The 10th IJCAR Automated Theorem Proving System Competition – CASC-J10

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

Abstract. The CADE ATP System Competition (CASC) is the annual evaluation of fully automatic, classical logic Automated Theorem Proving (ATP) systems. CASC-J10 was the twenty-fifth competition in the CASC series. Twenty-four 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) [26] 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 easily and usefully deployed in applications, provide an inspiring environment for personal interaction between ATP researchers, and expose ATP systems within and beyond the ATP community. CASC-J10 was held on 2nd July 2020, as part of the 10th International Joint Conference on Automated Reasoning[1], online due to the COVID-19 pandemic. CASC-J10 was the twenty-fifth competition in the CASC series; see [29] and citations therein for information about previous competitions. The CASC-J10 web site provides access to all the competition resources: http://www.tptp.org/CASC/J10.

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 a 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-J10.

Twenty-four ATP systems and system variants competed in the various divisions. The division winners of CASC-27 (the previous CASC) were automatically entered into the demonstration division, 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. The systems, the divisions in which they were entered, and their entrants, are listed in Table 2. System descriptions are in [28] and on the CASC-J10 web site.

[1]CADE was a constituent of the 10th International Joint Conference on Automated Reasoning, hence "J10" for the "10th Joint" conference.

Table 1
Divisions and Problem categories

Division	Problems	Problem categories. The example problems can be viewed online 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** – THF with **N**o **E**quality, e.g., `NUM738^1`. **TEQ** – THF 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 without a conjecture).	**FNN** – **FNT** with **N**o equality, e.g., `KRS173+1`. **FNQ** – **FNT** with e**Q**uality, e.g., `MGT033+2`.
UEQ	**U**nit **EQ**uality theorems in clause normal form (unsatisfiable clause sets).	E.g., `RNG026-7`.
LTB	Theorems (axioms with a provable conjecture) from **L**arge **T**heories, presented in **B**atches. A large theory typically has many functions and predicates, and large theory problems typically have many axioms of which only a few are required for a proof of the conjecture. 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. Each problem category might be accompanied by a set of training problems and their solutions that can be used for ATP system tuning during (typically at the start of) the competition.	**HL4** – Problems exported from **HOL4** [22]. This category was accompanied by training data. Eight versions of each problem were provided - two first-order (FOF) versions, two monomorphic typed first-order (TF0) versions, one polymorphic typed first-order (TF1) version, two monomorphic typed higher-order (TH0) versions, and one polymorphic typed higher-order (TH1) version. A solution to any version counted as a solution to the problem. The systems could attempt the problems and problem versions in any order, could attempt them concurrently, and could attempt each one multiple times. See Section 2.2 for details of the problems.

CASC-J10 was organized by Geoff Sutcliffe, and overseen by a panel consisting of Pascal Fontaine, Andre Platzer, and Christoph Weidenbach. The competition was run on computers provided by the StarExec project [23] at the University of Miami.

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-J10 design and organization is provided here; the details are in [28] and on the CASC-J10 web site. Important changes for CASC-J10 were (for readers already familiar with the general design of CASC; others can skip to Section 2.1):

- All divisions used a wall clock time limit, to promote use of all the cores on the CPU.
- The EPR division was put on hiatus.
- A new SotAC measure was adopted, to allow comparison of SotAC values between CASC editions.

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 (in the non-LTB divisions) or one batch (in the LTB division) at a time. Any command line parameters have to be the same for all problems/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, UEQ, and LTB divisions, and theorems to the systems in the FNT division. Claiming to have found a proof of a non-theorem or a disproof of a theorem indicates unsoundness. One system was found to be unsound before CASC-J10, and was repaired in time for the competition.

The systems in the competition divisions are ranked according to the number of problems solved with an

Table 2
The ATP systems and entrants

ATP System	Divisions	Entrant	Entrant's Affiliation
ATPBoost 1.0	LTB (demo)	Bartosz Piotrowski	University of Warsaw & Czech Technical University in Prague
CSE 1.3	FOF	Feng Cao	Southwest Jiaotong University
CSE_E 1.2	FOF	Feng Cao	Southwest Jiaotong University
CVC4 1.8	THF TFA FOF FNT	Andrew Reynolds	University of Iowa
E 2.4	UEQ (demo)	CASC	CASC-27 winner
E 2.5	FOF FNT UEQ LTB	Stephan Schulz	DHBW Stuttgart
Enigma 0.5.1	FOF	Jan Jakubuv	Czech Technical University in Prague
Etableau 0.2	FOF UEQ	John Hester	University of Florida
GKC 0.5	FOF UEQ LTB	Tanel Tammet	Tallinn University of Technology
iProver 3.3	FOF FNT (demo) EPR UEQ LTB	Konstantin Korovin	University of Manchester
lazyCoP 0.1	FOF UEQ	Michael Rawson	University of Manchester
leanCoP 2.2	FOF	Jens Otten	University of Oslo
LEO-II 1.7.0	THF	Alexander Steen	University of Luxembourg
Leo-III 1.4	LTB (demo)	CASC	CASC-27 winner
Leo-III 1.5	THF LTB	Alexander Steen	University of Luxembourg
MaLARea 0.9	LTB (demo)	Josef Urban	Czech Technical University in Prague
Prover9 1109a	FOF (demo)	CASC	CASC fixed point
PyRes 1.3	FOF FNT EPR	Stephan Schulz	DHBW Stuttgart
Satallax 3.4	THF (demo)	CASC	CASC-27 winner
Satallax 3.5	THF	Michael Färber	Inria, LSV, ENS Paris-Saclay
Twee 2.2.1	FOF UEQ	Nick Smallbone	Chalmers University of Technology
Vampire 4.4	TFA FOF FNT (all demo)	CASC	CASC-27 winner
Vampire 4.5	THF TFA FOF FNT UEQ	Giles Reger	University of Manchester
Zipperposition 2.0	THF FOF LTB	Petar Vukmirovic	Vrije Universiteit Amsterdam

acceptable proof/model output (see [28] for an explanation of what is "acceptable"). Ties are broken according to the average time taken over problems solved. Trophies are awarded to the competition divisions' winners.

In addition to the ranking, three other measures are presented in the results: The *state-of-the-art contribution* (SotAC) quantifies the unique abilities of the systems (excluding the previous year's winners that are earlier versions of competing systems). For each problem solved by a system, its SotAC for the problem is the fraction of systems that do not solve the problem, and a system's overall SotAC is the average over the problems it solves but which are not solved by all the systems. The *core usage* measures the extent to which the systems take advantage of multiple cores. It is the average of the ratios of CPU time to wall clock time used, over the problems solved. The competition ran on octa-core computers, thus the maximal core usage was 8.0. 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 reciprocal of the time taken to solve it), multiplied by the fraction of problems solved.

2.2. *The Competition Problems*

2.2.1. *Problems for the TPTP-based Divisions*

The problems for the THF, TFA, FOF, FNT, and UEQ divisions were taken from the Thousands of Problems for Theorem Provers (TPTP) problem library [25], v7.4.0. 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) [31]. Difficult problems with ratings in the range 0.21 to 0.99 are eligible. Problems of lesser and greater ratings are made eligible if there are not enough problems with ratings in that range. This was unnecessary in CASC-J10. 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 guided by 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 [24].
- 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.

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.

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-J10 had only one problem category (and hence only one source): the HL4 problem category, which used a set of 13431 problems exported from HOL4 [6, 22].[2] In CASC-27 the "bushy" variants of a previous export of problems were used, while in CASC-J10 the "chainy" variants of a new export were used. (See [1] for an explanation of how the two variants are formed.) In contrast to the bushy problems of CASC-27 that had from (only) 13 to 1389 axioms, the chainy problems had from 77 axioms to 126797 axioms, Recall from Section 1, eight versions of each problem were provided - two first-order (FOF) versions, two monomorphic typed first-order (TF0) versions, one polymorphic typed first-order (TF1) version, two monomorphic typed higher-order (TH0) versions, and one polymorphic typed higher-order (TH1) version. Solving any version of a problem counted as a solution to the problem.

In order to gauge the problems' difficulty, various ATP systems were run on the 107448 problem versions using a 60s CPU time limit. Of the 107448 problems, a proof for at least one version was found for 8559 problems, and at least one proof for every version was found for 740 problems. This provided assurance that the problems were of a reasonable difficulty. The competition used the chainy problems corresponding to the bushy problems used in CASC-27, i.e., 1000 problems and their solutions were made available as pre-release training data before the competition for developers to test their systems, another 1000 problems and their solutions were added as additional training data in the competition, and 10000 of the remaining 11431 problems were used in the competition.

The problems are given to the ATP systems in TPTP format, in the natural order of their source, i.e., for CASC-J10 in the order of their export from HOL4. The problem batch was *unordered*, which meant that the ATP systems could attempt the problems and problem versions in any order, could attempt them concurrently, and could attempt each one multiple times. This provided increased opportunities for sharing logical and control results between proof attempts.

[2]Special thanks to Thibault Gauthier and Chad Brown in Josef Urban's AI4REASON project at the Czech Institute of Informatics, Robotics and Cybernetics, who did the exports.

Table 3

Numbers of eligible and used problems

Division	THF		TFA		FOF		FNT		UEQ	LTB
Category	TNE	TEQ	TFI	TFE	FNE	FEQ	FNN	FNQ		HL4
Eligible	118	576	253	39	404	3625	90	192	536	13431
Usable	118	576	253	38	114	877	87	192	536	11431
Used	100	400	225	25	100	400	75	175	250	10000
New	0	51	0	0	0	52	6	2	0	-

2.3. Resources

The competition computers had an octa-core Intel(R) Xeon(R) E5-2667 3.20GHz CPU, 128GB memory, and ran the CentOS Linux release 7.4.1708 operating system kernel 3.10.0-957.12.2.el7.x86_64. One ATP system ran on one CPU at a time. Systems could use all the cores on the CPU.

In the non-LTB divisions a 120s wall clock time limit was imposed for each problem, and no CPU limits were imposed (so that it could be advantageous to use all the cores on the CPU). In the LTB division a 172800s wall clock time limit was imposed for the HL4 problem category, giving an average of 17.28s per problem. There was no wall clock time limit for each problem, and no CPU time limits. Time spent before starting the first problem in the batch, e.g., tuning on the training data 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 computers can be supplied by the entrant. The CASC-J10 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 an acceptable proof/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 wall clock time over the problems solved, the number of proofs/models 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-27 winner is *emphasized*. Detailed results, including the systems' output files, are available from the CASC-J10 web site.

3.1. The THF Division

Table 4 summarizes the results of the THF division. Zipperposition dominated the division. This excellent performance is of interest, and a brief system description of Zipperposition is provided in Section 4. In CASC-27 the THF division was won by Satallax 3.4, followed by Leo-III 1.4, Zipperposition 1.5, and Vampire 4.4. In CASC-J10 Satallax 3.4 still outperformed the new versions of those systems, and also the new Satallax 3.5. The Satallax developer explained that the changes from Satallax 3.4 to 3.5 were expected to "only minimally improve the performance", and that Satallax 3.4 outperformed 3.5 probably because of statistical fluctuations in the scheduling and the use of a different version of E as a subsystem. While the new versions of the other systems were still outperformed by Satallax 3.4, each did have some improvements:

• Zipperposition 2.0 improved on version 1.5 through a new unification algorithm that relies heavily on decidable fragments of higher-order unification [33], enhanced boolean reasoning capabilities [34], some use of Bhayat and Reger's combinatory calculus [3], and new heuristics tuned to solve problems from diverse subsets of the TPTP.

• Vampire 4.5 improved on version 4.4 through a rewrite based on a new superposition calculus [3], which uses a KBO-like ordering [4] for orienting combinator equations. Version 4.5 also has improved support for reasoning about booleans and choice.

• Leo-III 1.5 improved on version 1.4 through a change to a "nasty" parameter that caused version 1.4 to add redundancy in the search space, and very simple strategy scheduling that runs three different parameter settings in parallel. Leo-III does not have optimized parameter settings for strategy scheduling, but, as the developer said, it's "still better than none".

Table 4
THF division results

ATP System	THF	Avg	Prfs	Sot	μ	Core	TNE	TEQ
	/500	WC	out	AC	Eff.	usage	/100	/400
Zipperposition 2.0	424	14.8	424	0.35	382	4.44	83	341
Satallax 3.5	319	20.1	319	0.23	212	0.97	71	248
Vampire 4.5	299	3.7	299	0.19	400	4.57	64	235
Leo-III 1.5	287	14.7	287	0.17	90	2.47	65	222
CVC4 1.8	194	8.8	194	0.10	277	0.86	43	151
LEO-II 1.7.0	112	7.4	111	0.04	159	0.91	35	77
Demonstration division								
Satallax 3.4	323	20.0	323	-	214	0.99	72	251

After the competition the developers of Vampire reported that 320 problems could be solved by running the system with each strategy's time slice increased by a small amount. Analysis revealed that in multi-core mode the system's limited resource strategy [19] deleted more clauses because the search strategies made slower progress. The moral of the story is that, as one of the developers said, "running on multiple cores doesn't always just make things run faster!!".

Zipperposition and the Satallaxes were most effective at solving problems up to the time limit, particularly in the TEQ problem category - Zipperposition solved 27 problems in more than 60s, and Satallax 3.5 solved 28. Zipperposition's higher SotAC is due to its large number of unique solutions, as noted in the individual problem analysis below. Vampire has a higher efficiency due to it's lower time usage. Zipperposition and Vampire also made most use of the multiple cores available. Zipperposition used the cores by scheduling 50 strategies using the Python scheduler, which chose a batch of strategies at a time to execute on as many cores as possible. Vampire's use of multiple cores is explained in Section 3.3. The systems' performances in the two problem categories align with the overall ranking.

The individual problem results show that 39 problems were unsolved, 46 problems were solved by all the systems, 38 problems were solved by only one system, and 9 problems were solved by only the two versions of Satallax (these 9 are counted as unique solutions for Satallax 3.5). Of the 47 unique solutions, 28 were by Zipperposition, 9 by Satallax 3.5, 5 by CVC4, 2 each by Leo-III, and 1 each by Satallax 3.4, Vampire, and LEO-II. Many of the unique solutions took longer than average, so a simple portfolio that splits up the time equally is not effective. A portfolio giving Zipperposition 96s and Vampire 24s would solve 432 problems, which is a minimal gain.

After the competition the developer of Zipperposition found a bug in the implementation of a new side condition of the system's lambdaSup rule [2], which would lead to incorrect proofs. As the rule is used in only rare cases, this bug had escaped the developer's rigorous testing. An investigation showed that the rule is used in proofs of only two TPTP problems, neither of which were selected for the competition. All of Zipperposition's proofs in the competition were checked to ensure that the rule was not used. The competition panel agreed that the bug did not manifest itself in the competition, and thus according to the competition design no penalty was applied.

3.2. The TFA Division

Table 5 summarizes the results of the TFA division. The winner was the new Vampire 4.5, with the CASC-27 winner Vampire 4.4 close behind. The results parallel those of CASC-27, just with the systems' version numbers updated. The reader is thus referred to [29] for some insights into the systems' performances.

Vampire's SotAC was higher due to a large number of unique solutions, as noted in the individual problem analysis below. Vampire 4.5's efficiency was higher due to its lower time usage. None of the systems took advantage of the multiple cores. For Vampire this is simply an artifact of solving problems fast with one of the first strategies started. In CVC4's case, this is by design.

The individual problem results show that 16 problems were unsolved, 143 problems were solved by all the systems, 43 problems were solved by only one system, and 35 problems were solved by only the two versions of Vampire (these 35 are counted as unique solutions for Vampire 4.5). Of the 78 unique solutions, 41 were by Vampire 4.5, 31 by CVC4, and 6 by Vampire 4.4. A portfolio of CVC4 and Vampire 4.5, giving

Table 5

TFA division results

ATP System	TFA /250	Avg WC	Prfs out	Sot AC	μ Eff.	Core usage	TFI /125	TFE /75
Vampire 4.5	191	4.6	191	0.20	538	0.93	100	68
CVC4 1.8	187	17.9	187	0.18	302	0.83	87	72
Demonstration division								
Vampire 4.4	190	17.9	190	-	284	0.90	100	69

CVC4 95s and Vampire 25s, would solve 220 problems, i.e., almost complete coverage of the division.

As was the case in the TFA divisions of CASC-27 and CASC-J9, it was rather disappointing that effectively only two systems participated. The competition organizer had been hopeful that the TFA language and logic would be a sweet spot for ATP users [30]. Recent indications are that THF might be more popular, especially with the "hammers" [5, 13]. The TFA division will be placed in hiatus for CASC-28, awaiting further interest and development.

3.3. The FOF Division

Table 6 summarizes the results of the FOF division. The winner was the new Vampire 4.5. As has been the case for the past few years, Vampire performed strongly in terms of all measures - problems solved, average time, SotAC, and efficiency. The main changes to Vampire since version 4.4 were a major reimplementation of clause selection and proof recording that had a non-trivial downstream impact on proof search, running strategies in parallel, a much more aggressive normalisation and evaluation approach, the addition of subsumption demodulation [10], and layered clause selection [9] (however, the latter two changes were used minimally in the competition because their full power was not used in the main strategy schedules). The strong performance of Enigma is noteworthy. In CASC-27 Enigma 0.4 ran as a demonstration division system, and came in after E and CSE_E, which it now outperformed. The main improvements in Enigma 0.5.1 were a repair to a bug that caused it to crash during axiom selection, anonymous guidance that makes its machine learning phase invariant to consistent symbol renaming, and new models trained for clause selection. A brief system description of Enigma is provided in Section 4.

In CASC-27 the results divided the systems into four groups: the Vampires, the E-based systems, the group of CVC4/iProver/GKC, and the rest of the systems. A similar grouping had been noted in CASC-J9. In CASC-J10 the grouping was less pronounced, with Enigma filling the gap between the Vampires and the E-based systems, stronger performances by CVC4/iProver/GKC, and Zipperposition coming between CVC4/iProver/GKC and the rest of the systems. This smoother transition is evidence of continued interest and development of ATP systems for first-order logic.

After the competition one of the developers of Vampire noticed a coding mistake that had led to Vampire not using the full time limit: Vampire runs sets of schedules, but the piece of code that decided whether to run the next set got confused between "time spent" and "time remaining". This meant that after half of the time it would not run any more schedules, and as a result only four problems were solved after 60s. Interestingly, this shows that only the first 60s really matter to Vampire!

Two lower ranked systems with interesting results in the FOF division are leanCoP and lazyCoP. leanCoP, like Prover9, has not changed for many years. It, along with Prover9, provides a fixed point against which progress can be judged. For example, leanCoP beat CSE 1.2 in CASC-27, but was beaten by CSE 1.3 this year. lazyCoP is interesting as it was the only true newcomer to the competition. While it's performance was rather weak, and it did not output complete proofs (its proofs omitted the FOF to CNF conversion steps), it certainly did not disgrace itself. Further, as noted in Section 3.5, lazyCoP could make a useful contribution to a portfolio system for UEQ problems. A brief system description of lazyCoP is provided in Section 4.

The SotACs align with the ranking. Enigma had lower efficiency because is has a startup time of around 2s and thus no problems are solved very quickly. GKC had higher efficiency because it solved many problems very quickly. In terms of core usage, the systems split clearly into two groups: Vampire, Enigma, iProver, GKC, lazyCoP, and Twee had higher core usage, while the others' core usages were around 1.00. The developers provided short explanations of how their systems used the cores:

Table 6

FOF division results

ATP System	FOF /500	Avg WC	Prfs out	Sot AC	μ Eff.	Core usage	New /52	FNE /100	FEQ /400
Vampire 4.5	429	3.9	429	0.43	669	3.48	35	90	339
Enigma 0.5.1	401	10.2	401	0.37	183	4.40	25	85	316
E 2.5	351	15.2	351	0.30	398	0.95	22	83	268
CSE_E 1.2	316	9.0	316	0.26	387	1.05	21	77	239
iProver 3.3	312	10.3	312	0.25	296	5.99	26	85	227
GKC 0.5.1	289	3.1	289	0.22	406	4.13	19	75	214
CVC4 1.8	275	19.7	275	0.22	270	0.89	17	53	222
Zipperposition 2.0	237	22.6	237	0.16	174	0.96	17	57	180
Etableau 0.2	162	9.1	162	0.10	200	0.96	15	45	117
CSE 1.3	124	42.0	124	0.06	67	1.02	7	45	79
leanCoP 2.2	111	32.0	111	0.06	53	0.77	7	37	74
lazyCoP 0.1	94	8.8	0	0.04	111	4.45	4	38	56
Twee 2.2.1	68	9.8	68	0.04	70	5.33	7	10	58
PyRes 1.3	26	21.4	26	0.01	19	1.01	0	8	18
Demonstration division									
Vampire 4.4	416	12.1	416	-	440	0.92	32	90	326
Prover9 1109a	146	12.9	146	0.09	123	0.91	9	25	121

• Vampire 4.5 created a schedule of strategies with time limits, then started six processes each running one strategy. When a strategy ended without success the process took the next strategy from the schedule.
• Enigma first used one core to translate the problem to CNF and run LightGBM axiom selection. After that an instance of E was run on each available core: one in standard auto-schedule mode, another with the LightGBM axiom selection applied, and the rest with learned strategies (see Section 4).
• iProver created a schedule of 11 strategies with time limits, then started a process on each core. The first two processes ran the first two strategies for the full time limit, the other six processes ran the remaining nine strategies. If a strategy used up it's time allocation without finding a proof it was replaced by the next strategy in the schedule.
• GKC created a schedule of strategies with time limits, then started a process on each core running a subset of the strategies.
• lazyCoP maintained a priority queue of tableaux to be expanded. A worker thread was maintained on each core, independently exploring the tableau space in an A* fashion.
• Twee spawned a prover instance for each core, each using a different strategy.

There were 52 new problems in the FOF division, all in the FEQ problem category. Forty of them are from the TPTP's ITP domain, which contains problems that were also used in the LTB division. Of the 52 new problems, 13, all ITP problems, were not solved by any system. Those problems were eligible for the competition because they had been solved in earlier testing: E 2.4 had found that 7 of the 13 have contradictory axioms (see Section 3.6 for a discussion of this), and 4 others had been solved by iProver ... in disproving mode! Six new problems were solved by effectively only one system, 3 by both the Vampires, and 1 by each of Enigma, E, and CVC4.

The individual problem results show that 33 problems were unsolved, 1 problem was solved by all the systems, 18 problems were solved by only one system, and 13 problems were solved by only the two versions of Vampire (these 13 are counted as unique solutions for Vampire 4.5). Of the 31 unique solutions, 19 were by Vampire 4.5, 6 by Vampire 4.4, 3 by Enigma, and 1 by each of CVC4, E, and Prover9. Enigma provides a useful complement to Vampire 4.5, and a simple portfolio giving 60s to each would solve 445 problems.

3.4. The FNT Division

Table 7 summarizes the results of the FNT division. Vampire 4.5 came out well ahead. The effect of running Vampire on multiple cores is more pronounced for the FNT division because the first strategy is finite model finding that runs with a long time allocation, which holds back other strategies if only one core is used. With multiple cores Vampire can try other strategies in parallel with the finite model finding.

Vampire 4.5 had the highest SotAC due to the large number of problems that only it (8 problems) or it and Vampire 4.4 (37 problems) solved, and higher efficiency due to its lower time usage. Vampire and iProver were the only systems to use multiple cores (as discussed in Section 3.3).

iProver did better in the FNQ problem category than in previous years, thanks to a new system for developing efficient heuristics, along with general improvements in iProver. E and CVC4 inverted the ranking in the FNN problem category, and in both problem categories they solved quite different problems. In the FNN problem category there are 26 problems that E solved and CVC4 did not, and 10 problems that CVC4 solved and E did not. In the FNQ problem category there are 63 problems that CVC4 solved and E did not, and 10 problems that E solved and CVC4 did not. Between them they solved 42 FNN problems and 94 FNQ problems, for a combined total of 136 problems. E performed better in the FNN problem category because there are some classes of problems without equality for which E's calculus is a decision procedure, while for problems with equality the equational system very often has only an infinite saturation. CVC4 performed better in the FNQ category because, like all SMT solvers, it has efficient support for equality reasoning at the quantifier-free level. This is combined with finite model finding for universally quantified formulae (including those with equality).

There were 8 new problems in the division, and most of the systems were able to solve most of them. Six of the new problems are syntactic problems in the FNN problem category, generated using a single binary relation [15]. These were solved by the Vampires, CVC4, and iProver, all of which include a finite model finder. The other two new problems are in software verification [16], one of which has a finite model and was unsolved (but was eligible for the competition because it had been solved in earlier testing by Paradox [8]), and the other of which has only infinite models and was solved by only the Vampires with a saturation.

The individual problem results show that 10 problems were unsolved, 12 problems were solved by all the systems, 10 problems were solved by only one system, and 37 problems were solved by only the two versions of Vampire (these 37 are counted as unique solutions for Vampire 4.5). Of the 47 unique solutions, 45 were by Vampire 4.5, and 1 by each of Vampire 4.4 and iProver. A portfolio approach does not help.

An attentive observer of CASC might wonder why iProver ran in the demonstration division; here's the explanation: During the competition, in the FOF division iProver claimed that one problem was countersatisfiable (a non-theorem), which made iProver unsound wrt the FNT division. The bug was due to a last minute modification in a simplification index, which caused some clauses to be "lost", which in turn caused incompleteness for theorem proving and, correspondingly, unsoundness for disproving. The developer was alerted, and the bug was quickly fixed. The CASC-J10 panel disqualified iProver 3.3 from the FNT division, but agreed to having the fixed version run in the FNT demonstration division.

3.5. The UEQ Division

Table 8 summarizes the results of the UEQ division. The new E 2.5 won, significantly outperforming the previous version 2.4 that won in CASC-27. E 2.5 had a number of improvements over E 2.4, including: classification of clausal problems before expanding equational definitions - this moves equational definition expansion under control of the automatic mode, a stronger rewrite relation that optionally instantiates unbound variables on the right hand side of equations - this makes more instances orientable, and improved generation of strategy schedules with more strategies. Finally, more of the parameter space was evaluated on newer clause evaluation functions to find good strategies. The new version of Twee also beat E 2.4, firstly by fixing a completeness bug that was in Twee 2.2, but mainly by running multiple strategies in parallel.

The SotAC and efficiency values mostly align with the ranking, with the exception of Etableau's higher efficiency thanks to its lower time usage. Only E and Etableau did not use multiple cores; the ways in which the other systems used multiple cores is explained in Section 3.3. The developer of Etableau has promised to "add multiprocessing" for CASC-28.

The individual problem results show that 27 problems were unsolved, 7 problems were solved by all the systems, 26 problems were solved by only one system, and 3 problems were solved by only the two versions of E (these 3 are counted as unique solutions for E 2.5). Of the 29 unique solutions, 12 were by Twee, 11 by E 2.5, 3 by GKC, 2 by E 2.4, and 1 by Vampire. A portfolio giving Twee 49.5s, E 2.5 64s, and lazyCoP 6.5s, would solve 209 problems. This demonstrates how lower ranked systems can be usefully employed alongside top performing systems. In particu-

Table 7
FNT division results

ATP System	FNT /250	Avg WC	Mdls out	Sot AC	μ Eff.	Core usage	New /8	FNN /75	FNQ /175
Vampire 4.5	238	5.7	238	0.50	670	3.34	7	72	166
CVC4 1.8	98	24.8	98	0.13	211	0.91	6	14	84
E 2.5	63	10.2	63	0.08	157	0.97	0	32	31
PyRes 1.3	13	8.0	13	0.00	13	0.97	0	1	12
Demonstration division									
Vampire 4.4	226	8.6	226	-	299	0.98	7	65	161
iProver 3.3	183	9.7	183	0.32	371	5.63	6	56	127

Table 8
UEQ division results

ATP System	UEQ /250	Avg WC	Prfs out	Sot AC	μ Eff.	Core usage
E 2.5	202	7.8	202	0.27	595	0.95
Twee 2.2.1	197	6.0	197	0.26	532	4.41
Vampire 4.5	162	8.4	162	0.16	426	2.59
Etableau 0.2	148	2.6	148	0.14	465	0.93
GKC 0.5.1	128	8.3	128	0.10	289	5.05
iProver 3.3	124	7.4	124	0.09	293	5.31
lazyCoP 0.1	20	0.8	0	0.01	70	2.36
Demonstration division						
E 2.4	185	5.4	185	-	579	0.93

lar, it is nice to see that the lowest ranked lazyCoP can be helpful due to its ability to find some proofs very quickly.

3.6. The LTB Division

Table 9 summarizes the results of the LTB division. The LTB and CAX columns need explanation: After the competition the developer of MaLARea pointed out that some of its proofs (output by E, which is the underlying reasoning system within MaLARea) did not use the conjecture. Closer examination revealed that E had found the problem's axioms to be contradictory, and that MaLARea had "solved" 1532 problems using that contradiction (see the discussion of this issue at the end of this section). The developer of MaLARea explained, "once an unsoundness is found in the axioms included in many problems, MaLARea's learning will exploit it quite mercilessly". Thus the LTB column of Table 9 lists the numbers of problems solved, and the CAX column lists how many of those solutions were by finding the axioms contradictory. Even discounting the proofs by contradictory axioms, the performance of MaLARea is impressive, and had the system been in the competition division it would have won by a large margin.[3]

Table 9
LTB division results

ATP System	LTB /10000	CAX	Avg WC	Sot AC	WC μEff.	Core usage
E 2.5	3393	1	4.6	0.17	127	2.6
iProver 3.3	3163	3	5.5	0.15	98	6.0
Zipperposition 2.0	1699	0	38.5	0.06	31	4.1
Leo-III 1.5	1413	0	20.6	0.06	9	7.4
GKC 0.5.1	493	0	3.7	0.01	33	4.1
Demonstration division						
MaLARea 0.9	7054	1532	3.9	0.48	262	4.7
ATPBoost 1.0	1237	0	0.0	0.04	124	1.0
Leo-III 1.4	134	0	15.7	0.00	1	7.3

The strong performance of MaLARea, particularly in contrast to the performance of MaLARea 0.8 in CASC-27, is interesting. The key strength of MaLARea is it's ability to learn axiom selection from previous proofs [14], which was important in CASC-J10 because the problems had very many axioms of which most were unnecessary for a proof. Learning was done initially from the training data that was provided with the batch, and subsequently from each proof as it was found. While Leo-III 1.4 won the LTB division of CASC-27 by performing strongly on the TH1 v1 bushy problems that had between 7 and 345 axioms, MaLARea performed strongly in CASC-J10 on the FOF v1 chainy problems that had between 103 and 42675 axioms. As the developer of MaLARea said, "The THF people still need to learn some premise selection". In addition to strong axiom selection, MaLARea used specialized large-theory strategies developed automatically by the BliStr

[3]MaLARea was in the demonstration division because the developer heads the group that did the problem export from HOL4.

[32] and BliStrTune [12] systems, and learned internal guidance for E from previous proofs using the ENIGMA system [7]. Thanks to the BliStr/Tune strategies and ENIGMA guidance, some of the proofs found by MaLARea were far from trivial even after axiom selection. The longest proof consists of 358 inferences using 63 axioms out of the 128 selected from the problem's 21820 axioms.

After the competition the developer of ATPBoost realised that he had made a simple wrong assumption that every problem would have at least one axiom that was not in an included axiom file. This caused the machine learning phase after the first pass through the problems to fail, and the system stopped. A version of the system with that mistake corrected solved 1835 problems. Further, the developer had planned to use multiple cores, but he explained that "something went wrong with the way I used the parallel Perl script". These are simple bugs to fix, which provides a pathway forward for an improved entry in CASC-28. The poor performance of GKC was tracked down to a memory handling problem caused by process forks over the very many problems.

The SotAC and efficiency values align with the ranking, except for GKC's higher efficiency thanks to its lower time usage. The core usages are generally good, with Leo-III using almost all the cores on average thanks to its use of multiple external subsystems in parallel.

Table 10 shows the numbers of each version of the problems solved (considering only the problems not solved by contradictory axioms). There were 5967 problems solved across all systems and problem versions. There is quite a high degree of specialization to specific problem versions, with Zipperposition having the broadest range. Four systems solved only FOF versions, MaLARea solved only FOF v1 versions, Leo-III 1.5 solved mostly polymorphic versions, and E solved only FOF and TF0 v1 versions. Leo-III 1.4 solved only TH0 v2 versions, which was a contrast to CASC-27 when it won the division by solving TH0 v1, TH0 v2, and TH1 v1 versions. This year the TH1 v1 and TH0 v1 versions contained duplicate type declarations that Leo-III 1.4 could not deal with. There are some interesting complementary performances, e.g., Zipperposition with Leo-III 1.5 on TF1 v1 and TH0 v1 versions, and Zipperposition with Leo-III 1.4 on TH0 v2 versions.

Zipperposition solved a broad range of problem versions for a variety of reasons, including use of various higher-order features in different configurations, use of E as a backend, and being suited to particular problem versions' encodings. Zipperposition had a burst of 668 solutions in the 65s to 75s wall clock range[4]. Of the 834 TH0 v2 problems solved by Zipperposition, 656 have a final inference by "eprover", i.e., they were finally solved by the E backend. Of those 656, 646 have wall clock times in the 65s to 75s range. This burst of solutions is due to Zipperposition translating clauses with shallow proof depth to lambda-free higher-order logic, and running E in its auto-schedule mode.

The individual problem results (for any version of each problem, considering only the problems not solved by contradictory axioms) show that 4030 problems were unsolved, 107 problems were solved by all the systems, and 2100 problems were solved by only one system. No problems were solved by only the two versions of Leo-III (which would have counted as unique solutions for Leo-III 1.5). Of the 2100 unique solutions, 1778 were by MaLARea, 162 by Leo-III 1.5, 89 by E, 42 by iProver, 25 by Zipperposition, and 4 by GKC. As the systems have different approaches to solving the problems, a portfolio approach cannot be (easily) considered, but is noteworthy that even the lowly ranked GKC makes a small unique contribution.

The discovery that some of the LTB problems had contradictory axioms opened a can of worms. The contradiction was traced back through the new export (explained in Section 2.2.2) to axioms in HOL4. A HOL4 developer explained that those "were there to make extraction to SML/OCaml somewhat reasonable, and were not for a logical sense". The bug in HOL4 has now been fixed. At the same time, the process for exporting from HOL4 came under close scrutiny, and some further errors were revealed. An updated export of HOL4 with all the fixes will be available in the future. HOL4 is a very well established and highly used corpus, and it is a noteworthy success story for ATP to find the inconsistency after many years of no one noticing. Finding a contradiction in the axioms of large corpora is useful [21], but for users a proof by virtue of contradictory axioms is typically not what is desired. At best, one such proof is all that is needed, after which efforts are made to remove the contradiction from the axioms. For the LTB division in CASC-28 proofs by contradictory axioms will be recognized and applauded, but only one for each contradictory set will count for ranking.

[4]See the performance plot at http://www.tptp.org/CASC/J10/WWWFiles/ResultsPlots.html#HL4Problems

Table 10
Problems solved for each version

Ver	E	iPro'	Zip'n	Leo'5	ATPB'	GKC	Leo'4	MaL'	Union
FOF v1	3187	2945	0	0	455	382	0	5522	5695
FOF v2	143	218	0	0	782	111	0	0	1133
TF0 v1	62	0	0	0	0	0	0	0	62
TF0 v2	0	0	0	0	0	0	0	0	0
TF1 v1	0	0	32	52	0	0	0	0	83
TH0 v1	0	0	38	2	0	0	0	0	40
TH0 v2	0	0	834	0	0	0	134	0	963
TH1 v1	0	0	795	1359	0	0	0	0	1592
Sum	3392	3163	1669	1413	1237	493	134	5522	5970

4. System Descriptions

As was noted in Sections 3.1, 3.3, and 3.5, there were three systems of particular interest in CASC-J10: Zipperposition because of it's dominant performance in the THF division, Enigma because of it's strong performance in the FOF division, and lazyCoP as the only true newcomer to the competition that could also usefully contribute to a UEQ portfolio. The following brief system descriptions were written by their developers.

Zipperposition is a superposition-based theorem prover for typed first-order logic with equality and higher-order logic. The core architecture of the prover is based on saturation with an extensible set of rules for inferences and simplifications. Support for higher-order logic is based on the complete calculus for boolean-free higher-order logic [2]. To support efficient enumeration of complete sets of unifiers Zipperposition implements an efficient full unification procedure [33]. Pragmatic support for boolean reasoning [34] is another factor contributing to Zipperposition's high success rate. Part of this extension is lazy clausification, which enables Zipperposition to work directly on the formula level instead of the clause level. This makes it possible to simulate some of the inferences that Satallax (the CASC-27 winner, based on higher-order tableaux), performs. As a complementary higher-order approach, in some configurations Zipperposition uses the combinator-based calculus of Bhayat and Reger [3]. Various subsets of TPTP problems were used to create heuristic configurations that perform well on all types of TPTP problems.

Enigma[5] [11] is an **E**fficient lear**n**ing-based **i**nference **g**uiding **ma**chine, implemented as an extension of the E prover [20]. The core idea behind Enigma is to use machine learning methods on a corpus of proofs, to learn a model that classifies clauses as useful or not in proof search. The machine learning methods employed are Gradient Boosting Decision Trees (GBDT - implemented by the XGBoost and LightGBM frameworks) and Graph Neural Networks (GNN - implemented by the Tensorflow framework). GNN models are very complementary (different models solve different problems), and GBDT models can be used to memorize and generalise many GNN strategies into a single strategy. Only GBDT models were used in the competition because of their superior speed on standard hardware. Before the competition E was run on the TPTP with a portfolio of BliStrTune [12] strategies, to generate initial training data. Several iterations of training and testing were used to generate additional training data and to learn models with improved performance. Additionally, a LightGBM premise selection filter was trained for axiom selection. Based on the number of input formulae, a simple SInE filter was optionally applied for the learned strategies.

lazyCoP [18] is a connection-tableaux system for first-order logic with equality. lazyCoP implements the lazy paramodulation calculus [17], with some additional inferences such as "shortcut" strict rules and equality lemmas. The system implements well-known refinements of the predicate connection calculus, such as tautology elimination and strong regularity, lifted to equalities where appropriate. The resulting system appears to be complete, but no theoretical claim is made one way or another. This first version of the system explores a tableaux-level search space using a classic A* informed-search algorithm. The (admissible) heuristic function is the number of open branches. The system is implemented in Rust, allowing control over memory allocation and layout while avoiding some classes of memory/thread safety bugs. These safety guaran-

[5]Research supported by ERC Consolidator grant no. 649043 AI4REASON, the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466, and the European Regional Development Fund.

tees allow an easy implementation of multi-threading: lazyCoP uses all available cores to search, and scales nearly linearly. The system does not yet include a custom clausification routine - a recent build of Vampire is used for this purpose.

5. Conclusion

CASC-J10 was the twenty-fifth large scale competition for fully automatic, classical logic ATP systems. CASC-J10 fulfilled its objectives by evaluating the relative capabilities of current ATP systems, and stimulating development and interest in ATP.

The highlights of CASC-J10 were: a disappointing turnout in the TFA division (a lowlight), consistent proof and model output across all divisions, revelations of unsoundness and contradictory axioms, the emergence of machine learning as a potential game changer in ATP, and another fascinating LTB division.

The most impactful change in CASC-J10 was the use of wall clock time limits, which enabled the systems to take advantage of multiple cores. This change has been noted at various points in this paper. The most common approach was to create a schedule of strategies with time limits, then use the multiple cores to work through the strategies in competition parallel. This approach was used by Enigma, GKC, iProver, Twee, Vampire, and Zipperposition. A specific benefit of running mutiple strategies in competition parallel was that one of the initial strategies, but not the one that would be run first in a sequential approach, could find a solution without having to wait for preceding stratgies to complete. This was noticied particularly by Vampire 4.5 in the TFA and FNT divisions. Two systems took other approaches: lazyCoP explored its tableaux space in parallel, while Leo-III ran multiple external subsystems in parallel. Most systems "rolled their own" use of the multiple cores, by explicitly starting multiple processes. Two other approaches noted were lazyCoP's use of multi-threading in Rust, and Zipperposition's use of the Python scheduler. CASC-28 will be the third CASC edition with wall clock time limits. As the system developers refine their approaches and gain experience with the use of multiple cores, the impact of wall clock time limits is expected to increase, and core usage values will improve further.

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-28 are:

- The TFA division will be placed on hiatus.
- The Sledgehammer (SLH) division, previously run in CASC-26 [27], will be revived. The division's prize will be a "romantic hotel stay in Amsterdam", generously donated by Jasmin Blanchette's Matryoshka project[6].
- Only one proof by contradictory axioms, for each contradictory set, will count towards the ranking in the LTB division.

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.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1730419.

References

[1] J. Alama, T. Heskes, D. Külwein, E. Tsivtsivadze, and J. Urban. Premise Selection for Mathematics by Corpus Analysis and Kernel Methods. *Journal of Automated Reasoning*, 52(2):191–213, 2014.

[2] A. Bentkamp, J. Blanchette, P. Vukmirovic, and U. Waldmann. Superposition with Lambdas. In P. Fontaine, editor, *Proceedings of the 27th International Conference on Automated Deduction*, number 11716 in Lecture Notes in Computer Science, pages 55–73. Springer-Verlag, 2019.

[3] A. Bhayat and G. Reger. A Combinator-Based Superposition Calculus for Higher-Order Logic. In N. Peltier and V. Sofronie-Stokkermans, editors, *Proceedings of the 10th International Joint Conference on Automated Reasoning*, number 12166 in Lecture Notes in Artificial Intelligence, pages 278–296, 2020.

[4] A. Bhayat and G. Reger. A Knuth-Bendix-Like Ordering for Orienting Combinator Equations. In N. Peltier and V. Sofronie-Stokkermans, editors, *Proceedings of the 10th International Joint Conference on Automated Reasoning*, number 12166 in Lecture Notes in Artificial Intelligence, pages 259–277, 2020.

[5] J. Blanchette, C. Kaliszyk, L. Paulson, and J. Urban. Hammering Towards QED. *Journal of Formalized Reasoning*, 9(1):101–148, 2016.

[6] C. Brown, T. Gauthier, C. Kaliszyk, G. Sutcliffe, and J. Urban. GRUNGE: A Grand Unified ATP Challenge. In P. Fontaine, editor, *Proceedings of the 27th International Conference on Automated Deduction*, number 11716 in Lecture Notes in Computer Science, pages 123–141. Springer-Verlag, 2019.

[6]http://matryoshka.gforge.inria.fr

[7] K. Chvalovsky, J. Jakubuv, M. Suda, and J. Urban. ENIGMA-NG: Efficient Neural and Gradient-Boosted Inference Guidance for E. In P. Fontaine, editor, *Proceedings of the 27th International Conference on Automated Deduction*, number 11716 in Lecture Notes in Computer Science, pages 197–215. Springer-Verlag, 2019.

[8] K. Claessen and N. Sörensson. 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.

[9] B. Gleiss and M. Suda. Layered Clause Selection for Saturation-based Theorem Proving. In P. Fontaine, P. Rümmer, and S. Tourret, editors, *Proceedings of the 7th Workshop on Practical Aspects of Automated Reasoning*, page To appear, 2020.

[10] L. Gleiss, B. Kovacs and J. Rath. Subsumption Demodulation in First-Order Theorem Proving. In N. Peltier and V. Sofronie-Stokkermans, editors, *Proceedings of the 10th International Joint Conference on Automated Reasoning*, number 12166 in Lecture Notes in Computer Science, pages 297–315, 2020.

[11] J. Jakubuv, M. Chvalovský, K. Olsák, B. Piotrowski, M. Suda, and J. Urban. ENIGMA Anonymous: Symbol-Independent Inference Guiding Machine (System Description). In N. Peltier and V. Sofronie-Stokkermans, editors, *Proceedings of the 10th International Joint Conference on Automated Reasoning*, number 12167 in Lecture Notes in Artificial Intelligence, pages 448–463, 2020.

[12] J. Jakubuv and J. Urban. Hierarchical Invention of Theorem Proving Strategies. *AI Communications*, 31(3):237–250, 2018.

[13] J. Jakubuv and J. Urban. Hammering Mizar by Learning Clause Guidance. In *Proceedings of the 10th International Conference on Interactive Theorem Proving*, Leibniz International Proceedings in Informatics, page To appear. Dagstuhl Publishing, 2019.

[14] C. Kaliszyk, J. Urban, and J. Vyskocil. Machine Learner for Automated Reasoning 0.4 and 0.5. In S. Schulz, L. de Moura, and B. Konev, editors, *Proceedings of the 4th Workshop on Practical Aspects of Automated Reasoning*, number 31 in EPiC Series in Computing, pages 60–66. EasyChair Publications, 2015.

[15] T. Lampert and A. Nakano. Deciding Simple Infinity Axiom Sets with One Binary Relation by Superpostulates. In N. Peltier and V. Sofronie-Stokkermans, editors, *Proceedings of the 10th International Joint Conference on Automated Reasoning*, number 12166 in Lecture Notes in Artificial Intelligence, pages 201–217, 2020.

[16] D.L. Li and A. Tiu. Combining ProVerif and Automated Theorem Provers for Security Protocol Verification. In P. Fontaine, editor, *Proceedings of the 27th International Conference on Automated Deduction*, number 11716 in Lecture Notes in Computer Science, pages 354–365. Springer-Verlag, 2019.

[17] A. Paskevich. Connection Tableaux with Lazy Paramodulation. *Journal of Automated Reasoning*, 40(2-3):179–194, 2008.

[18] M. Rawson and G. Reger. lazyCoP 0.1. EasyChair Preprints 3926, 2020.

[19] A. Riazanov and A. Voronkov. Limited Resource Strategy in Resolution Theorem Proving. *Journal of Symbolic Computation*, 36(1-2):101–115, 2003.

[20] S. Schulz. E: A Brainiac Theorem Prover. *AI Communications*, 15(2-3):111–126, 2002.

[21] S. Schulz, G. Sutcliffe, J. Urban, and A. Pease. Detecting Inconsistencies in Large First-Order Knowledge Bases. In L. de Moura, editor, *Proceedings of the 26th International Conference on Automated Deduction*, number 10395 in Lecture Notes in Computer Science, pages 310–325. Springer-Verlag, 2017.

[22] K. Slind and M. Norrish. A Brief Overview of HOL4. In O. Mohamed, C. Munoz, and S. Tahar, editors, *Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics*, number 5170 in Lecture Notes in Computer Science, pages 28–32. Springer-Verlag, 2008.

[23] 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.

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

[25] 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.

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

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

[28] G. Sutcliffe. Proceedings of the 10th IJCAR ATP System Competition. Online, Earth, 2020. http://www.tptp.org/CASC/J10/Proceedings.pdf.

[29] G. Sutcliffe. The CADE-27 Automated Theorem Proving System Competition - CASC-27. *AI Communications*, 32(5-6):373–389, 2020.

[30] G. Sutcliffe and F.J. Pelletier. Hoping for the Truth - A Survey of the TPTP Logics. In Z. Markov and I. Russell, editors, *Proceedings of the 29th International FLAIRS Conference*, pages 110–115, 2016.

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

[32] J. Urban. BliStr: The Blind Strategymaker. In S. Autexier, editor, *Proceedings of the 1st Global Conference on Artificial Intelligence*, number 36 in EPiC Series in Computing, pages 312–319. EasyChair Publications, 2015.

[33] P. Vukmirovic, A. Bentkamp, and V. Nummelin. Efficient Full Higher-order Unification. In Z.M. Ariola, editor, *Proceedings of the 5th International Conference on Formal Structures for Computation and Deduction*, number 167 in Leibniz International Proceedings in Informatics, pages 5:1–5:20. Dagstuhl Publishing, 2020.

[34] P. Vukmirovic and V. Nummelin. Boolean Reasoning in a Higher-Order Superposition Prover. In P. Fontaine, P. Rümmer, and S. Tourret, editors, *Proceedings of the 7th Workshop on Practical Aspects of Automated Reasoning*, page To appear, 2020.