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 paper reviews these stepping stones in the development of the TPTP World.

1 Introduction

The TPTP World [20, 22] 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 [19], the TSTP solution library [20], the TPTP languages [28], the SZS ontologies [18], the Specialist Problem Classes (SPCs) and problem difficulty ratings [31], and the CADE ATP System Competition (CASC) [21]. SystemOnTPTP [16] and StarExec [15] 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 TPTP standards to happy users ...

- The TPTP language (see Section 2) is used for writing problems in the TPTP problem library and the TSTP solution library.
- The SZS ontologies (see Section 5) are used to specify properties of problems and solutions.
- The language form and SZS ontology values classify problems into Special Problem Classes (see Section 6).
- The TPTP problem library (see Section 3) is the central collection of test problems.
- The TSTP solution library (see Section 4) is the central collection of solutions to the TPTP library problems.
- The TPTP problems' difficulty ratings (see Section 7) are computed wrt each SPC, using data from the TSTP.
- The CADE ATP System Competition (see Section 9) is the annual evaluation of ATP systems - the world championship for such systems.
- SystemOnTPTP (see Section 8) provides online access to ATP systems and tools.
- The StarExec computers (see Section 8) are used to build the TSTP, and to run CASC.
- Users of the TPTP World (see Section 10) provide problems for the TPTP problem library, and ATP systems/tools for SystemOnTPTP and StarExec.

There is a 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, particularly building releases of the TPTP problem library, requires iteration until stability is reached.

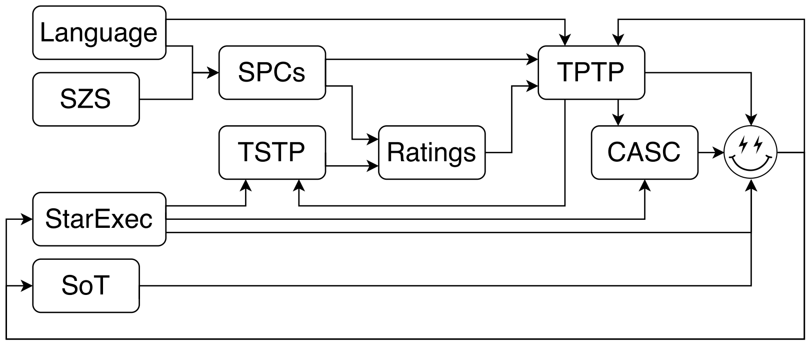

Figure 1: Dependencies between the Stepping Stones

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. 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 [23] 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) [30]. Over the years full first-order form (FOF) [19], typed first-order form (TFF) [27, 1], typed extended first-order form (TXF) [25], typed higher-order form (THF) [24, 4], and non-classical forms (NTF) [14] have been added. The standardisation of the language received a signifcant technical boost in 2006 when the BNF definition was revised to a state that a parser could be automatically generated using `lex/yacc` [33], and all the language forms are now quite precisely specified.[1] 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

[1]But note that the BNF "syntax" is not completely strict – the BNF uses an extended BNF form that relgates details to "semantic" rules that do not have to be encoded in a parser.

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.

```
%------------------------------------------------------------------------------
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

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. At that time a large number of interesting problems had accumulated in the ATP community, in both hardcopy [6, 35, 9, 2, 11, 7] and electronic form [5, 13].[2] The TPTP problem library was designed to centralize and unify these disparate collections, to provide a comprehensive library of ATP test problems in a simple, unambiguous, reference mechanism. 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

[2] To my knowledge, the first circulation of test problems was by Larry Wos in the late sixties.

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.

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 (MGT), 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.[3] 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 [30]	23	2295
v2.0.0	05/06/97	FOF [19] and ratings (Section 7)	28	3277
v3.0.0	11/11/04	New TPTP language [28]	32	7267
v4.0.0	04/07/09	TH0 (monomorphic typed higher-order) [24]	41	16512
v5.0.0	16/09/10	TF0 (monomorphic typed first-order) [27]	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) [4]	53	21851
v8.0.0	19/04/22	TXF (typed extended first-order) [25]	54	24785
v9.0.0	??/07/24	NTF (non-classical typed first-order) [14]	55	25598

Table 1: Overview of TPTP releases

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.

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

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

```
%------------------------------------------------------------------------------
% 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.

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 [17, 20]. 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. The TSTP started being built around 2005, using data provided by individual system developers. From 2010 to 2013 the TSTP was generated on the TPTP World servers at the University of Miami. Since 2014 the ATP systems have been run on StarExec, 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. 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. One use of the TSTP is for computing the TPTP problem difficulty ratings (see Section 7).

TSTP solution files have structure that mimics the TPTP problem files. The header section has four parts: the first part identifies the ATP system, the problem, and the system's runtime parameters; the second part provides information about the hardware, operating system, and resource limits; the third part provides the SZS result and output values (see Sec tion 5), and syntactic characteristics of the solution; the last part contains comments. The formula section contains annotated formulae.

For derivations, where formulae are derived from parent formulae, e.g., in proofs, refutations, etc., the *source* fields of the annotated formulae are used to capture parent-derived formulae relationships in the derivation DAG. This includes source of the formula – either the problem file or an inference. Inference data includes the name of the inference rule used, the semantic relationship between the parents and the derived formula as an SZS ontology value (see Section 5), and a list of the parent annotated formulae names. Figure 5 shows an example refutation from the E system [12] [slightly modified] for the problem formulae in Figure 2, and Figure 4 shows the corresponding header fields.

```
%------------------------------------------------------------------------------
% File     : E---3.0.04
% Problem  : Coffee
% Transfm  : none
% Format   : tptp:raw
% Command  : run_E %s %d THM

% Computer : quokka.cs.miami.edu
% Model    : x86_64 x86_64
% CPU      : Intel(R) Xeon(R) CPU E5-2609 v2 @ 2.50GHz
% Memory   : 64222MB
% OS       : Linux 3.10.0-1160.36.2.el7.x86_64
% CPULimit : 30s
% WCLimit  : 0s
% DateTime : Tue Feb 13 13:29:34 EST 2024

% Result   : Theorem 0.00s 0.08s
% Output   : Refutation 0.00s
% Verified :
% SZS Type : Refutation
%            Derivation depth      :    5
%            Number of leaves      :   10
% Syntax   : Number of formulae    :   19 (   8 unt;   7 typ;   0 def)
%            Number of atoms       :   16 (   7 equ;   0 cnn)
%            Maximal formula atoms :    2 (   1 avg)
%            Number of connectives :   42 (   6   ~;   2   |;   2   &;  32   @)
%                                         (   0 <=>;   0  =>;   0  <=;   0 <~>)
%            Maximal formula depth :    7 (   5 avg)
%            Number of types       :    3 (   2 usr)
%            Number of type conns  :   11 (  11   >;   0   *;   0   +;   0  <<)
%            Number of symbols     :    7 (   5 usr;   2 con; 0-2 aty)
%            Number of variables   :   15 (   0   ^  13   !;   2   ?;  15   :)

% Comments :
%------------------------------------------------------------------------------
```

Figure 4: Example derivation header

For interpretations (typically models) the annotated formulae are used to describe the domains and symbol mappings of Tarskian interpretations, or the formulae that induce Herbrand

```
%------------------------------------------------------------------------------
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(decl_27,type,         esk1_1: ( syrup > beverage ) > syrup ).
thf(decl_28,type,         esk2_1: ( syrup > beverage ) > syrup ).

thf(hot_coffee,conjecture,
    ? [X3: syrup > beverage] :
    ! [X2: syrup] :
      ( ( ( X3 @ X2 ) = coffee ) & ( hot @ ( X3 @ X2 ) ) ),
    file('Coffee.p',hot_coffee) ).

thf(heated_coffee_mix,axiom,
    ! [X2: syrup] :
      ( ( heated_mix @ coffee @ X2 ) = coffee ),
    file('Coffee.p',heated_coffee_mix) ).

thf(hot_mixture,axiom,
    ! [X1: beverage,X2: syrup] : ( hot @ ( heated_mix @ X1 @ X2 ) ),
    file('Coffee.p',hot_mixture) ).

thf(c_0_3,negated_conjecture,
    ~ ? [X3: syrup > beverage] :
      ! [X2: syrup] :
        ( ( ( X3 @ X2 ) = coffee ) & ( hot @ ( X3 @ X2 ) ) ),
    inference(assume_negation,[status(cth)],[hot_coffee]) ).

thf(c_0_4,negated_conjecture,
    ! [X16: syrup > beverage] :
      ( ( ( X16 @ ( esk1_1 @ X16 ) ) != coffee )
      | ~ ( hot @ ( X16 @ ( esk2_1 @ X16 ) ) ) ),
    inference(fof_nnf,[status(thm)],[inference(skolemize,[status(esa)],
[inference(variable_rename,[status(thm)],[inference(shift_quantors,[status(thm)],
[inference(fof_nnf,[status(thm)],[c_0_3])])])])]) ).

thf(c_0_10,negated_conjecture,
    ~ ( hot @ coffee ),
    inference(cn,[status(thm)],[inference(rw,[status(thm)],
[inference(spm,[status(thm)],[c_0_4,heated_coffee_mix]),heated_coffee_mix])]) ).

thf(c_0_11,plain,
    $false,
    inference(sr,[status(thm)],[inference(spm,[status(thm)],
[hot_mixture,heated_coffee_mix]),c_0_10]),[proof] ).
%------------------------------------------------------------------------------
```

Figure 5: Example derivation formulae

models. A TPTP format for interpretations with finite domains was previously been defined [28], and has served the ATP community adequately for almost 20 years. Recently the need for a format for interpretations with infinite domains, and for a format for Kripke interpretations, has led to the development of a new TPTP format for interpretations [29]. Figure 6 shows the problem formulae and a model that uses intergers as the domain. Please read [29] for lots more details!

```
%---- Problem formulae --------------------------------------------------------
tff(person_type,type,        person: $tType ).
tff(bob_decl,type,           bob: person ).
tff(child_of_decl,type,      child_of: person > person ).
tff(is_descendant_decl,type, is_descendant: (person * person) > $o ).

tff(descendents_different,axiom,
    ! [A: person,D: person] :
      ( is_descendant(A,D) => ( A != D ) ) ).

tff(descendent_transitive,axiom,
    ! [A: person,C: person,G: person] :
      ( ( is_descendant(A,C) & is_descendant(C,G) )
     => is_descendant(A,G) ) ).

tff(child_is_descendant,axiom,
    ! [P: person] : is_descendant(P,child_of(P)) ).

tff(all_have_child,axiom,
    ! [P: person] : ? [C: person] : C = child_of(P) ).
%------------------------------------------------------------------------------

%---- Model -------------------------------------------------------------------
tff(person_type,type,        person: $tType ).
tff(bob_decl,type,           bob: person ).
tff(child_of_decl,type,      child_of: person > person ).
tff(is_descendant_decl,type, is_descendant: ( person * person ) > $o ).

tff(int2person_decl,type,    int2person: $int > person ).

tff(people,interpretation,
%----Domain for type person is the integers
    ( ( ! [P: person] : ? [I: $int] : int2person(I) = P
%----The type promoter is a bijection (injective and surjective)
      & ! [I1: $int,I2: $int] :
          ( int2person(I1) = int2person(I2) => I1 = I2 ) )
%----Mapping people to integers. Note that Bob's ancestors will be interpreted
%----as negative integers.
    & ( bob = int2person(0)
      & ! [I: $int] : child_of(int2person(I)) = int2person($sum(I,1)) )
%----Interpretation of descendancy
    & ! [A: $int,D: $int] :
        ( is_descendant(int2person(A),int2person(D)) <=> $less(A,D) ) ) ).
%------------------------------------------------------------------------------
```

Figure 6: Example infinite model

Resource Limits: A common question, and often a misbelief, is whether the resource limits used should be increased to find more solutions. Analysis shows that increasing resource limits does not significantly affect which problems are solved by an ATP system. Figure 7 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 [10] 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, and if enough CPU time is 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.

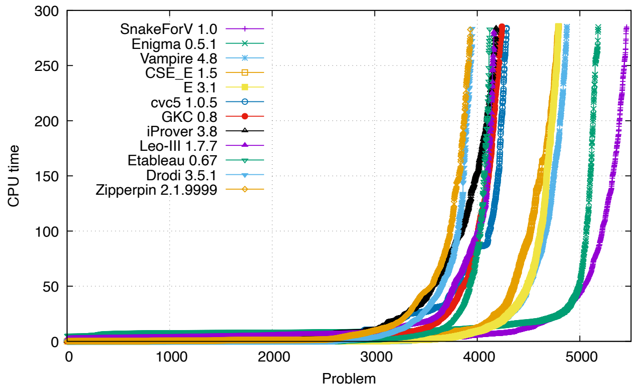

Figure 7: CPU times for `FOF_THM_RFO_*`

5 The SZS Ontologies

The SZS ontologies [18] (named "SZS" after the authors of the first presentation of the ontologies [32]) provide values to specify the logical status of problems, the status of ATP systems' (and other tools) outputs, and to describe logical data. Examples include whether a conjecture is a `Theorem` of the problem axioms, if an axiomatization is `Satisfiable`, that the output from a model finder is a `FiniteModel`, or that a reasoning system had a `TimeOut`. The Success ontology provides values for the logical status of a conjecture "C" with respect to a set of axioms "Ax", 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 (with no conjecture) is consistent should report `Satisfiable`. The Success ontology is also used to specify the semantic relationship between the parents "Ax" and inferred formulae "C" of an inference, as done in TPTP format derivations (see Section 4). 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`. Figure 8 shows some of the salient nodes of the ontologies. Their expanded names and their (abbreviated) definitions are listed in Figure 9.

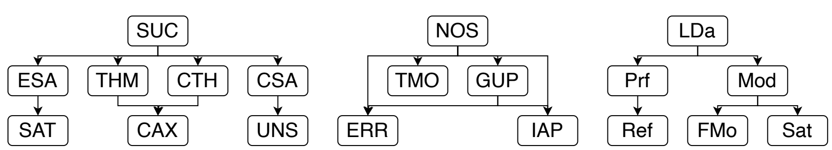

Figure 8: Extract of the SZS ontologies

The SZS standard also specifies the precise way in which the ontology values should be presented in ATP system output, in order to facilitate easy processing. For example, if an ATP system has established that a conjecture is not a theorem of the axioms, by finding a finite countermodel of the axioms and negated conjecture, the SZS format output would be (see

`SUC`	`Success`	The logical data has been processed successfully.
`ESA`	`Equisatisfiable`	There exists a model of Ax iff there exists a model of C.
`THM`	`Satisfiable`	Some interpretations are models of Ax.
`THM`	`Theorem`	All models of Ax are models of C.
`CTH`	`CounterTheorem`	All models of Ax are models of C.
`CAX`	`ContradictoryAxioms`	No interpretations are models of Ax.
`CSA`	`CounterSatisfiable`	Some models of Ax are models of C.
`UNS`	`Unsatisfiable`	All interpretations are models of Ax.
`NOS`	`NoSuccess`	The logical data has not been processed successfully.
`ERR`	`Error`	Stopped due to an error.
`TMO`	`Timeout`	Stopped because a time limit ran out.
`GUP`	`GaveUp`	Gave up of its own accord.
`IAP`	`Inappropriate`	Gave up because it cannot process this type of data.
`LDa`	`LogicalData`	Logical data.
`Prf`	`Proof`	A proof.
`Ref`	`Refutation`	A refutation (ending with $false$).
`Mod`	`Model`	A model.
`FMo`	`FiniteModel`	A model with a finite domain.
`Sat`	`Saturation`	A saturated set of formulae.

Figure 9: SZS ontology values

Section 4 for the format of the annotated formulae) ...

```
% SZS status CounterSatisfiable
% SZS output start FiniteModel
... annotated formulae for the finite model
% SZS output end FiniteModel
```

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 [34] performed differently on the ALC problems in the SYN domain of the TPTP. A data-driven test of homogeneity is also possible [3].

The characteristics currently used to define the SPCs in the TPTP are shown in Figure 10. Using these characteristics 223 SPCs are defined in TPTP v8.2.0. For example, the SPC `TF0_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 3) includes its SPC. The SPCs are used when computing the TPTP problems difficulty ratings, as explained in Section 7.

- TPTP language:
  CNF – Clause Normal Form; FOF – First-Order Form
  TF0/1 – Typed First-order Monomorphic/Polymorphic
  TX0/1 – Typed eXtended f-o Monomorphic/Polymorphic
  TH0/1 – Typed Higher-order Monomorphic/Polymorphic
  NX0/1 – Non-classical TX0/1; NH0/1 – Non-classical TH0/1
- SZS status:
  THM – Theorem; CSA – CounterSAtisfiable
  CAX – Contradictory AXioms
  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.

Figure 10: SPC characteristics

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 [31]. 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. CLEANING 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”).

The TPTP difficulty ratings of problems (that are unchanged) provide a way to assess progress in the field – as the problems are not actually getting easier decreases in the difficulty rating are evidence of progress in ATP systems. Figures 11 and 12 plot the average difficulty ratings overall and for each of the four language forms in the TPTP World (after some sensible data cleaning). Figure 11 is taken from [22], published in 2017. It plots the average ratings for the 14527 problems that had been unchanged and whose ratings had not been stuck at 0.00 or 1.00, from v5.0.0 that was released in 2010 to v6.4.0 that was released in 2016. Figure 12 is taken from [26], published in 2024. It plots the average ratings for the 16236 problems that had been unchanged and whose ratings had not been stuck at 0.00 or 1.00, from v6.3.0 that

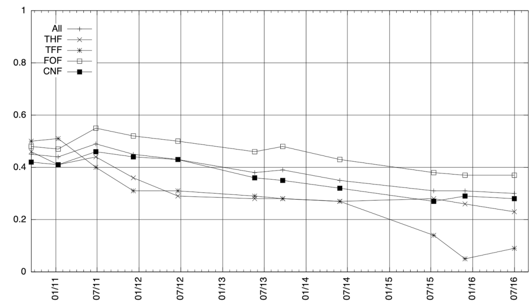

Figure 11: Ratings from v5.0.0 to v6.4.0

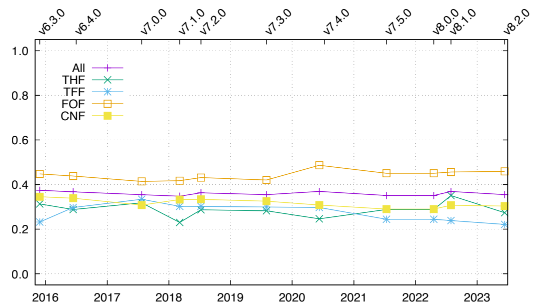

Figure 12: Ratings from v6.3.0 to v8.2.0

was released in 2016 to v8.2.0 that was released in 2023. The two figures' plots dovetail quite well, which gives confidence that they really are comparable (there are some minor differences caused by data cleaning and recent refinements to the difficulty rating calculations). The older plots show a quite clear downward trend both overall and for the four language forms, while the new plots do not. Possible reasons are discussed in [26].

8 SystemOnTPTP and StarExec

SystemOnTPTP - data on hits in 2023. Main user Sledgehammer.

Grant. Iowa then Miami. 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.

Number of users, jobs, job pairs in 2023.

Iowa being decommissioned, Miami will run longer. New containerised StarExec with containerised ATP systems, all in a Kubernetes framework on AWS.

9 The CADE ATP System Competition

The CADE ATP System Competition (CASC) [21] is the annual evaluation of fully automatic, classical logic, ATP systems - the world championship for such systems. 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. 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. Over the years CASC has been a catalyst for impressive improvements in ATP, stimulating both theoretical and implementation advances [8]. 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`.

CASC is run in divisions according to problem and system characteristics. 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, and some other sources. The systems are ranked according to the number of problems solved with an acceptable proof/model output. 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. The *efficiency measure* is a combined measure that balances the time taken for each problem solved against the number of problems solved. The *core usage* is the average of the ratios of CPU time to wall clock time used, over the problems solved.

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. ZZZZ 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. Some systems have emerged as dominant in some of the divisions, with Vampire being a well-known example. 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.

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: 1st CASC, with the CNF division.
- CASC-14, 1997: 2nd CASC, added the SAT division.
- CASC-15, 1998: 3rd CASC, added the FOF division.
- CASC-JC, 2001: 6th CASC, added the EPR division. Ranking based on proof output.
- CASC-21, 2007: 12th CASC, added the FNT division.
- CASC-J4, 2008: 13th CASC, added the LTB division.
- CASC-J5, 2010: 15th CASC, added the THF division.
- CASC-23, 2011: 16th CASC, added the TFA division.
- CASC-24, 2013: 18th CASC, removed the CNF division.
- CASC-J7, 2014: 19th CASC. Required use of the SZS ontology.
- CASC-25, 2015: 20th CASC, added the TFN division.
- CASC-26, 2017: 22nd CASC, added the SLH division.
- CASC-J12, 2024: 28th CASC, added the ICU division.

In the years from CASC-26 to CASC-J12 the competition stayed quite stable, but each year the various divisions, evaluations, etc., were optimized (as was also the case in prior years when step changes also peturbed the competition design).

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.

10 TPTP World Users

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.

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.

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.

Discuss people:

- Christian Suttner - TPTP problem library and CASC
- Adam Pease and Josef Urban - Large theory problems
- Stephan Schulz - Technical advice
- Jasmin Blanchette - Sledgehammer to SystemOnTPTP, TF1
- Allen van Gelder - BNF and parser
- asmin Blanchette and Andrei Paskevich - TF1
- Evgeny Kotelnikov - TXF
- Christoph Benzüller and Chad Brown - TH0
- Christoph Benzüller and Alexander Steen - NTF
- Koen Claessen and Peter Baumgartner and Stephan Schulz - TF0
- Cezary Kaliszyk and Florian Rabe - TH1

11 Conclusion

This paper

Currently this work is being extended to

None of this would have happened without the early strategic insights of Christian Suttner, with his willingness to let me do the organization without interference. Maybe his most wonderful contribution (which took him over two hours to produce when he was visiting me at James Cook University – I think he took a nap ☺) is his plain-language definition of automated theorem proving: "the derivation of conclusions that follow inevitably from known facts".

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] R. Boyer, Lusk E.L., W.W. McCune, R. Overbeek, M.E. Stickel, and L. Wos. Set Theory in First-Order Logic: Clauses for Godel's Axioms. *Journal of Automated Reasoning*, 2(3):287–327, 1986.

[3] 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.

[4] 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.

[5] Argonne National Laboratory. The Argonne National Laboratory Problem Collection. Previously available from http://info.mcs.anl.gov/.

[6] J. McCharen, R. Overbeek, and L. Wos. Problems and Experiments for and with Automated Theorem-Proving Programs. *IEEE Transactions on Computers*, C-25(8):773–782, 1976.

[7] W.W. McCune and L. Wos. Experiments in Automated Deduction with Condensed Detachment. In Kapur D., editor, *Proceedings of the 11th International Conference on Automated Deduction*, number 607 in Lecture Notes in Artificial Intelligence, pages 209–223. Springer-Verlag, 1992.

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

[9] F.J. Pelletier. Seventy-five Problems for Testing Automatic Theorem Provers. *Journal of Automated Reasoning*, 2(2):191–216, 1986.

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

[11] A. Quaife. Automated Deduction in von Neumann-Bernays-Godel Set Theory. *Journal of Automated Reasoning*, 8(1):91–147, 1992.

[12] S. Schulz, S. Cruanes, and P. Vukmirović. Faster, Higher, Stronger: E 2.3. In P. Fontaine, editor, *Proceedings of the 27th International Conference on Automated Deduction*, number 11716 in Lecture Notes in Computer Science, pages 495–507. Springer-Verlag, 2019.

[13] SPRFN. The Problem Collection Distributed with the SPRFN ATP System. https://www.cs.unc.edu/Research/mi/mi-provers.html.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] G. Sutcliffe, L. Kotthoff, C.R. Perrault, and Z. Khalid. An Empirical Assessment of Progress in Automated Theorem Proving. In C. Benzmüller, M. Heule, and R. Schmidt, editors, *Proceedings of the 12th International Joint Conference on Automated Reasoning*, Lecture Notes in Artificial Intelligence, page Submitted confidently, 2024.

[27] 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.

[28] 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.

[29] G. Sutcliffe, A. Steen, and P. Fontaine. The New TPTP Format for Interpretations. In K. Korovin, M. Rawson, and S. Schulz, editors, *Proceedings of the 15th International Workshop on the Implementation of Logics*, page Submitted confidently, 2024.

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

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

[32] 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.

[33] A. Van Gelder and G. Sutcliffe. Extending the TPTP Language to Higher-Order Logic with Automated Parser Generation. 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 156–161. Springer-Verlag, 2006.

[34] 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.

[35] G.A. Wilson and J. Minker. Resolution, Refinements, and Search Strategies: A Comparative Study. *IEEE Transactions on Computers*, C-25(8):782–801, 1976.