The 9th IJCAR Automated Theorem Proving System Competition – CASC-J9

Geoff Sutcliffe
Department of Computer Science
University of Miami, USA
geoff@cs.miami.edu

The CADE ATP System Competition (CASC) is the annual evaluation of fully automatic, classical logic Automated Theorem Proving (ATP) systems. CASC-J9 was the twenty-third competition in the CASC series. Twenty-three ATP systems and system variants competed in the various competition divisions. This paper presents an outline of the competition design, and a commentated summary of the results.

Keywords: automated theorem proving, competition

1. Introduction

The CADE ATP System Competition (CASC) [19] is the annual evaluation of fully automatic, classical logic Automated Theorem Proving (ATP) systems – the world championship for such systems. One purpose of CASC is to provide a public evaluation of the relative capabilities of ATP systems. Additionally, CASC aims to stimulate ATP research, motivate development and implementation of robust ATP systems that are useful and easily deployed in applications, provide an inspiring environment for personal interaction between ATP researchers, and expose ATP systems within and beyond the ATP community. CASC-J9 was held on 14th July 2018 in Oxford, United Kingdom, as part of the 9th International Joint Conference on Automated Reasoning, which was in turn part of the 2018 Federated Logic Conference (FLoC).[1] CASC-J9 was the twenty-third competition in the CASC series; see [20] and citations therein for information about individual previous competitions. The CASC-J9 web site provides access to all competition resources:

`http://www.tptp.org/CASC/J9`.

CASC is divided into divisions according to problem and system characteristics. There are competition divisions in which the systems are explicitly ranked, and a demonstration division in which systems demonstrate their abilities without being ranked (for systems that cannot be entered into the competition divisions for any reason, e.g., the system is experimental, or the entrant is a competition organizer). Each competition division uses problems that have certain logical, language, and syntactic characteristics, so that the systems that compete in the division are, in principle, able to attempt all the problems in the division. Some divisions are further divided into problem categories that make it possible to analyze, at a more fine-grained level, which systems work well for what types of problems. The demonstration division uses the same problems as the competition divisions, and the entry specifies which competition divisions' problems are to be used. Table 1 catalogs the divisions and problem categories of CASC-J9.

Twenty-three ATP systems and system variants listed in Tables 2 and 3 competed in the various divisions. The division winners of CASC-26 (the previous CASC) were automatically entered into the corresponding demonstration divisions, to provide benchmarks against which progress can be judged. Additionally, Prover9 1109a is entered into the FOF division each year, as a fixed point against which progress can be judged. System descriptions of the systems are in [21] and on the CASC-J9 web site.

CASC-J9 was organized by Geoff Sutcliffe, and overseen by a panel consisting of Martin Giese, Aart Middeldorp, and Florian Rabe. The competition was run on computers provided by the StarExec project [16] at the University of Iowa.

[1] The "FLoC Olympic Games" hosted 15 competitions in various areas of logic and reasoning, which provided the CASC-J9 entrants with opportunities to observe and interact with developers of other types of reasoning tools.

AI Communications
ISSN 0921-7126, IOS Press.

Division	Problems	Problem Categories. The examples can be viewed at `http://www.tptp.org/cgi-bin/SeeTPTP?Category=Problems`.
THF	Monomorphic **T**yped **H**igher-order **F**orm theorems (axioms with a provable conjecture).	**TNE** – **T**HF with **N**o **E**quality, e.g., `NUM738^1`. **TEQ** – **T**HF with **EQ**uality, e.g., `SET171^3`.
TFA	Monomorphic **T**yped **F**irst-order form theorems with **A**rithmetic (axioms with a provable conjecture).	**TFI** – **TF**A with only **I**nteger arithmetic, e.g., `DAT016=1`. **TFE** – **TF**A with only r**E**al arithmetic, e.g., `MSC022=2`.
FOF	**F**irst-**O**rder **F**orm theorems (axioms with a provable conjecture).	**FNE** – **F**OF with **N**o **E**quality, e.g., `COM003+1`. **FEQ** – **F**OF with **EQ**uality, e.g., `SEU147+3`.
FNT	**F**OF **N**on-**T**heorems (axioms with a countersatisfiable conjecture, and satisfiable axioms sets).	**FNN** – **FN**T with **N**o equality, e.g., `KRS173+1`. **FNQ** – **FN**T with e**Q**uality, e.g., `MGT033+2`.
EPR	**E**ffectively **PR**opositional theorems and non-theorems in clause normal form (unsatisfiable and satisfiable clause sets). *Effectively propositional* means that the problems are known to be reducible to propositional form, e.g., CNF problems that have no functions with arity greater than zero.	**EPT** – **E**ffectively **P**ropositional **T**heorems (unsatisfiable clause sets), e.g., `PUZ037-3`. **EPS** – **E**ffectively **P**ropositional non-theorems (**S**atisfiable clause sets), e.g., `GRP123-2.005`.
LTB	First-order form theorems (axioms with a provable conjecture) from **L**arge **T**heories, presented in **B**atches. A large theory typically has many functors and predicates, and many axioms of which only a few are required for the proof of a theorem. The problems in a batch are given to an ATP system all at once, and typically have a common core set of axioms. The batch presentation allows the ATP systems to load and preprocess the common core set of axioms just once, and to share logical and control results between proof searches.	**CML** – Problems exported from CakeML [6]. See Section 2.2 for details of the problems. Each problem category is accompanied by a set of training problems and their solutions, taken from the same source as the competition problems. The training data can be used for ATP system tuning during (typically at the start of) the competition.

Table 1
Divisions and Problem categories

This paper is organized as follows: Section 2 outlines the design and organization of the competition. Section 3 provides a commentated summary of the results. Section 4 contains short descriptions of three of the ATP systems. Section 5 concludes and discusses plans for future CASCs.

2. Outline of Design and Organization

The design and organization of CASC has evolved over the years to a sophisticated state. An outline of the CASC-J9 design and organization is provided here. The details are in [21] and on the CASC-J9 web site. Important changes for CASC-J9 were: (This summary of changes is most useful for readers who are already familiar with the general design of CASC. Other readers can skip to Section 2.1.)

- The TFR problem category of the TFA division was put into a hiatus state due to the lack of eligible problems.
- The SLH division was not run, as there was no sign of a changed work profile being produced by the Sledgehammer system.
- The previous year's winners were placed into the demonstration division, to ensure that one of the new systems wins. (Systems were not allowed to, and none tried to, circumvent this by making a trivial change, e.g., incrementing the system version number.)

2.1. System Delivery, Execution, and Evaluation

The ATP systems entered into CASC are delivered to the competition organizer as StarExec installation packages, which the organizer installs and tests on StarExec. Source code is delivered separately for archiving on the competition web site.

The ATP systems are required to be fully automatic. They are executed as black boxes, on one problem (the non-LTB divisions) or one batch (the LTB division) at a time. Any command line parameters have to be the same for all prob-

ATP System	Divisions	Entrant (Associates)	Entrant's Affiliation
CSE 1.0	FOF	Feng Cao (Yang Xu, Jun Liu, Shuwei Chen, Xiaomei Zhong, Peng Xu, Qinghua Liu, Huimin Fu, Xiaodong Guan, Zhenming Song)	Southwest Jiaotong University
CSE 1.1	FOF	Feng Cao (Yang Xu, Jun Liu, Shuwei Chen, Xingxing He, Xiaomei Zhong, Peng Xu, Qinghua Liu, Huimin Fu, Jian Zhong, Guanfeng Wu, Xiaodong Guan, Zhenming Song)	Southwest Jiaotong University
CSE_E 1.0	FOF	Feng Cao (Yang Xu, Stephan Schulz, Jun Liu, Shuwei Chen, Xingxing He, Xiaomei Zhong, Peng Xu, Qinghua Liu, Huimin Fu, Jian Zhong, Guanfeng Wu, Xiaodong Guan, Zhenming Song)	Southwest Jiaotong University
CVC4 1.6pre	TFA FOF FNT	Andrew Reynolds (Clark Barrett, Cesare Tinelli)	University of Iowa
E 2.2pre	FOF FNT EPR LTB	Stephan Schulz	DHBW Stuttgart
Geo-III 2018C	FOF FNT EPR	Hans de Nivelle	Nazarbayev University
Grackle 0.1	LTB	Jan Jakubuv (Cezary Kaliszyk, Stephan Schulz, Josef Urban)	Czech Technical University in Prague
iProver 2.6	EPR (demo)	CASC	CASC-26 winner
iProver 2.8	FOF FNT EPR LTB	Konstantin Korovin (Julio Cesar Lopez Hernandez)	University of Manchester
leanCoP 2.2	FOF	Jens Otten	University of Oslo
LEO-II 1.7.0	THF	Alexander Steen (Christoph Benzmüller)	Freie Universität Berlin
Leo-III 1.3	THF FOF EPR (demo)	Alexander Steen (Christoph Benzmüller, Max Wisniewski)	Freie Universität Berlin

Table 2: The ATP systems and entrants

ATP System	Divisions	Entrants (Associates)	Entrant's Affiliation
MaLARea 0.6	LTB	Josef Urban (Jan Jakubuv, Cezary Kaliszyk, Stephan Schulz)	Czech Technical University in Prague
nanoCoP 1.1	FOF	Jens Otten	University of Oslo
Princess 170717	TFA	Philipp Rümmer	Uppsala University
Prover9 1109a	FOF (demo)	CASC (William McCune, Bob Veroff)	CASC fixed point
Satallax 3.2	THF	CASC	CASC-26 winner
Satallax 3.3	THF	Michael Färber (Chad Brown)	Universität Innsbruck
Twee 2.2	FOF	Nick Smallbone (Koen Claessen)	Chalmers University of Technology
Vampire 4.0	LTB (demo)	CASC	CASC-26 winner
Vampire 4.1	TFA FNT (demo)	CASC	CASC-26 winner
Vampire 4.2	FOF (demo)	CASC	CASC-26 winner
Vampire 4.3	TFA FOF FNT EPR LTB	Giles Reger (Martin Suda, Andrei Voronkov, Evgeny Kotelnikov, Martin Riener, Laura Kovacs)	University of Manchester

Table 3: The ATP systems and entrants, continued

lems/batches in each division. The ATP systems are required to be sound, and are tested for soundness by submitting non-theorems to the systems in the THF, TFA, FOF, EPR, and LTB divisions, and theorems to the systems in the FNT and EPR divisions. Claiming to have found a proof of a non-theorem or a disproof of a theorem indicates unsoundness. For the three CASCs from CASC-J7 to CASC-J8 no systems failed soundness testing, but last year in CASC-26 five systems were found to be unsound. For CASC-J9 things returned to the pleasing state of no systems failing the soundness testing.

The THF, TFA, FOF, FNT, and LTB divisions are ranked according to the number of problems solved with an acceptable proof/model output. The EPR division is ranked according to the number of problems solved, but not necessarily accompanied by a proof/model (but systems that do output proofs/models are highlighted in the presentation of results). Ties are broken according to the average time taken over problems solved (CPU time or wall clock time, depending on the type of limit in the division). Trophies are awarded to the division winners. Additionally in CASC-J9, Jasmin Blanchette at the Vrije Universiteit Amsterdam contributed a travel prize for the THF division, with expenses covered by the Matryoshka project [2].

In addition to the ranking criteria, three other measures are made and presented in the results: The *state-of-the-art contribution* (SotAC) quantifies the unique abilities of each system. For each problem solved by a system, its SotAC for the problem is the reciprocal of the number of systems that solved the problem, so that if a system is the only one to solve a problem then its SotAC for the problem is 1.00, and if all the systems solve a problem their SotAC for the problem is the inverse of the number of systems. A system's overall SotAC is its average SotAC over the problems it solved. The *core usage* is the average of the ratios of CPU time to wall clock time used, over the problems solved. This measures the extent to which the systems take advantage of multiple cores. The competition ran on quad-core computers, thus the maximal core usage was 4.0. The *efficiency* measure combines the number of problems solved with the time taken. It is the average solution rate over the problems solved (the solution rate for one problem is the inverse of the time taken to solve it), multiplied by the fraction of problems solved. Efficiency is computed for both CPU time and wall clock time, to measure how efficiently the systems use one core and multiple cores respectively.

2.2. Competition Problems

2.2.1. Problems for the TPTP-based Divisions

The problems for the THF, TFA, FOF, FNT, and EPR divisions were taken from the Thousands of Problems for Theorem Provers (TPTP) problem library [18], v7.2.0. The TPTP version used for CASC is not released until after the competition has started, so that new problems in the release have not been seen by the entrants. The problems have to meet certain criteria to be eligible for selection:

- The TPTP tags problems that are designed specifically to be suited or ill-suited to a particular ATP system, calculus, or control strategy as *biased*. Biased problems are excluded from the competition.
- The problems have to be syntactically non-propositional.
- The TPTP uses system performance data in the Thousands of Solutions from Theorem Provers (TSTP) solution library to compute problem difficulty ratings in the range 0.00 (easy) to 1.00 (unsolved) [22]. Difficult problems with ratings in the range 0.21 to 0.99 are eligible. Problems of lesser and greater ratings might also be eligible in some divisions if there are not enough problems with ratings in that range. In CASC-J9 two FNQ problems of rating 1.00 were made eligible so that 100 problems were eligible for that problem category. Systems can be submitted before the competition so that their performance data is used in computing the problem ratings.

In order to ensure that no system receives an advantage or disadvantage due to the specific presentation of the problems in the TPTP, the problems are preprocessed to strip out all comment lines (in particular, the problem header), randomly reorder the formulae/clauses (`include` directives are left before the formulae, and type declarations are kept before the symbols' uses), randomly swap the arguments of associative connectives, randomly reverse implications, and randomly reverse equalities.

The numbers of problems used in each division and problem category are roughly proportional to the numbers of eligible problems. The problems used are randomly selected from the eligible problems based on a seed supplied by the competition panel:

- The selection is constrained so that no division or category contains an excessive number of very similar problems [17].
- The selection is biased to select problems that are new in the TPTP version used, until 50% of the problems in each problem category have been selected, after which random selection (from old and new problems) continues. The number of new problems used depends on how many new problems are eligible and the limitation on very similar problems.

Table 4 gives the numbers of eligible problems, the maximal numbers that could have been used after taking into account the limitation on very similar problems (in the context of the numbers of problems used), and the numbers of problems used, in each division and category. There were very few new problems in CASC-J9, because there had been very little growth of the TPTP since v7.1.0. (The ATP community is encouraged to contribute interesting new problems to the TPTP!)

The problems are given to the ATP systems in TPTP format, in increasing order of TPTP difficulty rating.

2.2.2. Problems for the LTB Division

The problems for the LTB division are taken from various sources, with each problem category being based on one source. CASC-J9 had one problem category[2]: the CML problem category used a set of 7592 problems exported from CakeML [6].[3] The problems used in the competition were selected by running E 2.1, iProver 2.7.5, SPASS 3.9, and Vampire 4.2 on the problems with a 300s CPU time limit. In those tests 1389 problems were proved to be theorems, 808 problems were found to be non-theorems, and 5395 were unsolved. The non-theorems were removed from the set, leaving 6784 problems. The 1000 problems with the most proofs (i.e., in some sense the easiest 1000 problems) were taken (with their proofs) for the training set. For the 1000 training problems there was a total of 3497 proofs. 784 problems were randomly removed from the remaining 5784 problems, to leave 5000 for the competition. In their exported form the problems have less than 100 axioms. In order to challenge the axiom selection capabilities of the ATP systems, the 5362 unique axioms in the 784 problems were added to the training and competition problems, to form the final problems for the competition. (While adding such "random" axioms can have unpredictable effects on the ability of ATP systems to find a proof, testing showed that the resultant problems were still appropriate for the competition.)

The problems are given to the ATP systems in TPTP format, in the natural order of their source, i.e., for CASC-J9 in the order of their export from CakeML. The problem batch was *unordered*, which meant that ATP systems could attempt the problems in any order, and could attempt each problem multiple times. This provides increased opportunities for sharing logical and control results between proof attempts.

2.3. Resources

The computers had four Intel(R) Xeon(R) E5-2609, 2.40GHz CPUs (a quad-core chip), 128GB memory, and ran the Red Hat Enterprise Linux Server release 7.2 (Maipo) operating system, kernel 3.10.0-514.16.1.el7.x86_64.

In the TPTP-based divisions a 300s CPU time limit was imposed for each problem. A wall clock time limit of 600s was also imposed to limit very high memory usage that causes swapping.

In the LTB division a 150000s wall clock time limit was imposed on the CML problem category, giving an average of 30s per problem. There was no wall clock time limit for each problem, and no CPU time limits (i.e., systems could benefit from using all cores available on the CPUs). Time spent before starting the first problem in the batch, e.g., learning from the training set and pre-loading the common axioms, and time spent between ending a problem and starting the next, e.g., learning from previous proofs, is not part of the time taken on problems. However, the time taken on such tasks is part of the overall time taken for the batch.

Demonstration division systems can run on the competition computers, or the computer(s) can be

[2] Usually the LTB division has multiple problem categories, but for CASC-J9 it was decided to have just one category, which allowed that category to have a very large number of problems.

[3] Thanks to Ramana Kumar for the export.

Division	THF		TFA		FOF		FNT		EPR		LTB
Category	TNE	TEQ	TFI	TFE	FNE	FEQ	FNN	FNQ	EPT	EPS	CML
Eligible	105	818	176	88	386	3470	104	98	137	167	6784
Usable	105	807	172	85	126	903	101	98	137	25	5784
New	0	0	0	0	3	0	2	0	0	0	-
Used	100	400	125	75	125	375	100	100	125	25	5000
New	0	0	0	0	3	0	2	0	0	0	-

Table 4
Numbers of eligible and used problems

supplied by the entrant. The CASC-J9 demonstration division systems all used the competition computers.

3. Results

For each ATP system, for each problem, four items of data were recorded: whether or not the problem was solved, the CPU time taken, the wall clock time taken, and whether or not a proof or model was output. This section summarizes the results, and provides commentary. The result tables below give the number of problems solved in the division, the average time over problems solved, whether or not proofs or models were output, the state-of-the-art contribution, the (micro-) efficiency, the core usage, the number of new problems solved, and the number of problems solved in each problem category. In each table the CASC-26 winner is *emphasized*. Detailed results, including the systems' output files, are available from the CASC-J9 web site.

3.1. The THF Division

Table 5 summarizes the results of the THF division. There were no great improvements to the systems since CASC-26, with Satallax 3.2 solving the most problems again, with Leo-III and LEO-II correspondingly behind. It was rather disappointing that effectively only three systems participated. As one non-entrant said when asked about interest in a THF-LTB division, "Probably won't participate (THF without LTB is already hard enough!)", while another long-standing THF entrant said "I don't think I'll be entering this year. Instead, I'd rather focus on the next generation of HO ATPs". Hopefully these entrants and others will return for CASC-27.

The average CPU times were notably higher than in CASC-26, e.g., Satallax 3.2's average CPU time in CASC-26 was 22.9s. An examination of the problems' difficulty ratings shows that the problems were harder than in CASC-26 - the TNE problems in CASC-26 had an average difficulty rating (at the time) of 0.49, and the TEQ problems had an average difficulty rating of 0.40. In CASC-J9 the averages were 0.53 and 0.43 respectively. This illustrates what is well known in ATP research - a small increase of difficulty can result in much higher CPU times for finding proofs. There were four problems (all from the CSR domain) that LEO-II solved without proofs, because the time limit was reached during proof generation.

The SotACs are quite even, with Leo-III's being the highest thanks to a high number of unique solutions (as noted in the individual problem analysis below), but no system showed any outstanding distinguishing capabilities. In contrast, the efficiencies are quite varied, with LEO-II having the highest efficiency thanks to its lower CPU times, and Leo-III having lower efficiency due to its higher CPU times. Leo-III's higher CPU times are due to (i) its implementation in Scala, which adds a fixed overhead for starting the JRE, (ii) running multiple external systems in parallel, and (iii) the Scala implementation doing garbage collection and JIT compilation in parallel with the main thread. On the upside, Leo-III is able to make good use of the cores available for the parallel processing.

The systems' performances in the two problem categories align with their overall performances, except that Leo-III had a slightly stronger performance on problems with equality. Beyond that there is no specialization to problems with or without equality.

The individual problem results show that 47 problems were unsolved, 172 problems were solved by all the systems, and 30 problems were solved

ATP System	THF /500	Avg CPU	Prfs out	Sot AC	μ Eff.	Core usage	TNE /100	TEQ /400
Satallax 3.3	401	32.2	401	0.33	262	0.97	70	331
Leo-III 1.3	355	38.5	355	0.35	134	2.62	56	299
LEO-II 1.7.0	213	17.9	209	0.29	290	0.76	37	176
Demonstration division								
Satallax 3.2	406	32.8	406	0.33	266	0.99	72	334

Table 5
THF division results

by only one system. Of the 30 unique solutions, 23 were by Leo-III, 4 by LEO-II, and 3 by Satallax 3.2. A portfolio of systems with higher SotAC values is often effective, and here a simple portfolio of Leo-III and Satallax 3.2 giving each 150s would solve 435 problems.

3.2. The TFA Division

Table 6 summarizes the results of the TFA division. The winner was the new Vampire 4.3, with the CASC-26 winner Vampire 4.1 close behind. Vampire 4.3 had a significantly lower average CPU time than Vampire 4.1 (as is also reflected in its higher efficiency). Vampire 4.3 improved over Vampire 4.1 thanks to improved theory instantiation [13] and extra heuristics to control the use of theory axioms in proof search [12]. The Vampire developers believe that Vampire 4.3 could have performed even better, as there was a bug in the detection of certain theory symbols in the problem that affected the addition of theory axioms. As a result Vampire 4.3 failed to solve some relatively easy problems, e.g., `ARI181=1` and `ARI634=1`. As was the case in the THF division, it was rather disappointing that effectively only three systems participated (including Princess which had not changed since CASC-26, but that's better than wimping out completely).

It was noted in the CASC-26 report that some of CVC4's proofs were vacuous (the output is "`No instantiations`"), which occurs when the proof is found using quantifier-free reasoning during preprocessing. Justifying preprocessing steps is a long term project in CVC4. The developers reported progress since CASC-26, but the work was not complete for CASC-J9.[4] CVC4 had a higher efficiency than Vampire 4.3 because it had lower CPU times for the first 67 problems solved (separately ordered by CPU time for each system) in the TFI problem category. CVC4 also performed strongly in the TFE problem category, thanks to improved support for unbounded non-linear arithmetic. Vampire 4.1 did worse in the TFE problem category because of a known inability to reason about quotients over the reals, as occurred in the six TFE problems that Vampire 4.3 solved and Vampire 4.1 did not, i.e., if that bug did not exist in Vampire 4.1 it would have solved the most problems.

The individual problem results show that only 1 problem was unsolved, 80 problems were solved by all the systems (40 in each problem category), and 23 problems were solved by only one system. Of the 23 unique solutions, 12 were by CVC4, 6 by Vampire 4.1, 3 by Vampire 4.3, and 2 by Princess. The best portfolio would be to give 100s to each of CVC4, Vampire 4.1, and Vampire 4.3, which would solve 194 problems, i.e., almost complete coverage of the division. The high fraction of problems solved by all systems, particularly in the TFE problem category, indicates that harder problems might be chosen for the TFA division.

3.3. The FOF Division

Table 7 summarizes the results of the FOF division. The new Vampire 4.3 beat the CASC-26 winner Vampire 4.2. Excluding Vampire 4.2, Vampire 4.3 also had the lowest average CPU time, highest SotAC, and highest efficiency. The improved performance of Vampire 4.3 is attributed to changed strategy schedules and (according to the developer) "getting lucky with problem selection", as

[4] It was a busy year for the CVC4 developers, as CVC4 was entered into and did well in multiple reasoning competitions. In particular, CVC4 had outstanding performances in SMT-COMP (`http://smtcomp.sourceforge.net/2018/`) and SyGuS (`http://www.sygus.org/SyGuS-COMP2018.html`).

ATP System	TFA /200	Avg CPU	Prfs out	Sot AC	μ Eff.	Core usage	TFI /125	TFE /75
Vampire 4.3	163	14,2	163	0.33	539	0.67	93	70
CVC4 1.6pre	157	15.0	157	0.36	613	0.61	85	72
Princess 170717	105	13.4	105	0.30	311	2.24	62	43
Demonstration division								
Vampire 4.1	162	23.1	162	0.35	491	0.75	98	64

Table 6
TFA division results

nothing significantly new was added to help solve FOF problems. The lefthand plot of Figure 1 shows the ordered solution times for the FOF division. It clearly divides the systems into five groups: the two Vampires; E and CSE_E; CVC4, Leo-II and iProver; leanCoP, nanoCoP, CSC1.1, CSC1.0, and Prover9; and finally Twee and Geo-III. The same grouping is evident in the righthand plot of Figure 1, which shows the fraction of problems solved, in the order that the problems were presented in the competition. The righthand plot additionally shows a different grouping for problems with lower difficulty ratings: the Vampires, CSE_E, E, CVC4, Leo-III, and iProver solving most of the easy problems; leanCoP, nanoCoP, and the CSEs solving less, and finally Prover9, Twee, and Geo solving the least (with gaps between those three systems). The consistent decrease in success rates after the first 200 problems also confirms that the TPTP problem ratings do reflect increasing difficulty of the problems for the systems.

It is admirable that the newcomer CSE_E did so well. It is clearly the case, and it is fully acknowledged, that CSE_E stands on the shoulders of the giant E (version 2.1). CSE_E first tries to solve the problem with E, and if unsuccessful it tries with CSE 1.0. If neither solves the problem then selected clauses inferred by CSE are combined with the original problem's clauses to form a new problem for E. The majority of CSE_E's solutions were found by the initial run of E. Another 3 problems were solved by CSE alone, and 11 more by E with CSE clauses added. None of the 3 problems solved by CSE alone, and only one of the 11 problems solved by E with CSE clauses added, were solved by the E 2.2pre entry in the competition. Evidently the clauses inferred by CSE help E. CSE_E was awarded the "best newcomer" prize, and a brief system description of the CSE family of systems is provided in Section 4.

CSE_E, iProver, and Twee each solved a problem for which they did not output a proof. In CSE_E's case the problem (`SYO525+1.015`) was solved in 169s by E, but CSE_E was unable to capture and reformat E's proof output. For iProver, the problem (`LCL666+1.010`) was solved in 146s, but subsequent investigation revealed that proof reconstruction takes about an hour, almost all within MiniSAT. The iProver developer classed it as "a weird problem". In Twee's case the problem (`GRP779+1`) was solved in 261s, but Twee ran out of time during post-processing that makes the proof human-readable. The developer of Twee has suggested that the generation of human-readable proofs "should perhaps be turned off" for the competition.

Twee and Geo, which were last ranked in the division, had high SotAC values. This demonstrates how quite different calculi can solve some different problems (as is noted in the individual problem analysis below). Twee converts FOF problems to CNF, then if they are Horn converts them to unit equality problems that it attempts to refute using unit equality reasoning. Geo-III is based on geometric resolution. Another system with a different calculus is nanoCoP, which is based on the non-clausal connection calculus. nanoCoP solved 16 problems not solved by its clausal predecessor leanCoP, but due to the additional overhead of dealing with the non-clausal formula structure, altogether slightly fewer problems than leanCoP. As interesting newcomers, brief system descriptions of Twee and nanoCoP are provided in Section 4.

There were three new problems in FOF division, all of which are encodings of simple calculations (e.g., $2 + 2 = 4$) in Robinson Arithmetic Q [15].[5] These problems were too hard for most of the systems.

[5]Those problems were bugfixed in the TPTP after the competition, but even in their buggy form they were still theorems.

ATP System	FOF /500	Avg CPU	Prfs out	Sot AC	μ Eff.	Core usage	New /3	FNE /125	FEQ /375
Vampire 4.3	461	16.4	461	0.20	483	0.83	2	115	346
CSE_E 1.0	363	26.4	362	0.15	333	0.95	1	90	273
E 2.2pre	350	25.6	350	0.15	339	0.84	1	92	258
CVC4 1.6pre	298	43.5	298	0.14	232	0.82	0	83	215
iProver 2.8	248	29.4	247	0.14	167	0.92	0	102	146
leanCoP 2.2	143	46.7	143	0.11	56	0.81	0	75	68
nanoCoP 1.1	133	48.6	133	0.10	55	0.81	0	71	62
CSE 1.1	126	54.9	126	0.10	61	0.97	0	62	64
CSE 1.0	123	50.2	123	0.10	64	0.99	0	61	62
Twee 2.2	74	62.2	73	0.17	50	0.87	0	16	58
Geo-III 2018C	50	40.1	50	0.18	21	0.95	0	14	36
Demonstration division									
Vampire 4.2	454	15.1	454	0.19	473	0.83	2	116	338
Leo-III 1.1	256	31.2	256	0.14	94	2.67	1	82	174
Prover9 1109a	122	29.7	122	0.13	113	0.87	0	28	94

Table 7
FOF division results

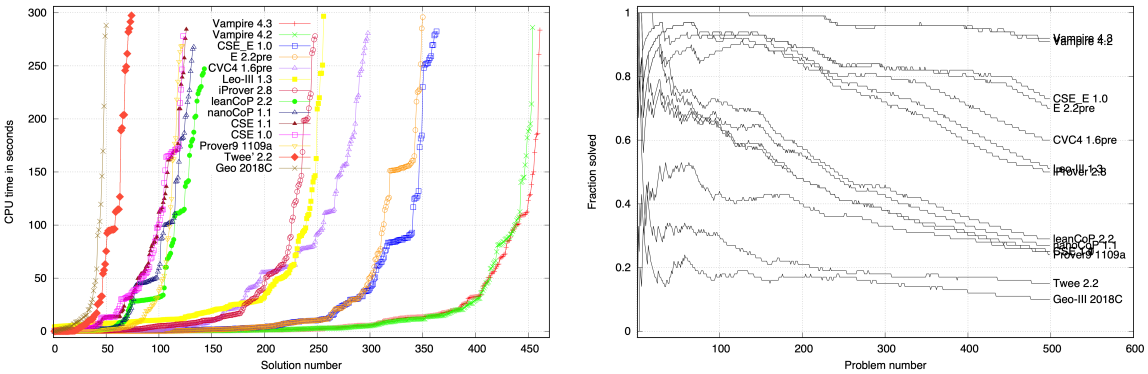

Fig. 1. Ordered solution times and success rates in the FOF division
The ordered solution times are for problems solved, individually sorted for each system in ascending order.
The success rates are for all problems in the order they were presented, i.e., in non-decreasing order of difficulty rating.

The individual problem results show that 14 problems were unsolved, 1 problem was solved by all the systems, and 17 problems were solved by only one system. Of the 17 unique solutions, 6 were by Vampire 4.3, 3 by Twee, 2 by Vampire 4.2, 2 by CSE_E, 2 by Geo-III, 1 by CVC4, and 1 by Leo-III. A simple portfolio of Vampire 4.3 and CSE_E, giving each 150s, would solve 467 problems.

3.4. The FNT Division

Table 8 summarizes the results of the FNT division. Vampire 4.3 came out well ahead. Vampire 4.3's improved performance came mainly from two new features in its finite model building component: clique detection to detect a minimal model size, and, with a greater effect, blocked clause elimination [5].

As in the THF division the average CPU times were higher than in CASC-26, and again increased problem difficulty was the cause. In this case the average difficulty rating was about the same for the FNQ problem category - 0.46 in CASC-26 and 0.47 in CASC-J9, but significantly higher in the FNN problem category - 0.19 in CASC-26 and 0.40 in CASC-J9. This resulted in increased average CPU times in the FNN problem category, e.g., Vampire 4.1's average CPU time jumped from 24.4s to

ATP System	FNT /200	Avg CPU	Mdls out	Sot AC	μ Eff.	Core usage	New /2	FNN /100	FNQ /100
Vampire 4.3	191	45.0	191	0.31	176	0.96	1	96	95
iProver 2.8	137	27.9	137	0.24	326	2.91	1	80	57
CVC4 1.6pre	116	29.5	116	0.23	208	0.83	1	69	47
E 2.2pre	38	21.1	38	0.25	105	0.94	0	28	24
Geo-III 2018C	38	23.5	38	0.20	113	0.86	1	14	10
Demonstration division									
Vampire 4.1	188	48.2	186	0.30	362	0.95	1	94	94

Table 8
FNT division results

70.9s.

Apart from the CPU times, the results parallel those of CASC-26, with the Vampires first, followed by iProver, CVC4, and E. Similarly, the SotACs, efficiencies, and core usages are pretty much the same. Finally, the Vampires again showed stronger performance in the FNQ problem category. The reader is referred to [20] for explanations. As was noted in Section 2.2, two problems of rating 1.0 were made eligible and used in the FNQ problem category. None of the systems solved either of them.

The individual problem results show that 7 problems were unsolved, 7 problems were solved by all the systems, and 6 problems were solved by only one system. Of the 6 unique solutions, 4 were by Vampire 4.3, and 2 by Vampire 4.1. A portfolio approach does not help.

3.5. The EPR Division

Table 9 summarizes the results of the EPR division. The winner, iProver 2.8, had better performance than its predecessor iProver 2.6. The main improvement in iProver 2.8 was the use of complete lookahead literal selection in the instantiation and resolution modules. If multiple literals are eligible for selection in the given clause, complete lookahead literal selection selects the literal for which the complexity of the inferred clauses, e.g., the number of inferred clauses, the number of symbols in inferred clauses etc., is minimal (similar to [4]). There is a significant gap from iProver and Vampire to the lower ranked systems, illustrating that special techniques are required for the EPR division.

It was pleasing to see that all the systems output proofs and models. However, both iProvers and Vampire solved some EPT problems without proofs. In all cases it was because the time limit was reached during proof production or output. For the one problem for which Vampire did not output a proof (`MSC015-1.022`), the proof was found after 280s, and when the time limit was reached the part of the proof that had been output already contained over 2.5 million inference steps. Subsequent testing revealed that the completed proof has over 6 million inference steps. Processing such large proofs, e.g., proof verification, is a challenge for ATP research. For the higher ranked systems the problem category rankings align with the overall ranking. For Geo-III and Leo-III there is a clear preference for the EPS and EPT problem categories, respectively.

The individual problem results show that 6 problems were unsolved, no problems were solved by all the systems, and 15 problems were solved by only one system. Of the 15 unique solutions, 10 were by Vampire, 4 by iProver 2.8, and 1 by iProver 2.6. A simple portfolio of iProver 2.8 and Vampire, giving each 150s, would solve 142 problems.

3.6. The LTB Division

Table 10 summarizes the results of the LTB division. The winner, MaLARea, was the only system to take full advantage of the features of the division, viz. learning from the training data, making multiple attempts on problems, learning from proofs found, and using all cores. Before starting on the competition problems MaLARea used the training data for learning axiom selection, and augmented that with incremental learning of axiom selection after each competition problem was solved. MaLARea made two passes through the problems (in principle it can make more, but the time limit of the division prevented another pass).

ATP System	EPR /150	Avg CPU	Pr out	Mo	Sot AC	μ Eff.	Core usage	EPT /125	EPS /25
iProver 2.8	133	29.2	✓	✓	0.34	169	0.97	108	25
Vampire 4.3	128	45.8	✓	✓	0.36	171	0.95	104	24
E 2.2.pre	27	47.6	✓	✓	0.23	64	0.87	19	9
Geo-III 2018C	10	41.7	✓	✓	0.23	44	0.76	0	10
Demonstration division									
iProver 2.6	126	37.5	✓	✓	0.32	163	0.96	101	25
Leo-III 1.3	17	124.1	✓	✓	0.24	4	2.57	17	0

Table 9
EPR division results

In the first pass it solved 828 problems, and in the second pass it solved another 48 problems by using higher time limits for each problem. MaLARea took advantage of the multiple cores by running various k-NN learners/premise-selectors and various E strategies in parallel.

Grackle was the only other system to make use of the training data. It used the training problems (but not the solutions) to develop strategies for E. It then used all the strategies to attempt each problem using E. Four problem-strategy pairs were run in parallel with a 20s wall clock time limit per pair, to make use of the four cores available.

Vampire 4.0 had a bug that caused it to terminate after 100000s wall clock time, at which point it had attempted only 3553 of the problems. Its success rate when it terminated was higher than that of Vampire 4.3 - 16% of attempted problems solved vs. 15%. Without the bug it might have outperformed Vampire 4.3.

ATP System	LTB /5000	Avg WC	Avg CPU	Prfs out	Sot AC	WC μEff.	Core usage
MaLARea 0.6	876	4.7	16.1	876	0.35	17	3.2
Vampire 4.3	757	1.8	6.0	757	0.25	97	2.1
iProver 2.8	613	14.1	55.9	613	0.21	2	4.0
E 2.2pre	458	16.1	7.4	458	0.19	13	0.5
Grackle 0.1	379	11.7	46.4	379	0.19	3	4.0
Demonstration division							
Vampire 4.0	594	2.9	10.9	594	0.24	88	2.8

Table 10
LTB division results

It is clear that the LTB problems were hard for the systems. The wall clock efficiency values are low compared to the other divisions of the competition, due to the low numbers of problems solved. The wall clock efficiency values are also highly varied. Vampire has higher efficiency due to its low wall clock times, helped by it's reasonable use of the cores available. Despite making maximal use of the cores available, iProver and Grackle have low efficiencies because of their higher wall clock times.

The individual problem results show that 4069 problems were unsolved, 249 problems were solved by all the systems, and 159 problems were solved by only one system. Of the 159 unique solutions, 131 were by MaLARea, 14 by Vampire 4.0, 13 by Vampire 4.2, and 1 by E. It is hard to judge whether or not a portfolio approach would help here, as the various systems impose different internal time limits on proof attempts.

4. System Descriptions

Three new systems were entered into the FOF division of CASC-J9: the CSE family consisting of CSE 1.0, CSE 1.1, and CSE_E 1.0; nanoCoP 1.1; and Twee 2.2. The following brief descriptions of these new systems were written by their developers.

CSE 1.0, **CSE 1.1**, and **CSE_E 1.0** are automated theorem provers for first-order logic. CSE is based mainly on a novel inference mechanism called Contradiction Separation Based Dynamic Multi-Clause Synergized Automated Deduction (S-CS) [23]. The S-CS inference rule takes two or more clauses as input, selects a subset of the literals from each input clause to build a contradictory set of sub-clauses, and infers the disjunction of the non-selected literals of the input clauses (binary resolution is thus just a special case). The number of clauses and the numbers of selected literals per inference are not fixed, and are adjusted dynamically for each inference. CSE has a number of strategies, including clause selection, literal selection, weight

strategy, and so on. CSE 1.0 and CSE 1.1 differ in the way that the contradiction in the sub-clauses is established. In CSE_E, E 2.1 [14] then CSE 1.0 are applied to the given problem. If either prover solves the problem, then the proof process completes. If not, some clauses inferred by CSE are fed to E as lemmas for further proof search. CSE is implemented mainly in C++, with Java being used for job dispatch in CSE_E. *Acknowledgement:* Development of the CSE family has been partially supported by the National Natural Science Foundation of China (NSFC) (Grant No.61673320) and the Fundamental Research Funds for the Central Universities in China (Grant No.2682018ZT10).

nanoCoP 1.1 [9] is an automated theorem prover for classical first-order logic with equality. It is a very compact implementation of the non-clausal connection calculus [8], and the core prover consists of only a few lines of Prolog code. The non-clausal connection calculus generalizes the clausal connection calculus by adding a decomposition rule and generalizing the extension rule to non-clausal formulae. So far, all popular and efficient proof calculi translate FOF problems into clausal form. This modifies the structure of the given formula, and it is difficult to translate the (clausal) proof back into a FOF proof. Using non-clausal matrices, the proof search of nanoCoP works directly on the original structure of the FOF problem, which avoids translation into and back from clausal form. This combines the advantages of more *natural* non-clausal sequent or tableau provers with the goal-oriented *efficiency* of connection provers. Additional inference rules and techniques include regularity, lemmata, restricted backtracking, and a fixed strategy scheduling [11]. The calculus and the implementation can be extended to first-order intuitionistic and several first-order modal logics [10].

Twee 2.2 is an equational theorem prover. It combines unfailing completion with strong criteria for checking redundancy of critical pairs, including ground joinability testing [7] and a novel test based on connectedness [1]. Twee is strong on problems with many unorientable equations or many critical pairs. Its main weaknesses is its simplistic search strategy: critical pairs are considered in order of weight, terms are always ordered by Knuth-Bendix ordering (KBO) with each function symbol having a weight of 1, and the proof search is not goal-directed. Twee converts FOF problems into clause normal form, discards non-Horn clauses, and then reasons about the Horn clauses by encoding them as equations; the technique is described in [3]. The heuristics described in [3] were inadvertently omitted from the CASC version of Twee - happily, this *improved* Twee's performance. As expected, Twee performed best in equation-heavy domains. Of the problems that only Twee was able to solve, all were heavily equational, none were unit equality, and only one was non-Horn. Twee is implemented in Haskell. It features a fast rewriting engine, compact storage of queued critical pairs, and an LCF-style kernel that certifies all generated rewrite rules as well as producing human-readable proofs.

5. Conclusion

CASC-J9 was the twenty-third large scale competition for fully automatic, classical logic ATP systems. The competition provided exposure for system builders both within and beyond the ATP community, and provided an overview of the implementation state of fully automatic classical logic ATP systems. CASC-J9 fulfilled its objectives by evaluating the relative abilities of current ATP systems, and stimulating development and interest in ATP.

The highlights of CASC-J9 were disappointing turnouts in the THF and TFA divisions (i.e., lowlights), three interesting new systems in the FOF division, and the re-emergence of MaLARea in the LTB division.

While the design of CASC is mature and stable, each year's experiences lead to ideas for changes and improvements. Some changes that are being considered for CASC-27 are:

- Two versions of the FOF and FNT divisions, one with a CPU time limit for each problem (as is currently the case), and another with a wall clock time limit for each problem, to promote the use of all cores on the CPU.
- The UEQ (Unit EQuality) division will come out of its hiatus state.
- A multi-format LTB category in which each problem is encoded in several of the available TPTP languages – TH1, TH0, TF1, TF0, and FOF. Systems will be able to attempt whichever versions they want, and a solution to any version will constitute a solution to the problem.

As always, the ongoing success and utility of CASC depends on ongoing contributions of problems to the TPTP. The automated reasoning community is encouraged to continue making contributions of all types of problems.

References

[1] L. Bachmair and N. Dershowitz. Critical Pair Criteria for Completion. *Journal of Symbolic Computation*, 6(1):1–18, 1988.

[2] J. Blanchette, P. Fontaine, S. Schulz, and U. Waldmann. Towards Strong Higher-Order Automation for Fast Interactive Verification. In G. Reger and D. Trayfel, editors, *Proceedings of the 1st International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements*, number 51 in EPiC Series in Computing, pages 16–23. EasyChair Publications, 2017.

[3] K. Claessen and N. Smallbone. Efficient Encodings of First-Order Horn Formulas in Equational Logic. In D. Galmiche, S. Schulz, and R. Sebastiani, editors, *Proceedings of the 9th International Joint Conference on Automated Reasoning*, number 10900 in Lecture Notes in Computer Science, pages 388–404, 2018.

[4] K. Hoder, G. Reger, M. Suda, and A. Voronkov. Selecting the Selection. In N. Olivetti and A. Tiwari, editors, *Proceedings of the 8th International Joint Conference on Automated Reasoning*, number 9706 in Lecture Notes in Artificial Intelligence, pages 313–329, 2016.

[5] B. Kiesl, M. Suda, M. Seidl, H. Tompits, and A. Biere. Blocked Clauses in First-Order Logic. In T. Eiter and D. Sands, editors, *Proceedings of the 21st International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, number 46 in EPiC Series in Computing, pages 31–48. EasyChair Publications, 2017.

[6] R. Kumar, M. Myreen, M. Norrish, and S. Owens. CakeML: A Verified Implementation of ML. In P Sewell, editor, *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 179–191. ACM Press, 2014.

[7] U. Martin and T. Nipkow. Ordered Rewriting and Confluence. In M.E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction*, number 449 in Lecture Notes in Artificial Intelligence, pages 366–380. Springer-Verlag, 1990.

[8] J. Otten. A Non-clausal Connection Calculus. In K. Brünnler and G. Metcalfe, editors, *Proceedings of the 20th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, number 6793 in Lecture Notes in Artificial Intelligence, pages 226–241. Springer-Verlag, 2011.

[9] J. Otten. nanoCoP: A Non-clausal Connection Prover. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Proceedings of the 7th International Joint Conference on Automated Reasoning*, number 8562 in Lecture Notes in Artificial Intelligence, pages 302–312, 2016.

[10] J. Otten. Non-clausal Connection Calculi for Non-classical Logics. In C. Nalon and R. Schmidt, editors, *Proceedings of the 26th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, number 10501 in Lecture Notes in Artificial Intelligence, pages 209–227. Springer-Verlag, 2017.

[11] J. Otten. Proof Search Optimizations for Non-clausal Connection Calculi. In B. Konev, P. Rümmer, and J. Urban, editors, *Proceedings of the 6th Workshop on Practical Aspects of Automated Reasoning*, number 2162 in CEUR Workshop Proceedings, pages 49–57, 2018.

[12] G. Reger and M. Suda. Set of Support for Theory Reasoning. In T. Eiter, D. Sands, G. Sutcliffe, and A. Voronkov, editors, *Proceedings of the IWIL Workshop and LPAR Short Presentations*, number 1 in Kalpa Publications in Computing, pages 124–134, 2017.

[13] G. Reger, M. Suda, and A. Voronkov. Unification with Abstraction and Theory Instantiation in Saturation-Based Reasoning. In D. Beyer and M. Huisman, editors, *Proceedings of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, number 10805 in Lecture Notes in Computer Science, pages 3–22. Springer-Verlag, 2018.

[14] S. Schulz. Simple and Efficient Clause Subsumption with Feature Vector Indexing. In M.P. Bonacina and M. Stickel, editors, *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*, number 7788 in Lecture Notes in Artificial Intelligence, pages 45–67. Springer-Verlag, 2013.

[15] P. Smith. *An Introduction to Goedel's Theorems*. Cambridge University Press, 2007.

[16] A. Stump, G. Sutcliffe, and C. Tinelli. StarExec: a Cross-Community Infrastructure for Logic Solving. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Proceedings of the 7th International Joint Conference on Automated Reasoning*, number 8562 in Lecture Notes in Artificial Intelligence, pages 367–373, 2014.

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

[18] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[19] G. Sutcliffe. The CADE ATP System Competition - CASC. *AI Magazine*, 37(2):99–101, 2016.

[20] G. Sutcliffe. The CADE-26 Automated Theorem Proving System Competition - CASC-26. *AI Communications*, 30(6):419–432, 2017.

[21] G. Sutcliffe. Proceedings of the 9th IJCAR ATP System Competition. Oxford, United Kingdom, 2018. http://www.tptp.org/CASC/J9/Proceedings.pdf.

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

[23] Y. Xu, S. Liu, J. Chen, X. Zhong, and X. He. Contradiction Separation Based Dynamic Multi-clause Synergized Automated Deduction. *Information Sciences*, 462:93–113, 2018.