The CADE-20 Automated Theorem Proving Competition

Geoff Sutcliffe
Department of Computer Science
University of Miami
USA
E-mail: geoff@cs.miami.edu

The CADE ATP System Competition (CASC) is an annual evaluation of fully automatic, first order Automated Theorem Proving systems. CASC-20 was the tenth competition in the CASC series. Seventeen ATP systems and system variants competed in the various competition and demonstration divisions. An outline of the competition design, and a commentated summary of the results, are presented.

Keywords: competition, automated theorem proving

1. Introduction

The CADE ATP System Competition (CASC) is an annual evaluation of fully automatic, first order Automated Theorem Proving (ATP) systems – the world championship for such systems. The primary purpose of CASC is a public evaluation of the relative capabilities of ATP systems. Additionally, CASC aims to stimulate ATP research in general, to stimulate ATP research towards autonomous systems, to motivate implementation of robust ATP systems, to provide an inspiring environment for personal interaction between ATP researchers, and to expose ATP systems within and beyond the ATP community. Fulfillment of these objectives provides stimulus and insight for the development of more powerful ATP systems, leading to increased and more effective usage.

CASC-20 was held on 26th July 2005, as part of the 20th International Conference on Automated Deduction (CADE-20). CASC-20 was the tenth competition in the CASC series; see [14] and citations therein for information about the individual previous competitions, and [15] for a historical overview and analysis of the first ten CASCs. Seventeen ATP systems and system variants, listed in Table 1, competed in the various competition and demonstration divisions. The division winners of CASC-J2 (the previous CASC) were automatically entered into CASC-20 to provide benchmarks against which progress can be judged.[1] Details of the CASC-20 design, and system descriptions for the entered systems, are in [13] and on the CASC-20 WWW site. The WWW site also provides access to all systems and competition resources: `http://www.tptp.org/CASC/20`

CASC-20 was organized by Geoff Sutcliffe, and was overseen by a panel consisting of Uli Furbach, Roberto Nieuwenhuis, and John Slaney. The competition computers were provided by the Department of Computer Science at the University of Manchester.

This paper is organized as follows: Sections 2 and 3 describe the divisions and organization of CASC-20. Section 4 provides a commentated summary of the results. Short descriptions of the winning systems are given in Section 5. Section 6 concludes and discusses plans for future CASCs.

2. Divisions

CASC is run in divisions according to system and problem characteristics. There are *competition divisions* in which systems are explicitly ranked, and a *demonstration division* in which systems demonstrate their abilities without being formally ranked.

Each competition division uses problems that have certain logical, language, and syntactic characteristics, so that the ATP systems that com-

[1] Gandalf c-2.6-SAT, the CASC-J2 SAT division assurance class winner, was withdrawn from CASC-20 after unsoundness was detected in testing against new problems in TPTP v3.1.0. Paradox 1.0, the CASC-J2 SAT division model class winner had solved only one less problem, and thus provided an adequate benchmark for CASC-20.

AI Communications
ISSN 0921-7126, IOS Press.

Table 1
The ATP systems and entrants

ATP System	Divisions	Entrants	Affiliation
Darwin 1.2	MIX SAT* EPR	Alexander Fuchs, Peter Baumgartner, Cesare Tinelli	University of Iowa, Max-Planck-Institut für Informatik, University of Iowa
DCTP 10.21p	EPR	CASC	*CASC-J2 EPR winner*
E 0.9pre3	MIX FOF SAT EPR UEQ	Stephan Schulz	Technische Universität München
EP 0.9pre3	MIX* FOF*		*E 0.9pre variant*
Mace2 2.2	SAT*	William McCune	Argonne National Laboratory
Mace4 0705D	SAT*	William McCune	Argonne National Laboratory
MathServ 0.62	MIX FOF SAT EPR UEQ (demo)	Jürgen Zimmer Serge Autexier	Universität des Saarlandes
MUSCADET 2.5	FOF	Dominique Pastre	Université René Descartes - Paris
Octopus JN05	MIX FOF (demo)	Monty Newborn, Zongyan Wang	McGill University
Otter 3.3	MIX* FOF UEQ	William McCune	Argonne National Laboratory
Paradox 1.0	SAT*	CASC	*CASC-J2 SAT winner*
Paradox 1.3	SAT*	Koen Claessen, Niklas Sörensson	Chalmers University of Technology
Prover9 0705	MIX* FOF* UEQ	William McCune	Argonne National Laboratory
THEO JN05	MIX (comp) FOF (demo)	Monty Newborn	McGill University
Vampire 7.0	MIX* FOF*	CASC	*CASC-J2 MIX*, FOF* winner*
Vampire 8.0	MIX* FOF* EPR UEQ	Andrei Voronkov	University of Manchester
Waldmeister 704	UEQ	CASC	*CASC-J2 UEQ winner*

MIX* indicates participation in the MIX division proof class, FOF* indicates participation in the FOF division proof class, and SAT* indicates participation in the SAT division model class - see Section 2.

pete in the division are, in principle, able to attempt all the problems in the division. The systems are ranked according to the numbers of problems solved, with ties decided by average CPU times over problems solved. Some divisions are further divided into problem categories, which make it possible to analyze, at a more fine grained level, which systems work well for what types of problems. The categories have no effect on the competition rankings, which are made at only the division level. The demonstrate division uses the same problems as the competition divisions - he entry specifies which competition divisions' problems are to be used. The results for the demonstration division are presented along with the competition divisions' results, but may not be comparable with those results.

Table 2 shows the divisions and problem categories used in CASC-20. *Mixed* means Horn and non-Horn problems, with or without equality, but not unit equality problems (see the UEQ division), *really non-propositional* means with an infinite Herbrand universe (so that the problems cannot necessarily be solved by finite saturation methods), and *effectively propositional* means non-propositional with a finite Herbrand universe.

The MIX, FOF, and SAT divisions each had two ranking classes: an **assurance** class - ranked according to the number of problems solved (a "yes" output, giving an *assurance* of the existence of a proof/model), and a **proof/model** class - ranked according to the number of problems solved with an *acceptable proof/model* output to `stdout`. The competition panel judged whether or not each system's proof/model format is acceptable.

3. Organization

The CASC-20 competition divisions ran on 44 AMD Athlon XP 2200+ computers, each having a 1.8 GHz CPU, 512 MB memory, and the Linux 2.4 operating system. Demonstration division systems can run on the competition computers, or the com-

Table 2
Divisions and Problem categories

Division	Problems	Problem Categories
MIX	Mixed CNF really non-propositional theorems (unsatisfiable clause sets)	**HNE** - Horn with No Equality **HEQ** - Horn with some (not pure) Equality **NNE** - Non-Horn with No Equality **NEQ** - Non-Horn with some (not pure) Equality **PEQ** - Pure Equality
FOF	First-order form non-propositional theorems (axioms with a provable conjecture)	**FNE** - FOF with No Equality **FEQ** - FOF with Equality
SAT	CNF really non-propositional non-theorems (satisfiable clause sets)	**SNE** - SAT with No Equality **SEQ** - SAT with Equality
EPR	Effectively propositional CNF theorems and non-theorems (clause sets)	**EPT** - Effectively Propositional Theorems (unsatisfiable clause sets) **EPS** - Effectively Propositional non-theorems (Satisfiable clause sets)
UEQ	Unit equality CNF really non-propositional theorems (unsatisfiable clause sets)	—

puter(s) can be supplied by the entrant: MathServ ran on a computer with an Intel Xeon 2.8 GHz CPU, 2 GB memory, and the SuSE Linux 9.1 operating system. Octopus ran on a network of 152 computers, each having an Intel P3 or P4 CPU, at least 512 MB memory, and a version of either the FreeBSD or Linux operating system. THEO ran on the competition computers.

The problems were taken from the TPTP problem library [16], v3.1.0. The TPTP version used for the competition is not released until after the competition, so that new problems have not been seen by the entrants. Problems with a TPTP difficulty rating in the range 0.21 to 0.99 (solvable by some but not all state-of-the-art ATP systems [17]), and not tagged as "biased" (not designed specifically to be suited or ill-suited to some ATP system, calculus, or control strategy) by the TPTP, were eligible for use. In TPTP v3.1.0 there are very few EPR or FNE problems with ratings other than 0.00 and 1.00 (i.e., the problems can either be easily solved or are not solvable by current ATP systems). Therefore in the EPR division, and FNE category, problems with rating 0.00 to 0.20 were also made eligible for use. The problems used were randomly selected from the eligible problems at the start of the competition, based on a seed provided by the competition panel. The random selection was subject to a limitation on the number of very similar problems in each division and category [12], and biased to ensure (if possible) the selection of at least 50% new problems in each division and category. Problems with rating 0.00 were selected after problems with greater rating. The numbers of problems used in each division's problem categories were (roughly) proportional to the numbers of eligible problems, after taking into account the limitation on very similar problems. Table 3 gives the numbers of eligible problems, the maximal numbers that could be used after taking into account the limitation on very similar problems, and the numbers of problems used, in each division and category. In contrast to many previous CASCs, all the divisions and categories had sufficient eligible problems to use all the computing time available for the number of systems entered (in the EPR division and FNE category this was because of the eligibility of easy problems). The PEQ, FEQ, and SEQ categories had sufficient new problems to permit the bias to 50% new problems to be fulfilled, and the FNE and UEQ categories also had reasonable numbers of new problems. These numbers reflect the increase in equality and FOF problems in the TPTP, and correspondingly in the general use of ATP. To ensure that no system received an advantage or disadvantage due to the specific presentation of the problems in the TPTP, the `tptp2X` utility was used to rename all predicate and function symbols, randomly reorder the formulae and the clauses' literals, and randomly reverse the equalities in the UEQ problems.

The ATP systems were delivered to the competition organizers in source code form, and installed (by what whatever compilation, etc, process necessary) on the competition computers by the organizers. The ATP systems were required to be sound and fully automatic. The organizers

Table 3
Numbers of eligible and used problems

Division	MIX					FOF		SAT		EPR		UEQ
Category	HNE	HEQ	NNE	NEQ	PEQ	FNE	FEQ	SNE	SEQ	EPT	EPS	
Eligible	90	142	191	660	285	39	681	155	169	429	209	180
Eligible new	0	0	0	0	43	11	220	1	29	0	0	33
Max usable	64	63	69	537	207	39	681	155	169	429	76	180
Max new	0	0	0	0	43	11	220	1	29	0	0	33
Used	20	20	20	60	30	35	115	60	60	60	60	120
New used	0	0	0	0	15	11	58	1	29	0	0	33

tested the systems for soundness by submitting non-theorems to the systems participating in the MIX, FOF, EPR, and UEQ divisions, and theorems to the systems participating in the SAT and EPR divisions. Claiming to have found a proof of a non-theorem or a disproof of a theorem indicates unsoundness. No systems failed this test. However, after the competition the entrant of THEO discovered that THEO was unsound for FOF problems, and THEO was thus retrospectively disqualified from the FOF division of the competition by the competition panel. It should be noted that the unsoundness occurred only for certain types of problems - with more than 256 constants, and that there was no intention to deceive. THEO was retrospectively entered into the demonstration FOF division, and the results amended to report failure by THEO for the seven affected problems. Fully automatic operation meant that any command line switches had to be the same for all problems.

A 600s CPU time limit was imposed on each solution attempt. A wall clock time limit of double the CPU time limit was imposed in the competition divisions, to limit very high memory usage that causes swapping.

4. Results

For each ATP system, for each problem, three items of data were recorded: whether or not the problem was solved, the CPU time taken, and whether or not a proof or model was output. This section summarizes the results, and provides some commentary. Detailed results, including the systems' output files, are available from the CASC-20 WWW site. In each of the results summary tables below, the CASC-J2 winner is *emphasized*.

4.1. The MIX Division

Tables 4 and 5 summarize the results in the MIX division. As Vampire outputs refutations, Vampire was the winner of both the ranking classes. Only the newer version of Vampire beat last year's winner, confirming the dominance of Vampire in this division. The improved performance of Vampire 8.0 over Vampire 7.0 is due to bugfixes in the orderings used. The Vampires ware the only strategy scheduling systems (i.e., running a selected sequence of search strategies, each with a fraction of the time limit, until one finds a solution) in the MIX division of CASC-20. This is a change from the previous few years, where strategy scheduling systems have dominated. The benefits of strategy scheduling, in the context of a large enough time limit for multiple strategies to each be allocated enough CPU time to be effective, is evident. A naive strategy scheduling of Vampire 8.0, E, and Prover9, in which each is given 200s (one third of the 600s limit), would have resulted in the solution of 141 problems. E and EP have lower average times than the Vampires, due to the use of a single strategy rather than strategy scheduling.

As has been the case in the last two CASCs, there is a gap between the top systems and the lower ranked systems, here coming after Prover9. The strong performance of Prover9, a new system, is noteworthy. Its strong performances in MIX, FOF, and UEQ divisions won it the "Outstanding newcomer" award for CASC-20.

Only the Vampires performed weakly on the new problems relative to their overall performance. All of the new problems were PEQ lattice theory problems that are unit equality problems with the exception of a single non unit axiom. There is no significant correlation between the performances on these near-UEQ lattice theory problems and the

Table 4
MIX division results

ATP System	MIX /150	Avg time	Prfs out	New /15
Vampire 8.0	137	40.6	137	9
Vampire 7.0	133	45.5	133	11
E 0.9pre3	117	24.1	0	13
EP 0.9pre3	117	30.0	108	13
Prover9 0705	100	45.6	100	12
THEO JN05	52	48.7	52	0
Darwin 1.2	32	13.1	0	0
Otter 3.3	27	46.4	27	0
Demonstration division				
MathServ 0.62	121	38.1	0	13
Octopus JN05	79	24.8	79	1

Table 5
MIX category results

ATP System	HNE /20	HEQ /20	NNE /20	NEQ /60	PEQ /30
Vampire 8.0	19	18	19	58	23
Vampire 7.0	19	17	19	53	25
E 0.9pre3	15	13	16	47	26
EP 0.9pre3	15	13	16	47	26
Prover9 0705	17	13	10	34	26
THEO JN05	10	0	14	27	1
Darwin 1.2	5	0	10	14	3
Otter 3.3	9	0	3	10	5
Demonstration division					
MathServ 0.62	16	15	17	47	26
Octopus JN05	12	3	16	44	4

fully UEQ lattice theory problems (see Section 4.5) that have a very similar axiomatization. This indicates that different techniques are necessary for near-UEQ and fully UEQ problems.

The rankings in the categories align quite closely with the division ranking, with the exception of Prover9's relatively strong performance in the HNE category and the Vampires' relatively weaker performances in the PEQ category. Prover9's strong HNE performance was due to a strategy that examines the depths of the positive and negative literals in non-unit clauses, and decides whether a forward search (positive hyper-resolution) or a backward search (negative hyper-resolution) should be used. The Vampires' weaker PEQ performances were due to the weaker performances on the new problems.

In the demonstration division MathServ performed fairly well. MathServ selects an existing (publicly available prior to CASC-20) ATP system to run on each problem according to syntactic characteristics of the problem. In the MIX division MathServ always chose EP 0.82, thus the MathServ results reflect what EP 0.82 would have achieved in the MIX division - slightly better than the newer EP 0.9pre3! The reason for this reversal is that E 0.9pre3 is not a significantly better system than E 0.82[2], and MathServ ran on a faster computer with more memory than E 0.9pre3 in the competition.

[2]The improvement in version 0.9pre3 over version 0.82 was less than expected, due to a bug in version 0.9pre3's problem classification. This was repaired in the E 0.9 public release.

The individual problem results show that three problems were solved by all the systems. Four problems, `LAT036-1`, `LAT192-1`, `LCL423-1`, and `SYN314-1.002.001`, were unsolved. These problems were eligible because they have been solved by Gandalf c-2.6 [18], Vampire 8.0 on a faster computer, Darwin 1.2, and Bliksem 1.12 [2], respectively. Eleven problems were solved by only Vampires (eight by both versions and three by only version 8.0). Other unique solutions were by E and Prover9, each of which solved one problem that no other system solved.

As has been observed in previous CASCs, EP failed to produce refutations for some of the problems it solved, due to the separate post-production of refutations after their existence has been assured.

4.2. The FOF Division

Table 6 summarizes the results in the FOF division. As Vampire outputs proofs, Vampire 8.0 was the winner of both the ranking classes. All the systems except MUSCADET work by converting to CNF and producing a refutation. MUSCADET is a natural deduction system, best suited to mathematical, especially set theory, problems. The Vampires' use of CNF conversion means that the improved performance of Vampire 8.0 over Vampire 7.0 in the MIX division carries over to the FOF division. Additionally, Vampire 8.0's CNF conversion was improved by a naming technique that results in fewer and shorter clauses, and a better Skolemization algorithm that sometimes makes Skolem functions with fewer arguments.

Table 6
FOF division and category results

ATP System	FOF /150	Avg time	Prfs out	New /69	FNE /35	FEQ /115
Vampire 8.0	131	31.6	131	54	35	96
Vampire 7.0	129	26.7	129	55	34	95
E 0.9pre3	122	12.4	0	50	34	88
EP 0.9pre3	122	15.2	117	50	34	88
Prover9 0705	113	33.2	110	53	34	79
Otter 3.3	66	63.0	66	28	33	79
MUSCADET 2.5	31	0.0	0	4	13	18
Demonstration division						
MathServ 0.62	113	33.2	0	48	35	79
Octopus JN05	121	22.2	120	47	34	87
THEO JN05	93	19.4	93	34	33	60

The higher ranked systems performed well on the large number of new problems. These new problems come from a range of domains, including common sense reasoning, graph theory, arithmetic, and software verification. The successful solution of such a range of new problems indicates that the systems have strategies that are general purpose and that extend usefully to new unseen problems.

The individual problem results show that fourteen problems were solved by all the systems, with thirteen being in the FNE category. This large number of undifferentiating problems was to be expected due to the use of easy problems in the FNE category. Three problems, `CSR001+1`, `MGT035+2`, and `SWV115+1`, were unsolved. These problems were eligible because they have been solved by SPASS 2.1 [19], E 0.82, and SPASS 2.1, respectively. `MGT035+2` was also solved in the demonstration division by MathServ, because MathServ uses E 0.82. Thirty-two of the new problems came from the `SWV` domain, obtained from the software certification process described in [3]. Of these seven were solved by only Prover9, indicating some specialist capability of this new system.

Both EP and Prover9 solved some problems for which they did not output proofs. For Prover9 this is due to some problems being solved in the conversion from FOF to CNF, and those steps not being documented in the output, and hence the problems are not counted as having a proof output.

4.3. The SAT Division

Table 7 summarizes the results in the SAT division. As Paradox outputs models, Paradox was the winner of both the ranking classes. The Paradox systems dominated the division, with the new Paradox improving over the older version that had won the previous two CASCs, especially in terms of time taken. The improvement in Paradox 1.3 is due to the use of a better SAT solver, a faster and smarter clause instantiation implementation, and a preprocessing step that detects and removes unnecessary clauses.

There were 30 new problems in the SAT division, all except for one of which are lattice theory problems. The results show that the top three systems had no difficulty with these, while the lower ranked systems could solve none of them. Of the three problems not solved by Paradox 1.3, `LCL415-1` and `PUZ049-1` were solved by E. `PUZ049-1` has a finite model with a very large domain. It is unknown if `LCL415-1` has a finite model.[3] The problems solved by Paradox necessarily have finite models. It would be desirable to have more test problems with infinite models, and the community is encouraged to contribute such problems to the TPTP.

Table 7
SAT division and category results

ATP System	SAT /120	Avg time	Mdls out	New /30	SNE /60	SEQ /60
Paradox 1.3	117	3.3	117	30	58	59
Paradox 1.0	116	8.5	116	29	58	58
Mace4 0705D	83	2.4	83	29	29	54
Mace2 2.2	40	10.9	40	0	16	24
Darwin 1.2	20	4.5	20	0	20	0
E 0.9pre3	11	4.6	0	0	9	2
Demonstration division						
MathServ 0.62	7	10.6	0	0	7	0

4.4. The EPR Division

Table 8 summarizes the results in the EPR division. The winner, DCTP 10.21p, was the winner of the previous CASC. Darwin 1.2, in second place and the outstanding newcomer of CASC-J2, has a notably lower average solution time. This is because DCTP is a strategy scheduling system, while Darwin is monolithic. Only Vampire output proofs, and only Darwin and Paradox output mod-

[3] After correspondence with the researchers who proposed this problem.

els. It would be desirable to have a system that outputs both.

The individual problem results show that many problems were solved by all systems. This was due to the use of easy problems. No problems were unsolved, but two problems, `NLP213-1` and `NLP216`, were solved by only Paradox, and one problem, `PUZ037-3`, was solved by only the two weakest systems in the division, E and Vampire. These distinctive solutions indicate some useful unique capabilities of these weaker systems.

Table 8
EPR division and category results

ATP System	EPR /120	Avg time	Ps/Ms output	New /0	EPT /60	EPS /60
DCTP 10.21p	117	14.9	0/0	0	59	58
Darwin 1.2	113	1.3	0/54	0	59	54
Paradox 1.3	111	7.6	0/55	0	56	55
E 0.9pre3	96	1.5	0/0	0	57	39
Vampire 8.0	91	1.1	59/0	0	59	32
Demonstration division						
MathServ 0.62	86	8.4	0/0	0	55	31

4.5. The UEQ Division

Table 9 summarizes the results in the UEQ division. The winner, Waldmeister 704, was the winner of the previous CASC. Waldmeister's performance is significantly better than that of the other systems. In particular, Waldmeister performed much better on the new problems (all in lattice theory) than the other systems.

The individual problem results show that four problems, `LAT145-1`, `LAT162-1`, `LAT164-1`, and `LAT169-1` (all new lattice theory problems), were unsolved. These problems were eligible because they had been solved by SCOTT 6.1 [5], EQP 0.9d [8] and Gandalf c-2.6, Scott 6.1 and SOS 1.0 [11], and Bliksem 1.12, respectively. One problem, `GRP196-1`, was solved by only Vampire, and three problems, `LAT152-1`, `LAT154-1`, and `LAT156-1` (all new problems), were solved by only Otter. The new lattice theory problems evidently provided new challenges for the ATP systems.

Table 9
UEQ division results

ATP System	UEQ /120	Avg time	Prfs out	New /33
Waldmeister 704	110	56.5	110	24
Prover9 0705	86	40.1	86	5
Vampire 8.0	71	48.3	71	0
E 0.9pre3	63	2.5	0	0
Otter 3.3	23	20.0	23	5
Demonstration division				
MathServ 0.62	64	30.0	0	1

5. Descriptions of the Winning Systems

Vampire 8.0 [10], is an automatic theorem prover for classical first-order logic. Its kernel implements the calculi of ordered binary resolution with superposition for handling equality. Knuth-Bendix and lexicographic path ordering are available; Knuth-Bendix ordering was used for CASC-20. The splitting rule and negative equality splitting are simulated by the introduction of new predicate definitions and dynamic folding of such definitions. A number of standard redundancy criteria and simplification techniques are used for pruning the search space: subsumption, tautology deletion (optionally modulo commutativity), subsumption resolution, rewriting by ordered unit equalities, and a lightweight basicness. The shell of Vampire accepts problems in full FOF syntax, clausifies, and performs a number of useful transformations before passing the result to the kernel. When a refutation is found, the system produces a verifiable proof, which validates both the clausification phase and the refutation of the CNF. The automatic mode of Vampire 8.0 was derived from extensive experimental data obtained on problems from TPTP v3.0.1. Input problems are classified taking into account simple syntactic properties, such as being Horn or non-Horn, presence of equality, etc. Additionally, the presence of some important kinds of axioms, such as set theory axioms, associativity, and commutativityi, is taken into account. Every class of problems is assigned a fixed schedule consisting of a number of kernel strategies, which are called one by one with different time limits. The main improvements to Vampire from version 7.0 to 8.0 are bugfixes in the orderings used, the availability of lexicographic path ordering, a naming technique in the CNF transformation, a better Skolemization algorithm, the ability

to parse the new TPTP input syntax as well as the KIF syntax, the possibility of working with multiple knowledge bases, and a query answering mode. Vampire is implemented in C++.

Paradox 1.3 [1], the SAT division winner, is a finite-domain model generator that produces human readable models. Paradox 1.3 is basically the same solver as Paradox 1.0, with a few minor modifications. The main algorithm is based on a MACE-style [6] flattening and instantiation of the first order clauses into propositional clauses that encode the existence of a model of a fixed size. These propositional problems are generated for increasing domain sizes and given to a SAT solver. Features that already existed in Paradox 1.0 include polynomial-time clause splitting heuristics, the use of incremental SAT, static symmetry reduction techniques, and the use of sort inference. New features in Paradox 1.3 are the simplification of problems containing pure predicate symbols (appearing positively (resp. negatively) in all clauses where they occur), and a new fast clause instantiation algorithm. Another notable change from Paradox 1.0 is the use of the incremental SAT-solver MiniSat [4] as its SAT-engine. The main part of Paradox is implemented in Haskell using the GHC compiler, linked together with the MiniSat library which is written in C++.

DCTP 10.21p, the EPR division winner, and **Waldmeister 704**, the UEQ division winner, were the winners in CASC-J2, and were described in the CASC-J2 report [14].

Prover9 0705 [9], the best newcomer, is a saturation-style Otter-loop prover for first-order and equational logic. It descends from Otter [7] and from other provers associated with Argonne, in particular, ITP, AURA, and the TP series. Prover9 has available ordered resolution, ordered paramodulation, UR resolution, hyperresolution, and several variants of those inference rules. The term structure is not shared, and several types of term, literal, and clause indexing are used for the inference rules, rewriting, and subsumption. Strategies for the automatic mode used in CASC-2005 include (1) term ordering by LPO, with a simple rule for symbol precedence, (2) selective use of non-orientable equations as rewrite rules, (3) a rule based on term depth for determining inference rules for Horn sets, (4) a limited-resource strategy for limiting the number of clauses kept, and (5) a method for attempting to reduce FOF problems to independent subproblems. Experimentation with the CASC-J2 problems led to the weighting function used for selecting given clauses. Strategies *not* used include recognition of theories (e.g., for determining the term ordering or symbol precedence) and multiple searches with different inference rules or strategies. Prover9 is implemented in C.

6. Conclusion

CASC-20 was the tenth large scale competition for first order ATP systems. The results showed small but significant improvements over last year's systems. The consistent strong performances of a few systems in the last few CASCs has established these systems as the preferred general purpose ATP systems for research and application. CASC makes these systems publicly available, and thus provides starting points from which new developers can leverage the knowledge of highly experienced ATP system developers. Section 4 has highlighted some gaps in the characteristics of the ATP systems in the competition, and new developers are encouraged to take up the challenge of filling them.

For CASC-J3 (CASC will be part of the 3rd IJCAR, as part of FLoC, in 2006) the FOF division will be promoted to the primary place. This change reflects the increased number of FOF contributions to the TPTP (550 new FOF problems between TPTP v3.0.1 and TPTP v3.1.0, in contrast with only 168 new CNF problems), and the corresponding increased use of FOF in applications. The FOF problems in CASC-J3 will use the full set of FOF operators defined in the TPTP syntax, e.g., <= and <~>, and no standardizing preprocessing will be performed.

CASC-20 fulfilled its objectives, by evaluating the relative abilities of current ATP systems, and stimulating development of and interest in ATP systems. The competition highlighted areas of ATP where progress was made in the last year. Through the continuity of the event and consistency in the the reporting of the results, performance comparisons with previous and future years are easily possible. The competition provided exposure for system builders both within and outside of the community, and provided an overview of the implementation state of running, fully automatic, first order ATP systems.

References

[1] K. Claessen and N. Sorensson. New Techniques that Improve MACE-style Finite Model Finding. In P. Baumgartner and C. Fermueller, editors, *Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications*, 2003.

[2] H. de Nivelle. Bliksem Resolution Prover. http://www.mpi-sb.mpg.de/ nivelle, URL.

[3] E. Denney, B. Fischer, and J. Schumann. Using Automated Theorem Provers to Certify Auto-generated Aerospace Software. In M. Rusinowitch and D. Basin, editors, *Proceedings of the 2nd International Joint Conference on Automated Reasoning*, number 3097 in Lecture Notes in Artificial Intelligence, pages 198–212, 2004.

[4] N. Eén and N. Sörensson. MiniSat - A SAT Solver with Conflict-Clause Minimization. In F. Bacchus and T. Walsh, editors, *Posters of the 8th International Conference on Theory and Applications of Satisfiability Testing*, 2005.

[5] K. Hodgson and J.K. Slaney. System Description: SCOTT-5. In R. Gore, A. Leitsch, and T. Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning*, number 2083 in Lecture Notes in Artificial Intelligence, pages 443–447. Springer-Verlag, 2001.

[6] W.W. McCune. Mace4 Reference Manual and Guide. Technical Report ANL/MCS-TM-264, Argonne National Laboratory, Argonne, USA, 2003.

[7] W.W. McCune. Otter 3.3 Reference Manual. Technical Report ANL/MSC-TM-263, Argonne National Laboratory, Argonne, USA, 2003.

[8] W.W. McCune. EQP: Equational Prover. http://www-unix.mcs.anl.gov/AR/eqp/, URL.

[9] W.W. McCune. Prover9. http://www.mcs.anl.gov/ mccune/prover9/, URL.

[10] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[11] J.K. Slaney, A. Binas, and D. Price. Guiding a Theorem Prover with Soft Constraints. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 221–225, 2004.

[12] G. Sutcliffe. The CADE-16 ATP System Competition. *Journal of Automated Reasoning*, 24(3):371–396, 2000.

[13] G. Sutcliffe. Proceedings of the CADE-20 ATP System Competition. Tallinn, Estonia, 2005.

[14] G. Sutcliffe. The IJCAR-2004 Automated Theorem Proving Competition. *AI Communications*, 18(1):33–40, 2005.

[15] G. Sutcliffe and C. Suttner. The State of CASC. *AI Communications*, page To appear, 2006.

[16] G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[17] G. Sutcliffe and C.B. Suttner. Evaluating General Purpose Automated Theorem Proving Systems. *Artificial Intelligence*, 131(1-2):39–54, 2001.

[18] T. Tammet. Towards Efficient Subsumption. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction*, number 1421 in Lecture Notes in Artificial Intelligence, pages 427–440. Springer-Verlag, 1998.

[19] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobald, and D. Topic. SPASS Version 2.0. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in Lecture Notes in Artificial Intelligence, pages 275–279. Springer-Verlag, 2002.