An Empirical Assessment of Progress in Automated Theorem Proving

Geoff Sutcliffe[1(✉)], Christian Suttner[2], Lars Kotthoff[3], C. Raymond Perrault[4], and Zain Khalid[1]

[1] University of Miami, Miami, USA
geoff@cs.miami.edu , zsk17@miami.edu
[2] Miami, USA
[3] University of Wyoming, Laramie, USA
larsko@uwyo.edu
[4] SRI International, Menlo Park, USA
ray.perrault@sri.com

Abstract. The TPTP World is a well established infrastructure that supports research, development, and deployment of Automated Theorem Proving (ATP) systems. This work uses data in the TPTP World to assess progress in ATP from 2015 to 2023.

Keywords: Automated Theorem Proving · Empirical Evaluation · Progress

1 Introduction

The TPTP World [69] (www.tptp.org) is a well established infrastructure that supports research, development, and deployment of Automated Theorem Proving (ATP) systems. The TPTP World includes the TPTP problem library, the TSTP solution library, standards for writing ATP problems and reporting ATP solutions, tools and services for processing ATP problems and solutions, and it supports the CADE ATP System Competition (CASC). This work uses data in the TPTP World to assess progress in ATP from 2015 to 2023.

Any meaningful assessment of progress in ATP must refer to the ability of ATP systems to solve problems. As the systems improve over time, the problems that they must solve also change to meet the demands of applications (with a fixed set of problems the systems can simply be finely tuned to the set, with inevitable asymptotic progress towards solving all the problems [52]). The TPTP problem library provides an evolving set of ATP problems that reflect the needs of ATP users, and is an appropriate basis for assessing the changing ability of ATP systems (the library is almost monotonically growing, but occasionally problems are removed – see Sect. 4.1). Alongside the TPTP problem library, the TSTP solution library provides data about ATP systems' abilities to solve the problems in the TPTP problem library. This paper examines progress in ATP,

C. Suttner—Deceased.

C. Benzmüller et al. (Eds.): IJCAR 2024, LNAI 14739, pp. 53–74, 2024.
https://doi.org/10.1007/978-3-031-63498-7_4

based on the data from TPTP v6.3.0 released on 28th November 2015 to TPTP v8.2.0 released on 13th June 2023. It is important to differentiate between evaluations at instances in time, such as provided by competitions, and evaluations over time. At instances of time the test problems used for evaluation, the systems being evaluated, and the hardware/software platform used, are static, e.g., as done in [20]. That provides a clean basis for a detailed comparison between systems. In contrast, evaluation over time is complicated by changing test problems, changing systems, and changing hardware/software. This dynamic evaluation environment requires additional control to provide meaningful results. The analyses done in this work do not explicitly factor in the resources needed to find solutions, e.g., hardware, time limits, etc.; Sect. 3.1 explains why this makes sense in the ATP context.

Related Work: The use of system performance data to evaluate a field of endeavour is common. In the realm of logic-based systems, examples include the various competitions [6] for logic-based systems (e.g., CASC [68], the SAT Competition [36], SMT-COMP [5], the ASP Competition [18]), longitudinal surveys of competitions [20,75], the Technical Performance chapter of Stanford University's AI Index Annual Report [45], the use of Shapley values to evaluate algorithmic improvements in SAT solving [25,41], comparison of algorithmic and hardware advances (in SAT solving) [24], and other more specialized benchmarking, e.g., [89]. A general examination of the requirements for such benchmarking is provided by [9]. An ontology of artificial intelligence benchmarks is described in [14]. [52] provides an insightful analysis of the global dynamics of using benchmark sets in computer vision and natural language processing, and the takeaway messages are broadly applicable, including to benchmark sets for logic-based systems. In all cases the common measures for evaluation are (i) the ability of systems to solve problems, and (ii) the resources required by the systems to solve the problems. In order for results to be relevant, test problems must be representative of the problems the systems will face in applications, and the resource measurements must be appropriate for the availability and demands of the applications.

Summary of Findings: There has been progress in the last eight years, with stronger progress from v6.3.0 (2015) up to v7.1.0 (2018), but then a period of quiet until some more signs of progress in v8.2.0 (2023). There have been some first solutions of problems that are of direct interest to humans, a quite large number of first ATP solutions of problems from the TPTP, and some noteworthy improvements in individual ATP systems. There has been an apparent slowing of progress compared to the five years prior to 2015.

Paper Structure: Sections 2 and 3 provide a brief background to the TPTP problem library and TSTP solution library, highlighting features relevant to this work. Section 4 describes how the TPTP and TSTP data was prepared for analysis, and describes the measures used. Section 5 is the core of the paper, giving the results with commentary. Section 6 concludes.

Table 1. Overview of TPTP releases

Release	Date	Changes	Size	Analysed
v6.3.0	28/11/15	New TFO problems with arithmetic	20762	20168
v6.4.0	31/06/16	New problems	20897	20839
v7.0.0	24/07/17	First TH1 problems	21851	21310
v7.1.0	06/03/18	TXF syntax specified	22011	21893
v7.2.0	10/07/18	New problems	22026	21909
v7.3.0	02/08/19	New problems	22686	22570
v7.4.0	10/07/20	New problems	23291	23118
v7.5.0	13/07/21	New problems	24098	24027
v8.0.0	19/04/22	First TXF problems	24785	24027
v8.1.0	30/07/22	New problems	25257	25103
v8.2.0	13/06/23	New problems	25474	25325

2 The TPTP Problem Library

The core of the TPTP World is the TPTP problem library [66]; it is the de facto standard set of test problems for classical logic ATP systems. The problems can be browsed online[1] and documentation is available[2] Each release of the problem library is identified by a number in the form *version.edition.patch*. The current release at the time of writing was v8.2.0. Section 3 explains why the analyses of progress presented in this paper start at v6.3.0. Table 1 gives some summary data about the editions from v6.3.0 to v8.2.0. The Size column gives the number of problems in the release at the time of the release, while the Analysed column gives the number of problems left for analysis after the data cleaning described in Sect. 4.1. The acronyms for problem types are given in Sect. 2.1.

Each TPTP problem file has a header section (as comments) that contains information for users in four parts: the first part identifies and describes the problem; the second part provides information about occurrences of the problem in the literature and elsewhere; the third part provides semantic and syntactic characteristics of the problem; the last part contains comments and bugfix information. The third part is most relevant to this work. It contains the problem's SZS status [77] that provides the semantic status of the problem, e.g., if it is a `Theorem`, a `Satisfiable` set of formulae, a problem whose status is `Unknown`, etc. It also includes statistics about the problem's syntax, e.g., the number of formulae, the numbers of symbols, the use of equality and arithmetic, etc. The SZS status and the syntactic characteristics are used to form the Specialist Problem Class of the problem, as explained in Sect. 2.1.

[1] www.tptp.org/cgi-bin/SeeTPTP?Category=Problems.
[2] www.tptp.org/cgi-bin/SeeTPTP?Category=Documents.

2.1 Specialist Problem Classes

The problems in the TPTP library are divided into Specialist Problem Classes (SPCs) – classes of problems that are homogeneous wrt recognizable logical, language, and syntactic characteristics. Evaluation of ATP systems within SPCs makes it possible to say which systems work well for what types of problems. Empirically, homogeneity is ensured by examining the patterns of system performance across the problems in each SPC. For example, the separation of "essentially propositional" problems was motivated by observing that SPASS [85] performed differently on the ALC problems in the SYN domain of the TPTP. A data-driven test of homogeneity is also possible [26].

The characteristics used to define the SPCs in TPTP v8.2.0 are ...

1. TPTP language:
   `CNF` – Clause Normal Form `FOF` – First-Order Form
   `TFO` – Typed Monomorphic First-order form
   `TF1` – Typed Polymorphic First-order form
   `TX0` – Typed Monomorphic eXtended First-order form
   `TX1` – Typed Polymorphic eXtended First-order form
   `TH0` – Typed Monomorphic Higher-order form
   `TH1` – Typed Polymorphic Higher-order form
2. SZS status:
   `THM` – Theorem `CSA` – CounterSAtisfiable
   `CAX` – Contradictory AXioms (merged with `THM` in this work)
   `UNS` – UNSatisfiable `SAT` – SATisfiable
   `UNK` – UNKown `OPN` – OPeN
3. Order (for `CNF` and `FOF`):
   `PRP` – PRoPositional
   `EPR` – Effectively PRopositional (known to be reducible to `PRP`)
   `RFO` – Real First-Order (not known to be reducible to `PRP`)
4. Equality:
   `NEQ` – No EQuality `EQU` – EQUality (some or pure)
   `SEQ` – Some (not pure) EQUality `PEQ` – Pure EQUality
   `UEQ` – Unit EQUality `CNF` `NUE` – Non-Unit Equality `CNF`
5. Hornness (for `CNF`):
   `HRN` – HoRN `NHN` – Non-HorN
6. Arithmetic (for `T*` languages):
   `NAR` – No ARithmetic `ARI` – ARIthmetic.

Using these characteristics 223 SPCs are defined in TPTP v8.2.0. For example, the SPC `TFO_THM_NEQ_ARI` contains typed monomorphic first-order theorems that have no equality but include arithmetic. Combinations of SPCs are written using UNIX globbing, e.g., `TFO_THM_*_NAR` is the combination of `TFO_THM_EQU_NAR` and `TFO_THM_NEQ_NAR` – typed monomorphic higher-order theorems problems, either with or without equality, but no arithmetic.

The SPCs are used when computing the TPTP problems difficulty ratings, as explained in Sect. 2.2.

2.2 TPTP Problem Ratings

Each TPTP problem has a difficulty rating that provides a well-defined measure of how difficult the problem is for current ATP systems [76]. The ratings are based on performance data in the TSTP (see Sect. 3), and are updated in each TPTP edition. Rating is done separately for each SPC. First, a partial order between systems is determined according to whether or not a system solves a strict superset of the problems solved by another system. If a strict superset is solved, the first system is said to *subsume* the second. Then the fraction of non-subsumed systems that fail on a problem is the difficulty rating for the problem. Problems that are solved by all of the non-subsumed systems get a rating of 0.00 ("easy"); problems that are solved by some of the non-subsumed systems get a rating between 0.00 and 1.00 ("difficult"); problems that are solved by none of the non-subsumed systems get a rating of 1.00 ("unsolved").

3 The TSTP Solution Library

The complement of the problem library is the TSTP solution library [65,67]. The TSTP is built by running all the ATP systems that are available in the TPTP World on all the problems in the TPTP problem library. At the time of writing this paper, the TSTP contained the results of running 87 ATP systems and system variants on all the problems in the TPTP that they could attempt. This produced 1091026 runs, of which 432718 (39.6%) solved the problem. One use of the TSTP is for ATP system developers to examine solutions to problems and thus understand how they can be solved, leading to improvements to their own systems. The use considered here is for TPTP problem ratings.

Prior to 2010 the data in the TSTP came from results submitted by ATP system developers, who performed testing on their own hardware. From 2010 to 2013 the data was generated on the TPTP World servers at the University of Miami. Since 2014 the ATP systems have been run on StarExec [63], initially on the StarExec Iowa cluster, and since 2018 on the StarExec Miami cluster. StarExec has provided stable platforms that produce reliably consistent and comparable data in the TSTP. The analyses presented in Sect. 4 start at TPTP v6.3.0, which was released in November 2015. By that time the problem ratings were based on data produced on the StarExec computers.

The StarExec Iowa computers have a quad-core Intel Xeon CPU E5-2609 CPU running at 2.40 GHz, 128 GiB memory, and the CentOS Linux release 7.9.2009 operating system. The StarExec Miami computers have an octa-core Intel Xeon E5-2667 v4 CPU running at 2.10 GHz, 128 GiB memory, and the CentOS Linux release 7.4.1708 operating system. One ATP system is run on one CPU at a time, with a 300 s CPU time limit and a 128GiB memory limit (see Sect. 3.1). The minor differences between the Iowa and Miami configurations can be ignored for the task of "solving problems", as is explained in Sect. 3.1.

3.1 Resource Limits

Analysis shows that increasing resource limits does not significantly affect which problems are solved by an ATP system. Fig. 1 illustrates this point; it plots the CPU times taken by several contemporary ATP systems to solve the TPTP problems for the `FOF_THM_RFO_*` SPCs, in increasing order of time taken. The relevant feature of these plots is that each system has a point at which the time taken to find solutions starts to increase dramatically. This point is called the system's Peter Principle [55] Point (PPP), as it is the point at which the system has reached its level of incompetence. Evidently a linear increase in the computational resources beyond the PPP would not lead to the solution of significantly more problems. The PPP thus defines a realistic computational resource limit for the system. Therefore, provided that enough CPU time and memory are allowed for an ATP system to reach its PPP, a usefully accurate measure of what problems it can solve is achieved. The performance data in the TSTP is produced with adequate resource limits.

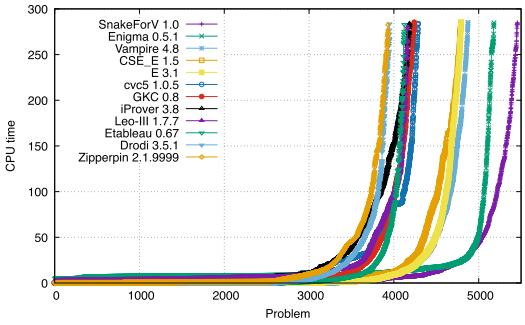

Fig. 1. CPU times for `FOF_THM_RFO_*`

4 Analysis Processes

4.1 Analysis Data

The analyses performed in this assessment use the TPTP problem ratings, and historical data about which ATP systems solved which problems in each TPTP release. The data was extracted from the `ProblemAndSolutionStatistics` file that accompanies each TPTP release, which summarizes information from the header fields of the TPTP problem files and corresponding TSTP solution files. As explained in Sect. 3, TSTP data starting from TPTP v6.3.0 in November 2015 has been used, taking snapshots at each TPTP edition up to v8.2.0.

Before analysis the rating data was cleaned as follows:

Cleaning for Bias: The TPTP tags problems that are designed specifically to be suited or ill-suited to some ATP system, calculus, or control strategy as *biased*. The biased problems were excluded from the analyses.

Cleaning for Bugfixes: Over time some problems have had to be removed from the TPTP because they are renamed, duplicates, wrongly formulated, etc. Such problems in a TPTP release are thus not in subsequent releases. The removed problems were excluded from the analyses.

Cleaning for the Past: Problems are added to the TPTP in each release, and corresponding TSTP data is generated using the available ATP systems. As it is not possible to run all previously available ATP systems on new problems when they are added to the TPTP, it has been (quite reasonably) assumed that if a problem was unsolved by the current ATP systems when it was added to the TPTP (initial rating 1.00), then it would have been unsolved by previously available ATP systems. The rating data was thus augmented for problems that were added after v6.3.0 and had an initial rating of 1.00, by setting the problems' ratings in the prior TPTP releases to 1.00. There were 1854 such problems. This, however, can lead to an unfairly optimistic view of progress, because those retrospectively added 1.00 ratings increase the average problem rating in the past. For problems that were solved when they were added to the TPTP (initial rating less than 1.00), it is unknown if the previously available ATP systems would have been able to solve them. Augmenting the rating data by setting the problems' ratings in prior TPTP releases to their initial rating of less than 1.00 could lead to an optimistic or pessimistic view of progress, depending if the rating was greater or less than the average in the past releases. In this work the rating data was augmented for problems that were added after v6.3.0 and had an initial rating less than 1.00, by setting the problems' ratings in the prior TPTP releases to their initial rating. There were 2632 such problems. The optimistic/pessimistic effect gets stronger when rating data is augmented for problems that were added in more recent TPTP releases. A total of $1854 + 2632 = 4486$ problems had their initial ratings propagated backwards, starting from the various releases over the eight years of analysis. Overall this could have had a slightly optimistic impact in the analyses.

Cleaning for Change: A counterintuitive feature of an individual problem's difficulty ratings is that they sometimes increase with time. It is counterintuitive because the problem has not changed. (This was also noted in a prior analysis [69].) Increases are caused by new ATP systems or system versions becoming available. If a new system is not subsumed then its TSTP data is used in the rating process: the ratings of problems that it solves decrease, but at the same time the ratings of problems that it does not solve increase – you have to "pay the

piper".[3] A common instance of this phenomenon is a new system that can solve some previously unsolved (rating 1.00) problems, but that cannot solve a substantial number of problems that are solved by other systems (rating less than 1.00). In this work the anomaly is resolved by additionally looking at *monotonic ratings*: if a problem's rating in a TPTP release is greater than its previous rating, the monotonic rating is set to the previous lower rating. Monotonized ratings make clear sense in the case of problems that were unsolved (rating 1.00) and were later solved by a new system (the rating drops to less than 1.00) – if a problem is solved, it cannot become unsolved – the solving system still exists in principle. In cases where the rating is less than 1.00 monotonized ratings might be considered to be optimistic because ratings do have to "pay the piper".

4.2 Coherent SPC Sets

Five of the analyses performed (see Sect. 4.3) require data from sets of problems with similar characteristics, so that the analysis results are wrt that type of problem. The basis for such sets is the SPCs (see Sect. 2.1), which provide a fine-grained partitioning of the TPTP problems so that each SPC is coherent. Some SPCs that capture compatible problem characteristics can be merged to form a *coherent SPC set*.

The coherent SPC sets used for the analyses are listed in Table 2. The SPC set column lists the SPCs that are in the set, using the abbreviations given in Sect. 2.1. Some noteworthy exclusions are: typed extended first-order problems, because they were added to the TPTP only in v8.0.0; typed polymorphic first-order and higher-order problems, because too few systems are capable of attempting the problems and generating the necessary TSTP data; some SPCs that have too few problems, e.g., `TFO_CSA_*_NAR` and `TFO_SAT_*_NAR`, which combined have only 154 problems.

4.3 Six Analyses

The cleaned TPTP problems ratings and historical TSTP data has been used for six analyses of progress in ATP. Individual problem ratings are used for the first analysis. The other five analyses are wrt the coherent SPC sets described in Sect. 4.2.

First Solutions: Arguably the most successful use of ATP comes from the "hammers" [15] associated with Interactive Theorem Proving (ITP) systems, where the individual problems being solved are typically not of direct interest to the human users who are focussed on the larger task being addressed in the ITP system. In contrast, the use of ATP by practitioners to solve individual problems

[3] Conversely, if a system that was not subsumed becomes unavailable, it no longer contributes TSTP data for new problems. This phenomenon is rare (e.g., Isabelle ran fine on StarExec Iowa but did not port to StarExec Miami in 2018) and has not materially impacted the analyses of progress.

Table 2. Coherent SPC sets

SPC set	Description
CNF_UNS_RFO_PEQ_UEQ	Unsatisfiable really-first-order unit equality clauses
CNF_UNS_RFO_NEQ_* ∪ CNF_UNS_RFO_SEQ_* ∪ CNF_UNS_RFO_PEQ_NUE	Unsatisfiable really first-order clauses that are not unit equality
CNF_SAT_RFO_*	Satisfiable really-first-order clauses
FOF_*_PRP ∪ FOF_*_EPR_* ∪ CNF_*_PRP ∪ CNF_*_EPR_*	Unsatisfiable and satisfiable propositional and effectively propositional clauses. Un/Satisfiable is coherent because the problems are decidable
FOF_THM_RFO_*	Really first-order theorems, with or without equality
FOF_CSA_RFO_* ∪ FOF_SAT_RFO_*	Really first-order non-theorems and satisfiable sets, with and without equality
TFO_THM_*_NAR	Typed monomorphic first-order theorems, with and without equality, no arithmetic
TFO_THM_*_ARI	Typed monomorphic first-order theorems, with and without equality, with arithmetic
THO_THM_*_NAR	Typed monomorphic higher-order theorems, with and without equality, no arithmetic

that have resisted manual approaches is less common and possibly less successful, but the sparsity makes successes particularly noteworthy. First solutions of problems that are of direct interest to humans are indications of progress. Such problems are identifiable by (i) the rating decreasing from 1.00, and (ii) evidence that the problem is of direct interest to some humans.

Average Difficulty Ratings: This is the average problem difficulty rating, and the average monotonized difficulty rating. (This approach was used in [73].) As the problems are unchanged (they are not actually getting easier), decreases are evidence of progress in ATP systems.

Never-Solved: This is the fraction of problems that were unsolved (rating 1.00) in all TPTP releases up to each TPTP release, relative to the number in v6.3.0. (The converse of this is plotted in [78].) Decreases are evidence of progress.

Solved: For the given system and a given TPTP release, this is ...

$$\frac{ProblemSolvedInRelease - LeastSolvedAcrossAllReleases}{MostSolvedAcrossAllReleases - LeastSolvedAcrossAllReleases}$$

The releases with a 1.00 value are those in which the most problems were solved, and those with 0.00 had the least number solved. Increases are evidence of progress.

Always-Easy: This is the converse of Never-solved – the fraction of problems that were easy (rating 0.00) in all TPTP releases back to each TPTP release, relative to the number in v8.2.0. Increases are evidence of progress.

Shapley Value: A State-of-the-Art (SotA) ATP system for a TPTP release is defined as one that solves the union of the problems solved by the individual ATP systems, e.g., by using competition parallelism [79]. The Shapley value [87] is the average of the marginal contributions (how much the SotA system improves when adding each given system) over all systems added to all possible subsets of other systems. First, temporal Shapley analysis [41] is used to measure the SotA systems' contributions to progress, normalized by the number of previously unsolved problems so that 0.0 means no previously unsolved problems were solved and 1.0 means all previously unsolved problems were solved. Peaks indicate stronger progress. Next, (non-temporal) Shapley analysis [25] is used to measure the contributions of the individual systems in each release. Finally, temporal Shapley analysis for all systems in all releases is used to measure the contributions of the individual system versions when they were introduced. The latter two analyses were used to provide insights for the commentary about the systems' performances (they are not plotted in Sect. 5).

5 Evidence of Progress

5.1 First Solutions

There are some nice examples of ATP systems finding first solutions to problems that are of direct interest to humans ...

- Model finding ATP systems were used to solve previously open problems concerning the existence of quasigroups satisfying certain additional conditions [61]. Many examples are in the `GRP` domain of the TPTP.
- The solution of the Robbins problem[4] by the specialist ATP system EQP [47] in 1996 was a noteworthy success, as the problem had defied the efforts of eminent mathematicians [29]. It is `ROB001-1` in the TPTP, and still has a rating of 1.00 because it has not been solved by a non-specialist ATP system.

[4] The Robbins problem was posed in personal communications between Edward Huntington, Herbert Robbins, and Alfred Tarski. The background is given in en.wikipedia.org/wiki/Robbins_algebra.

- The first inner five-segment theorem of Tarski's geometry [60] was first automatically proved by E [58] in 2019, after being posed by Quaife in 1989 [56]. It is problem `GEO033-2` in the TPTP.
- The proof of the consistency of an encoding of a large fragment of a high school textbook on biology [19] by iProver [38] in 2021 showed how new techniques could be used to find models in large theories. It is problem `BIO001+1` in the TPTP.
- Larry Wos' challenge to find a "circle of pure proofs" that shows the equivalence of the four Moufang identities [86] was met by careful application [81] of Otter [48] in 2021. While those specific problems are not in the TPTP, many related problems are in the ring theory (`RNG`) domain of the TPTP.

5.2 Solutions and Ratings

A total of 25325 problems were analysed over the coherent SPCs, of which 19762 (78%) were solved in TPTP v6.3.0, increasing to 20227 (80%) in v8.2.0. Of the 25325 problems, 5563 (22%) were unsolved when they were added to the TPTP, of which 1009 (4%) were solved in some release by v8.2.0. Conversely, there were 8984 problems (35%) that had a rating of 0.00 in v8.2.0, of which 2965 (12%) had a higher rating in some preceding release. These overall figures provide evidence of overall progress, but the contributions vary across the coherent SPC sets. Figures 2, 3, 4, 5, 6, 7, 8, 9 and 10 plot the values for each coherent SPC set for the latter five analyses described in Sect. 4.3.[5] The captions provide the numbers of 'P'roblems in TPTP v8.2.0, the number left for analysis after the data cleaning, and the numbers of 'N'ever-solved, 'S'olved, and 'A'lways-easy problems in releases v6.3.0-v8.2.0.

Figures 2, 3 4, 5, 6 and 7 plot the values for the CNF- and FOF-based coherent SPC sets. CNF is now the "assembly language" of most ATP systems, which typically translate more expressive logics down to CNF. As such, progress in CNF typically contributes to progress in other SPCs.

`CNF_UNS_RFO_PEQ_UEQ` showed progress in v6.4.0 due to the strong performance of Twee 2.0 [62], which made a lot more problems always-easy by v7.0.0. Also in v6.4.0, Waldmeister 710 [44] solved five problems that had never been solved before. In v7.4.0 E 2.5 made a strong contribution, then in v8.1.0 Twee 2.4 made another strong contribution, alongside CSE_E 1.3 [88]. Waldmeister 710 had the highest Shapley value across all the releases, but in v8.1.0 both Twee and CSE_E solved more problems than Waldmeister. The lowest number of problems solved was in v7.5.0 and v8.0.0, when 23 fewer problems were solved than in v7.4.0 – not many in the context of the 1034 solved in v7.4.0. The only discernible common feature of those 23 problems is that they had ratings over 0.90 in v7.4.0. Apparently some changes in the ATP system versions from v7.4.0 to v7.5.0 made the problems unsolvable in v7.5.0, and further changes reversed the situation for v8.1.0 when 1043 problems were solved.

[5] Data: github.com/GeoffsPapers/ATPProgress2024/raw/master/DataForAnalysis.

CNF_UNS_RFO_*_NUE had a small but quite consistent decline in the problem ratings, indicating some progress. The big advances were in v7.0.0 when Vampire 4.2 performed well, including solving 33 problems that had never been solved before. In v8.2.0 SnakeForV 1.0 solved 26 problems that had never been solved before. The biggest drop in problems solved was between v7.2.0 and v7.3.0, when 66 fewer problems were solved. The largest increase in problems solved was between v8.1.0 and v8.2.0, when 50 more problems were solved. SnakeForV was again the big contributor to the increase. SnakeForV is interesting, as it is a variant of Vampire with an independent reimplementation of Spider-style [82] strategy discovery and schedule construction that factors in prover randomization [64].

CNF_SAT_RFO_* had only one high point, in v6.4.0 when Vampire 4.0.5 made a strong contribution, including solving four problems that had never been solved before. The sudden drop in problems solved in v7.0.0 was due to Prover9 1105 [46] data not being available; the reason is lost in the mists of time, but it is interesting to note that the older system was able to solve some problems that other systems could not. By v7.1.0 new systems had taken up the slack. The plots are all quite stable from v7.1.0 onwards.

{FOF,CNF}_*_EPR_* had two points of progress, the first in v7.0.0 and the second in v7.3.0. In v7.0.0 the progress came from iProver 2.6 that had integrated an abstraction-refinement framework [30], and Vampire 4.2 that had some changes in its model building. Between them they solved five problems that had never been solved before. In v7.3.0 iProver 3.0 integrated superposition [23]. The number of problems solved increased continuously until v8.2.0. The drop in v8.2.0 was due to poorer performances by the new iProver 3.7, SnakeForV 1.0, and Vampire 4.7. These systems share the same FOF to CNF translator, which might have been the source of the common change.

FOF_THM_RFO_* is the best known of the FOF-based SPCs, with the most ATP systems able to attempt the problems, and is the target of most new systems. The problem difficulty ratings are quite flat, but the number of problems solved increased quite regularly, from 6086 in v6.3.0 to 6235 in v8.2.0. The largest step of progress came in v7.0.0 when Vampire 4.2 solved 72 problems that had never been solved before, thanks to improvements in preprocessing. ET 0.2 [37] also contributed to the progress in v7.0.0. In v7.4.0 Enigma 0.4 [32,33] was a new system that made a strong contribution to progress. Vampire 4.5 also contributed to progress in v7.4.0, with a new layered clause selection approach [27] and a new subsumption demodulation rule [28].

FOF_{CSA,SAT}_RFO_* is also well known, and along with its typed first-order counterpart (not analysed due to insufficient data) is important for applications, e.g., [22]. The largest sign of progress was in v6.4.0. The main contributors were Vampire 4.0.5 with improvements to its satisfiability checking, and iProver 2.5 with restructured core data structures and improved preprocessing including predicated elimination. Vampire 4.0.5 solved 10 problems that had never been solved before. There is a drop of 10 problems solved from v8.0.0 to v8.2.0. As in CNF_UNS_RFO_PEQ_UEQ, there is no discernible common feature of those 10

problems, and their ratings were at most 0.75. This again shows that the set of problems solved by evolving versions of systems does not grow monotonically.

Figures 8, 9 and 10 plot the values for the TFF- and THF-based coherent SPC sets. `TFO_THM_*_NAR` uses the simplest of the typed TPTP languages. In v7.0.0 there was progress thanks to Vampire 4.2 and CVC4 1.5.2 [4]. In v8.2.0 there was progress thanks to SnakeForV 1.0. In between those points of progress there was a drop in the number of problems solved, from 282 in v7.4.0 down to 260 in v7.5.0, apparently due to poorer performance of CVC4 1.9 in v7.5.0 compared to that of CVC4 1.7 in v7.4.0.

`TFO_THM_*_ARI` is important because it uses the simplest TPTP language that includes arithmetic, which occurs naturally in application areas [16,39,53]. There was clearly some significant progress in v6.4.0 as many problems were solved for the first time by Vampire 4.0.5, which had integrated Z3 [50] since Vampire 4.0. This contributed to the increase in the number of problems solved, from 915 in v6.3.0 to 1009 in v6.4.0. CVC4 1.5 [4] and Princess 150706 [57] also performed well.

`THO_THM_*_NAR` uses typed higher-order logic, and despite using a more expressive language than the `TFO_*` SPCs, has been the focus of ATP system development longer [70,74]. The problem ratings declined moderately, and there were bursts of progress in v7.0.0 and v7.5.0. The progress in v7.0.0 was largely thanks to Satallax 3.2 [17], which included a SInE-like [31] procedure for premise selection that enabled it to solve some large problems that were previously out of reach. That progress increased the number of always-easy problems by v7.1.0. In v7.5.0 Zipperposition 2.0 [8] improved over the previous version, and solved 18 problems that had never been solved before.

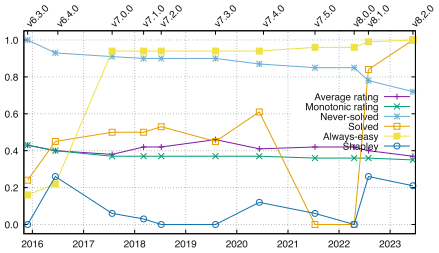

Fig. 2. `CNF_UNS_RFO_PEQ_UEQ` P:1140-1140 N:120-86 S:1020-1049 A:38-233

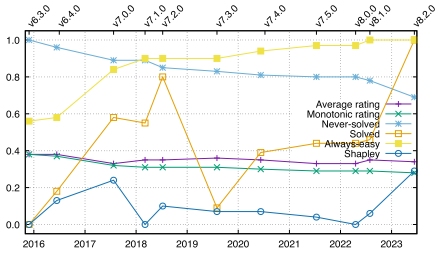

Fig. 3. `CNF_UNS_RFO_*_NUE` P:4445-4441 N:569-391 S:3873-3966 A:1004-1780

Figure 11 was presented (verbatim) in a prior analysis done at TPTP release v6.4.0 [69]. The figure plotted the average ratings for the 14527 problems that were unchanged in the TPTP since v5.0.0, and whose ratings had not been stuck at 0.00 or 1.00 since v5.0.0. It was noted in [69]: "The ratings generally show a downward trend - there has been progress!". Figure 12 shows the same done at TPTP release v8.2.0, for the 16236 problems that were unchanged in the TPTP since v6.3.0, and whose ratings have not been stuck at 0.00 or 1.00 since v6.3.0.

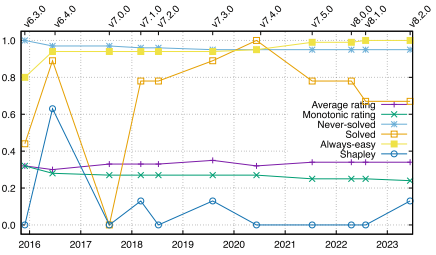

Fig. 4. `CNF_SAT_RFO_*` P:1044-1042 N:155-147 S:887-889 A:476-598

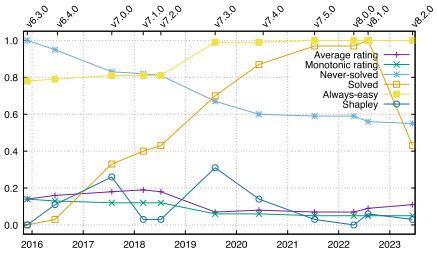

Fig. 5. `{FOF,CNF}_*_EPR_*` P:1457-1425 N:78-43 S:1347-1360 A:1027-1311

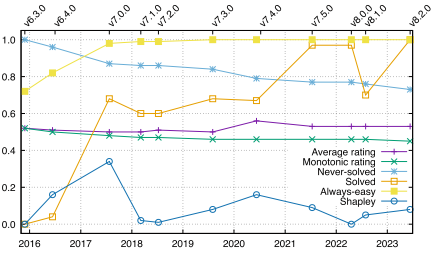

Fig. 6. `FOF_THM_RFO_*` P:7204-7202 N:1116-818 S:6086-6235 A:696-971

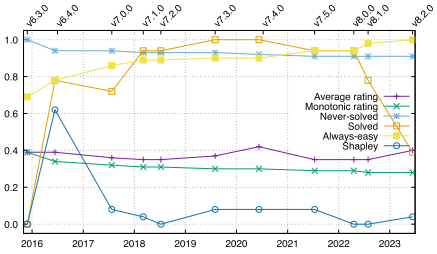

Fig. 7. `FOF_{CSA,SAT}_RFO_*` P:1329-1028 N:282-256 S:746-753 A:481-709

The two figures' plots dovetail quite well, which gives confidence that they really are comparable (there are some minor differences caused by the data cleaning done for this work, and recent refinements to the rating calculations [71,72]). The older plots show a quite clear downward trend both overall and for the four types of problems, while the new plots do not. Possible reasons are discussed in the conclusion (Sect. 6).

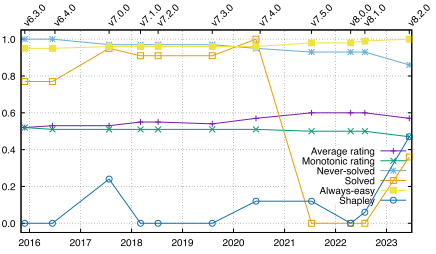

Fig. 8. `TFO_THM_*_NAR` P:400-397 N:120-103 S:277-268 A:117-123

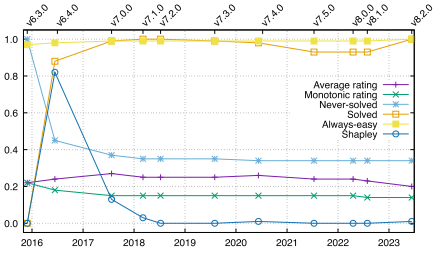

Fig. 9. `TFO_THM_*_ARI` P:1176-1087 N:172-58 S:915-1022 A:763-785

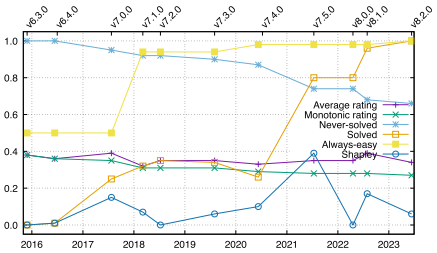

Fig. 10. `THO_THM_*_NAR` PA:3189-3183 N:461-305 S:2722-2814 A:617-1244

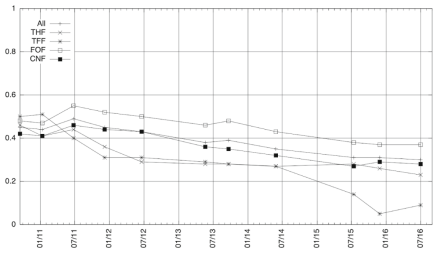

Fig. 11. Ratings from v5.0.0 to v6.4.0

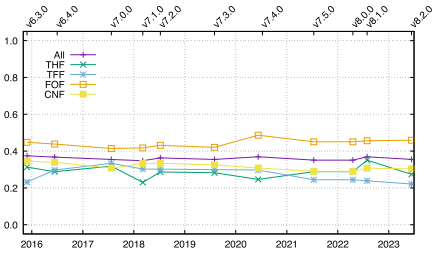

Fig. 12. Ratings from v6.3.0 to v8.2.0

6 Conclusion

This paper has presented an empirical assessment of progress in ATP, using data from the TPTP World in TPTP v6.3.0 in 2015 to v8.2.0 in 2023. The assessment has been in terms of six measures, divided into nine coherent SPC sets of problems that are reasonably homogeneous for ATP systems. The assessment shows that there has been progress in the last eight years, with stronger progress from v6.3.0 (2015) up to v7.1.0 (2018), but then a period of quiet until some more signs of progress in v8.2.0 (2023). There have been some first solutions of problems that are of direct interest to humans, and a quite large number of first ATP solutions of problems from the TPTP. The coherent SPCs with the strongest signs of progress were `CNF_UNS_RFO_PEQ_UEQ` and `THO_THM_*_NAR`.

In addition to overall trends, it is worth noting some of the salient improvements in individual ATP systems, extracted from Sect. 5 ...

- The development of EQP leading to the solution of the Robbins problem in 1996.
- The release of Waldmeister in 1997 (before the period of analysis), which dominated UEQ problem solving until the arrival of Twee 2.4 in 2021.
- The release of Satallax 2.8 in 2015 with strong performance on THF problems, improving up to Satallax 3.2 in 2017.
- The release of Vampire 4.0.5 in 2016, with arithmetic capability included.
- The release of Vampire 4.2 in 2017 with significantly improved performance on many types of problems, including NUE, EPR, FOF, and TF0_NAR.

- The release of iProver versions 2.5 to 2.8 between 2016 and 2018, with strong performance on EPR problems.
- the release of Zipperposition 2.0 in 2020, with strong performance on THF problems.
- The release of the Vampire-based SnakeForV 1.0 in 2022, which outperformed Vampire on many types of problems.

In terms of problem difficulty ratings, the monotonized ratings necessarily went down but the trend was not dramatic, and the raw ratings were generally stable. This is in contrast to the clearly decreasing ratings from 2011 to 2016. The reasons for that apparent slowing of progress are not definitely known, but we have thought of the following possible reasons:

- System developers have expended effort adding breadth of capability at the expense of depth, e.g., E – processed only CNF and FOF up to 2015, added TF0 in 2017 [59], TX0 and TH0 in 2019 [83]; iProver – processed only CNF and FOF up to 2015, added TF0 with arithmetic in 2021 (unpublished); Vampire – processed only CNF, FOF, and TF0 up to 2015, added TX0 in 2016 [40], TF1 and TX1 in 2020 [13], and THF in several incarnations from 2019 to 2023 [10–12].
- The entry barrier to building new high-performance ATP systems is high, because top systems dominate the field and attract the best developer talent. In Maria Paola Bonacina's welcoming address at the Dagstuhl Seminar "The Next Generation of Deduction Systems: from Composition to Compositionality"[6] she referred to this as a "crisis of growth".
- New systems that take new approaches that solve different subsets of SPCs have an impact on problem difficulty ratings. For examples: CSE_E [88] was new in 2018, combining the S-CS calculus with E; Zipperposition [7] was new in 2019, extending superposition to higher-order logic; Twee [62] was new in 2018, solving CNF and FOF problems by transformation to UEQ.
- Time spent on machine learning based techniques, for axiom selection, e.g., [42,80], given clause selection, e.g., [1,21,34,49]), learning for large problem corpora, e.g., [3,35,43], and use of large language models to improve ATP performance [2,84], is focussed largely on sets of many quite similar problems over one fixed signature. The progress made in that usage does not contribute directly to general progress in solving individual problems with different signatures, as measured in this work.
- SMT solvers have been in existence since the late 1970s [51], blossomed fully in the early 2000s, and has attracted ever-increasing interest since then. Some ATP systems have been adapted to solving SMT problems, e.g., Vampire has been entered into SMT-COMP since 2016, and iProver since 2021. This is all good work, but has possibly diverted developer energy from ATP to SMT.
- The divisions of CASC cause developers to put extra effort into solving the types of problems in the division. For examples, the Effectively Propositional (EPR) division was run from CASC-JC in 2001 to CASC-27 in 2019, and

[6] www.dagstuhl.de/23471.

during those years several ATP systems were optimized for EPR problems, most notably iProver. Putting a division on hiatus leads to less development in that aspect of ATP.
– In [72] it was noted that CASC might be causing incremental development of ATP systems. This concern has been expressed as far back as CASC-JC in 2001 [54]. In response to this concern CASC-J12 will have a new ICU (I Challenge yoU) division that focusses on solving hard problems rather than solving more problems, hoping to stimulate new developments and progress.

This assessment of progress is based on ATP systems' abilities to solve problems. Evaluation of other performance measures would be interesting, e.g., stability of proof search modulo perturbations of the input, and some have been done in other evaluations of logic-based systems. These include measures such as resource usage and verifiability of proofs/models. Evaluation of non-performance measures is often ignored, but for users might be just as necessary. These include measures such as the range of logics covered, ease of building and deploying, portability to different hardware and operating system environments, availability of source code, quality of source code and its documentation, licensing that permits a required level of use or modification, availability of user documentation, and (maybe most importantly!) developer support. These are topics for future assessments.

References

1. Aygün, E., et al.: Proving theorems using incremental learning and hindsight experience replay. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning, pp. 1198–1210. No. 162 in Proceedings of Machine Learning Research (2022)
2. Azerbayev, Z., et al.: Llemma: An Open Language Model For Mathematics (2023). arXiv:2310.10631
3. Bansal, K., Loos, S., Szegedy, C., Wilcox, S.: HOList: an environment for machine learning of higher-order theorem proving. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, pp. 454–463 (2019)
4. Barrett, C., et al.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_14
5. Barrett, C., de Moura, L., Stump, A.: SMT-COMP: satisfiability modulo theories competition. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 20–23. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_4
6. Bartocci, E., et al.: TOOLympics 2019: an overview of competitions in formal methods. In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) TACAS 2019. LNCS, vol. 11429, pp. 3–24. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17502-3_1
7. Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P.: Superposition for full higher-order logic. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 396–412. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_23

8. Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P., Waldmann, U.: Superposition with lambdas. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 55–73. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_4
9. Beyer, D., Löwe, S., Wendler, P.: Reliable benchmarking: requirements and solutions. Int. J. Softw. Tools Technol. Transfer **21**, 1–29 (2019)
10. Bhayat, A.: Automated theorem proving in higher-order logic. Ph.D. thesis, Faculty of Science and Engineering, University of Manchester, Manchester, United Kingdom (2020)
11. Bhayat, A., Rawson, M., Schoisswohl, J.: Superposition with delayed unification. In: Pientka, B., Tinelli, C. (eds.) CADE 2023. LNCS, vol. 14132, pp. 23–40. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38499-8_2
12. Bhayat, A., Reger, G.: Restricted combinatory unification. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 74–93. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_5
13. Bhayat, A., Reger, G.: A polymorphic vampire. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 361–368. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_21
14. Blagec, K., Barbosa-Silva, A., Ott, S., Samwald, M.: A curated, ontology-based, large-scale knowledge graph of artificial intelligence tasks and benchmarks. Sci. Data **9**(322), 1–10 (2022)
15. Blanchette, J., Kaliszyk, C., Paulson, L., Urban, J.: Hammering towards QED. J. Formaliz. Reason. **9**(1), 101–148 (2016)
16. Bobot, F., Filliâtre, J.C., Marché, C., Paskevich, A.: Let's verify this with why3. Int. J. Softw. Tools Technol. Transfer **17**(6), 709–727 (2015)
17. Brown, C.E.: Satallax: an automatic higher-order prover. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 111–117. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_11
18. Calimeri, F., Ianni, G., Krennwallner, T., Ricca, F.: The answer set programming competition. AI Mag. **33**(4), 114 (2012)
19. Chaudri, V., Dinesh, N., Inclezan, D.: Three lessons in creating a knowledge base to enable explanation, reasoning and dialog. In: Klenk, M., Laird, J. (eds.) Proceedings of the 2nd Annual Conference on Advances in Cognitive Systems, pp. 187–203 (2013)
20. Cok, D., Stump, A., Weber, T.: The 2013 evaluation of SMT-COMP and SMT-LIB. J. Autom. Reason. **55**(1), 61–90 (2015)
21. Crouse, M., et al.: A deep reinforcement learning approach to first-order logic theorem proving. In: Leyton-Brown, K., Mausam (eds.) Proceedings of the 35th AAAI Conference on Artificial Intelligence, vol. 35, no. 7, pp. 6279–6287. AAAI Press (2021)
22. D'Silva, V., Kroening, D., Weissenbacher, G.: A survey of automated techniques for formal software verification. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(7), 1165–1178 (2008)
23. Duarte, A., Korovin, K.: Implementing superposition in iprover (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 388–397. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_24
24. Fichte, J.K., Hecher, M., Szeider, S.: A time leap challenge for SAT-solving. In: Simonis, H. (ed.) CP 2020. LNCS, vol. 12333, pp. 267–285. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58475-7_16

25. Fréchette, A., Kotthoff, L., Michalak, T., Rahwan, T., Hoos, H., Leyton-Brown, K.: Using the shapley value to analyze algorithm portfolios. In: Schuurmans, D., Wellman, M. (eds.) Proceedings of the 30th AAAI Conference on Artificial Intelligence, pp. 3397–3403. AAAI Press (2016)
26. Fuchs, M., Sutcliffe, G.: Homogeneous sets of ATP problems. In: Haller, S., Simmons, G. (eds.) Proceedings of the 15th International FLAIRS Conference, pp. 57–61. AAAI Press (2002)
27. Gleiss, B., Suda, M.: Layered clause selection for theory reasoning. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12166, pp. 402–409. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51074-9_23
28. Gleiss, B., Kovács, L., Rath, J.: Subsumption demodulation in first-order theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12166, pp. 297–315. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51074-9_17
29. Henkin, L., Monk, J., Tarski, A.: Cylindrical Algebras, vol. Part 1. North-Holland (1971)
30. Hernandez, J., Korovin, K.: Towards an abstraction-refinement framework for reasoning with large theories. In: Eiter, T., Sands, D., Schulz, S., Urban, J., Sutcliffe, G., Voronkov, A. (eds.) Proceedings of the IWIL Workshop and LPAR Short Presentations. No. 1 in Kalpa Publications in Computing (2017)
31. Hoder, K., Voronkov, A.: Sine qua non for large theory reasoning. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS (LNAI), vol. 6803, pp. 299–314. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22438-6_23
32. Jakubův, J., Chvalovský, K., Olšák, M., Piotrowski, B., Suda, M., Urban, J.: ENIGMA anonymous: symbol-independent inference guiding machine (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 448–463. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_29
33. Jakubuv, J., Urban, J.: BliStrTune: hierarchical invention of theorem proving strategies. In: Bertot, Y., Vafeiadis, V. (eds.) Proceedings of Certified Programs and Proofs 2017, pp. 43–52. ACM (2017)
34. Jakubův, J., Urban, J.: ENIGMA: efficient learning-based inference guiding machine. In: Geuvers, H., England, M., Hasan, O., Rabe, F., Teschke, O. (eds.) CICM 2017. LNCS (LNAI), vol. 10383, pp. 292–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62075-6_20
35. Jakubuv, J., Urban, J.: Hammering mizar by learning clause guidance. In: Proceedings of the 10th International Conference on Interactive Theorem Proving. Leibniz International Proceedings in Informatics, Dagstuhl Publishing (2019)
36. Järvisalo, M., Le Berre, D., Roussel, O., Simon, L.: The international SAT solver competitions. AI Mag. **33**(1), 89–92 (2012)
37. Kaliszyk, C., Schulz, S., Urban, J., Vyskočil, J.: System description: E.T. 0.1. In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 389–398. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_27
38. Korovin, K.: iProver – an instantiation-based theorem prover for first-order logic (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 292–298. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71070-7_24
39. Korovin, K., Kovac, L., Reger, G., J., S., Voronkov, A.: ALASCA: Reasoning in Quantified Linear Arithmetic (Extended Version) (2023). https://easychair.org/publications/preprint/KJX2

40. Kotelnikov, E., Kovacs, L., Reger, G., Voronkov, A.: The vampire and the FOOL. In: Avigad, J., Chlipala, A. (eds.) Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, pp. 37–48. ACM (2016)
41. Kotthoff, L., Fréchette, A., Michalak, T., Rahwan, T., Hoos, H., Leyton-Brown, K.: Quantifying algorithmic improvements over time. In: Lang, J. (ed.) Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 5165–5171 (2018)
42. Külwein, D., Blanchette, J.: A survey of axiom selection as a machine learning problem. In: Geschke, S. (ed.) Computability and Metamathematics: Festschrift Celebrating the 60th birthdays of Peter Koepke and Philip Welch, pp. 1–15. College Publications (2014)
43. Kumar, R., Myreen, M., Norrish, M., Owens, S.: CakeML: a verified implementation of ML. In: Sewell, P. (ed.) Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 179–191. ACM Press (2014)
44. Loechner, B., Hillenbrand, T.: A phytography of waldmeister. AI Commun. **15**(2/3), 127–133 (2002)
45. Maslej, N., et al.: The AI Index 2023 Annual Report. Institute for Human-Centered AI, Stanford University (2023)
46. McCune, W.: Prover9. http://www.cs.unm.edu/~mccune/prover9/
47. McCune, W.: Solution of the robbins problem. J. Autom. Reason. **19**(3), 263–276 (1997)
48. McCune, W.: Otter 3.3 reference manual. Technical report, ANL/MSC-TM-263, Argonne National Laboratory, Argonne, USA (2003)
49. McKeown, J., Sutcliffe, G.: Reinforcement learning for guiding the e theorem prover. In: Ae Chun, A., Franklin, M. (eds.) Proceedings of the 36th International FLAIRS Conference (2023). https://doi.org/10.32473/flairs.36.133334
50. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
51. Nelson, G., Oppen, D.: Simplification by cooperating decision procedures. ACM Trans. Program. Lang. Syst. **1**(2), 245–257 (1979)
52. Ott, S., Barbosa-Silva, A., Blagec, K., Brauner, J., Samwald, M.: Mapping global dynamics of benchmark creation and saturation in artificial intelligence. Nat. Commun. **13**(6793), 1–11 (2022)
53. Paulson, L., Blanchette, J.: Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In: Sutcliffe, G., Ternovska, E., Schulz, S. (eds.) Proceedings of the 8th International Workshop on the Implementation of Logics, pp. 1–11. No. 2 in EPiC Series in Computing, EasyChair Publications (2010)
54. Pelletier, F., Sutcliffe, G., Suttner, C.: The development of CASC. AI Commun. **15**(2–3), 79–90 (2002)
55. Peter, L., Hull, R.: The Peter Principle. Souvenir Press (1969)
56. Quaife, A.: Automated development of Tarski's geometry. J. Autom. Reason. **5**(1), 97–118 (1989)
57. Rümmer, P.: A constraint sequent calculus for first-order logic with linear integer arithmetic. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 274–289. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89439-1_20

58. Schulz, S.: System description: E 1.8. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR 2013. LNCS, vol. 8312, pp. 735–743. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45221-5_49
59. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 495–507. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_29
60. Schwabbauser, W., Szmielew, W., Tarski, A.: Metamathematische Methoden in der Geometrie. Springer, Heidelberg (1983)
61. Slaney, J., Fujita, M., Stickel, M.: Automated reasoning and exhaustive search: quasigroup existence problems. Comput. Math. Appl. **29**(2), 115–132 (1995)
62. Smallbone, N.: Twee: an equational theorem prover. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 602–613. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_35
63. Stump, A., Sutcliffe, G., Tinelli, C.: StarExec: a cross-community infrastructure for logic solving. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS (LNAI), vol. 8562, pp. 367–373. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08587-6_28
64. Suda, M.: Vampire getting noisy: will random bits help conquer chaos? (system description). In: Blanchette, J., Kovacs, L., Pattinson, D. (eds.) IJCAR 2022. LNCS, vol. 13385, pp. 659–667. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10769-6_38
65. Sutcliffe, G.: TPTP, TSTP, CASC, etc. In: Diekert, V., Volkov, M.V., Voronkov, A. (eds.) CSR 2007. LNCS, vol. 4649, pp. 6–22. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74510-5_4
66. Sutcliffe, G.: The TPTP problem library and associated infrastructure. The FOF and CNF parts, v3.5.0. J. Autom. Reason. **43**(4), 337–362 (2009)
67. Sutcliffe, G.: The TPTP world – infrastructure for automated reasoning. In: Clarke, E.M., Voronkov, A. (eds.) LPAR 2010. LNCS (LNAI), vol. 6355, pp. 1–12. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17511-4_1
68. Sutcliffe, G.: The CADE ATP system competition - CASC. AI Mag. **37**(2), 99–101 (2016)
69. Sutcliffe, G.: The TPTP problem library and associated infrastructure. From CNF to TH0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017)
70. Sutcliffe, G., Benzmüller, C.: Automated reasoning in higher-order logic using the TPTP THF infrastructure. J. Formaliz. Reason. **3**(1), 1–27 (2010)
71. Sutcliffe, G., Desharnais, M.: The 11th IJCAR automated theorem proving system competition - CASC-J11. AI Commun. **36**(2), 73–91 (2023)
72. Sutcliffe, G., Desharnais, M.: The CADE-29 automated theorem proving system competition - CASC-29. AI Commun. (2024, to appear)
73. Sutcliffe, G., Fuchs, M., Suttner, C.: Progress in automated theorem proving, 1997-2001. In: Hoos, H., Stützle, T. (eds.) Proceedings of the IJCAI'01 Workshop on Empirical Methods in Artificial Intelligence, pp. 53–60 (2001)
74. Sutcliffe, G., Schulz, S., Claessen, K., Baumgartner, P.: The TPTP typed first-order form with arithmetic. In: Bjørner, N., Voronkov, A. (eds.) LPAR 2012. LNCS, vol. 7180, pp. 406–419. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28717-6_32
75. Sutcliffe, G., Suttner, C.: The state of CASC. AI Commun. **19**(1), 35–48 (2006)
76. Sutcliffe, G., Suttner, C.: Evaluating general purpose automated theorem proving systems. Artif. Intell. **131**(1–2), 39–54 (2001)

77. Sutcliffe, G., Zimmer, J., Schulz, S.: Communication formalisms for automated theorem proving tools. In: Sorge, V., Colton, S., Fisher, M., Gow, J. (eds.) Proceedings of the Workshop on Agents and Automated Reasoning, pp. 52–57 (2003)
78. Suttner, C., Sutcliffe, G., Perrault, R.: Technical performance of automated theorem proving (ATP). In: Zhang, D., et al. (eds.) The AI Index 2021 Annual Report, pp. 34–35. Human-Centered AI Institute, Stanford University (2021)
79. Suttner, C., Schumann, J.: Parallel automated theorem proving. In: Kanal, L., Kumar, V., Kitano, H., Suttner, C. (eds.) Parallel Processing for Artificial Intelligence 1, pp. 209–257. Elsevier Science (1994)
80. Urban, J.: MPTP 0.2: design, implementation, and initial experiments. J. Autom. Reason. **37**(1-2), 21–43 (2006)
81. Veroff, R.: A Wos challenge met. J. Autom. Reason. **66**, 565–574 (2022)
82. Voronkov, A.: Spider: Learning in the Sea of Options (2023). https://easychair.org/smart-program/Vampire23/2023-07-05.html
83. Vukmirović, P., Blanchette, J.C., Cruanes, S., Schulz, S.: Extending a brainiac prover to lambda-free higher-order logic. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 192–210. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_11
84. Wang, H., et al.: LEGO-Prover: Neural Theorem Proving with Growing Libraries (2023). arXiv:2310.00656
85. Weidenbach, C., et al.: System description: Spass version 1.0.0. In: Ganzinger, H. (ed.) CADE 1999. LNCS (LNAI), vol. 1632, pp. 378–382. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48660-7_34
86. Wos, L.: From the AAR President, Larry Wos. AAR Newsletter 129-2019-10 (2019)
87. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Evaluating component solver contributions to portfolio-based algorithm selectors. In: Cimatti, A., Sebastiani, R. (eds.) SAT 2012. LNCS, vol. 7317, pp. 228–241. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31612-8_18
88. Xu, Y., Liu, J., Chen, S., Zhong, X., He, X.: Contradiction separation based dynamic multi-clause synergized automated deduction. Inf. Sci. **462**, 93–113 (2018)
89. Zheng, K., Han, J., Polu, S.: miniF2F: a cross-system benchmark for formal olympiad-level mathematics. In: Liu, Y., Finn, C., Choi, Y., Deisenroth, M. (eds.) Proceedings of the 10th International Conference on Learning Representations (2022)