J Autom Reasoning
DOI 10.1007/s10817-017-9407-7

The TPTP Problem Library and Associated Infrastructure

From CNF to TH0, TPTP v6.4.0

Geoff Sutcliffe[1]

Received: 29 August 2016 / Accepted: 7 February 2017

Abstract This paper describes the TPTP problem library and associated infrastructure, from its use of Clause Normal Form (CNF), via the First-Order Form (FOF) and Typed First-order Form (TFF), through to the monomorphic Typed Higher-order Form (TH0). TPTP v6.4.0 was the last release prior to the introduction of the polymorphic Typed Higher-order Form, and thus serves as the exemplar. This paper summarizes the aims and history of the TPTP, documents its growth up to v6.4.0, reviews the structure and contents of TPTP problems, and gives an overview of TPTP-related infrastructure.

Keywords TPTP · ATP system evaluation · ATP standards and tools

1 Introduction

The Thousands of Problems for Theorem Provers (TPTP) problem library is the de facto standard set of test problems for classical logic Automated Theorem Proving (ATP) systems. The TPTP supplies the ATP community with a comprehensive library of the test problems that are available today, in order to provide an overview and a simple, unambiguous reference mechanism. Since its first release in 1993, many researchers have used the TPTP as an appropriate and convenient basis for ATP system evaluation. Over the years the TPTP has also increasingly been used as a conduit for ATP users to provide samples of their problems to ATP system developers—this exposes the problems to the developers, who then improve their systems' performances on the problems, which completes a cycle to provide the users with more effective tools. The TPTP is supported by a rich infrastructure of standards, tools, and linked projects. All together these have become known colloquially as the "TPTP world".

The TPTP problem library is managed in the manner of a software product, in the sense that fixed releases are made. Each release of the TPTP is identified by a release number in

✉ Geoff Sutcliffe
geoff@cs.miami.edu

[1] Department of Computer Science, University of Miami, Coral Gables, Florida, USA

Published online: 17 February 2017

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. The $Patch$ level is incremented each time errors, found in the current edition, are corrected. The TPTP web site http://www.tptp.org provides access to all components of the TPTP world, including the TPTP problem library download package, a quick-guide introduction and detailed documentation for the TPTP, online browsing of the TPTP problems and their solutions, an interface for running ATP systems on problems and post-processing the solutions, and links to a range of related projects.

This paper describes the TPTP problem library and associated infrastructure, from its use of Clause Normal Form (CNF), via the First-Order Form (FOF) and Typed First-order Form (TFF, consisting of the monomorphic TF0 and the polymorphic TF1), through to the monomorphic Typed Higher-order Form (TH0). TPTP v6.4.0 was the last release prior to the introduction of the polymorphic Typed Higher-order Form (TH1, which together with TH0 forms THF), and thus serves as the exemplar. Section 2 provides the background and motivation for the TPTP, and gives guidelines for obtaining and using the TPTP. Section 3 summarizes the history of the TPTP, and documents the growth of the TPTP up to TPTP v6.4.0. Section 4 explains various features of the TPTP that provide structure for, and useful information about, TPTP problems and solutions. Section 5 reviews the contents of TPTP problems. Section 6 gives an overview of TPTP-related infrastructure. Section 7 concludes, discussing current developments and future directions.

This paper is a natural successor to the 1998 paper that described the CNF part of the TPTP using TPTP v1.2.1 as the exemplar [44], and the 2009 paper that described the FOF and CNF parts of the TPTP using TPTP v3.5.0 as the exemplar [34]. This paper is not fully self-contained—it assumes some familiarity with the TPTP world. Neophytes are encouraged to read [34] and [35] to obtain a useful background for this paper.

2 Aims of the TPTP Project

Prior to the development of the TPTP a large number of interesting problems had accumulated in the ATP community, in print form, e.g., [24,28,54], and as electronic collections, e.g., the Argonne National Laboratory collection [19] and the SPRFN collection [26]. Although some of these early collections provided significant support to researchers, and formed the early core of the TPTP, none, with the possible exception of the Argonne collection, were designed to serve as a common basis for ATP research. The goal for building the TPTP was to overcome the previous drawbacks, and to centralize the burden of problem collection and maintenance to one place. In particular, the TPTP is easy to discover and obtain, is easy to use, is comprehensive, and is up-to-date. It is independent of any particular ATP system, spans a diversity of subject matters, is large enough for statistically significant testing, and is well structured and documented. The TPTP has an unambiguous naming scheme for the problems, documents each problem, contains problems varying in difficulty with a difficulty rating for each problem, provides a classification of problems according to their semantic and syntactic characteristics, and provides statistics for each problem and the library as a whole. The TPTP is supported with mechanisms for adding new problems and correcting errors in existing problems.

Appropriate use of the TPTP allows developers and users to meaningfully evaluate ATP systems. To that end, the following guidelines for using the TPTP and presenting results should be followed:

- The TPTP release number must be stated.
- Each problem must be referenced by its unambiguous TPTP name.
- The formulae should, as far as is possible, not be changed in any way. Any changes made (addition, removal, reordering, reformatting, etc.) must be explicitly noted.
- Any information given to the ATP system, other than the formulae, must be explicitly noted. All system switches and settings must be recorded. The header information in problems (which includes the provability status and syntactic measures of the problem formulae—see Sect. 5) may not be used by the ATP system without explicit notice.

There are several projects that are analogous to the TPTP, for different logics but with similar aims and operation. Examples include the now defunct SATLIB library of propositional satisfiability problems [14], the QBFLib library of quantified boolean formulae problems [20], the CSPLib library of constraint satisfaction problems [12], and the SMT-LIB of satisfiability modulo theories problems [1]. The SMT-LIB is probably the most similar to the TPTP, most closely to TF0 with arithmetic. SMT-LIB problems are specified in part by formulae, and in part by one or more background theories (which might not be finitely axiomatizable). Example theories are arrays, integer arithmetic, real arithmetic, and bit vectors. The arithmetic theories are almost the same as the TPTP arithmetic. SMT-LIB problems' formulae use a slightly richer form of first-order logic that the TPTP, very similar to the FOOL logic that has been proposed as an extension to the TPTP [17]. The SMT-LIB logics are classified according to the theories and features of the logic that are used, e.g., the AUFLIA logic uses closed formulae over the theory of linear integer arithmetic and arrays, extended with free sort and function symbols but restricted to arrays with integer indices and values. Syntactically, SMT-LIB problems follow LISP conventions, in the same way that TPTP problems follow Prolog conventions (see Sect. 5). TPTP formulae can be output in SMT format, e.g., by using the TPTP4X utility (see Sect. 6), and for some SMT-LIB logics SMT-LIB problems can be translated to TPTP format, e.g., using the SMTtoTPTP tool [2].

3 TPTP History and Growth

The development of the TPTP started 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. A beta release, TPTP v0.5.0, was made to selected researchers on 15th April 1993. The first public release, TPTP v1.0.0, was made on 12th November 1993. The exemplar release of this paper, TPTP v6.4.0, was made on 13th June 2016, and is the forty-second TPTP release.

In the early years of the TPTP (the early 1990s) most ATP systems were based on the resolution inference rule, using the clause normal form of first-order logic. As such, the early releases of the TPTP contained only CNF problems [44], and there was no syntax or support for full first-order form problems. As more ATP users became aware of the TPTP, an increasing number of full first-order form problems were submitted to the TPTP developers, and from TPTP v2.0.0 (released June 1997) the TPTP included FOF problems [34]. TPTP v3.0.0 introduced a new (the current) TPTP language [41], which is now the defacto standard for writing problems for and solutions from TPTP-compliant ATP systems and tools. Thanks to the sup-

port of a European Community grant[1] TH0 problems [36] were introduced in TPTP v4.0.0 (released July 2009). TF0 problems [40] were introduced in TPTP v5.0.0 (released September 2010), with the addition of TF1 problems [7] in TPTP v6.0.0 (released September 2013).

Table 1 lists the editions of the TPTP up to v6.4.0, the major changes, the number of problem domains (see Sect. 4), and the number of problems in each form. The attentive reader might note that many releases have been made in July/August. This is because the CADE ATP System Competition (CASC—see Sect. 6), has had an influence on the release cycle of the TPTP. From the sixth CASC in 2001, the TPTP release used for the competition was not made available until after the competition, to make over-tuning for TPTP problems disadvantageous [43]. This forced releases to be made after each CADE or IJCAR conference (in July or August), from TPTP v2.4.0 (released June 2001).

Figure 1 graphs the number of problems in the TPTP, over time. The trend away from CNF and towards FOF increased significantly from TPTP v3.1.0 (i.e., since mid-2005). The large increase in the number of FOF problems in v3.4.0 (released March 2008) was due to contributions from the Cyc project [18] and the MPTP [48]. The large increase in the number of FOF problems in v4.0.0 (released August 2009) was due to contributions from the SAD project [52], the SUMO ontology [23], the ILTP library [29], and the work of Peter Höfner, e.g., [13]. The large increase in the number of CNF problems in v4.1.0 (released June 2010) was due to contributions from the Judgement Day suite [8] and the work of David Stanovsky, e.g., [25]. The introduction of THF problems in TPTP v4.0.0 augmented the continued growth in the number of FOF problems. The introduction of the TF0 language in TPTP v5.0.0 (released September 2010) helped maintain the overall growth.

Table 2 gives some overview statistics for TPTP v6.4.0, with key characteristics. The problems are mostly known or believed to be theorems or unsatisfiable sets of formulae. Equality occurs in the majority of problems, as it occurs naturally in many domains. There are still relatively few problems that use arithmetic, because the TF0 language that supports arithmetic was introduced only relatively recently, in TPTP v5.0.0. The development of TPTP-compatible ATP systems that support reasoning with arithmetic has progressed significantly since then. The number of TFF problems with arithmetic is expected to grow, because, like equality, arithmetic occurs naturally in many domains. It has been suggested that TFF with arithmetic might become the language of choice for ATP users [38]. The addition of THF problems with arithmetic is in its infancy in the TPTP, and more THF problems with arithmetic are expected in the future.

Table 3 provides problem-level statistics for TPTP v6.4.0. The statistics show the syntactic diversity of the problems in the TPTP. The large FOF problems come from an export of part of the Cyc project's knowledge base [18], while the largest TFF and CNF problems verify properties of the SPARCT2 RTL hardware design [11]. The THF problems are generally smaller, with the largest being number theory problems generated by Sledgehammer [22]. Generally, there is an inverse relationship between the size of TPTP problems and the expressivity of the language (with THF being most expressive, going down to TFF, FOF, and lastly CNF). Part of the reason for this is the trade-off between representational and reasoning reliabilities of the languages [38]. Other factors include the ability of mature CNF and FOF-based systems to cope with large numbers of formulae, and the use of strong axiom selection techniques when exporting THF problems from automated components of ITP systems, e.g., Isabelle's Sledgehammer [6] and the HOL(y)Hammer for HOL Light [16].

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

[1] Seventh Framework Programme FP7/2007-2013, grant agreement PIIF-GA-2008-219982

Table 1 Overview of TPTP releases

Release	Date	Significant changes	Domains	All	THF		TFF		FOF		CNF	
v0.5.0	15/04/93	Beta release			0	0%	0	0%	0	0%	1695	100%
v1.0.0	12/11/93	First public release	23	2295	0	0%	0	0%	0	0%	2295	100%
v1.1.0	08/04/94	TPTP2X overhauled	25	2652	0	0%	0	0%	0	0%	2652	100%
v1.2.0	30/08/95	Generators introduced	25	2752	0	0%	0	0%	0	0%	2752	100%
v2.0.0	05/06/97	FOF and ratings introduced	28	3277	0	0%	0	0%	217	7%	3060	93%
v2.1.0	17/12/97	German distribution stopped	28	3622	0	0%	0	0%	347	10%	3275	90%
v2.2.0	11/02/99	TPTP2X upgraded	28	4004	0	0%	0	0%	670	17%	3334	83%
v2.3.0	16/11/99	Ratings made more accurate	28	4229	0	0%	0	0%	671	16%	3558	84%
v2.4.0	24/06/01	Releases sync'ed with CASC	30	5882	0	0%	0	0%	1463	25%	4419	75%
v2.5.0	28/07/02	VerySimilarProblems listed	30	6672	0	0%	0	0%	1491	22%	5181	78%
v2.6.0	20/08/03		31	6973	0	0%	0	0%	1500	22%	5473	78%
v2.7.0	15/08/04		32	7267	0	0%	0	0%	1745	24%	5522	76%
v3.0.0	11/11/04	New TPTP language	32	7267	0	0%	0	0%	1745	24%	5522	76%
v3.1.0	25/07/05		33	8013	0	0%	0	0%	2324	29%	5689	71%
v3.2.0	19/07/06	Language BNF updated	35	8984	0	0%	0	0%	2724	30%	6260	70%
v3.3.0	28/06/07		35	9894	0	0%	0	0%	3644	37%	6250	63%
v3.4.0	04/03/08	Very large problems added	35	11279	0	0%	0	0%	5029	45%	6250	55%
v3.5.0	13/08/08	Last FOF and CNF release	35	11395	0	0%	0	0%	5049	44%	6346	56%
v3.6.0	16/12/08	Alpha release for THF	35	11680	220	2%	0	0%	5051	43%	6348	54%
v3.7.0	08/03/09	Beta release for THF	36	12674	1275	10%	0	0%	5051	40%	6348	50%
v4.0.0	04/07/09	THF formally introduced	41	16512	2729	17%	0	0%	6983	42%	6800	41%
v4.1.0	15/06/10	SPC field added to headers	43	17663	2892	16%	0	0%	7137	40%	7634	43%
v5.0.0	16/09/10	TF0 introduced	45	18480	2892	16%	817	4%	7137	39%	7634	41%
v5.1.0	12/01/11	Cleaned up THF duplicates	46	18460	2798	15%	891	5%	7137	39%	7634	41%
v5.2.0	26/06/11		47	19137	2860	15%	891	5%	7674	40%	7712	40%

Table 1 continued

Release	Date	Significant changes	Domains	All	THF		TFF		FOF		CNF	
v5.3.0	07/12/11	Language BNF updated	47	19446	2926	15%	970	5%	7807	40%	7743	40%
v5.4.0	19/06/12		47	19459	2929	15%	970	5%	7807	40%	7753	40%
v5.5.0	21/05/13		47	19769	3025	15%	986	5%	7923	40%	7835	40%
v6.0.0	21/09/13	TF1 introduced	48	20306	3025	15%	1523	8%	7923	39%	7835	39%
v6.1.0	29/05/14		50	20646	3036	15%	1757	9%	7971	39%	7882	38%
v6.2.0	14/07/15		51	20654	3041	15%	1757	9%	7974	39%	7882	38%
v6.3.0	28/11/15		51	20762	3041	15%	1863	9%	7976	38%	7882	38%
v6.4.0	31/06/16	Last CNF to TH0 release	51	20897	3077	15%	1877	9%	8045	38%	7898	38%

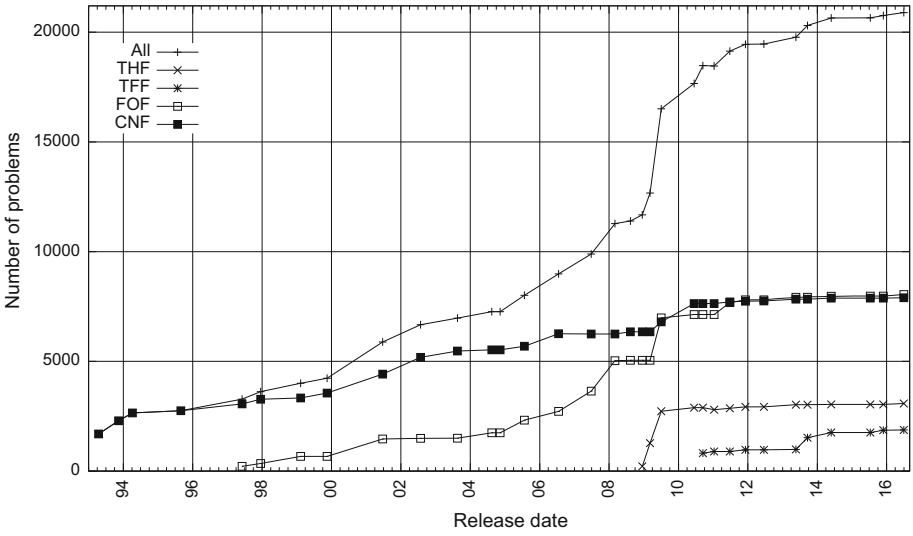

Fig. 1 The growth of the TPTP

Table 2 Statistics for TPTP v6.4.0

	All		THF		TFF		FOF		CNF	
Domains	51		28	55%	18	35%	40	78%	35	68%
Problems	20,897		3077	15%	1877	9%	8044	38%	7899	38%
Known theorems/unsat.	16,591	79%	2492	80%	1451	77%	6753	83%	5895	74%
Known non-theorems/sat.	2619	11%	393	12%	107	5%	940	11%	1179	14%
Problems with equality	15,942	76%	2363	76%	1532	81%	6329	78%	5718	72%
Problems with arithmetic	1700	8%	0	0%	1700	90%	0	0%	0	0%

The percentages for "Domains" and "Problems" are of "All'. The other percentages are for the "Problems" of that form

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 (see Sect. 4). 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 TPTP Features

The problems in the TPTP are classified into *domains* that reflect the natural hierarchy of scientific domains. The classification is based mainly on the Dewey Decimal Classification (DDC) [9] and the Mathematics Subject Classification (MSC) [30]. Seven main fields are

Table 3 Statistics over the TPTP v6.4.0 problems

	Min	Max	Avg	Med
THF				
Number of formulae	1	5639	102	11
Number of atoms	1	63,737	723	75
Number of symbols	1	1442	44	9
Number of variables	0	11,290	152	19
TFF				
Number of formulae	1	340,918	4579	19
Number of atoms	1	1,986,853	15,796	163
Number of predicate symbols	1	271,855	3488	17
Number of function symbols	0	9977	146	16
Number of variables	0	246,172	3379	154
Number of arithmetic symbols	0	30,626	296	3
FOF				
Number of formulae	1	3,341,984	30,886	59
Number of atoms	1	5,328,211	62,125	273
Number of predicate symbols	0	204,678	3395	000
Number of function symbols	0	1,050,014	10,119	12
Number of variables	0	972,236	15,622	127
CNF				
Number of clauses	1	2,332,428	2344	44
Number of literals	1	6,570,884	6978	99
Number of predicate symbols	1	480,215	404	5
Number of function symbols	0	9978	47	12
Number of variables	0	4,034,129	6531	73

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, as shown in Table 4. The range of domains covered shows the semantic diversity of the problems in the TPTP. The most populated domains are set theory and software verification. The large number of set theory problems reflects the ease with which set theory can be encoded in classical logic, most elegantly in THF [4]. The use of both interactive and automated reasoning systems for software verification has been a focus of attention in the automated reasoning community for the last decade, leading to a large number of software verification problems in the TPTP. Many of these are produced as a byproduct of interactive software verification, e.g., by Isabelle's Sledgehammer subsystem [6].

Each TPTP problem has a difficulty rating that provides a well-defined measure of how difficult the problem is for current ATP systems [45]. The ratings are based on performance data in the TSTP [32], generated by currently available ATP systems (see Sect. 6). The ratings range from 0.00 (easy problems) to 1.00 (problems that are unsolved by any current ATP system), with increasing ratings inbetween (difficult problems). The ratings help users select problems that are appropriate for their needs. Figure 2 shows the problem ratings for TPTP v6.4.0 sorted into increasing order, and Table 5 shows the distribution over the three

Table 4 The domain structure of TPTP v6.4.0

Field	Domain and mnemonic		Prob's	THF	TFF	FOF	CNF
Logic	Combinatory logic	COL	239	0	0	0	239
	Logic calculi	LCL	1377	149	100	438	688
	Henkin models	HEN	67	0	0	0	67
Mathematics	Set Theory	SET	3588	1403	0	1378	801
	Graph theory	GRA	127	93	0	33	1
	Algebras						
	Relational algebra	REL	220	0	0	111	109
	Boolean algebra	BOO	140	0	0	1	139
	Robbins algebra	ROB	45	0	0	0	45
	Left distributive	LDA	50	0	0	0	50
	Lattices	LAT	733	0	0	413	320
	Quantales	QUA	21	21	0	0	0
	Kleene algebra	KLE	241	0	0	241	0
	Groups	GRP	1091	1	0	202	888
	Rings	RNG	263	0	0	157	106
	Fields	FLD	281	0	0	0	281
	Homological alg	HAL	10	0	0	10	0
	General algebra	ALG	545	88	0	286	171
	Number theory	NUM	1344	204	176	643	321
	Topology	TOP	131	0	0	107	24
	Analysis	ANA	91	0	0	0	91
	Geometry	GEO	676	0	1	421	254
	Category theory	CAT	131	1	0	68	62
Computer	Computing theory	COM	178	1	102	61	14
Science	Knowledge representation	KRS	305	6	1	268	30
	Natural language processing	NLP	531	11	0	262	258
	Planning	PLA	76	2	0	17	57
	Agents	AGT	76	23	0	53	0
	Commonsense reasoning	CSR	868	82	0	786	0
	Semantic web	SWB	204	0	0	204	0
	Data structures	DAT	113	3	110	0	0
	Software creation	SWC	847	1	0	423	423
	Software verification	SWV	2170	76	380	638	1076
Science and	Biology	BIO	4	0	0	4	0
Engineering	Hardware creation	HWC	6	0	0	0	6
	Hardware verification	HWV	511	0	200	108	203
	Medicine	MED	12	0	0	12	0
	Processes	PRO	72	0	0	72	0
	Products	PRD	3	0	0	3	0

Table 4 continued

Field	Domain and mnemonic		Prob's	THF	TFF	FOF	CNF
Social sciences	Social choice theory	SCT	300	9	105	85	101
	Management	MGT	157	0	0	79	78
	Geography	GEG	24	18	5	1	0
Arts and Humanities	Philosophy	PHI	17	17	0	0	0
Other	Arithmetic	ARI	667	0	667	0	0
	Syntactic	SYN	2091	800	10	411	870
	Puzzles	PUZ	211	61	11	37	102
	Miscellaneous	MSC	43	5	3	11	24

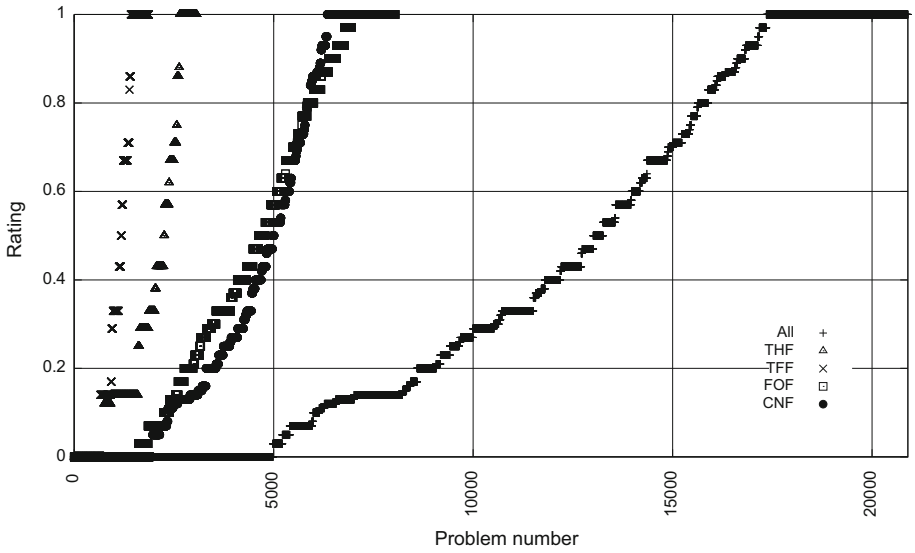

Fig. 2 TPTP v6.4.0 problem ratings

discrete difficulty ranges. The plots and data show that the TPTP contains problems with well distributed difficulty, thus providing test problems that are easy enough for testing newly developed ATP systems, while also providing significant challenges for mature systems. A weakness in the current TPTP is the relatively small number of difficult TFF problems, which is expected to change as the use of TFF increases. Over time, decreasing difficult ratings of individual TPTP problems provide an indication of progress in the field [37]. Figure 3 plots the average ratings for the 14527 problems that have been unchanged in the TPTP since v5.0.0, and whose ratings have not been stuck at 0.00 or 1.00 since v5.0.0.

The ratings generally show a downward trend - there has been progress! Note that ratings can also increase when data from new systems is added to the TSTP. The TPTP infrastructure uses the three SZS ontologies to help facilitate automatic processing of TPTP problems and ATP systems' output [33] (named "SZS" after the authors of the first presentation of the ontologies [46]). The ontologies provide values to specify the logical status of problems, and

Table 5 TPTP v6.4.0 problem difficulties

	All		THF		TFF		FOF		CNF	
Problems	20,897		3077		1877		8044		7899	
Easy	4982	24%	748	24%	655	35%	1615	20%	1964	25%
Difficult	12,368	59%	1893	62%	764	41%	5334	66%	4377	55%
Unsolved	3547	17%	436	14%	458	24%	1095	14%	1558	20%

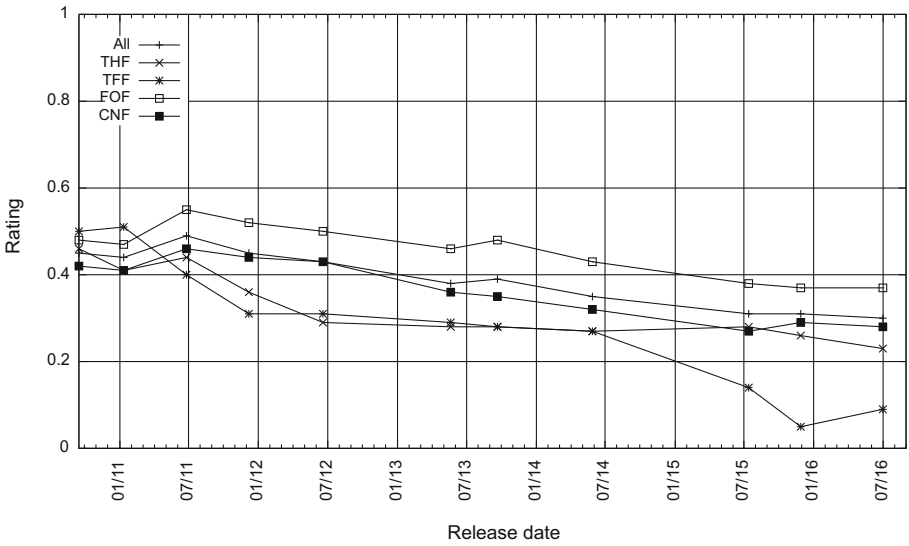

Fig. 3 Ratings decline from TPTP v5.0.0 to TPTP v6.4.0

to describe logical data. Figure 4 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 Fig. 5), and a model finder that establishes that a set of axioms is consistent should report `Satisfiable`. The Success ontology can also be used to specify the semantic relationship between the parents and inferred formula of an inference, as done in TPTP format derivations [41] (see Sect. 5 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.

A useful classification of ATP problems is the TPTP's Specialist Problem Classes (SPCs). An SPC is a syntactically identifiable class of problems that is suited to some known ATP techniques and systems. Every TPTP problem is annotated with its SPC. The SPCs help ATP system developers select test problems that are appropriate for their needs. Examples of SPCs are "TF0 theorems with equality and arithmetic"—the `TF0_THM_EQU_ARI` class (as in Fig. 5), and "FOF satisfiable sets of formulae that are not known to be reducible to propositional problems, with some (not pure) equality"—the `FOF_SAT_RFO_SEQ` class.

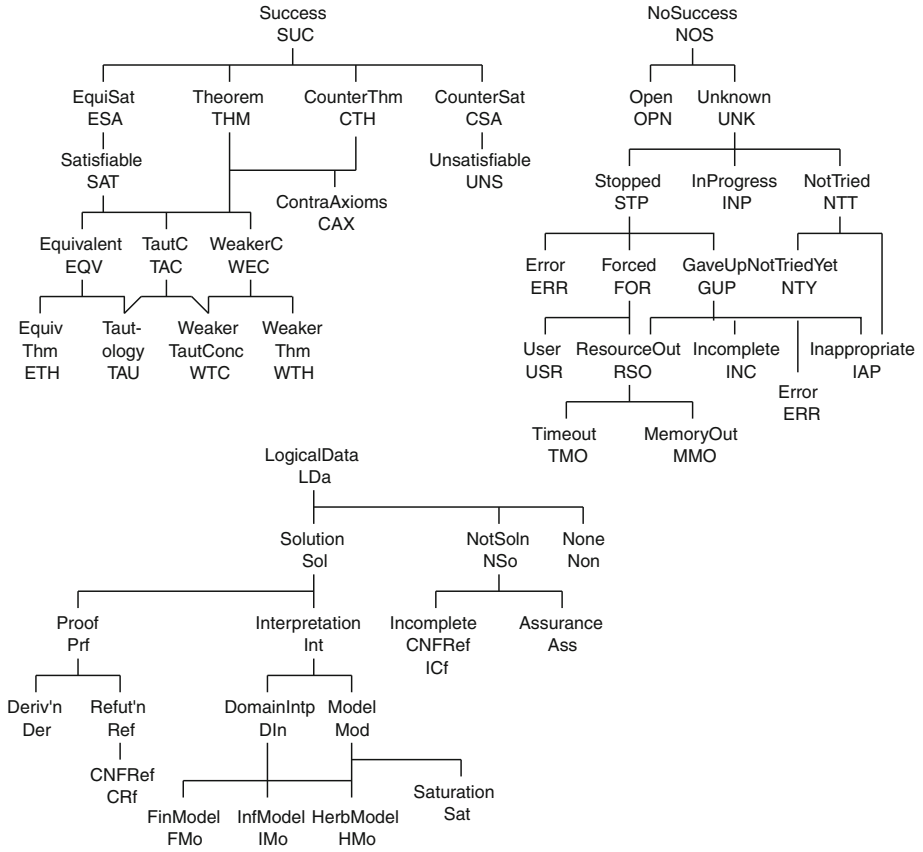

Fig. 4 The SZS ontologies

As well as being directly useful, the SPCs are used in the computation of the TPTP problem difficulty ratings.

5 TPTP Problems

TPTP problem files present the logical formulae in a format that is both human and machine readable, and additionally provide useful information for users. Each file has three sections. Figure 5 is an example of the *header* section, and Fig. 6 is an example of the *include* and *formula* sections.

The header section of a TPTP problem file 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 of a TPTP problem file is optional, and if used contains `include` directives for axiom files. This avoids the need for duplication of the formulae of commonly used axiomatizations.

```
%------------------------------------------------------------------------------
% File     : HWV087=1 : TPTP v6.4.0. Bugfixed v6.2.0.
% Domain   : Hardware Verification
% Problem  : dmu_dmc property 3 cone of influence 5_b20
% Version  : Especial.
% English  : Verification of a property of the SPARCT2 RTL hardware design.

% Refs     : [Kha14] Khasidashvili (2014), Email to Geoff Sutcliffe
% Source   : [Kha14]
% Names    : dmu_dmc_prop3_cone5_b20 [Kha14]

% Status   : Theorem
% Rating   : 0.57 v6.4.0, 0.33 v6.3.0, 0.71 v6.2.0
% Syntax   : Number of formulae    : 1777 ( 238 unit; 672 type)
%            Number of atoms       : 3682 (  21 equality)
%            Maximal formula depth :  128 (   4 average)
%            Number of connectives : 3051 ( 474   ~; 124   |; 731   &)
%                                         (1407 <=>; 315  =>;   0  <=;   0 <~>)
%                                         (   0  ~|;   0  ~&)
%            Number of type conns  :  773 ( 649   >; 124   *;   0   +;   0  <<)
%            Number of predicates  : 1328 ( 677 propositional; 0-2 arity)
%            Number of functors    :  399 ( 399 constant; 0-0 arity)
%            Number of variables   : 1023 (   0 sgn;1023   !;   0   ?)
%                                         (1023   :;   0  !>;   0  ?*)
%            Maximal term depth    :    1 (   1 average)
%            Arithmetic symbols    :  477 (   1 prd;   0 fun; 378 num;  98 var)
% SPC      : TFO_THM_EQU_ARI

% Comments : Copyright 2013 Moshe Emmer and Zurab Khasidashvili
%            Licensed under the Apache License, Version 2.0 (the "License");
%            you may not use this file except in compliance with the License.
% Bugfixes : v6.2.0 - Fixed type declarations.
%------------------------------------------------------------------------------
```

Fig. 5 Header of TPTP problem `HWV087=1`

The logical formulae in a TPTP problem file are wrapped in *annotated formulae* of the form

$$language(name, role, formula, source, useful\_info).$$

The *language*s supported in TPTP v6.4.0 are `thf`, `tff`, `fof`, and `cnf` - formulae in THF, TFF, FOF, and CNF respectively. The *name* identifies the annotated formula. The *role* gives the user semantics of the *formula*, e.g., `axiom`, `lemma`, `conjecture`, and hence defines its use in an ATP system. The logical *formula*e are written in an easily understood and easily parsed notation [41], using only standard ASCII characters. The syntax of atoms and terms is that of Prolog: variables start with upper case letters; uninterpreted predicates and functors either start with lower case and contain only alphanumerics and underscore, or are ‘ single quoted’; uninterpreted first-order atoms and terms are written in prefix notation. The language also supports interpreted symbols that are syntactically distinct from uninterpreted ones, e.g., numbers (which are constants interpreted as themselves), and arithmetic operators such as `$less`. The *source* field is used to record where an annotated formula came from, e.g., from the problem file, or as the result of an inference. The *useful_info* field is used to store arbitrary information about the formula. The *useful_info* is optional, and if it is not used then the *source* becomes optional. An example of a (fictional) inferred annotated formula that uses the *source* and *useful_info* fields is:

```
%------------------------------------------------------------------------------
%----Basic set theory definitions
include('Axioms/SET008^0.ax').
%------------------------------------------------------------------------------
thf(is_function_type,type,(
    is_function: ( $i > $o ) > ( $i > $i ) > ( $i > $o ) > $o )).

thf(is_function,axiom,
    ( is_function
    = ( ^ [X: $i > $o,F: $i > $i,Y: $i > $o] :
        ! [E: $i] :
          ( ( X @ E )
         => ( Y @ ( F @ E ) ) ) ) )).

thf(injection_type,type,(
    injection: ( $i > $o ) > ( $i > $i ) > ( $i > $o ) > $o )).

thf(injection,axiom,
    ( injection
    = ( ^ [X: $i > $o,F: $i > $i,Y: $i > $o] :
          ( ( is_function @ X @ F @ Y )
          & ! [E1: $i,E2: $i] :
              ( ( ( X @ E1 )
                & ( X @ E2 )
                & ( ( F @ E1 )
                  = ( F @ E2 ) ) )
             => ( E1 = E2 ) ) ) ) )).

   ... some similar formulae omitted ...

thf(prove,conjecture,(
    ! [A: $i > $o,Ap: $i > $o,B: $i > $o,Bp: $i > $o] :
      ( ( ( equinumerous @ A @ Ap )
        & ( equinumerous @ B @ Bp )
        & ( ( intersection @ Ap @ Bp )
          = emptyset ) )
     => ( embedding @ ( union @ A @ B ) @ ( union @ Ap @ Bp ) ) ) )).

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

Fig. 6 `include` directive and annotated formulae of TPTP problem `NUM863^1`

```
tff(made_up_07,lemma,(
? [X_1: set,X_2: set] :
( intersection(X_1,X_2) = emptyset
& $greater(cardinality(X_1),0) ) ),
inference(sup_right,[status(thm)],[injection,made_up_03]),
[interestingness(0.72),proved_by(bprover)]).
```

The *source* field records that the formula was inferred using the inference rule `sup_right`, and is a theorem (using the SZS mnemonic) of the two parent formulae of the inference, `injection` and `made_up_03`. The *useful_info* field records a measure of how interesting the formula is [27], and that it was inferred by the ATP system `bprover`.

The TPTP language is one of the keys to the success of the TPTP. The initial, and still an ongoing, principle that guided the development of the TPTP language was that the language should maintain compatibility with Prolog. This compatibility makes it easy to quickly develop ATP software for the TPTP world, e.g., by developers prototyping ATP systems, e.g., [21], or by users writing tools to process their TPTP format data, e.g., [10]. Later, with the specification of the TPTP language in an extended BNF [51], it became a requirement that

the BNF should be easy to translate into valid lex/yacc/flex/bison input, so that construction of parsers can be a reasonably easy task. The same TPTP language is used both for writing problems and for writing solutions (proofs, models, etc.). This makes it easy to connect TPTP-compliant ATP systems and tools, and enables convenient communication between researchers.

6 Other Parts of the TPTP World

While the TPTP problem library is at the core of the TPTP world, there are several other projects that make important contributions to the infrastructure. Several of these are described in detail in [34], and reviewed here:

The Thousands of Solutions from Theorem Provers (TSTP) solution library [32], is a library of ATP systems' solutions to TPTP problems. The solutions are not validated, but are checked against the SZS status of the problems to establish a high level of confidence in their correctness. A major use of the TSTP is by ATP system developers, who examine solutions to understand how problems can be solved, leading to improvements in their own systems. The TSTP also provides the performance data used to compute the TPTP problems' difficulty ratings. At the time of writing the TSTP contained the performance data from 71 ATP systems, for a total of 668679 runs on TPTP v6.4.0 (systems are run on only those SPCs that they can attempt in principle). 262830 (39%) of the runs solved the problem, and 175680 (26%, which is 69% of those solved) of those include a proof or model output.

The TPTP utilities and tools provide facilities for selecting and processing TPTP format problems and solutions. TPTP2T is used to select problems with specified features and solutions. TPTP2X and TPTP4X are used to generate, transform, analyze, and reformat TPTP format files. GDV [31] is able to verify TPTP format derivations. IDV [47] provides graphical rendering and analysis of TPTP format derivations.

The system and tool execution harness SystemOnTPTP allows a TPTP format problem or solution to be easily and quickly submitted to ATP systems and tools. The utility uses a suite of currently available ATP systems and tools, whose properties (input format, command line, reporting of result status, etc.) are stored in a simple text database. SystemOnTPTP is most commonly accessed online (http://www.tptp.org/cgi-bin/SystemOnTPTP), but is also available as a standalone utility.

The CADE ATP System Competition (CASC) [42] is held annually at each CADE (or IJCAR, of which CADE is a constituent) conference. CASC evaluates the performance of sound, fully automatic ATP systems - it is the world championship for such (TPTP-compatible) systems. The design and implementation of CASC is closely linked to the TPTP: the divisions and problem categories of CASC are based on the TPTP's SPCs, the problems used in CASC are taken from the TPTP problem library, and the problem difficulty ratings are used to select appropriately difficult problems.

Two new components of the TPTP infrastructure have been released in the last few years:

The TPTP Process Instruction (TPI) language provides command and control instructions for manipulating logical formulae in the context of ATP systems. The TPI commands have the same syntactic structure as TPTP annotated formulae, and are designed to be used in problem files to instruct ATP systems how to proceed. Examples of TPI commands are to read a file of formulae, place formulae into groups, activate and deactivate groups, and execute reasoning processes on active formulae either synchronously or asynchronously. The TPI language was used in the highly publicized automatic proof of Gödel's ontological argument [5].

The Thousands of Models for Theorem Provers (TMTP) model library [39] is a library of models for axiomatizations built from axiom sets in the TPTP. The TMTP is thus similar to the TSTP - it extends the TSTP by storing multiple models for each axiomatization, but is restricted to contain only models (while the TSTP also contains proofs). The library is supported by tools for efficiently interpreting ground terms and closed formulae, interfaces for visualizing interpretations, a GDV-like utility for verifying models, etc. The TMTP supports the development of semantically guided ATP systems, provides models that can be used to guide axioms selection in large theories, and provides insights into the semantic structure of axiomatizations.

One new TPTP-based project is under development:

The Thousands of Problems for Artificially Intelligent Theorem Provers (TPAP) library is a library of problems, solutions, and structural data that supports the development of ATP systems that use artificial intelligence techniques to improve their performance. A key idea is to use machine learning on existing proofs and structural data (e.g., the order in which the problems naturally occur in the underlying theory) to improve axiom selection in large theories (e.g., [6], and to tune the search parameters of ATP systems (e.g., [16]). TPAP adopts problem representation and organization from the TPTP, solution representation and organization from the TSTP, and adds structural information about the problems, the solutions, and their underlying theories.

7 Conclusion

The TPTP problem library provides a basis for meaningful empirical evaluation of classical logic ATP systems, and provides a support infrastructure that enables developers and users to take advantage of the TPTP problems and their solutions. The TPTP has had significant effects on ATP research. The original goals for building the TPTP have been largely met - performance results based on the TPTP accurately reflect capabilities of the ATP systems being considered. The common TPTP format for data, and the SZS ontologies, have become standards for communication between components of complex reasoning systems, e.g., [16, 22,49,50]

The next development in the TPTP is the addition of problems in polymorphic Typed Higher-order Form (TH1) [15], which will be introduced in TPTP v7.0.0. TH1 is an extension of TH0 with TF1-style rank-1 polymorphism. Many existing ATP and ITP systems for higher-order logic already implement some kind of polymorphism, and TH1 has been designed in collaboration with those systems' developers. TH1 opens the door to useful middleware, such as monomorphizers and other translation tools that encode polymorphism in FOF or TH0, and can provide an inter-operability layer for ATP systems. The hope is that TH1 will be implemented in many popular ATP systems for typed higher-order logic.

A future development is to extend the TPTP language to support non-classical logics, e.g., quantified multi-modal logic, building on ideas proposed in [53]. A logic will be specified in a manner similar to the theories of SMT-LIB [1], leveraging TH1 polymorphism. The representation will provide support for automated reasoning in these logics, including embedding approaches [3].

The keys to sustaining the value of the TPTP in the future are its continued growth, and the automated reasoning community's continued adoption of the TPTP language, standards, and tools for building automated reasoning software. ATP system developers and users are encouraged to build TPTP-compliant ATP software, write ATP problems using the TPTP language, and to contribute ATP problems to the TPTP.

The TPTP is currently hosted in the Department of Computer Science at the University of Miami, on hardware funded by an NSF grant.[2] Some TPTP users have generously donated to the TPTP project, which provides some further support. A robust source of funding would also help keep the project alive in the future. In the long term, there is a need for the TPTP to be adopted, either by another individual, or by a community group.

Acknowledgements Many people have contributed to this work. Most salient are: Christian Suttner, the codeveloper of the TPTP library and CASC; Stephan Schulz and Koen Claessen who influenced the development of the TPTP language; Allen Van Gelder who wrote the core of the language BNF; Jasmin Blanchette, Andrei Paskevich, and Christoph Benzmüller who contributed significantly to the higher order and polymorphic parts of the TPTP; Andrei Voronkov for useful ideas and lots of support; and the automated reasoning community for contributing problems, writing ATP systems, and using the TPTP world.

References

1. Barrett, C., Stump, A., Tinelli, C.: The SMT-LIB standard: Version 2.0. In: Gupta, A., Kroening, D. (eds.) Proceedings of the 8th International Workshop on Satisfiability Modulo Theories (2010)
2. Baumgartner, P.: SMTtoTPTP-A converter for theorem proving formats. In: Felty, A., Middeldorp, A. (eds.) Proceedings of the 25th International Conference on Automated Deduction, number 9195 in Lecture Notes in Computer Science, pp. 285–294. Springer (2015)
3. Benzmüller, C., Paulson, L.: Quantified multimodal logics in simple type theory. Log. Univ. **7**(1), 7–20 (2013)
4. Benzmüller, C., Sorge, V., Jamnik, M., Kerber, M.: Combined reasoning by automated cooperation. J. Appl. Log. **6**(3), 318–342 (2008)
5. Benzmüller, C., Paleo, B.W.: Automating Gödel's ontological proof of God's existence with higher-order automated theorem provers. In: Schaub, T. (ed.) Proceedings of the 21st European Conference on Artificial Intelligence, pp. 93–98 (2014)
6. Blanchette, J., Greenaway, D., Kaliszyk, C., Kühlwein, D., Urban, J.: A learning-based fact selector for Isabelle/HOL. J. Autom. Reason. **57**(3), 219–244 (2016)
7. Blanchette, J., Paskevich, A.: TFF1: The TPTP typed first-order form with rank-1 polymorphism. In: Bonacina, M.P. (ed.) Proceedings of the 24th International Conference on Automated Deduction, number 7898 in Lecture Notes in Artificial Intelligence, pp. 414–420. Springer (2013)
8. Böhme, S., Nipkow, T.: Sledgehammer: judgement day. In: Giesl, J., Haehnle, R. (ed) Proceedings of the 5th International Joint Conference on Automated Reasoning, number 6173 in Lecture Notes in Artificial Intelligence, pp. 107–121 (2010)
9. Comaromi, J.P., Beall, J., Matthews, W.E., New, G.R.: Dewey Decimal Classification and Relative Index, 20th edn. Forest Press, Cinderford (1989)
10. Denney, E., Fischer, B., Schumann, J.: Using automated theorem provers to certify auto-generated aerospace software. In: Rusinowitch, M., Basin, D. (eds.) Proceedings of the 2nd International Joint Conference on Automated Reasoning, number 3097 in Lecture Notes in Artificial Intelligence, pp. 198–212 (2004)

[2] NSF CI-EN grant 1405674: SystemOnTPTP - Online Services for Automated Theorem Proving in Classical Logic.

11. Emmer, M., Khasidashvili, Z., Korovin, K., Voronkov, A.: Encoding industrial hardware verification problems into effectively propositional logic. In: Bloem, R., Sharygina, N. (eds.) Proceedings of the 10th International Conference on Formal Methods in Computer-Aided Design, pp. 137–144. IEEE Press (2010)
12. Gent, I., Walsh, T.: CSPLib: a benchmark library for constraints. In: Jaffar, J. (ed.), Proceedings of the 5th International Conference on the Principles and Practice of Constraint Programming, number 1713 in Lecture Notes in Computer Science, pp. 480–481. Springer (1999)
13. Höfner, P., Struth, G.: Automated reasoning in Kleene Algebra. In: Pfenning, F. (ed.) Proceedings of the 21st International Conference on Automated Deduction, number 4603 in Lecture Notes in Artificial Intelligence, pp. 279–294. Springer (2007)
14. Hoos, H., Stützle, T.: SATLIB: an online resource for research on SAT. In: Gent, I., van Maaren, H., Walsh, T. (eds.) Proceedings of the 3rd Workshop on the Satisfiability Problem, pp. 283–292. IOS Press (2000)
15. Kaliszyk, C., Sutcliffe, G., Rabe, F.: TH1: The TPTP typed higher-order form with rank-1 polymorphism. In: Fontaine, P., Schulz, S., Urban, J. (eds.) Proceedings of the 5th Workshop on the Practical Aspects of Automated Reasoning, number 1635 in CEUR Workshop Proceedings, pp. 41–55 (2016)
16. Kaliszyk, C., Urban, J.: Learning-assisted automated reasoning with Flyspeck. J. Autom. Reason. **53**(2), 173–213 (2014)
17. Kotelnikov, E., Kovacs, L., Voronkov, A.: A first class boolean sort in first-order theorem proving and TPTP. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V. (eds.) Proceedings of the International Conference on Intelligent Computer Mathematics, number 9150 in Lecture Notes in Computer Science, pp. 71–86. Springer (2015)
18. Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J.: An introduction to the syntax and content of Cyc. In: Baral, C. (ed.) Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, pp. 44–49 (2006)
19. McCune, W.W.: Otter 3.3 Reference Manual. Technical Report ANL/MSC-TM-263, Argonne National Laboratory, Argonne, USA (2003)
20. Narizzano, M., Pulina, L., Tacchella, A.: The QBFEVAL Web Portal. In: Fischer, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) Proceedings of the 10th European Conference on Logics in Artificial Intelligence, pp. 494–497 (2006)
21. Otten, J.: leanCoP 2.0 and ileancop 1.2: high performance lean theorem proving in classical and intuitionistic logic. In: Baumgartner, P., Armando, A., Dowek, G. (eds.) Proceedings of the 4th International Joint Conference on Automated Reasoning, number 5195 in Lecture Notes in Artificial Intelligence, pp. 283–291 (2008)
22. 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, number 2 in EPiC, pp. 1–11 (2010)
23. Pease, A., Sutcliffe, G.: First order reasoning on a large ontology. In: Urban, J., Sutcliffe, G., Schulz, S. (eds.) Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, number 257 in CEUR Workshop Proceedings, pp. 61–70 (2007)
24. Pelletier, F.J.: Seventy-five problems for testing automatic theorem provers. J. Autom. Reason. **2**(2), 191–216 (1986)
25. Phillips, J.D., Stanovsky, D.: Automated theorem proving in loop theory. In: Sutcliffe, G., Colton, S., Schulz, S. (eds.) Proceedings of the CICM Workshop on Empirically Successful Automated Reasoning in Mathematics, number 378 in CEUR Workshop Proceedings, pp. 42–53 (2008)
26. Plaisted, D.A.: Non-Horn clause logic programming without contrapositives. J. Autom. Reason. **4**(3), 287–325 (1988)
27. Puzis, Y., Gao, Y., Sutcliffe, G.: Automated generation of interesting theorems. In: Sutcliffe, G., Goebel, R. (eds.) Proceedings of the 19th International FLAIRS Conference, pp. 49–54. AAAI Press (2006)
28. Quaife, A.: Automated deduction in von Neumann-Bernays-Godel set theory. J. Autom. Reason. **8**(1), 91–147 (1992)
29. Raths, T., Otten, J., Kreitz, C.: The ILTP problem library for intuitionistic logic—release v1.1. J. Autom. Reason. **38**(1–2), 261–271 (2007)
30. American Mathematical Society. Mathematical Subject Classification. American Mathematical Society (1992)
31. Sutcliffe, G.: Semantic derivation verification: techniques and implementation. Int. J. Artif. Intell. Tools **15**(6), 1053–1070 (2006)

32. Sutcliffe, G.: TPTP, TSTP, CASC, etc. In: Diekert, V., Volkov, M., Voronkov, A. (eds.) Proceedings of the 2nd International Symposium on Computer Science in Russia, number 4649 in Lecture Notes in Computer Science, pp. 6–22. Springer (2007)
33. Sutcliffe, G.: The SZS ontologies for automated reasoning software. In: Sutcliffe, G., Rudnicki, P., Schmidt, R., Konev, B., Schulz, S. (eds.) 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, pp. 38–49 (2008)
34. 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)
35. Sutcliffe, G.: The TPTP world - infrastructure for automated reasoning. In: Clarke, E., Voronkov, A. (eds.) Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, number 6355 in Lecture Notes in Artificial Intelligence, pp. 1–12. Springer (2010)
36. Sutcliffe, G., Benzmüller, C.: Automated reasoning in higher-order logic using the TPTP THF infrastructure. J. Formaliz. Reason. **3**(1), 1–27 (2010)
37. 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)
38. Sutcliffe, G., Pelletier, F.J.: Hoping for the truth—a survey of the TPTP logics. In: Markov, Z., Russell, I. (eds.) Proceedings of the 29th International FLAIRS Conference, pp. 110–115 (2016)
39. Sutcliffe, G., Schulz, S.: The thousands of models for theorem provers (TMTP) model library - first steps. In: Konev, B., Schulz, S., Simon, L. (eds.) Proceedings of the 11th International Workshop on the Implementation of Logics, number 40 in EPiC Series in Computing, pp. 106–121. EasyChair Publications (2016)
40. Sutcliffe, G., Schulz, S., Claessen, K., Baumgartner, P.: The TPTP typed first-order form with arithmetic. In: Bjørner, N., Voronkov, A. (eds.) Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, number 7180 in Lecture Notes in Artificial Intelligence, pp. 406–419. Springer (2012)
41. Sutcliffe, G., Schulz, S., Claessen, K., Van Gelder, A.: Using the TPTP language for writing derivations and finite interpretations. In: Furbach, U., Shankar, N. (eds.) Proceedings of the 3rd International Joint Conference on Automated Reasoning, number 4130 in Lecture Notes in Artificial Intelligence, pp. 67–81 (2006)
42. Sutcliffe, G., Suttner, C.: The state of CASC. AI Commun. **19**(1), 35–48 (2006)
43. Sutcliffe, G., Suttner, C., Pelletier, F.J.: The IJCAR ATP system competition. J. Autom. Reason. **28**(3), 307–320 (2002)
44. Sutcliffe, G., Suttner, C.B.: The TPTP problem library: CNF release v1.2.1. J. Autom. Reason. **21**(2), 177–203 (1998)
45. Sutcliffe, G., Suttner, C.B.: Evaluating general purpose automated theorem proving systems. Artif. Intell. **131**(1–2), 39–54 (2001)
46. 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, 18th International Joint Conference on Artificial Intelligence, pp. 52–57 (2003)
47. Trac, S., Puzis, Y., Sutcliffe, G.: An interactive derivation viewer. In: Autexier, S., Benzmüller, C. (eds.) Proceedings of the 7th Workshop on User Interfaces for Theorem Provers, 3rd International Joint Conference on Automated Reasoning, volume 174 of Electronic Notes in Theoretical Computer Science, pp. 109–123 (2007)
48. Urban, J.: MPTP 0.2: design, implementation, and initial experiments. J. Autom. Reason. **37**(1–2), 21–43 (2006)
49. Urban, J., Rudnicki, P., Sutcliffe, G.: ATP and presentation service for Mizar formalizations. J. Autom. Reason. **50**(2), 229–241 (2013)
50. Urban, J., Sutcliffe, G., Pudlak, P., Vyskocil, J.: MaLARea SG1: machine learner for automated reasoning with semantic guidance. In: Baumgartner, P., Armando, A., Dowek, G. (eds.) Proceedings of the 4th International Joint Conference on Automated Reasoning, number 5195 in Lecture Notes in Artificial Intelligence, pp. 441–456. Springer (2008)
51. Van Gelder, A., Sutcliffe, G.: Extending the tptp language to higher-order logic with automated parser generation. In: Furbach, U., Shankar, N. (eds.) Proceedings of the 3rd International Joint Conference on Automated Reasoning, number 4130 in Lecture Notes in Artificial Intelligence, pp. 156–161. Springer (2006)
52. Verchinine, K., Lyaletski, A., Paskevich, A.: System for automated deduction (SAD): a tool for proof verification. In: Pfenning, F. (ed.) Proceedings of the 21st International Conference on Automated Deduction, number 4603 in Lecture Notes in Artificial Intelligence, pp. 398–403. Springer (2007)

53. Wisniewski, M., Steen, A., Benzmüller, C.: TPTP and beyond: representation of quantified non-classical logics. In: Benzmüller, C., Otten, J. (eds.) Proceedings of the 2nd International Workshop on Automated Reasoning in Quantified Non-Classical Logics, number 1770 in CEUR Workshop Proceedings, pp. 51–65 (2016)
54. Wos, L.A., Overbeek, R.A., McCharen, J.D.: Problems and experiments for and with automated theorem-proving programs. IEEE Trans. Comput. **C–25**(8), 773–782 (1976)