The TPTP Problem Library

Geoff Sutcliffe[1], Christian Suttner[2], Theodor Yemenis[2]

[1] Dep't of Computer Science, James Cook University, Townsville, Australia
geoff@cs.jcu.edu.au, +61 77 815085
[2] Institut für Informatik, Technische Universität München, Germany
{suttner,gemenis}@informatik.tu-muenchen.de, +49 89 521098

Abstract. This paper provides a description of the TPTP library of problems for automated theorem provers. The library is available via Internet, and is intended to form a common basis for the development of and experimentation with automated theorem provers. To support this goal, this paper provides:

- the motivations for building the library;
- a description of the library structure including overview information;
- a description of the tptp2X utility program;
- guidelines for obtaining and using the library.

1 Introduction

This paper describes the TPTP (Thousands of Problems for Theorem Provers) Problem Library. The TPTP is a library of problems for automated theorem proving (ATP) systems, for 1st order logic. The TPTP is comprehensive, and thus provides an overview, and a simple, unambiguous reference mechanism for ATP problems. All problems in the TPTP are presented in an unambiguous format, and automatic conversion to other known ATP formats is provided.

The principal motivation for this project is to move the testing and evaluation of ATP systems from the present ad hoc situation onto a firm footing. This is necessary, as results currently being published seldom provide an accurate reflection of the intrinsic power of the ATP system being considered. A common library of problems is necessary for meaningful system evaluations, meaningful system comparisons, repeatability of testing, and the production of statistically significant results. Guidelines for using TPTP problems and presenting results are given in Section 4.2.

1.1 State of the Art

A large number of interesting problems have accumulated over the years in the ATP community. Besides publishing particularly interesting individual problems, from early on researchers have collected problems in order to obtain a basis for experimentation. The first major publication[3] in this regard was [MOW76],

[3] The first circulation of problems for testing theorem provers to our knowledge is due to L. Wos in the late sixties.

which provides an explicit listing of clauses for 63 problems, many of which are still relevant today. In the same year Wilson and Minker [WM76] tested six resolution strategies on a collection of 86 named problems. The problem clauses are not supplied in [WM76], however. A second major thrust was provided by [Pel86], which lists 75 problems. Other more recent collections are [BLM$^+$86], [Qua92a], [MW92], and [McC93], to name a few. Also, the Journal of Automated Reasoning's Problem Corner regularly provides interesting challenge problems. However, problems published in hardcopy form are often not suitable for testing ATP systems, because they have to be transcribed to electronic form. This is a cumbersome, error-prone process, and is feasible for only very small numbers of problems. A problem library in electronic form was made publicly available by Argonne National Laboratories (Otter format, [McC90]) in 1988 [ANL]. This library has been a major source of problems for ATP researchers. Other electronic collections of problems are available, but have not been announced officially (e.g., that distributed with the SPRFN ATP system [SPR]). Although some of these collections provide significant support to researchers, and formed the early core of the TPTP library, none (with the possible exception of the ANL library) was specifically designed to serve as a common basis for ATP research. Rather, these collections typically were built in the course of research into a particular ATP system. As a result, there are several factors that limit their usefulness as a common basis for research. In particular, existing problem collections are often:

- hard to discover and obtain,
- limited in scope and size,
- outdated,
- formatted and tuned for a particular ATP system,
- inconsistent in their presentation of equally named problems,
- undocumented,
- provided without a regular update service for new problems,
- provided without a reliable error correction service.

As a consequence, several problems arise. Firstly, system development and system evaluations typically rely on a limited range of problems, depending on the collections of problems available to the researcher. Secondly, the presentation format used in a collection may not be appropriate for the desired purpose, and a comparatively large effort is required just to make the problems locally usable (which in practice often means that such a collection of problems is simply ignored). Thirdly, using a particular collection may lead to biases in results, because the problems have been designed and tuned specifically for a particular ATP system. Fourthly, the significance and difficulty of a problem, with respect to the current state-of-the-art in ATP systems, is hard to assess by newcomers to the field. Existing test problems are often not adequate anymore (e.g., Schubert's Steamroller [Sti86]), while others may be solvable only with specialized techniques (e.g., LIM+ [Ble90]) and therefore are much too hard to start with. Finally, many copies and variants of the same "original" problem may exist in different libraries. Coupled with a lack of documentation, this means that unambiguous identification of problems, and therefore a clear interpretation of performance figures for given problems, has become difficult.

The problem of meaningfully interpreting results is even worse than indicated. Commonly a few problems are selected and hand-tuned (clauses and literals are arranged in a special order, irrelevant clauses are omitted, lemmas are added in, etc) specifically for the ATP system being tested. However, the presentation of a problem can significantly affect the nature of the problem, and changing the clauses clearly makes a different problem altogether. Nevertheless the problem may be referenced under the same name as it was presented elsewhere. As a consequence the experimental results reveal little. Some researchers avoid this ambiguity by listing the clause sets explicitly. But obviously this usually cannot be done for a large number of problems or for large individual problems. The only satisfactory solution to these issues is a common and stable library of problems, which the TPTP provides.

1.2 What is Required?

The goal for building the TPTP has been to overcome the current drawbacks, and to centralize the burden of problem collection and maintenance to one place. The TPTP tries to address all relevant issues. In particular, the TPTP

- is available by anonymous ftp.
  The TPTP is thus easily available to the research community. Awareness of the TPTP is assured by extensive formal and informal announcements.
- spans a diversity of subject matters.
  This reduces biases in the development and testing of theorem provers, which arise from the use of a limited range of problems. It also provides an overview of the domains that ATP is used in. The significance of each domain may be measured by the number of problems available.
- is large enough for statistically significant testing.
  In contrast to common practise, an ATP system should be evaluated over a large number of problems, rather than a small set of judiciously selected examples. The large size of the TPTP makes this possible.
- is comprehensive and up-to-date.
  The TPTP contains all problems known to the community. There is no longer a need to look elsewhere.
- is presented in a well structured and documented format.
  All problems in the TPTP are presented in an unambiguous format. Automatic conversion to other known formats is provided, thus eliminating the necessity for transcription. The design and arrangement of the problems is independent of any particular ATP system.
- provides a common, independent, source for problems.
  This provides unambiguous problem reference, and makes the comparison of results meaningful.
- contains problems varying in difficulty, from very simple problems through to open problems.

This allows all interested researchers, from newcomers to experts, to rely on the same problem library.
- provides statistics for each problem and for the library as a whole.
  This gives information about the syntactic nature of the problems.
- will provide a rating for the difficulty of each problem.
  This is important for several reasons: (1) It simplifies problem selection according to the user's intention. (2) It allows the quality of an ATP system to be judged. (3) Over the years, changes in the problem ratings will provide an indicator of the advancement in ATP. The problem ratings are currently being worked on, and will be part of a future TPTP release.
- documents each problem.
  This contributes to the unambiguous identification of each problem.
- provides standard axiomatizations that can be used in new problems.
  This simplifies the construction of new problems.
- is a channel for making new problems available to the community, in a simple and effective way.
- removes the necessity for the duplicated effort of maintaining many libraries.
- specifies guidelines for its use for evaluating ATP systems.
  As the TPTP is a standard library of problems, it is possible to set ATP evaluation guidelines which make reported results meaningful. This will in turn simplify and improve system comparisons, and allow ATP researchers to accurately gauge their progress.

The TPTP problem library is an ongoing project, with the aim to provide all of the desired properties.

Current Limitations of the TPTP. The current version of the TPTP library is limited to problems expressed in 1st order logic, presented in clause normal form. In particular, there are no problems for induction and nonclassical theorem proving. However, see Section 5 for upcoming and planned extensions.

2 Inside the TPTP

Scope. Release v1.0.0 of the TPTP contains 1577 abstract problems, which result in 2295 ATP problems, due to alternative presentations (see Section 2.2). Tables 1, 2, and 3 provide some statistics about release v1.0.0 of the TPTP.

The problems have been collected from various sources. The two principal sources have been existing problem collections and the ATP literature. Other sources include logic programming, mathematics, puzzles, and correspondence with ATP researchers. Many people and organizations have contributed towards the TPTP. In particular, the foundations of the TPTP were laid with David Plaisted's SPRFN collection [SPR]; many problems have been taken from Argonne National Laboratory's ATP problem library [ANL] (special thanks to Bill McCune here); Art Quaife has provided several hundred problems in set theory and algebra [Qua92b]; the Journal of Automated Reasoning, CADE Proceedings,

Table 1. Statistics on the TPTP.

Number of problem domains	23	
Number of abstract problems	1577	
Number of problems	2295	
Number of non-Horn problems	1449	(63%)
Number of problems with equality	1805	(79%)
Number of propositional problems	111	(5%)
... being non-Horn	56	(50%)
Total number of clauses	322156	
Total number of literals	970456	

Table 2. Statistics for non-propositional TPTP problems.

Measure	Minimum	Maximum	Average	Median
Number of clauses	2	6404	124	68
Percentage of non-Horn clauses	0%	99%	6%	4%
... in non-Horn problems	0%	99%	9%	4%
Percentage of unit clauses	0%	100%	32%	22%
Number of literals	2	18966	276	162
Percentage of equality literals	0%	100%	47%	47%
... in equality problems	4%	100%	57%	47%
Number of predicate symbols	1	4042	12	4
Number of function symbols	0	94	27	13
Percentage of constants	0%	100%	46%	33%
Number of variables	0	32000	311	271
Percentage of singletons	0%	100%	6%	7%
Maximal clause size	1	25	4	5
Maximal term depth	0	51	4	6

Table 3. Statistics for propositional TPTP problems.

Measure	Minimum	Maximum	Average	Median
Number of clauses	2	8192	444	36
Percentage of non-Horn clauses	0%	99%	20%	1%
... in non-Horn problems	1%	99%	40%	45%
Percentage of unit clauses	0%	100%	16%	1%
Number of literals	2	106496	3303	132
Number of predicate symbols	1	840	69	13
Maximal clause size	1	21	5	3

and Association for Automated Reasoning Newsletters have provided a wealth of material; smaller numbers of problems have been provided by a number of further contributors (see also the Acknowledgements at the end of Section 5).

The problems in the TPTP are syntactically diverse, as is indicated by the ranges of the values in Tables 2 and 3. The problems in the TPTP are also semantically diverse, as is indicated by the range of domains that are covered. The problems are grouped into 23 domains, covering topics in the fields of logic, mathematics, computer science, and more. The domains are presented and discussed in Section 2.1.

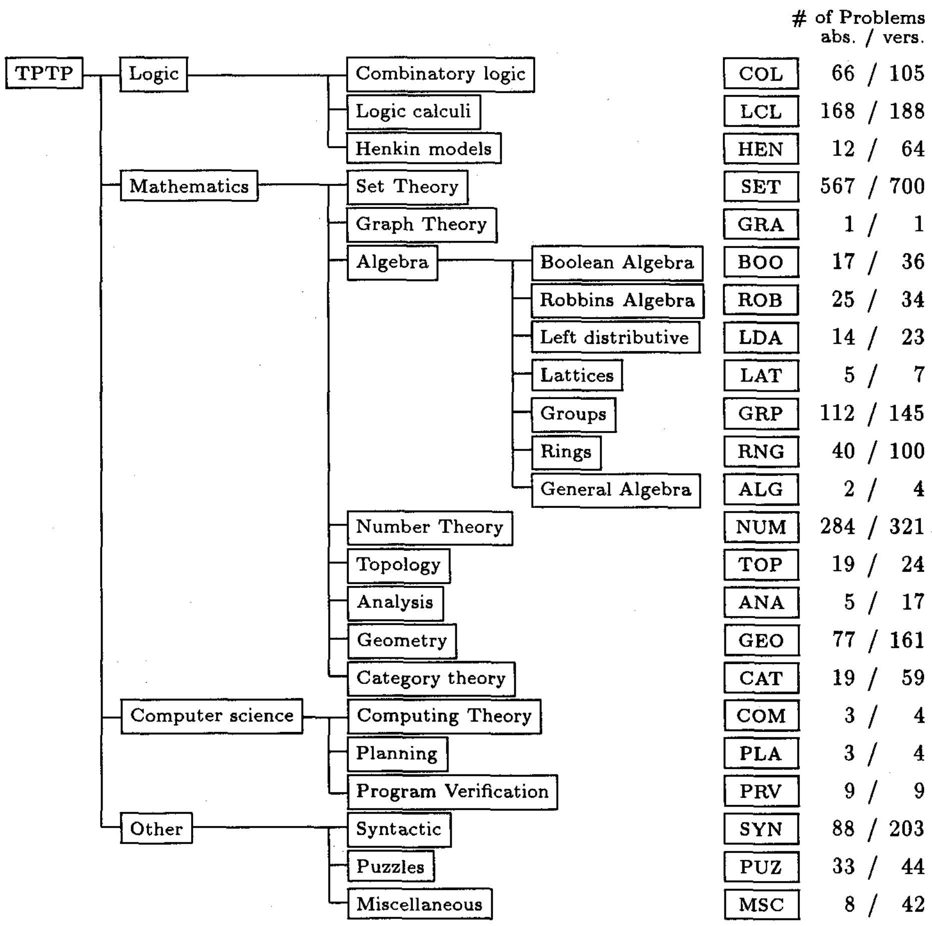
Releases. The TPTP is managed in the manner of a software product, in the sense that fixed releases are made. Each release of the TPTP is identified by a release number, in the form v<Version>.<Edition>.<Patchlevel>. The Version number enumerates major new releases of the TPTP, in which important new features have been added. The Edition number 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. All changes are recorded in a history file, as well as in the file for an affected problem.

2.1 The TPTP Domain Structure

An attempt has been made to classify the totality of the TPTP problems in a systematic and natural way. The resulting domain scheme reflects the natural hierarchy of scientific domains, as presented in standard subject classification literature. The current classification is based mainly on the Dewey Decimal Classification [Dew89] and the Mathematics Subject Classification used for the Mathematical Reviews by the American Mathematical Society [MSC92]. We define four main fields: logic, mathematics, computer science, and other. Each field contains further subdivisions, called *domains.* These TPTP domains constitute the basic units of our classification. The full classification scheme is shown in Figure 1.

2.2 Problem Versions and Standard Axiomatizations.

There are often many ways to formulate a problem for presentation to an ATP system. Thus, in the TPTP, there are often alternative presentations of a problem. The alternative presentations are called *versions* of the underlying *abstract problem.* As the problem versions are the objects that ATP systems must deal with, they are referred to simply as problems, and the abstract problems are referred to explicitly. Each problem is stored in a separate physical file. The primary reason for different versions of an abstract problem is the use of different axiomatizations. This issue is discussed below. A secondary reason is the use of different formulations of the theorem to be proven.



		# of Problems abs. / vers.
Combinatory logic	COL	66 / 105
Logic calculi	LCL	168 / 188
Henkin models	HEN	12 / 64
Set Theory	SET	567 / 700
Graph Theory	GRA	1 / 1
Boolean Algebra	BOO	17 / 36
Robbins Algebra	ROB	25 / 34
Left distributive	LDA	14 / 23
Lattices	LAT	5 / 7
Groups	GRP	112 / 145
Rings	RNG	40 / 100
General Algebra	ALG	2 / 4
Number Theory	NUM	284 / 321
Topology	TOP	19 / 24
Analysis	ANA	5 / 17
Geometry	GEO	77 / 161
Category theory	CAT	19 / 59
Computing Theory	COM	3 / 4
Planning	PLA	3 / 4
Program Verification	PRV	9 / 9
Syntactic	SYN	88 / 203
Puzzles	PUZ	33 / 44
Miscellaneous	MSC	8 / 42

Fig. 1. The domain structure of the TPTP.

Different axiomatizations. Commonly, different axiomatizations of a theory exist, and, in general, most are equally acceptable. In the TPTP an axiomatization is acceptable if it is complete (in the sense that it captures some closed theory) and it has not had any lemmas added. Such axiomatizations are called *standard* axiomatizations. Within the ATP community some problems have been created with *non-standard* axiomatizations. An axiomatization is non-standard if it has been *reduced* (i.e., axioms have been removed) and the result is an *incomplete* axiomatization, or if it has been *augmented* (i.e., lemmas have been added) and the result is a *redundant* axiomatization. Non-standard axiomatizations are typically used to find a proof of a theorem (based on the axiomatization) using a particular ATP system. In any 'real' application of an ATP system, a standard axiomatization of the application domain would typically have to be used, at least initially. Thus the use of standard axiomatizations is desirable, because it reflects such 'real' usage. In the TPTP, for each collected problem that uses a non-standard axiomatization, a new version of the problem is created with a standard axiomatization. The standard axiomatization is created by reducing or augmenting the non-standard axiomatization appropriately.

The standard axiomatizations used in the TPTP are kept in separate axiom files, and are included in problems as appropriate. If an axiomatization uses equality, the required axioms of substitution are kept separate from the theory specific axioms. The equality axioms of reflexivity, symmetry, and transitivity, which are also required when equality is present, are also kept separately, as they are often used independently of the substitution axioms.

2.3 Problem and Axiomatization Naming

Providing unambiguous names for all problems is necessary in a problem library. A naming scheme has been developed for the TPTP, to provide unique, stable names for abstract problems, problem versions, and axiomatizations. File names are in the form DDDNNN-V[MMM]`.p` for problem files, and DDDNNN-E.TT for axiom files. DDD is the domain name abbreviation, NNN is the index within the domain, V is the version number, MMM is an optional generation parameter, E is the axiomatization extension number, `.p` indicates a problem file, and TT is either `.ax` for theory specific axioms or `.eq` for equality substitution axioms. The complete file names are unique within the TPTP.

Abstract problems and axiomatizations have also been allocated semantic names. The semantic names can be used to augment file names, so as to give an indication of the contents. While these names are provided for users who like to work with mnemonic names, only the standard syntactic names are guaranteed to provide unambiguous reference. The semantic names are formed from a set of specified abbreviations. The semantic names can be added to the syntactic file names using a script that is provided.

2.4 Problem Presentation

The physical presentation of the TPTP problem library is such that ATP researchers can easily use the problems. The TPTP file format, for both problem files and axiom files, has three main sections. The first section is a header section that contains information for the user. The second section contains `include` instructions for axiom files. The last section contains the clauses that are specific to the problem.

The syntax of the problem and axiom files is that of Prolog. This conformance makes it trivial to manipulate the files using Prolog. All information in the files that is not for use by ATP systems is formatted as Prolog comments, with a leading `%`. All the information for ATP systems is formatted as Prolog facts. A utility is provided for converting TPTP files to other known ATP system formats (see Section 3). A description of the information contained in TPTP files is given below. Full details are given in [SSY93].

The Header Section

The header contains information about the problem, for the user. It is divided into four parts. The first part identifies and describes the problem. The second part provides information about occurrences of the problem. The third part gives the problem's ATP status and a table of syntactic measurements made on the problem. The last part contains general information about the problem. An example of a TPTP header, extracted from the problem file GRP039-7.p, is shown in Figure 2.

```
%--------------------------------------------------------------------------
% File     : GRP039=SubGI2Norm-7 : Released v0.0.0, Updated v0.11.5.
% Domain   : Group Theory (Subgroups)
% Problem  : Subgroups of index 2 are normal
% Version  : [McCharen, et al., 1976] (equality) axioms : Augmented.
% English  : If O is a subgroup of G and there are exactly 2 cosets in
%            G/O, then O is normal [that is, for all x in G and y in O,
%            x*y*inverse(x) is back in O].

% Refs     : McCharen J.D., Overbeek R.A., Wos L.A. (1976), Problems and
%            Experiments for and with Automated Theorem Proving Programs
%            IEEE Transactions on Computers C-25(8), 773-782.
% Source   : [McCharen, et al., 1976]
% Names    : GP2 [McCharen, et al., 1976]

% Status   : unsatisfiable
% Syntax   : Number of clauses          :  29 (  2 non-Horn)(17 units)
%            Number of literals         :  47 ( 32 equality)
%            Number of predicate symbols :  2 (  0 propositions)
%            Number of function symbols :   8 (  5 constants)
%            Number of variables        :  42 (  0 singletons)
%            Maximal clause size        :   4
%            Maximal term depth         :   3

% Comments : element_in_O2(A,B) is A^-1.B. The axioms with element_in_O2
%            force index 2.
% Bugfixes : v0.8.1 - Clauses multiply_inverse_* and multiply_identity_*
%            fixed.
%--------------------------------------------------------------------------
```

Fig. 2. Example of a problem file header (GRP039-7.p).

The header fields contain the following information:

- The % File field contains the problem's syntactic and semantic names, the TPTP release in which the problem first appeared, and the TPTP release in which the problem was last updated.

- The `% Domain` field identifies the domain from which the problem is drawn.
- The `% Problem` field provides a one line, high-level description of the abstract problem.
- The `% Version` field gives information that differentiates this version of the problem from other versions of the problem.
- The `% English` field provides a full description of the problem if the one line description is too terse.
- The `% Refs` field provides a list of references to items in which the problem has been presented.
- The `% Source` field acknowledges, with a citation, where the problem was (physically) obtained from.
- The `% Names` field lists existing known names for the problem (as it has been named in other problem collections or the literature).
- The `% Status` field gives the problem's ATP status [4], one of `satisfiable`, `unsatisfiable`, `open`, or `broken`.
- The `% Syntax` field lists various syntactic measures of the problem's clauses.
- The `% Comments` field contains free format comments about the problem.
- The `% Bugfixes` field documents any changes which have been made to the clauses of the problem.

The Include Section

The include section contains `include` instructions for TPTP axiom files. An example of an include section, extracted from the problem file `GRP039-7.p`, is shown in Figure 3.

```
%--------------------------------------------------------------------------
%----Include the axioms of equality
include('Axioms/EQU001-0.ax').
%----Include the equality axioms for group theory in equality form
include('Axioms/GRP004-0.ax').
%----Include the subgroup axioms in equality formulation
include('Axioms/GRP004-1.ax').
%--------------------------------------------------------------------------
```

Fig. 3. Example of a problem file include section (`GRP039-7.p`).

Each of the `include` instructions indicates that the clauses in the named axiom file should be included at that point. Axiom files are presented in the same format as problem files, and `include` instructions may also appear in axiom files. Full versions of TPTP problems (without `include` instructions) can be created by using the tptp2X utility (see Section 3).

[4] This field has not yet been filled in all problems of the TPTP.

The Clauses Section

TPTP problems are presented in clausal normal form. The literals that make up a clause are presented as a Prolog list of terms. Each literal is a term whose functor is either ++ or --, indicating a positive or negative literal respectively. The single argument of the sign, i.e., the atom of the literal, is a Prolog term. The signs ++ and -- are assumed to be defined as prefix operators in Prolog.

Each clause has a name, in the form of a Prolog atom. Each clause also has a status, one of `axiom`, `hypothesis`, or `theorem`. The `hypothesis` and `theorem` clauses are those that are derived from the negation of the theorem to be proved. The status `hypothesis` is used only if the clauses can clearly be determined as such; otherwise their status is `theorem`. The name, status, and literal list of each clause are bundled as the three arguments of a Prolog fact, whose predicate symbol is `input_clause`. These facts are in the clauses section of the problem file.

Two examples of clauses, extracted from the problem file `GRP039-1.p`, are shown in Figure 4.

```
%--------------------------------------------------------------------------
%----Definition of subgroup of index 2
input_clause(an_element_in_O2,axiom,
    [++subgroup_member(element_in_O2(A,B)),
     ++subgroup_member(B),
     ++subgroup_member(A)]).

input_clause(prove_d_is_in_subgroup,theorem,
    [--subgroup_member(d)]).
%--------------------------------------------------------------------------
```

Fig. 4. Examples of problem file clauses (GRP039-1.p).

2.5 Physical Organization

The TPTP is physically organized into five subdirectories:

- The `Problems` directory contains a subdirectory for each domain, as shown in Figure 1. Each subdirectory contains the problem files for that domain.
- The `Axioms` directory contains the axiom files.
- The `TPTP2X` directory contains the tptp2X utility, described in Section 3.
- The `Scripts` directory contains some useful C shell scripts.
- The `Documents` directory contains comprehensive online information about the TPTP, in specific files. This provides quick access to relevant overview data. In particular, it provides a simple means for selecting problems with specific properties, by using standard system tools (e.g., `grep`, `awk`).

3 The tptp2X Utility

The tptp2X utility converts TPTP problems from the TPTP format to formats used by existing ATP systems. Currently, the tptp2X utility supports the following output formats:

- the MGTP format [FHKF92];
- the Otter `.in` format [McC90] (the set of support and inference rules to be used are also specified, and are included in the output file);
- the PTTP format [Sti84];
- the SETHEO `.lop` format [STvdK90];
- the SPRFN format [Pla88];
- the TPTP format, substituting `include` instructions with the actual clauses.

It is simple to add new formatting capabilities to the tptp2X utility. The tptp2X utility can also be used to rearrange the clauses and literals in a problem. This facilitates testing the sensitivity of an ATP system to the order in which the clauses and literals are presented. Full details of the tptp2X utiltity are given in [SSY93].

4 You and the TPTP

4.1 How to FTP the TPTP

The TPTP can be obtained by anonymous ftp from the Department of Computer Science, James Cook University, or the Institut für Informatik, Technische Universität München. The ftp hosts are `coral.cs.jcu.edu.au` (137.219.17.4) and `flop.informatik.tu-muenchen.de` (131.159.8.35), respectively. At both sites the TPTP is in the directory `pub/tptp-library`. There are three files. Information about the TPTP is in the ReadMe file (9.3 KByte), a technical report [SSY93] is in TR-v1.0.0.ps.Z (357 KByte), and the library itself is packaged in TPTP-v1.0.0.tar.Z (2.4 MByte). Please read the ReadMe file, as it contains up-to-date information about the TPTP.

4.2 Important: Using the TPTP

By providing this library of ATP problems, and a specification of how these problems should be presented to ATP systems, it is our intention to place the testing, evaluation, and comparison of ATP systems on a firm footing. For this reason, you should abide by the following conditions when using TPTP problems and presenting your results:

- The specific version, edition, and patch level of the TPTP, used as the problem source, must be stated (see the introduction of Section 2).
- Each problem must be referenced by its unambiguous syntactic name.

- No clauses/literals may be added/removed without explicit notice. (This holds also for removing equality axioms when built-in equality is provided by the prover.)
- The clauses/literals may not be rearranged without explicit notice. If clause or literal reversing is done by the tptp2X utility, the reversals must be explicitly noted.
- The header information in each problem may not be used by the ATP system without explicit notice. Any information that is given to the ATP system, other than that in the `input_clauses`, must be explicitly noted (including any system switches or default settings).

Abiding by these rules will allow unambigous identification of the problem, the arrangement of clauses, and further input to the ATP system which has been used.

5 Present and Future Activity

5.1 The Present

Users' experiences with the TPTP will tell if we have achieved the design goals described in Section 1, except for the problem ratings. The problem ratings are currently being worked on. Note that these ratings should decrease as advances in automated theorem proving are made. Therefore the long term maintenance of the individual problem ratings will provide an objective measure of progress in the field.

An upcoming addition to the TPTP is a set of Prolog programs for generating arbitrary sizes of generic problems (e.g., the "pigeon holes" problem). This will implicitly provide any size of such problems, and also reduce the physical size of the TPTP. These programs are being worked on. In order to ensure the usability of the programs, they will be tested on a public domain Prolog.

The tptp2X utility will soon be modified to be capabile of applying arbitrary sequences of transformations to problems. The transformations will include those currently in tptp2X, and others that become available (e.g., Mark Stickel has provided a transformation for his upside-down meta-interpretation approach [Sti94]). Automatic removal of equality axioms will also be provided, to ease the life of researchers whose ATP systems have built in mechanisms for equality reasoning.

5.2 The Future

We have several short and long term plans for further development of the TPTP. The main ideas are listed here.

- Performance tables for popular ATP systems.
  Extensive experimental results, for publically available ATP systems, are currently being collected (based on all the TPTP problems). These will be published in some form.

- ATP system evaluation guidelines.
  General guidelines outlining the requirements for ATP system evaluation will be produced.
- The BSTP Benchmark Suite.
  A benchmark suite (the BSTP) will be selected from the TPTP. The BSTP will be a small collection of problems, and will provide a minimal set of problems on which an ATP system evaluation can be based. The BSTP will be accompanied by specific guidelines for computing a performance index for an ATP system.
- Full 1st order logic.
  The TPTP will be extended to include problems in full 1st order logic notation. ATP systems with automatic conversion to clausal form then can derive additional information regarding the problem, such as which functors are Skolem functors.
- Various translators.
  Translators between various logical forms will be provided.
  - from full 1st order logic to clausal normal form
  - from non-Horn to Horn form
  - from 1st order to propositional form
- Non-classical and higher order logics.
  In the longer term, the TPTP may be extended to include problems expressed in non-classical and higher order logics.

5.3 Acknowledgements

We are indebted to the following people and organizations who have contributed problems and helped with the construction of the TPTP:

Geoff Alexander, the ANL group (especially Bill McCune), the Automated Reasoning group in Munich, Dan Benanav, Woody Bledsoe, Maria Poala Bonacina, Heng Chu, Tom Jech, Reinhold Letz, Thomas Ludwig, Xumin Nie, Jeff Pelletier, David Plaisted, Joachim Posegga, Art Quaife, Alberto Segre, John Slaney, Mark Stickel, Bob Veroff, and TC Wang.

References

[ANL] ANL Problem Library. Available by anonymous ftp from info.msc.anl.gov, Maths and Computer Science Division, Argonne National Laboratory, Argonne, IL.

[Ble90] W.W. Bledsoe. Challenge Problems in Elementary Calculus. *Journal of Automated Reasoning*, 6:341 - 359, 1990.

[BLM$^+$86] R. Boyer, E. Lusk, W. McCune, R. Overbeek, M. Stickel, and L. Wos. Set Theory in First-Order Logic: Clauses for Gödel's Axioms. *Journal of Automated Reasoning*, 2:287 - 327, 1986.

[Dew89] M. Dewey. *Dewey Decimal Classification and Relative Index.* Forest Press, 20 edition, 1989.

[FHKF92] M. Fujita, R. Hasegawa, M. Koshimura, and H. Fujita. Model Generation Theorem Provers on a Parallel Inference Machine. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 357–375, 1992.

[McC90] W.W. McCune. Otter 2.0 Users Guide. Technical Report ANL-90/9, Argonne National Laboratory, 1990.

[McC93] W.W. McCune. Single Axioms for Groups and Abelian Groups with Various Operations. *Journal of Automated Reasoning*, 10(1):1–13, 1993.

[MOW76] J. McCharen, A. Overbeek, and L. Wos. Problems and Experiments for and with Automated Theorem-Proving Programs. *IEEE Transactions on Computers*, C-25(8):773 – 782, August 1976.

[MSC92] *Mathematical Subject Classification.* American Mathematical Society, Provedence, RI, 1992. Mathematical Reviews, Subject Index.

[MW92] W. McCune and L. Wos. Experiments in Automated Deduction with Condensed Detachment. In *Proceedings of CADE-11*, pages 209–223, Saratoga Springs, USA, 1992. Springer LNAI 607.

[Pel86] F.J. Pelletier. Seventy–five Problems for Testing Automated Theorem Provers. *Journal of Automated Reasoning*, 2:191–216, 1986.

[Pla88] D.A. Plaisted. Non-Horn Clause Logic Programming without Contrapositives. *Journal of Automated Reasoning*, 4(3):287–325, 1988.

[Qua92a] A. Quaife. Automated Deduction in von Neumann-Bernays-Godel Set Theory. *Journal of Automated Reasoning*, 8(1):91–147, 1992.

[Qua92b] A. Quaife. Problems based on NBG Set Theory. Emailed from A. Quaife to G. Sutcliffe, 1992.

[SPR] SPRFN. The problem collection distributed with the SPRFN ATP system [Pla88].

[SSY93] C.B. Suttner, G. Sutcliffe, and T. Yemenis. The TPTP Problem Library (TPTP v1.0.0). Technical Report FKI-184-93, Institut für Informatik, Technische Universität München, Munich, Germany; Technical Report 93/11, Department of Computer Science, James Cook University, Townsville, Australia, 1993.

[Sti84] M.E. Stickel. A Prolog Technology Theorem Prover. *New Generation Computing*, 2(4):371–383, 1984.

[Sti86] M.E. Stickel. Schubert's Steamroller Problem: Formulations and Solutions. *Journal of Automated Reasoning*, 2:89 – 101, 1986.

[Sti94] M.E. Stickel. Upside-Down Meta-Interpretation of the Model Elimination Theorem-Proving Procedure for Deduction and Abduction. *Journal of Automated Reasoning*, to appear, 1994.

[STvdK90] J. Schumann, N. Trapp, and M. van der Koelen. SETHEO/PARTHEO Users Manual. Technical Report SFB Bericht 342/7/90 A, Institut für Informatik, TU München, 1990.

[WM76] G.A. Wilson and J. Minker. Resolution, Refinements, and Search Strategies: A Comparative Study. *IEEE Transactions on Computers*, C-25(8):782–801, 1976.