Stepping Stones in the TPTP World

Geoff Sutcliffe

University of Miami, Miami, USA
geoff@cs.miami.edu

Abstract

The first release of the TPTP problem library was made on Friday 12th November 1993. Since then the TPTP World (once gently referred to as the "TPTP Jungle") has evolved into a well established infrastructure that supports research, development, and deployment of ATP systems. There have been some key developments that helped make the TPTP World a success: the first TPTP problem library that was first released in 1993, the CADE ATP System Competition (CASC) that was conceived after CADE-12 in Nancy in 1994, the problem difficulty ratings that were added in 1997, the current TPTP language that was adopted in 2003, the SZS ontologies that were specified in 2004, the TSTP solution library that was built starting around 2005, the Specialist Problem Classes (SPCs) used to classify problems from 2010, the SystemOnTPTP service that was offered from 2011, and the StarExec service that started in 2013. This talk reviews these stepping stones in the development of the TPTP World.

1 Introduction

The TPTP World [12, 14] is a well established infrastructure that supports research, development, and deployment of ATP systems. Salient components of the TPTP World are the TPTP problem library [11], the TSTP solution library [12], the TPTP languages [19], the SZS ontologies [10], the Specialist Problem Classes (SPCs) and problem difficulty ratings [21], and the CADE ATP System Competition (CASC) [13]. StarExec [7] and SystemOnTPTP [8] provide computational support for the TPTP World. There are dependencies between these parts of the TPTP World, as shown in Figure 1, forming a series of "stepping stones" from key starting points through to happy users, who contribute to TPTP problem library. There is another cycle of dependencies: from the TPTP problem library, to the TSTP solution library, to the problem difficulty ratings, and back to the TPTP problem library. This cycle means that building these components, releases of the TPTP problem library in particular, requires iteration until stability is reached.

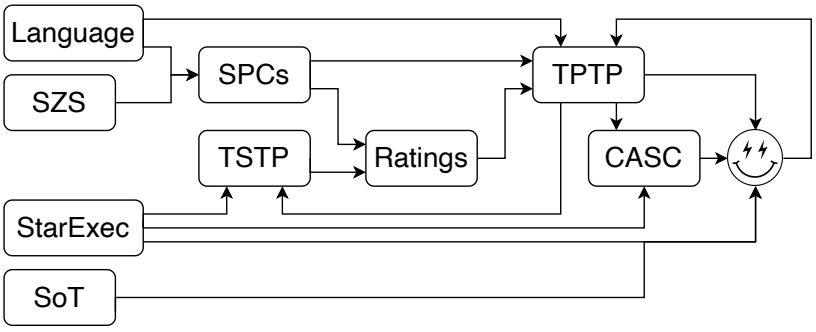

Figure 1: Dependencies between the Stepping Stones

Various parts of the TPTP World have been deployed in a range of applications, in both academia and industry. Since the first release of the TPTP problem library in 1993, many researchers have used the TPTP World as an appropriate and convenient basis for ATP system research and development. Over the years the TPTP World has provided a platform upon which ATP users have presented their needs to ATP system developers, who have then adapted their ATP systems to the users' needs. The web page `https://www.tptp.org` provides access to all components.

This paper is organized as follows:

2 The TPTP Languages

The TPTP language [15] is one of the keys to the success of the TPTP World. The language is used for writing both problems and solutions, which enables convenient communication between systems. Originally the TPTP World supported only first-order clause normal form (CNF) [20]. Over the years full first-order form (FOF) [11], typed first-order form (TFF) [18, 1], typed extended first-order form (TXF) [17], typed higher-order form (THF) [16, 3], and non-classical forms (NTF) [6] have been added. A general principle of the TPTP language is "we provide the syntax, you provide the semantics". As such, there is no a priori commitment to any semantics for the languages, although in almost all cases the intended logic and semantics are well known.

The formulae of problems solutions are built from *annotated formulae*, which have the form ...

language`(`*name*`,` *role*`,` *formula*`,` *source*`,` *useful_info*`)`

The *language*s supported are `cnf` (clause normal form), `fof` (first-order form), `tff` (typed first-order form), and `thf` (typed higher-order form). The *role*, e.g., `axiom`, `lemma`, `conjecture`, defines the use of the formula in an ATP system. In a *formula*, terms and atoms follow Prolog conventions – functions and predicates start with a lowercase letter or are `'single quoted'`, and variables start with an uppercase letter. The language also supports interpreted symbols, which either start with a `$`, e.g., the truth constants `$true` and `$false`, or are composed of non-alphabetic characters, e.g., integer/rational/real numbers such as 27, 43/92, -99.66. The logical connectives in the TPTP language are `!`, `?`, `~`, `|`, `&`, `=>`, `<=`, `<=>`, and `<~>`, for the mathematical connectives $\forall$, $\exists$, $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$, and $\oplus$ respectively. Equality and inequality are expressed as the infix operators `=` and `!=`. The *source* and *useful_info* are optional. Figure 2 shows an example with typed higher-order formulae.

3 The TPTP Problem Library

The development of the TPTP World started with the TPTP problem library in mid-1992, as a collaboration between Geoff Sutcliffe at the University of Western Australia (James Cook University from 1993), and Christian Suttner at the Technische Universität München. The TPTP problem library is managed in the manner of a software product, in the sense that fixed releases are made. Each release is identified by a release number in the form v$Version.Edition.Patch$: the $Version$ enumerates major new releases of the TPTP in which important new features have been added, the $Edition$ is incremented each time new problems are added to the current version, and The $Patch$ level is incremented each time errors, found in the current edition, are corrected. The first public release, TPTP v1.0.0, was made on 12th November 1993.

```
%--------------------------------------------------------------------------
thf(beverage_decl,type,   beverage: $tType ).
thf(syrup_decl,type,      syrup: $tType ).
thf(coffee_type,type,     coffee: beverage ).
thf(mix_type,type,        mix: beverage > syrup > beverage ).
thf(heat_type,type,       heat: beverage > beverage ).
thf(heated_mix_type,type, heated_mix: beverage > syrup > beverage ).
thf(hot_type,type,        hot: beverage > $o ).

thf(heated_mix,axiom,
    ( heated_mix
    = ( ^ [B: beverage,S: syrup] : ( heat @ ( mix @ B @ S ) ) ) ) ).

thf(hot_mixture,axiom,
    ! [B: beverage,S: syrup] : ( hot @ ( heated_mix @ B @ S ) ) ).

thf(heated_coffee_mix,axiom,
    ! [S: syrup] : ( ( heated_mix @ coffee @ S ) = coffee ) ).

thf(hot_coffee,conjecture,
    ? [Mixture: syrup > beverage] :
      ~ ? [S: syrup] :
          ( ( ( Mixture @ S ) = coffee )
          & ( hot @ ( Mixture @ S ) ) ) ).
%--------------------------------------------------------------------------
```

Figure 2: THF annotated formulae

The problems in the TPTP are classified into *domains* that reflect the natural hierarchy of scientific domains. Seven main fields are defined: logic, mathematics, computer science, science & engineering, social sciences, arts & humanities, and other. Each field is subdivided into domains, each identified by a three-letter mnemonic, e.g., the social science field has three domains: Social Choice Theory (SCT), Management (NGT), and Geography (GEG).

Table 1 lists the versions of the TPTP up to v9.0.0, with the new feature added, the number of problem domains, and the number of problems.[1] The number of domains indicates the semantic diversity of the TPTP problems, while the number of problems indicates the size of the TPTP problem library. The attentive reader might note that many releases have been made in July/August. This is because the CADE ATP System Competition (CASC - see Section 9), has an influence on the release cycle of the TPTP.

Release	Date	Changes	Domains	Problems
v1.0.0	12/11/93	First public release, only CNF [20]	23	2295
v2.0.0	05/06/97	FOF [11] and ratings (Section 7)	28	3277
v3.0.0	11/11/04	New TPTP language [19]	32	7267
v4.0.0	04/07/09	TH0 (monomorphic typed higher-order) [16]	41	16512
v5.0.0	16/09/10	TF0 (monomorphic typed first-order) [18]	45	18480
v6.0.0	21/09/13	TF1 (polymophic typed first-order) [1]	48	20306
v7.0.0	24/07/17	TH1 (polymophic typed higher-order) [3]	53	21851
v8.0.0	19/04/22	TXF (typed extended first-order) [17]	54	24785
v9.0.0	??/07/24	NTF (non-classical typed first-order) [6]	55	25598

Table 1: Overview of TPTP releases

[1] The data for v9.0.0 is an estimate, because this paper was written before the release was finalised.

TPTP problem files present the logical formulae in a format that is both human and machine readable, and additionally provide useful information for users. The file names are built from the domain acronym, a 3 digit problem number, a separator that indicates the syntax (- for CNF, + for FOF, _ for TFF, ^ for THF), optional digits for the problem size, and a problem version number. Each file has three sections: a header, optional includes, and the formulae.

The header section contains information for users, formatted as comments 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 include section is optional, and if used contains `include` directives for axiom files, which in turn have the same three-part format as problem files. Their inclusion avoids the need for duplication of the formulae in commonly used axiomatizations. The formula section contains annotated formulae, as described in Section 2. Figure 3 shows an example header and `include` section. The header fields are self-explanatory, but of particular interest are the `Status` field that is explained in Section 5, the `Rating` field that is explained in Section 7, and the `SPC` field that is explained in Section 6.

```
%------------------------------------------------------------------------------
% File     : DAT016_1 : TPTP v8.2.0. Bugfixed v5.1.0.
% Domain   : Data Structures
% Problem  : Some element is 53
% Version  : [PW06] axioms.
% English  : Show that some element of the array has the value 53.

% Refs     : [PW06]  Prevosto & Waldmann (2006), SPASS+T
%          : [Wal10] Waldmann (2010), Email to Geoff Sutcliffe
% Source   : [Wal10]
% Names    : (40) [PW06]

% Status   : Theorem
% Rating   : 0.25 v8.2.0, 0.12 v7.5.0, 0.30 v7.4.0, 0.12 v7.3.0, etc.
% Syntax   : Number of formulae    :    6 (   1 unt;   3 typ;   0 def)
%            Number of atoms       :   12 (   5 equ)
%            Maximal formula atoms :    4 (   2 avg)
%            Number of connectives :    4 (   0   ~;   1   |;   1   &)
%                                       (   0 <=>;   2  =>;   0  <=;   0 <~>)
%            Maximal formula depth :    6 (   5 avg)
%            Maximal term depth    :    3 (   1 avg)
%            Number of FOOLs       :    5 (   5 fml;   0 var)
%            Number arithmetic     :   16 (   2 atm;   2 fun;   5 num;   7 var)
%            Number of types       :    2 (   1 usr;   1 ari)
%            Number of type conns  :    5 (   2   >;   3   *;   0   +;   0  <<)
%            Number of predicates  :    3 (   1 usr;   0 prp; 2-2 aty)
%            Number of functors    :    9 (   2 usr;   5 con; 0-3 aty)
%            Number of variables   :   10 (   9   !;   1   ?;  10   :)
% SPC      : TF0_THM_EQU_ARI

% Comments : The array contains integers.
% Bugfixes : v5.1.0 - Fixed conjecture
%------------------------------------------------------------------------------
%----Includes axioms for arrays
include('Axioms/DAT001_0.ax').
%------------------------------------------------------------------------------
```

Figure 3: Header of problem `DAT016_1`.

This section has naturally focused on the successful parts of the TPTP history. There have also been some failed developments and suboptimal (in retrospect) decisions ☹. For example,

in 2015 there was an attempt to develop a description logic form for the TPTP language. While some initial progress was made, it ground to a halt without support from the description logic community. A suboptimal design decision, rooted in the early days of the TPTP, is the naming scheme used for problem files. The naming scheme uses three digits to number the problems in each domain, thus setting a limit of 1000 problems, which failed to anticipate the numbers of problems that would be contributed to some of the problem domains. This has been overcome by creating multiple domain directories where necessary, but if it were to be done again, six or eight digit problem numbers shared across all domains would be an improvement.

4 The TSTP Solution Library

The complement of the problem library is the TSTP solution library [9, 12]. 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 (therefore, e.g., systems that do model finding for FOF are not run on THF problems). This produced 1091026 runs, of which 432718 (39.6%) solved the problem.

Since 2014 the ATP systems have been run on StarExec [7], initially on the StarExec Iowa cluster, and since 2018 on the StarExec Miami cluster (see Section 8. StarExec has provided stable platforms that produce reliably consistent and comparable data in the TSTP. Analysis shows that increasing resource limits does not significantly affect which problems are solved by an ATP system. Figure 4 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 data was taken from the TSTP, i.e., using the StarExec Miami computers. 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 [5] 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.

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. Another use is for computing the TPTP problem ratings (see Section 7).

5 The SZS Ontologies

The TPTP infrastructure uses the three SZS ontologies to help facilitate automatic processing of TPTP problems and ATP systems' output [10] (named "SZS" after the authors of the first presentation of the ontologies [22]). The ontologies provide values to specify the logical status of problems, and to describe logical data. Figure 5 shows some of the salient nodes of the ontologies. The Success ontology provides values for the logical status of a conjecture with respect to a set of axioms, e.g., a TPTP problem whose conjecture is a logical consequence of the axioms is tagged as a `Theorem` (as in Figure 3), and a model finder that establishes that a set of axioms is consistent should report `Satisfiable`. The Success ontology can also be used

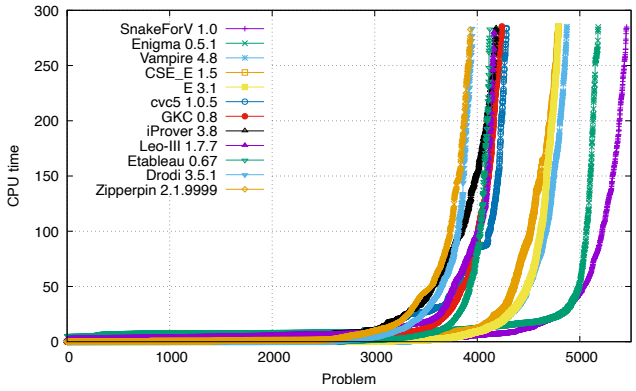

Figure 4: CPU times for `FOF_THM_RFO_*`

to specify the semantic relationship between the parents and inferred formula of an inference, as done in TPTP format derivations [19] (see Section 3 for an example). The NoSuccess ontology catalogs reasons why an ATP system/tool has failed, e.g., an ATP system might report `Timeout`. The Dataform ontology provides values for describing the form of logical data, as might be output from an ATP system/tool, e.g., a model finder might output a `FiniteModel`. The SZS standard also recommends the precise way in which the ontology values should be presented in ATP system output, in order to facilitate easy processing.

6 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. The appropriate level of subdivision for SPCs is that at which less subdivision would merge SPCs for which ATP systems have distinguishable behaviour, and at which further subdivision would unnecessarily split an SPC for which ATP systems have reasonably homogeneous behaviour. 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 [23] performed differently on the ALC problems in the SYN domain of the TPTP. A data-driven test of homogeneity is also possible [2].

The characteristics currently used to define the SPCs in the TPTP are ...

- TPTP language:
  `CNF` – Clause Normal Form `FOF` – First-Order Form
  `TF0` – 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
- SZS status:

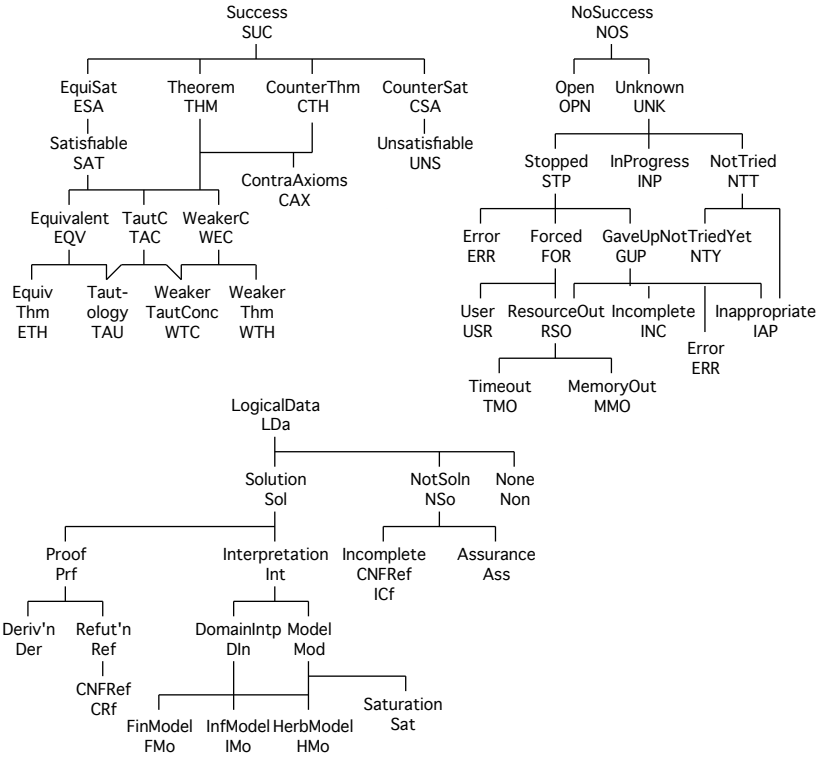

Figure 5: The SZS ontologies

`THM` – Theorem `CSA` – CounterSAtisfiable
`CAX` – Contradictory AXioms (merged with `THM` in this work)
`UNS` – UNSatisfiable `SAT` – SATisfiable
`UNK` – UNKown `OPN` – OPeN

- 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`)
- 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`
- Hornness (for `CNF`):
  `HRN` – HoRN `NHN` – Non-HorN
- 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. The header section of each problem in the TPTP problem library (see Section **??**) includes its SPC.

The SPCs are used when computing the TPTP problems difficulty ratings, as explained in Section 7.

7 Problem Difficulty Ratings

Each TPTP problem has a difficulty rating that provides a well-defined measure of how difficult the problem is for current ATP systems [21]. The ratings are based on performance data in the TSTP (see Section 4), 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").

8 StarExec and SystemOnTPTP

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 300s CPU time limit and a 128GiB memory limit.

9 The CADE ATP System Competition

The CADE ATP System Competition (CASC) [13] is the annual evaluation of fully automatic, classical logic, 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 can be 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 evaluates the performance of the ATP systems in terms of the number of problems solved, the number of acceptable solutions (proofs or models) output, and the average time taken for problems solved, in the context of a bounded number of eligible problems and specified time limits.

CASC is held at each CADE (the International Conference on Automated Deduction) and IJCAR (the International Joint Conference on Automated Reasoning) conference – the major forums for the presentation of new research in all aspects of automated deduction. Over the years CASC has been a catalyst for impressive improvements in ATP, stimulating both theoretical and implementation advances [4]. It has provided a forum at which empirically successful implementation efforts are acknowledged and applauded, and at the same time provides a focused meeting at which novice and experienced developers exchange ideas and techniques. The CASC web site provides access to all the details: `www.tptp.org/CASC`.

The first CASC was held at CADE-13 in 1996, at DIMACS in Rutgers University, USA, in collaboration with Christian Suttner. Of particular interest for this IJCAR is that CASC was conceived of in 1994 after CADE-12 in Nancy, when we were sitting on a bench under a tree in Parc de la Pépinière, burning time before our train departures.

CASC is run in divisions according to problem and system characteristics. Over the years the following divisions have existed (those marked with an * are the divisions for CASC-25):

- THF*: **T**yped **H**igher-order **F**orm theorems (axioms with a provable conjecture).
- THN*: **T**yped **H**igher-order form **N**on-theorems (axioms with a countersatisfiable (i.e., unprovable) conjecture, and satisfiable axiom sets).
- TFA*: **T**yped **F**irst-order with **A**rithmetic theorems (axioms with a provable conjecture).
- TFN*: **T**yped **F**irst-order with arithmetic **N**on-theorems (axioms with a countersatisfiable conjecture, and satisfiable axiom sets).
- FOF*: **F**irst-**O**rder **F**orm theorems (axioms with a provable conjecture).
- FNT*: **F**irst-order form **N**on-**T**heorems (axioms with a countersatisfiable conjecture, and satisfiable axiom sets).
- CNF: **C**lause **N**ormal **F**orm theorems (unsatisfiable clause sets) that are not effectively propositional[2], and not unit equality problems (see the UEQ division below).
- SAT: **C**lause **N**ormal **F**orm non-theorems (satisfiable clause sets) that are not effectively propositional, and not unit equality problems (see the UEQ division below).
- EPR*: **E**ffectively **PR**opositional theorems and non-theorems (unsatisfiable and satisfiable clause sets).
- UEQ: **U**nit **EQ**uality theorems (unsatisfiable clause sets) that are not effectively propositional.
- SEM: FOF theorems based on a specified axiomatization of a specified **SEM**antic domain.
- LTB*: First-order form theorems (axioms with a provable conjecture) from **L**arge **T**heories, presented in **B**atches with a shared time limit.

The different logics and syntactic characteristics of the problems in the various divisions provide different challenges for ATP systems. The tasks of proving theorems and showing unsatisfiability (which can be treated similarly) are quite distinct from establishing non-provability and satisfiablility (which can also be treated similarly).

Problems for CASC are taken from the TPTP Problem Library. The TPTP version used for CASC is released after the competition, so that new problems have not been seen by the entrants. In some divisions the systems are ranked according to the number of problems solved with an acceptable proof/model output, and some divisions the systems are ranked according to the number of problems solved but not necessarily accompanied by a proof or model (thus giving only an assurance of the existence of a proof/model). Ties are broken according to the average time over problems solved. Division winners are announced and prizes are awarded. In addition to the ranking criteria, three other measures are made and presented in the results: The *state-of-the-art (SoTA) contribution* quantifies the unique abilities of each system. For each problem solved by a system, its SoTA contribution for the problem is the inverse of the number of systems that solved the problem, and it's overall SoTA contribution is the average SoTA contribution over the problems it solved. The *efficiency measure* is a combined measure that balances the time taken for each problem solved against the number of problems solved. It is the average of the inverses of the times for problems solved, This can be interpreted intuitively as the average of the solution rates for problems solved, multiplied by the fraction of problems solved. The *core usage* is the average of the ratios of CPU time to wall clock time used, over the problems solved. This measures the extent to which the systems take advantage of multiple cores.

[2] *Effectively propositional* means that the problem is known to be reducible to a propositional problem, e.g., a CNF problem that has no functions with arity greater than zero.

CASC typically has 20 to 30 ATP systems entered. For each CASC the division winners of the previous CASC are automatically entered to provide benchmarks against which progress can be judged. Additionally, a fixed version (initially v3.2, later v3.3) of the well known Otter ATP system was entered in every CASC from 2002 to 2011, as a fixed point against which progress could be judged. By 2011 Otter was no longer competitive, and was replaced by Prover9 2009-11A in 2012. Over all 20 CASCs so far 99 distinct ATP systems have been entered. Almost all the ATP systems have come from academia, partially due to the CASC requirement that all source code must be published on the CASC web site. The most popular divisions have been the FOF, FNT, CNF, SAT, EPR, and UEQ divisions. Some systems have emerged as dominant in some of the divisions: Satallax in the THF division, Vampire in the FOF and CNF divisions, Paradox in the FNT and SAT divisions (with iProver now coming on strong), iProver in the EPR division, and Waldmeister in the UEQ division. The strengths of these systems stem from four main areas: solid theoretical foundations, significant implementation efforts (in terms of coding and data structures), extensive testing and tuning, and an understanding of how to optimize for CASC. For example, Vampire is founded on the theoretical principles of superposition, has a highly efficient implementation in C++ using code trees and advanced structures for representing logical data, is repeatedly tested and tuned on the TPTP problem library, and has special modes for the various divisions of CASC. Technical information about these systems, and the techniques they employ, can be found on the individual CASC web pages.

The design and organization of CASC has evolved over the years to a sophisticated state. Decisions made for CASC (alongside the TPTP) have had an influence on the directions of development in ATP. It is interesting to look back on some of the key decisions that have helped bring the competition to its current state.

- CASC-13, 1996: The first CASC stimulated research towards robust, fully automatic systems that take only logical formulae as input. It increased the visibility of systems and developers, and rewarded implementation efforts.
- CASC-14, 1997: Introduced the SAT division, stimulating the development of model finding systems for CNF.
- CASC-15, 1998: Introduced the FOF division, starting the slow demise of CNF to becoming just the "assembly language" of ATP.
- CASC-16, 1999: Changes to the problem selection process motivated the development of techniques for automatic tuning of ATP systems' search parameters.
- CASC-JC, 2001: Introduced ranking based on proof output, starting the the trend towards ATP systems that efficiently output proofs and models. Introduced the EPR division, stimulating the development of specialized techniques for this important subclass of problems.
- CASC-20, 2005: Required systems to develop builtin equality reasoning, by removing the equality axioms from all TPTP problems.
- CASC-J3, 2006: The FOF division was promoted as the most important, stimulating development of ATP systems for full first-order logic.
- CASC-21, 2007: Introduced the FNT division, further stimulating the development of model finding systems.
- CASC-J4, 2008: Introduced the LTB division, stimulating the development of techniques for automatically dealing with very large axiom sets.
- CASC-J5, 2010: Introduced the THF division, stimulating development of ATP systems for higher-order logic.

- CASC-23, 2011: Introduced the TFA division, stimulating development of ATP systems for full first-order logic with arithmetic.
- CASC-J6, 2012: Otter replaced by Prover9 as the "fixed-point" in the FOF division, demonstrating the progress in ATP.
- CASC-24, 2013: Removed the CNF division, confirming the demise of CNF.
- CASC-J7, 2014: Required use of the SZS ontology, so the ATP systems unambiguously report what they have established about the problem.
- CASC-25, 2015: Introduced the THN and TFN divisions, stimulating development of model finding for the THF and TFA logics.

Over the years the TPTP and CASC have increasingly been used as a conduit for ATP users to provide samples of their problems to ATP system developers. Users' problems that are contributed to the TPTP are eligible for use in CASC. The problems are then exposed to ATP system developers, who improve their systems' performances on the problems, in order to perform well in CASC. This completes a cycle that provides the users with more effective tools for solving their problems.

10 TPTP World Users

11 Conclusion

This paper

Currently this work is being extended to

References

[1] J. Blanchette and A. Paskevich. TFF1: The TPTP Typed First-order Form with Rank-1 Polymorphism. In M.P. Bonacina, editor, *Proceedings of the 24th International Conference on Automated Deduction*, number 7898 in Lecture Notes in Artificial Intelligence, pages 414–420. Springer-Verlag, 2013.

[2] M. Fuchs and G. Sutcliffe. Homogeneous Sets of ATP Problems. In S. Haller and G. Simmons, editors, *Proceedings of the 15th International FLAIRS Conference*, pages 57–61. AAAI Press, 2002.

[3] C. Kaliszyk, G. Sutcliffe, and F. Rabe. TH1: The TPTP Typed Higher-Order Form with Rank-1 Polymorphism. In P. Fontaine, S. Schulz, and J. Urban, editors, *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning*, number 1635 in CEUR Workshop Proceedings, pages 41–55, 2016.

[4] R. Nieuwenhuis. The Impact of CASC in the Development of Automated Deduction Systems. *AI Communications*, 15(2-3):77–78, 2002.

[5] L.J. Peter and R. Hull. *The Peter Principle*. Souvenir Press, 1969.

[6] A. Steen, D. Fuenmayor, T. Gleißner, G. Sutcliffe, and C. Benzmüller. Automated Reasoning in Non-classical Logics in the TPTP World. In B. Konev, C. Schon, and A. Steen, editors, *Proceedings of the 8th Workshop on Practical Aspects of Automated Reasoning*, number 3201 in CEUR Workshop Proceedings, page Online, 2022.

[7] 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.

[8] G. Sutcliffe. SystemOnTPTP. In D. McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction*, number 1831 in Lecture Notes in Artificial Intelligence, pages 406–410. Springer-Verlag, 2000.

[9] G. Sutcliffe. TPTP, TSTP, CASC, etc. In V. Diekert, M. Volkov, and A. Voronkov, editors, *Proceedings of the 2nd International Symposium on Computer Science in Russia*, number 4649 in Lecture Notes in Computer Science, pages 6–22. Springer-Verlag, 2007.

[10] G. Sutcliffe. The SZS Ontologies for Automated Reasoning Software. In G. Sutcliffe, P. Rudnicki, R. Schmidt, B. Konev, and S. Schulz, editors, *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and the 7th International Workshop on the Implementation of Logics*, number 418 in CEUR Workshop Proceedings, pages 38–49, 2008.

[11] 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.

[12] G. Sutcliffe. The TPTP World - Infrastructure for Automated Reasoning. In E. Clarke and A. Voronkov, editors, *Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, number 6355 in Lecture Notes in Artificial Intelligence, pages 1–12. Springer-Verlag, 2010.

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

[14] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[15] G. Sutcliffe. The Logic Languages of the TPTP World. *Logic Journal of the IGPL*, 31(6):1153–1169, 2023.

[16] G. Sutcliffe and C. Benzmüller. Automated Reasoning in Higher-Order Logic using the TPTP THF Infrastructure. *Journal of Formalized Reasoning*, 3(1):1–27, 2010.

[17] G. Sutcliffe and E. Kotelnikov. TFX: The TPTP Extended Typed First-order Form. In B. Konev, J. Urban, and S. Schulz, editors, *Proceedings of the 6th Workshop on Practical Aspects of Automated Reasoning*, number 2162 in CEUR Workshop Proceedings, pages 72–87, 2018.

[18] G. Sutcliffe, S. Schulz, K. Claessen, and P. Baumgartner. The TPTP Typed First-order Form with Arithmetic. In N. Bjørner and A. Voronkov, editors, *Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, number 7180 in Lecture Notes in Artificial Intelligence, pages 406–419. Springer-Verlag, 2012.

[19] G. Sutcliffe, S. Schulz, K. Claessen, and A. Van Gelder. Using the TPTP Language for Writing Derivations and Finite Interpretations. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, number 4130 in Lecture Notes in Artificial Intelligence, pages 67–81. Springer, 2006.

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

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

[22] G. Sutcliffe, J. Zimmer, and S. Schulz. Communication Formalisms for Automated Theorem Proving Tools. In V. Sorge, S. Colton, M. Fisher, and J. Gow, editors, *Proceedings of the Workshop on Agents and Automated Reasoning, 18th International Joint Conference on Artificial Intelligence*, pages 52–57, 2003.

[23] C. Weidenbach, B. Afshordel, U. Brahm, C. Cohrs, T. Engel, E. Keen, C. Theobalt, and D. Tpoic. System Description: SPASS Version 1.0.0. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 378–382. Springer-Verlag, 1999.