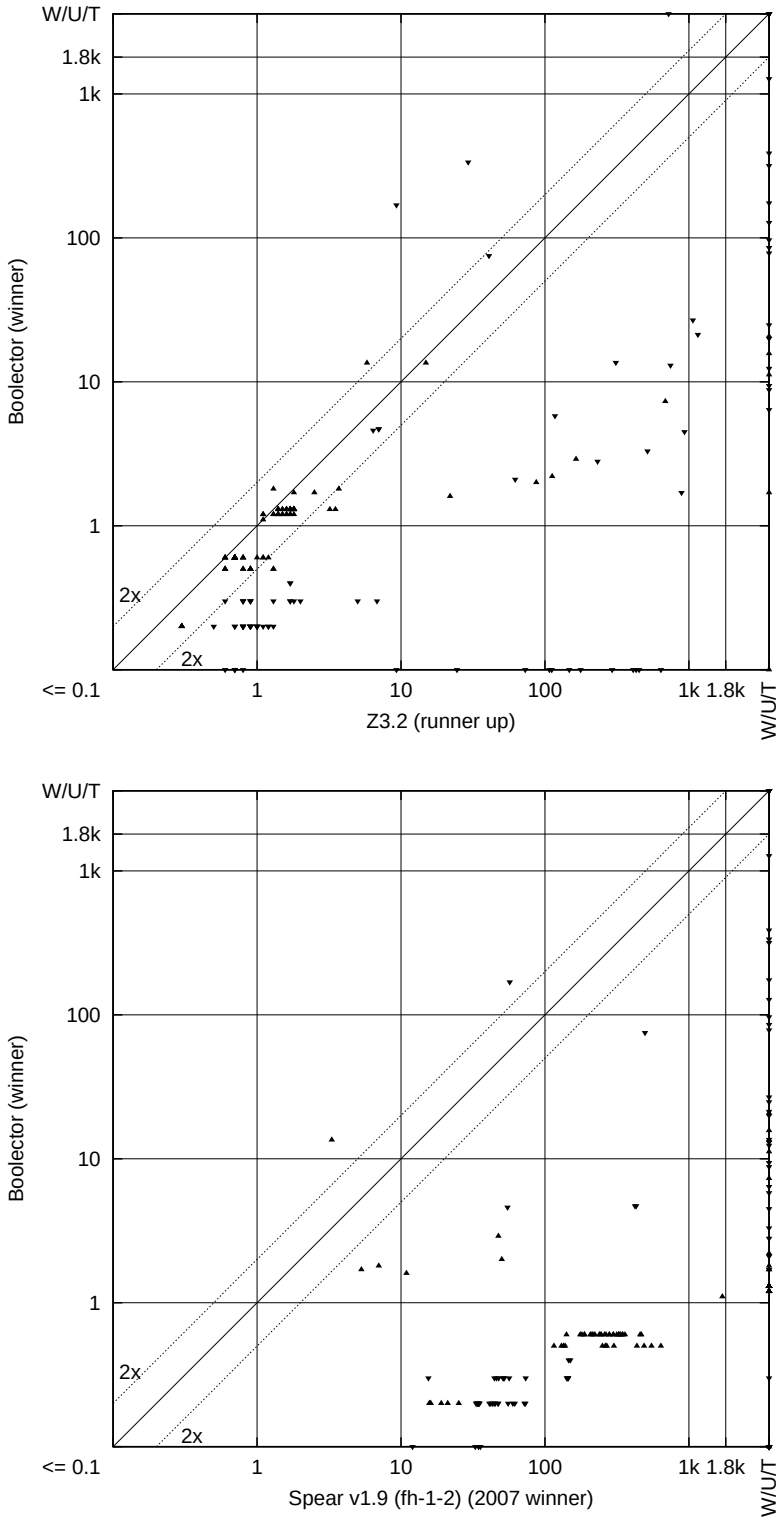
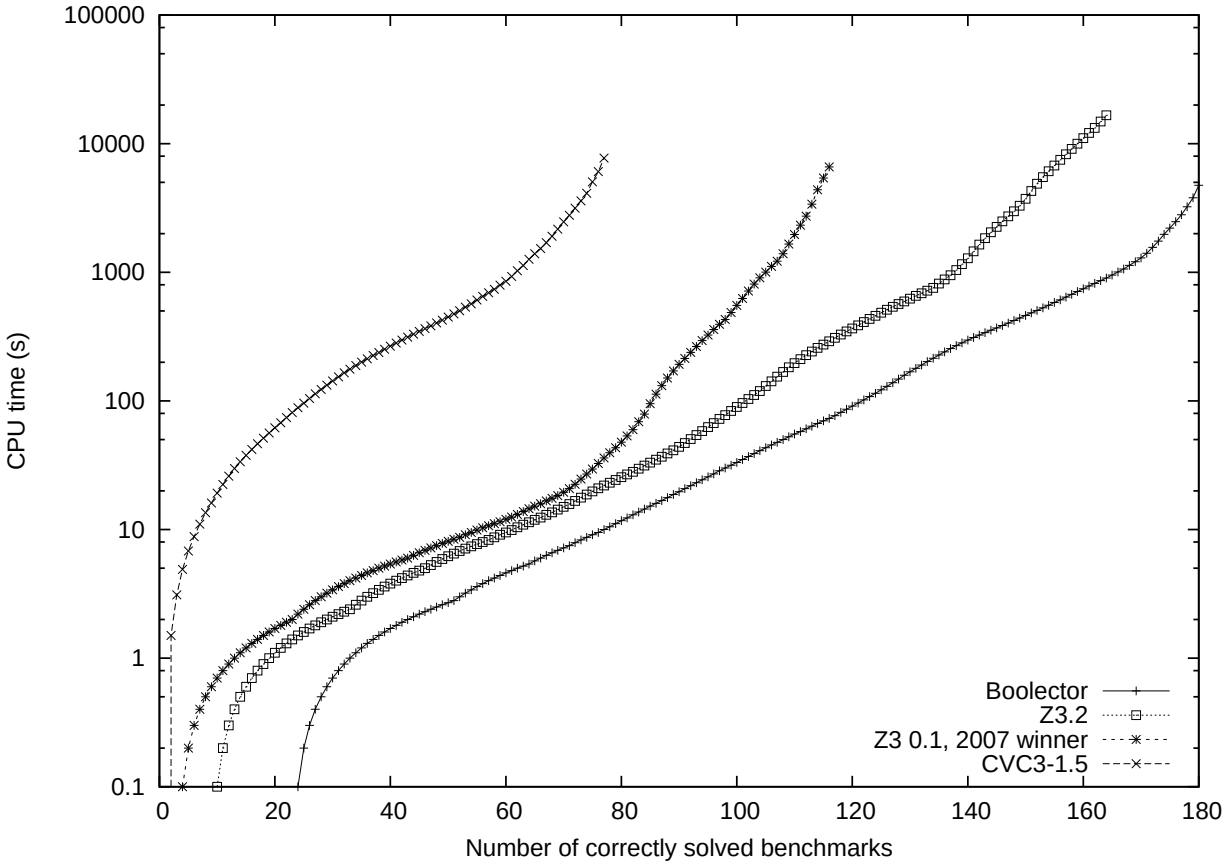


Figure 8: Benchmark comparisons of (above) the top two contenders in the QF_BV division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Boolector	180	4752.8	88	92	0	20	0
Z3.2	164	16650.4	79	85	3	33	0
CVC3-1.5	77	7728.9	47	30	69	54	0
Z3 0.1, 2007 winner	116	6589.6	60	56	4	80	0

Figure 9: Results in the QF_AUFBV division.

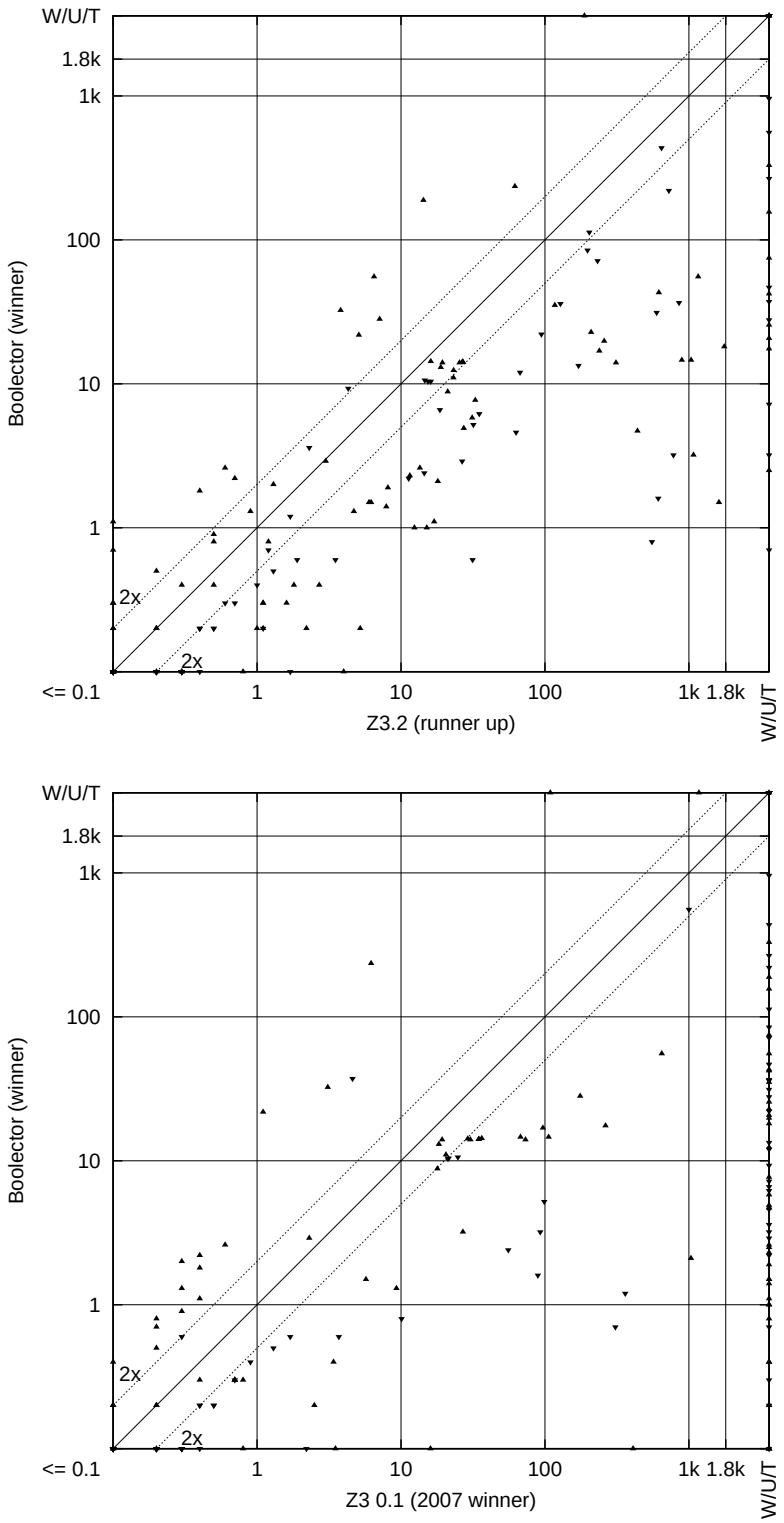
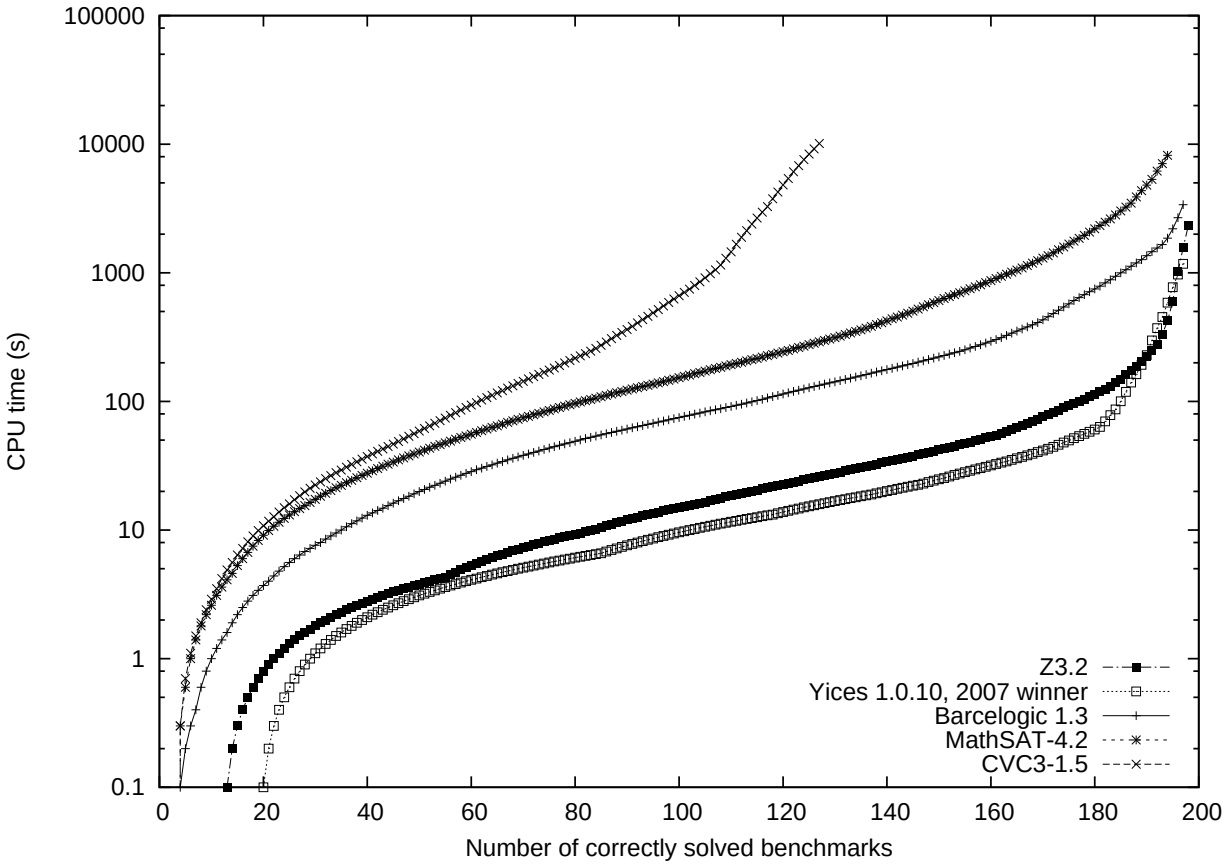


Figure 10: Benchmark comparisons of (above) the top two contenders in the QF_AUFBV division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	198	2334.4	149	49	0	5	0
Barcelogic 1.3	197	3385.5	148	49	0	6	0
MathSAT-4.2	194	8210.2	146	48	0	9	0
CVC3-1.5	127	10140.5	84	43	0	76	0
Yices 1.0.10, 2007 winner	197	1173.3	148	49	0	6	0

Figure 11: Results in the QF_UFIDL division.

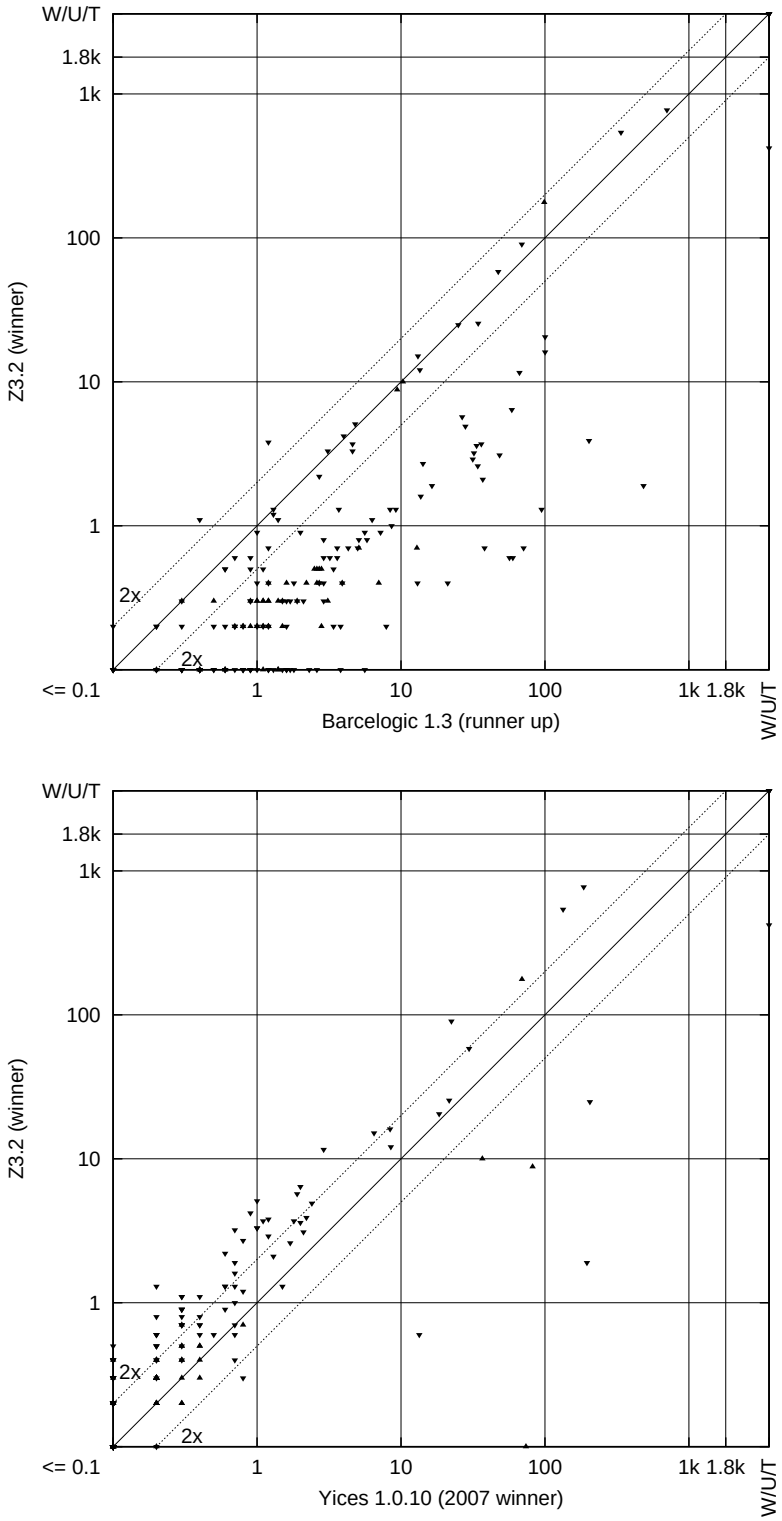
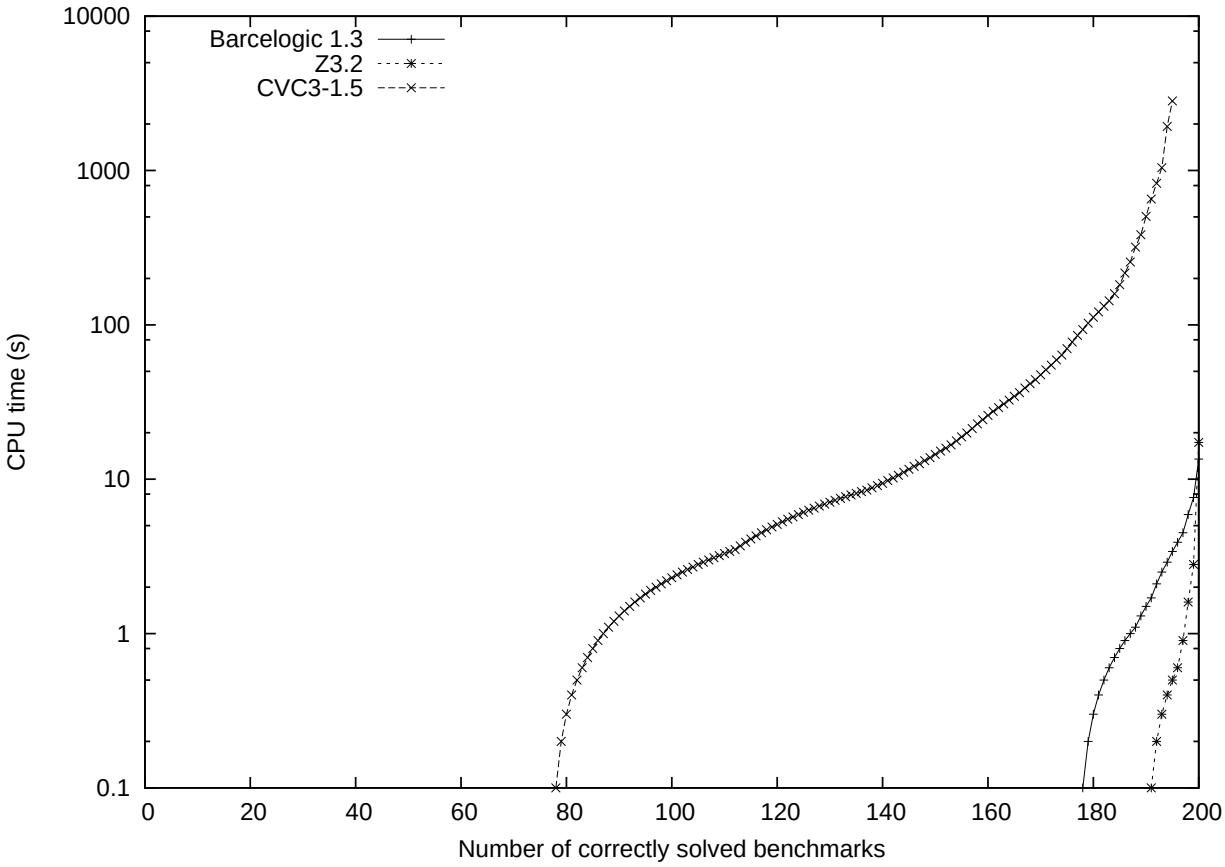


Figure 12: Benchmark comparisons of (above) the top two contenders in the QF_UFIDL division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Barcelogic 1.3	200	13.5	100	100	0	0	0
Z3.2	200	17.3	100	100	0	0	0
CVC3-1.5	195	2820.2	95	100	4	1	0

Figure 13: Results in the QF_AX division.

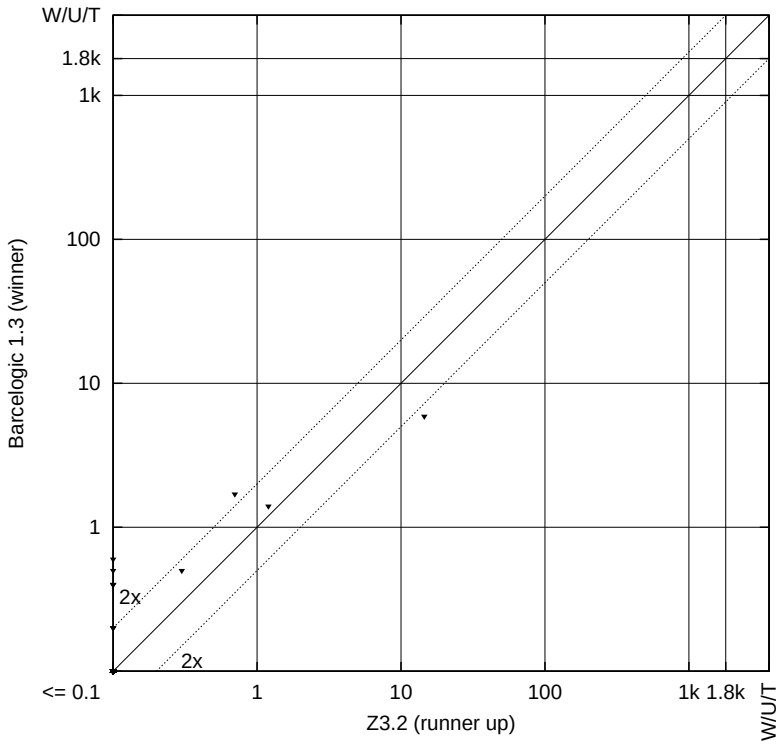
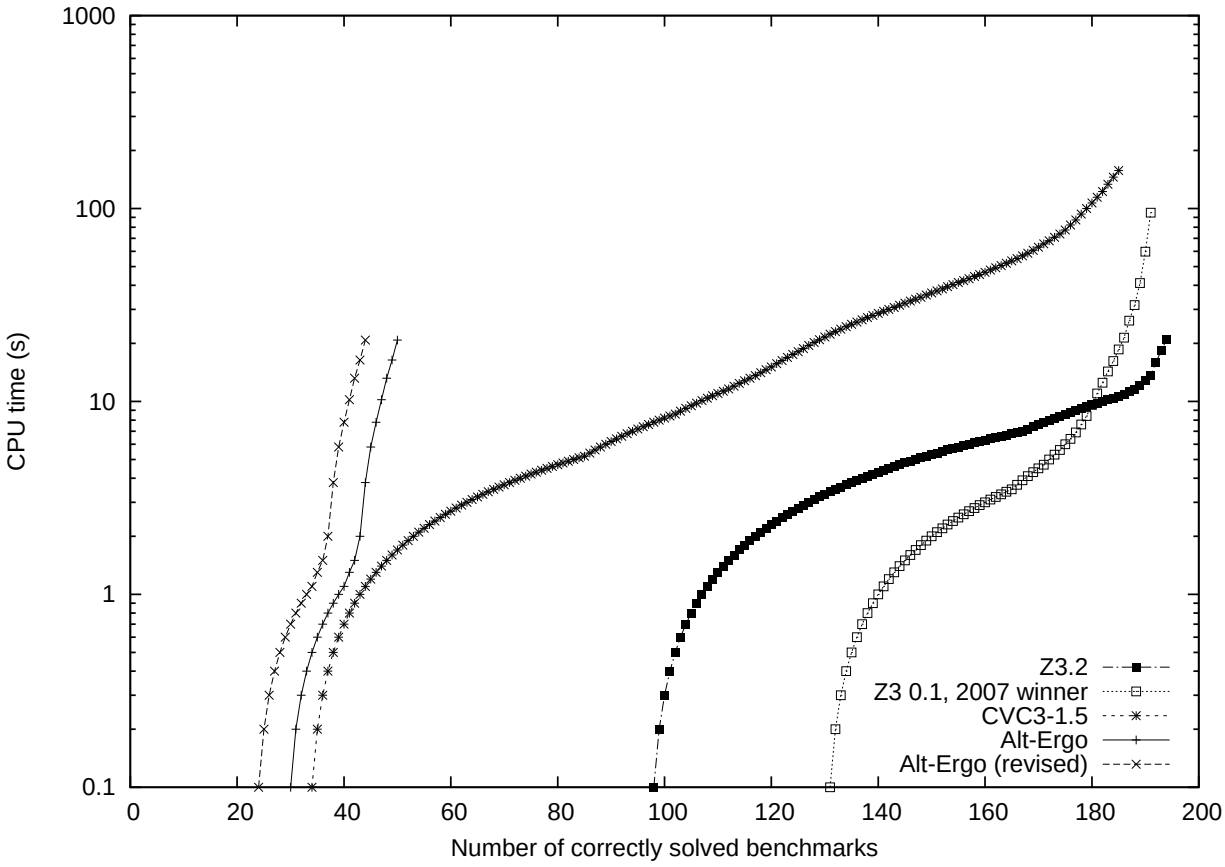


Figure 14: A benchmark comparison of the top two contenders in the QF_AX division. This division is new in this year's competition.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	194	20.9	192	2	5	2	0
CVC3-1.5	185	157.4	185	0	9	7	0
Z3 0.1, 2007 winner	191	95.2	191	0	5	5	0
Alt-Ergo (revised)	44	20.8	44	0	30	127	0
Alt-Ergo *disqualified*	-86	21.0	44 (17)	6	7	127	17

Figure 15: Results in the AUFLIA+p division.

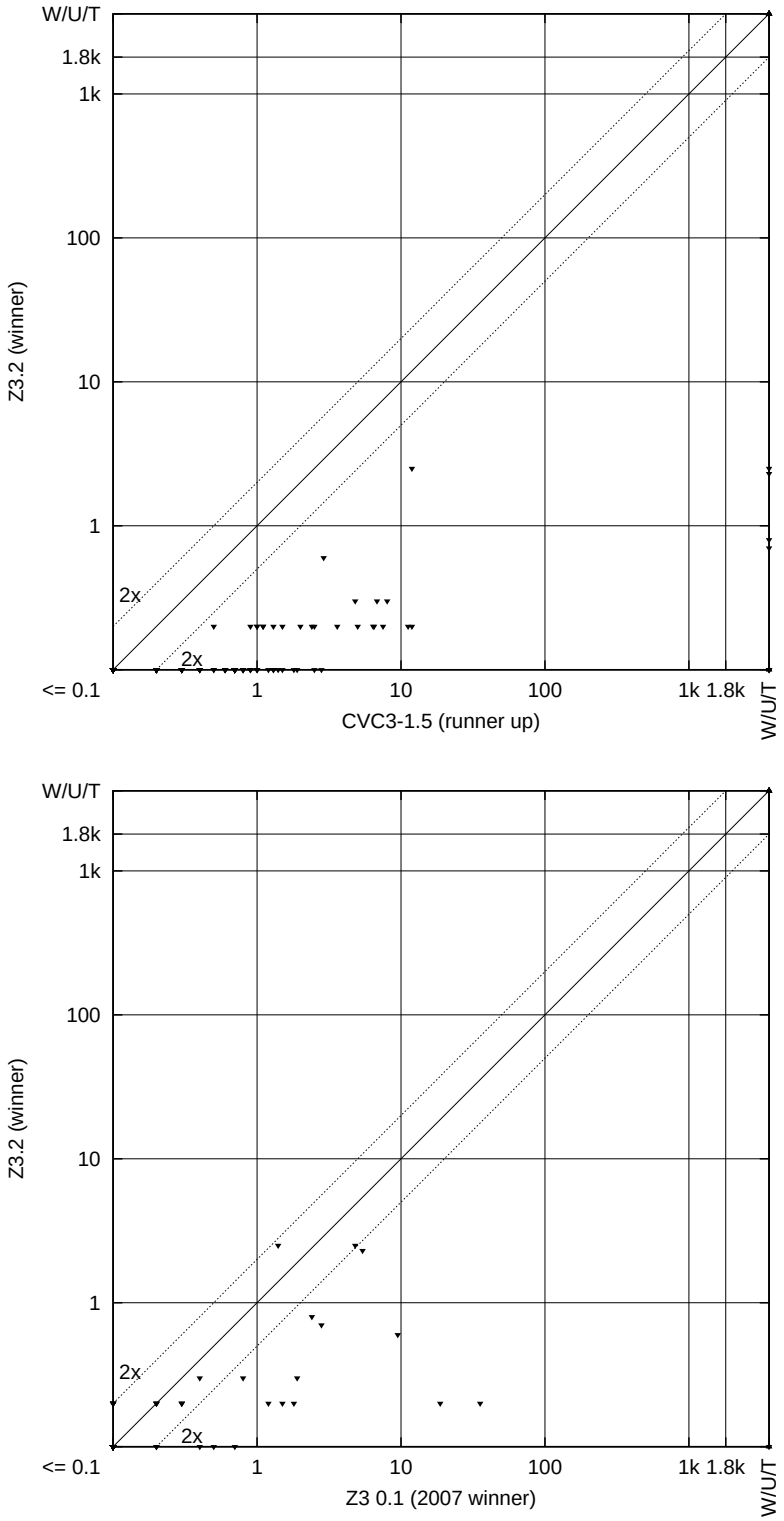
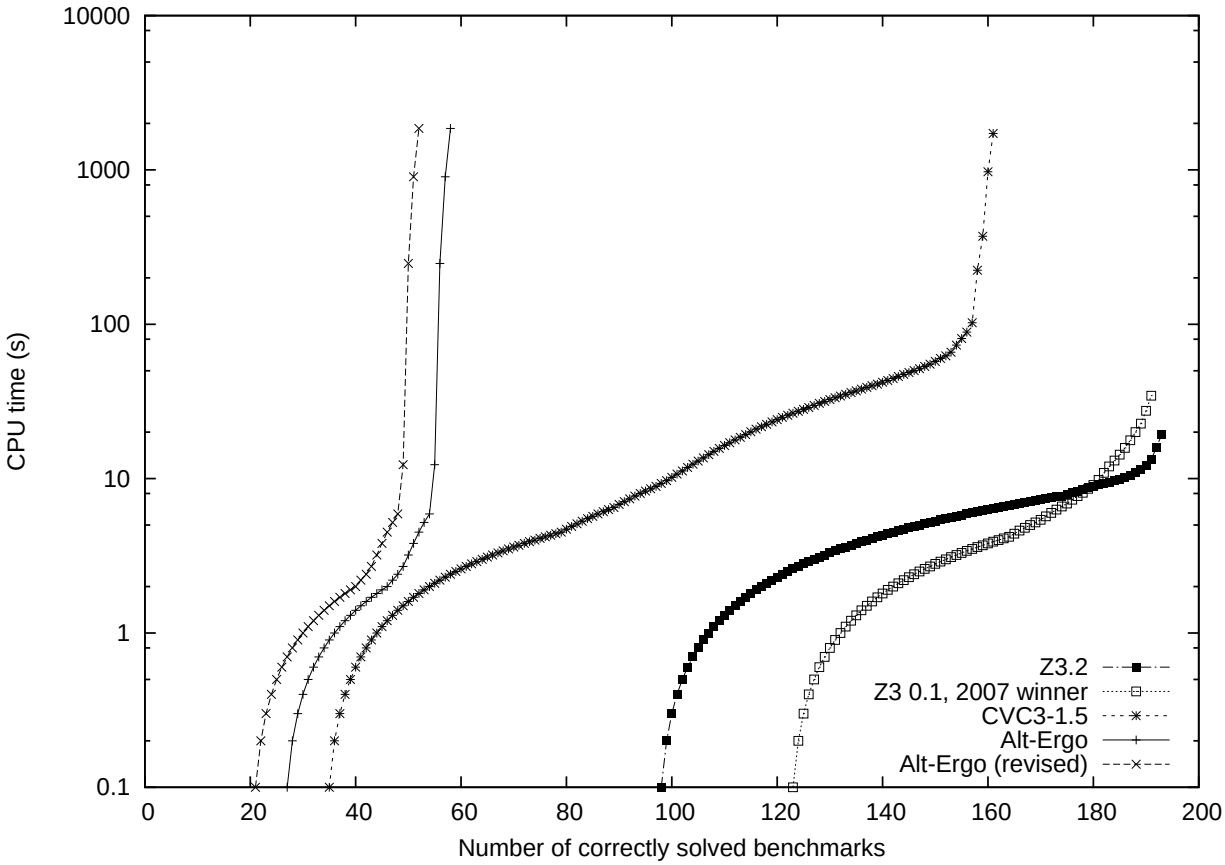


Figure 16: Benchmark comparisons of (above) the top two contenders in the AUFLIA+p division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	193	19.4	191	2	6	2	0
CVC3-1.5	161	1721.0	161	0	9	31	0
Z3 0.1, 2007 winner	191	34.4	191	0	5	5	0
Alt-Ergo (revised)	52	1858.1	52	0	24	125	0
Alt-Ergo *disqualified*	-54	1858.1	52 (14)	6	4	125	14

Figure 17: Results in the AUFLIA$-p$ division.

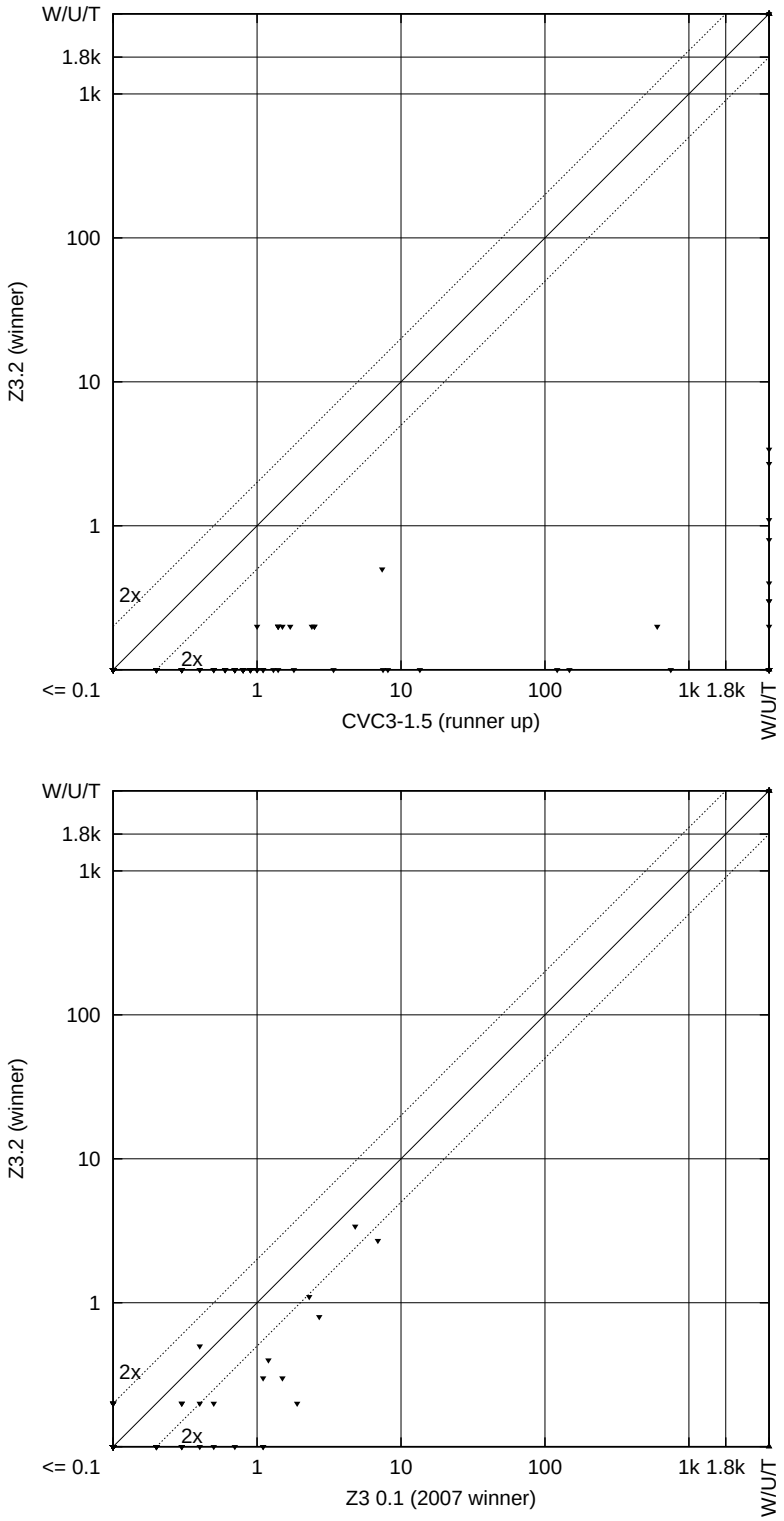
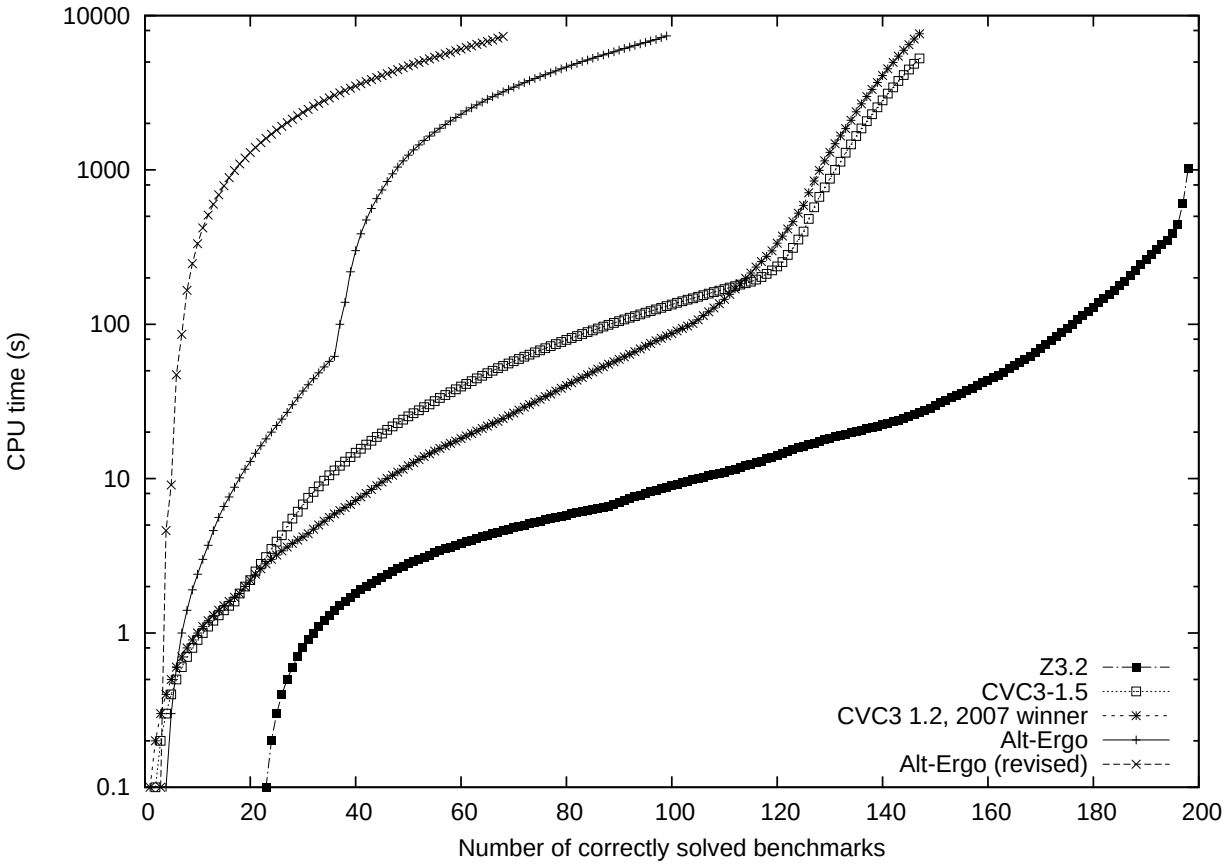


Figure 18: Benchmark comparisons of (above) the top two contenders in the AUFLIA$-p$ division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	198	1018.0	168	30	1	1	0
CVC3-1.5	147	5275.6	147	0	46	7	0
CVC3 1.2, 2007 winner	147	7619.1	147	0	34	19	0
Alt-Ergo (revised)	68	7334.5	68	0	83	49	0
Alt-Ergo *disqualified*	-317	7461.3	68 (52)	31	0	49	52

Figure 19: Results in the AUFLIRA division.

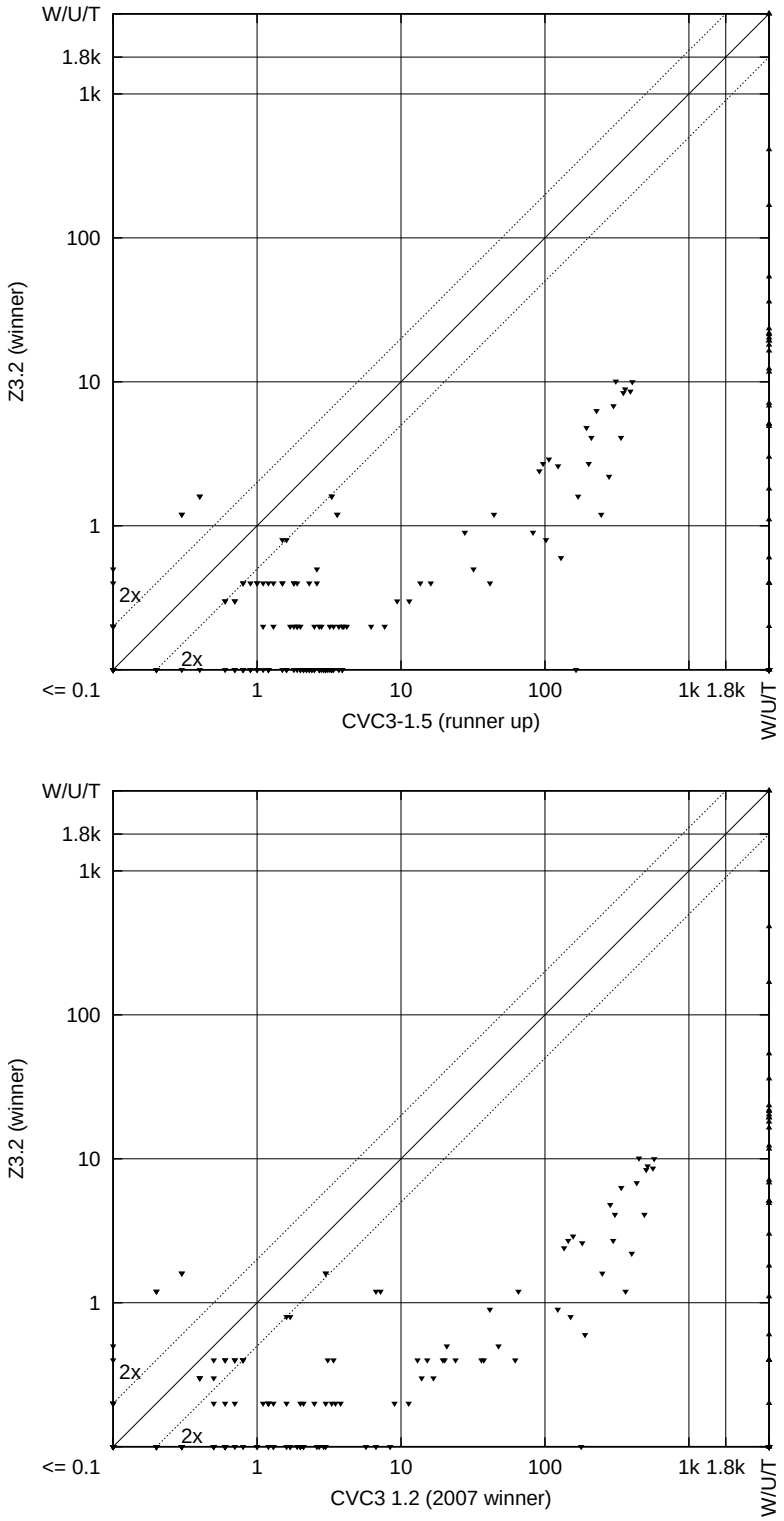
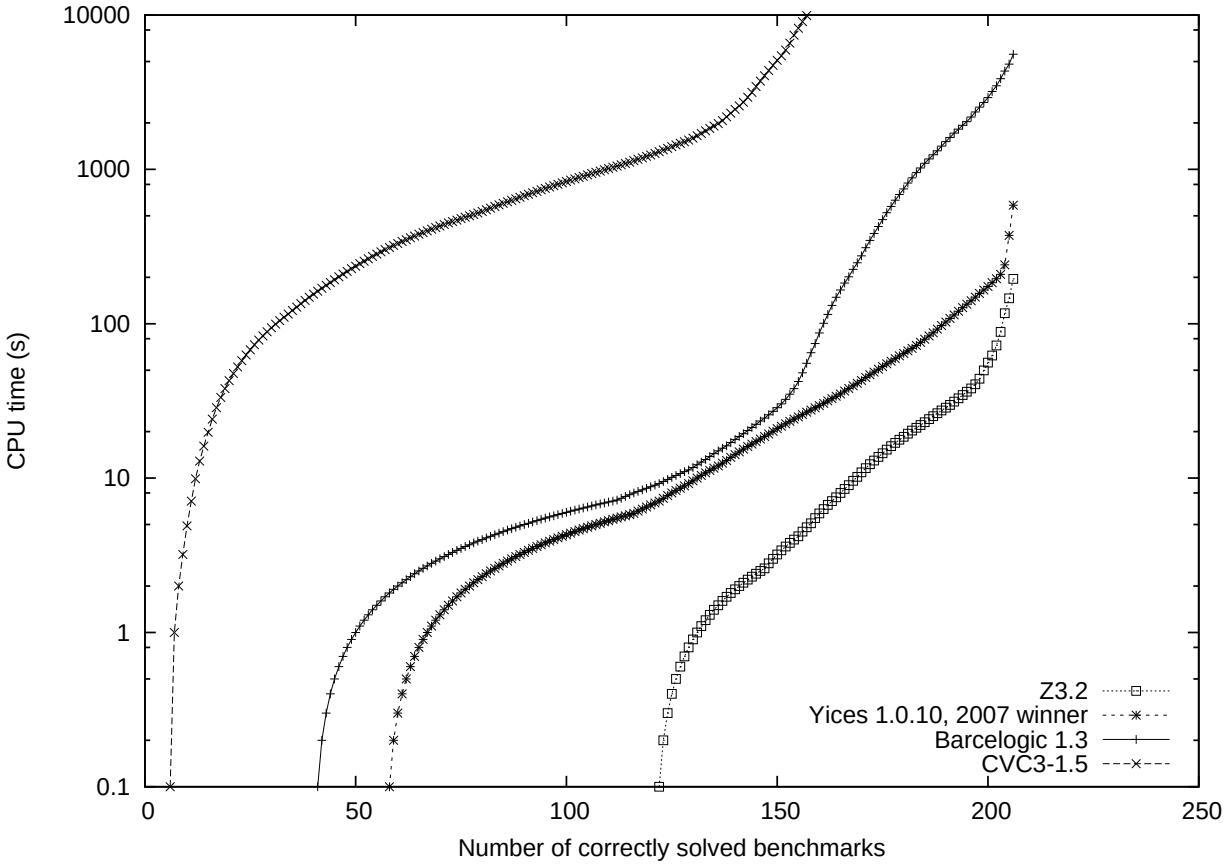


Figure 20: Benchmark comparisons of (above) the top two contenders in the AUFLIRA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	206	195.3	107	99	0	0	0
Barcelogic 1.3	206	5572.5	107	99	0	0	0
CVC3-1.5	157	9930.7	80	77	35	14	0
Yices 1.0.10, 2007 winner	206	585.8	107	99	0	0	0

Figure 21: Results in the QF_AUFLIA division.

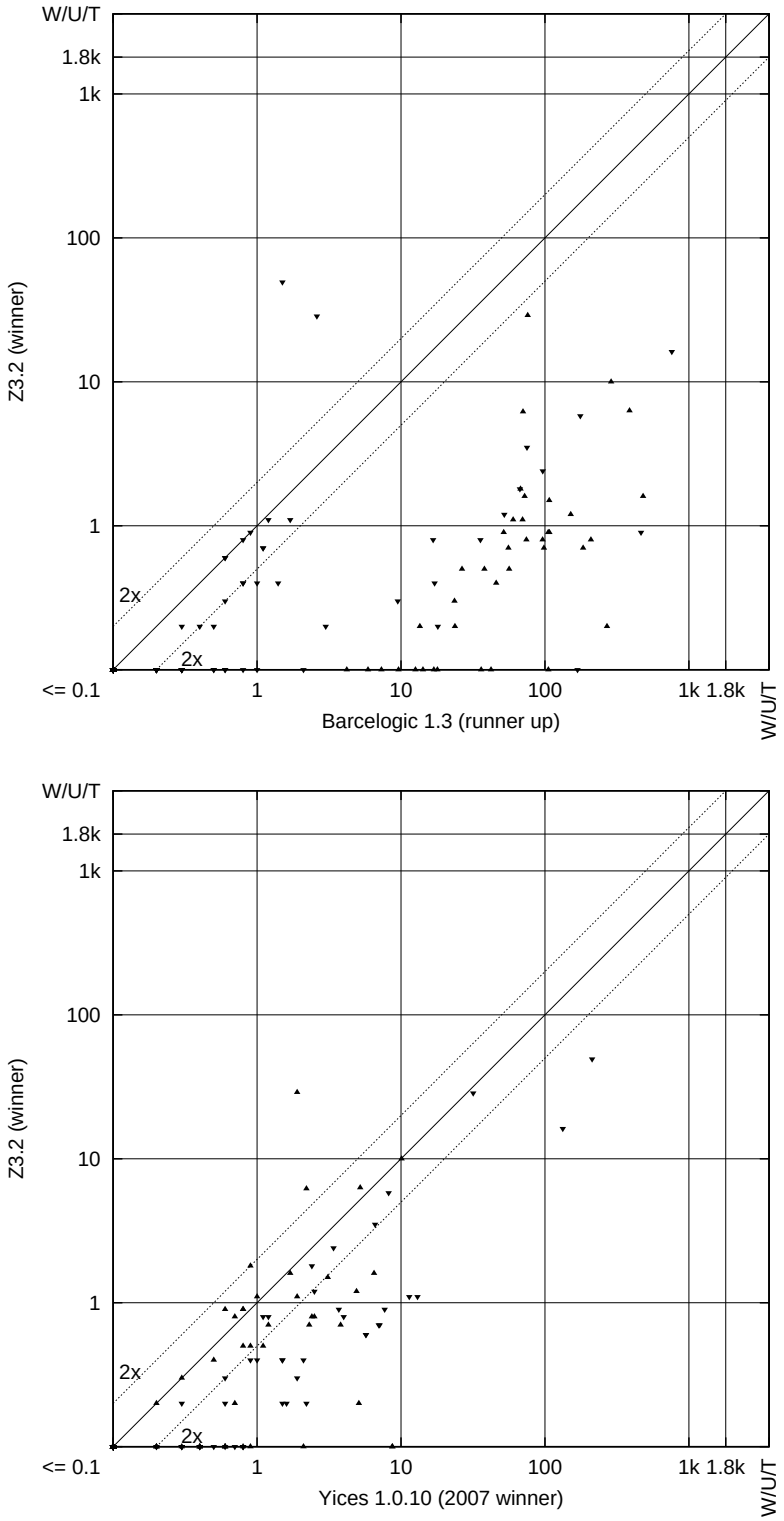
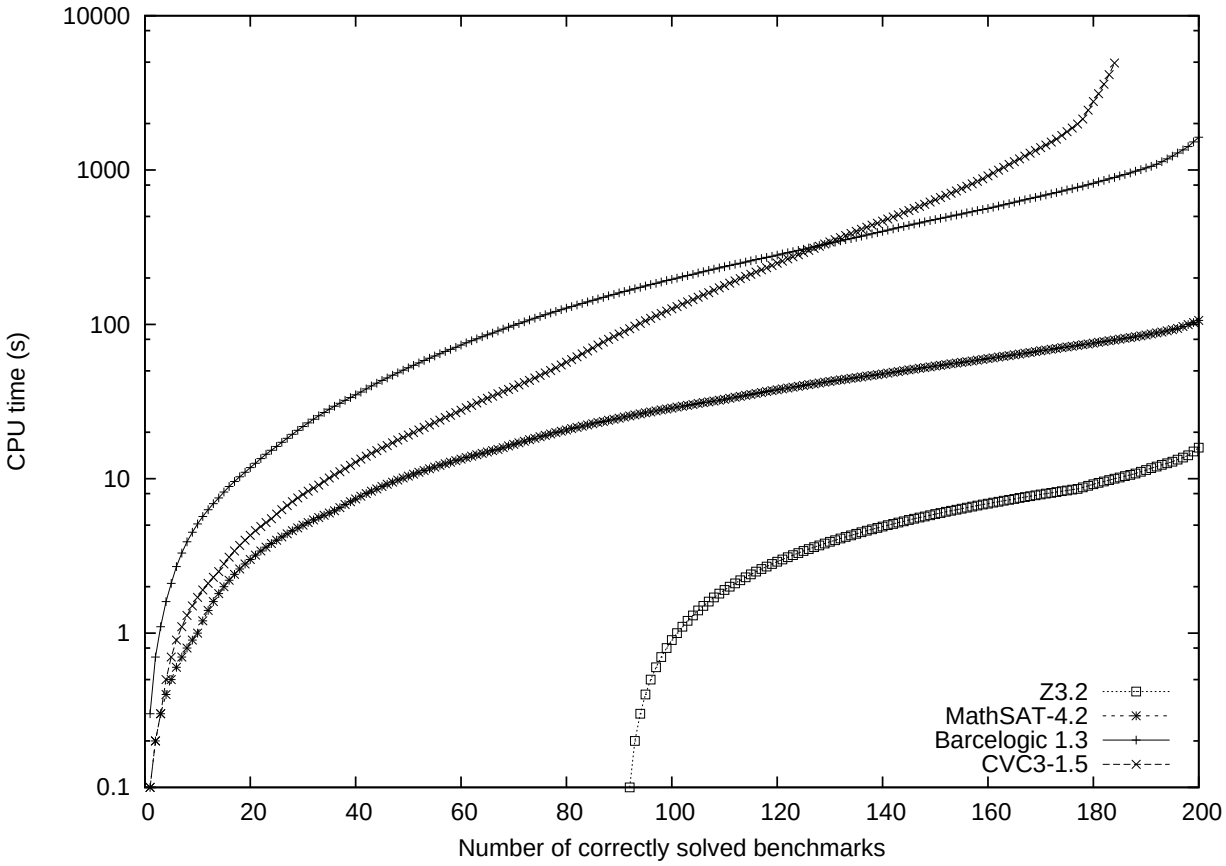


Figure 22: Benchmark comparisons of (above) the top two contenders in the QF_AUFLIA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	200	15.9	82	118	0	0	0
MathSAT-4.2	200	106.1	82	118	0	0	0
Barcelogic 1.3	200	1630.2	82	118	0	0	0
CVC3-1.5	184	4941.7	80	104	11	5	0

Figure 23: Results in the QF_UFLRA division.

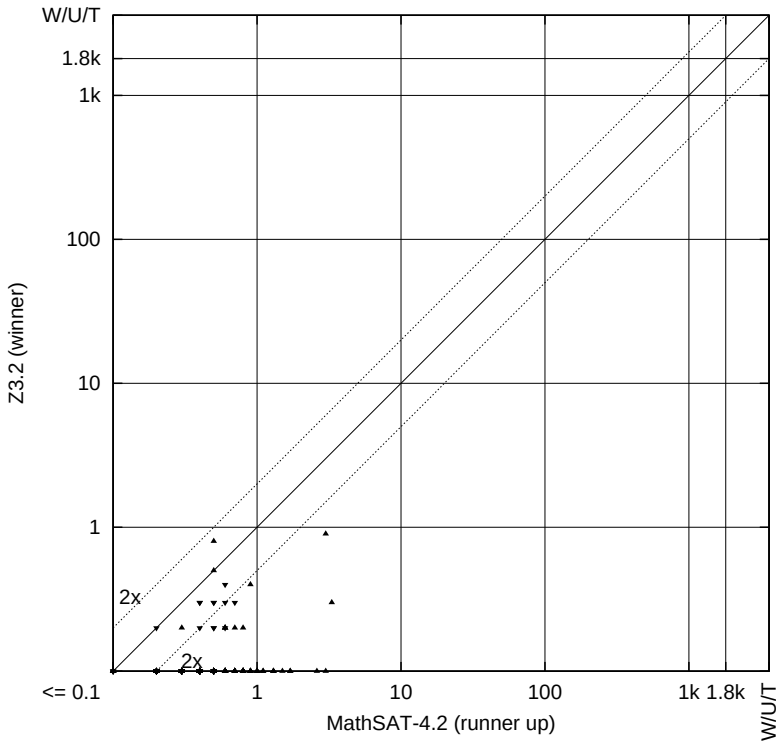
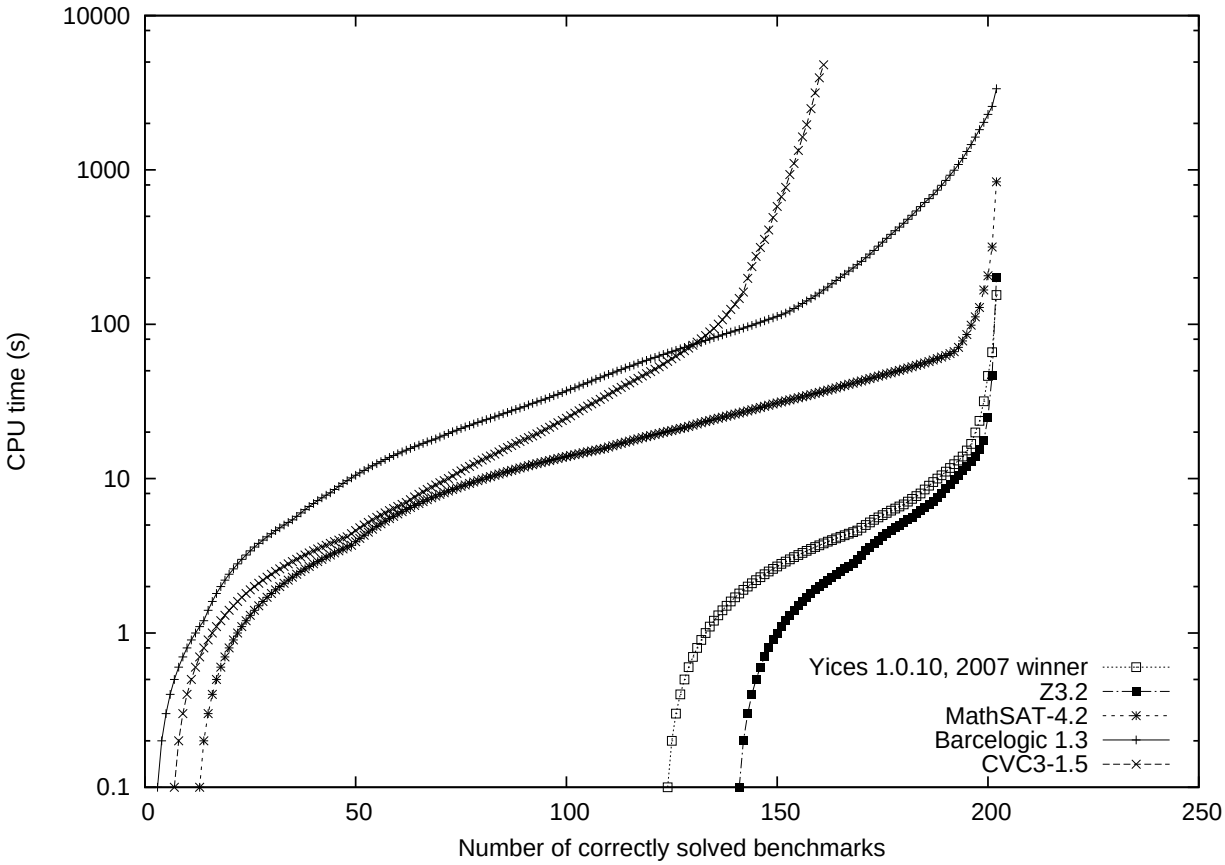


Figure 24: A benchmark comparison of the top two contenders in the QF_UFLRA division. This division is new in this year's competition.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	202	201.0	63	139	0	0	0
MathSAT-4.2	202	836.6	63	139	0	0	0
Barcelogic 1.3	202	3353.5	63	139	0	0	0
CVC3-1.5	161	4793.3	56	105	0	41	0
Yices 1.0.10, 2007 winner	202	154.7	63	139	0	0	0

Figure 25: Results in the QF_UFLIA division.

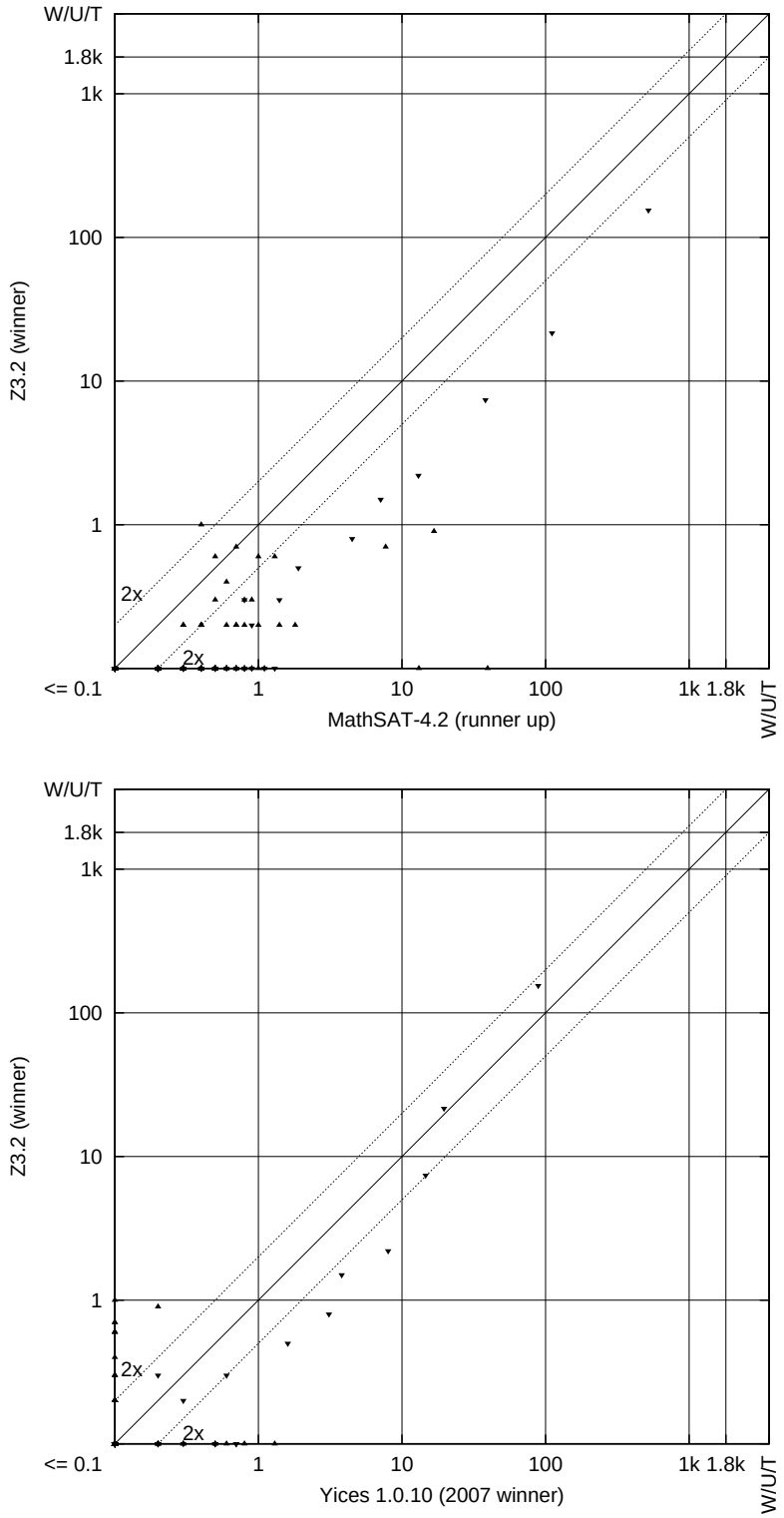
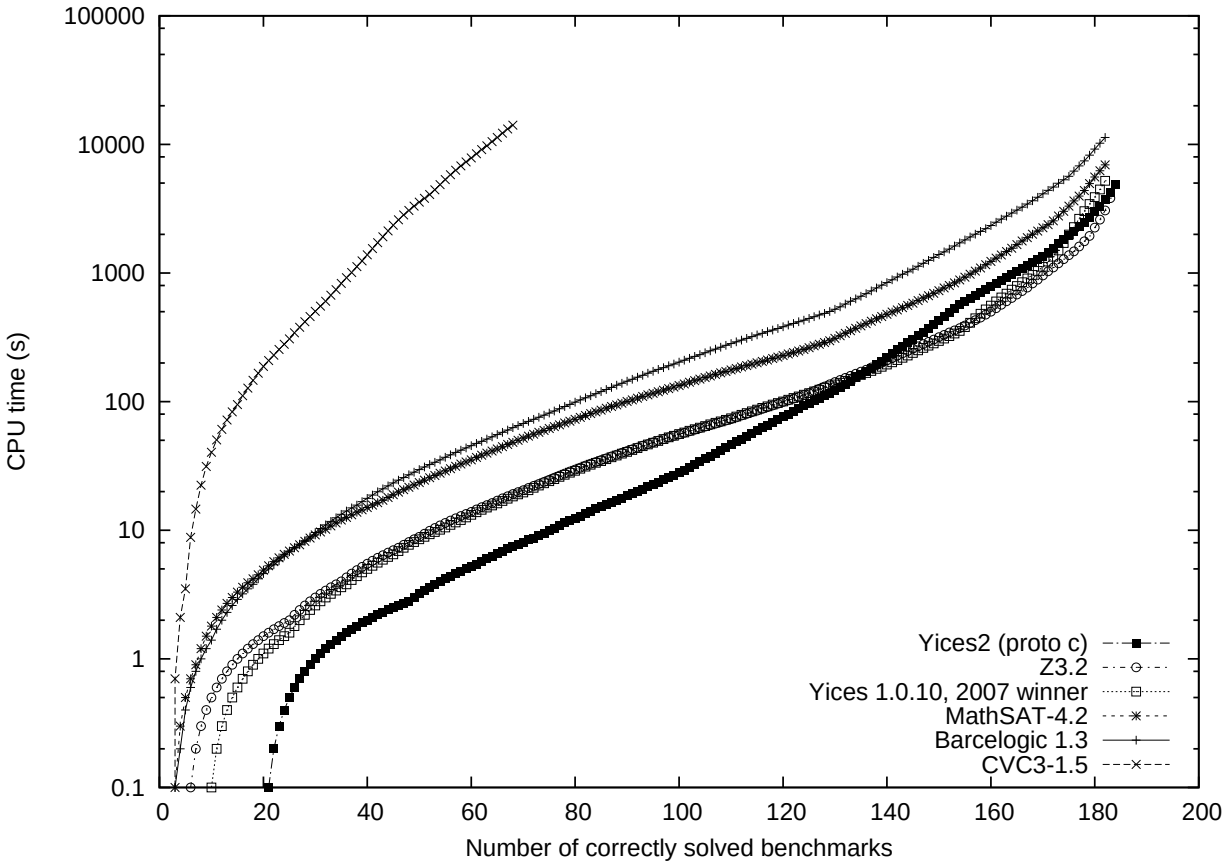


Figure 26: Benchmark comparisons of (above) the top two contenders in the QF_UFLIA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Yices2 (proto c)	184	4899.1	90	94	0	18	0
Z3.2	183	3817.6	91	92	0	19	0
MathSAT-4.2	182	6941.6	91	91	0	20	0
Barcelogic 1.3	182	11316.7	90	92	0	20	0
CVC3-1.5	68	14121.1	25	43	14	120	0
Yices 1.0.10, 2007 winner	182	5214.5	90	92	0	20	0

Figure 27: Results in the QF_LRA division.

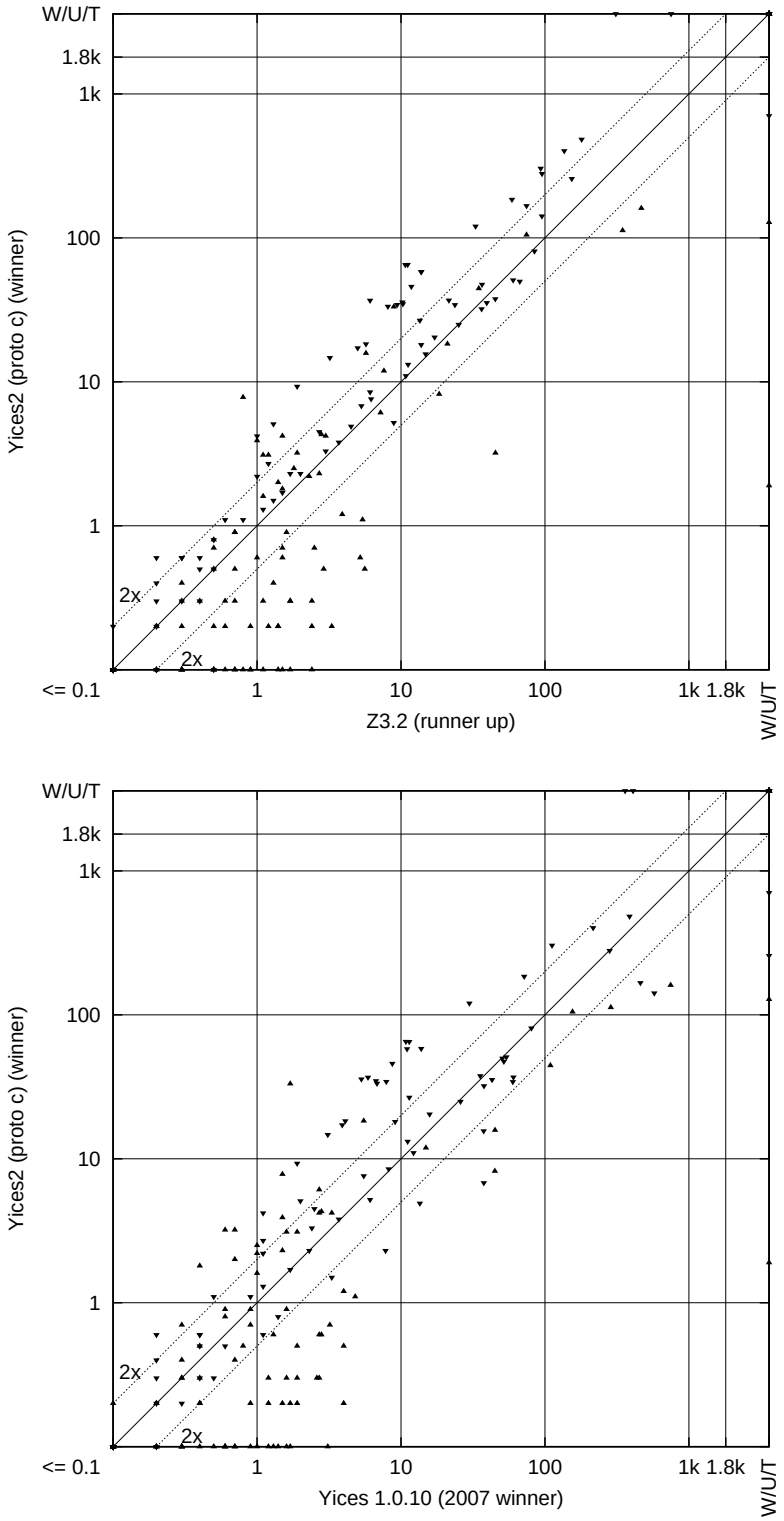
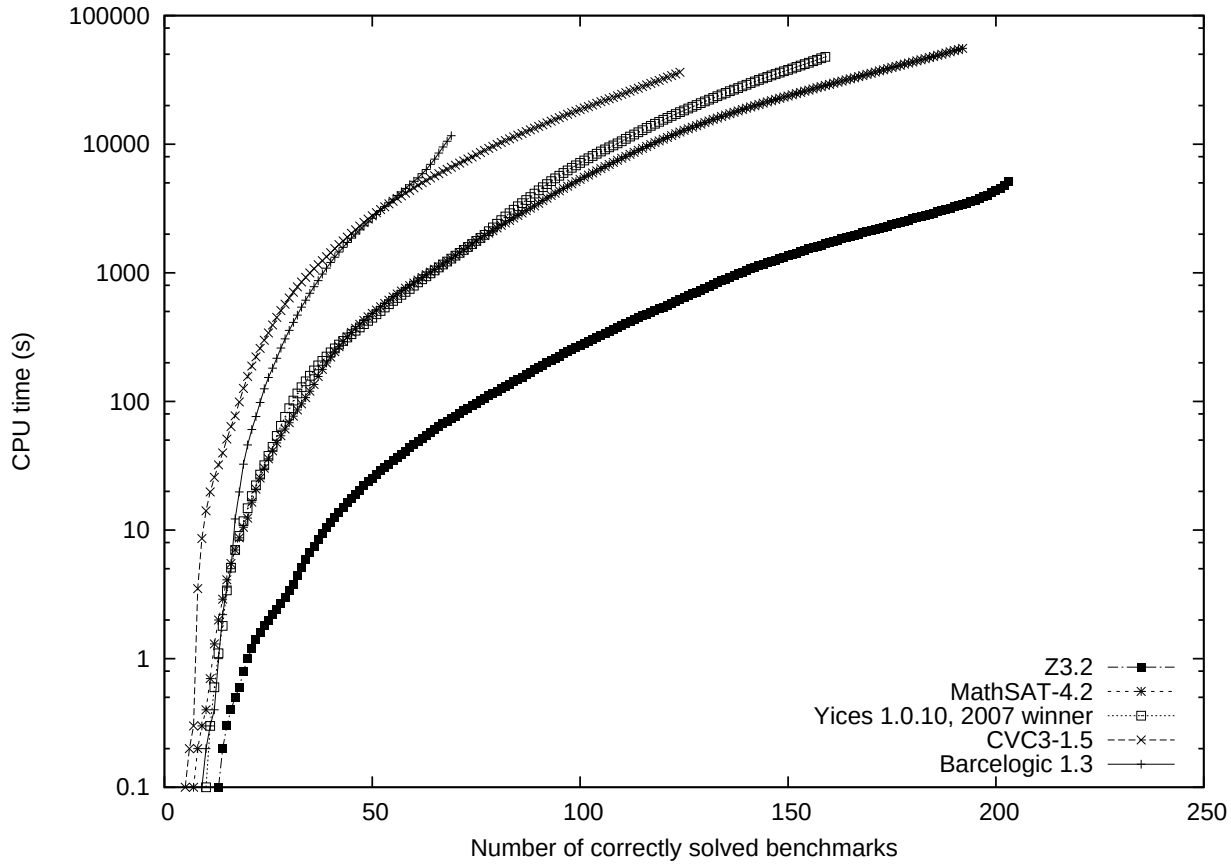


Figure 28: Benchmark comparisons of (above) the top two contenders in the QF_LRA division this year, and (below) last year's and this year's winners.



Solver	Score	Time (s)	Unsat	Sat	Unknown	Timeout	Wrong
Z3.2	203	5104.5	112	91	0	2	0
MathSAT-4.2	192	55673.3	107	85	0	13	0
CVC3-1.5	124	36294.4	64	60	18	63	0
Barcelogic 1.3	69	11636.0	46	23	0	136	0
Yices 1.0.10, 2007 winner	159	47754.8	81	78	0	46	0

Figure 29: Results in the QF_LIA division.

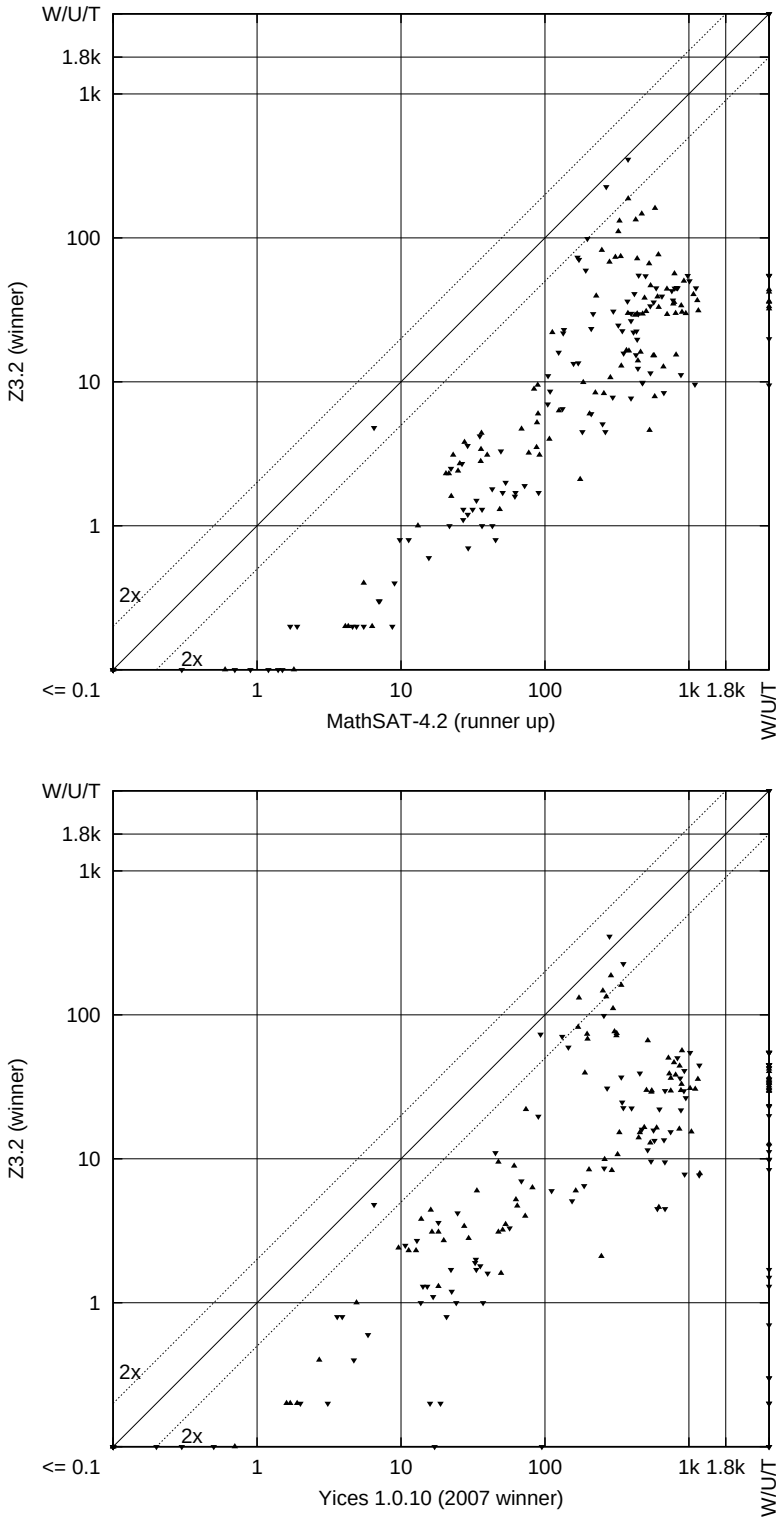


Figure 30: Benchmark comparisons of (above) the top two contenders in the QF_LIA division this year, and (below) last year's and this year's winners.