Journal of Automated Reasoning **21:** 177–203, 1998.

The TPTP Problem Library

CNF Release v1.2.1

GEOFF SUTCLIFFE
Department of Computer Science, James Cook University, Townsville, Australia.
e-mail: geoff@cs.jcu.edu.au

CHRISTIAN SUTTNER
Institut für Informatik, TU München, Munich, Germany.
e-mail: suttner@informatik.tu-muenchen.de

(Received: 19 November 1996)

Abstract. This paper provides a detailed description of the CNF part of the TPTP Problem Library for automated theorem-proving systems. The library is available via the Internet and forms a common basis for development and experimentation with automated theorem provers. This paper explains the motivations and reasoning behind the development of the TPTP (thus implicitly explaining the design decisions made) and describes the TPTP contents and organization. It also provides guidelines for obtaining and using the library, summary statistics about release v1.2.1, and an overview of the tptp2X utility program. References for all the sources of TPTP problems are provided.

Key words: TPTP, problem library, experimental evaluation.

1. Introduction

The TPTP (Thousands of Problems for Theorem Provers) is a library of problems for automated theorem proving (ATP) systems. The principal motivation for the TPTP project is to move the testing and evaluation of ATP systems from the previously ad hoc situation onto a firm footing. This became necessary because results being published do not always accurately reflect the capabilities 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. The TPTP is such a library. The TPTP provides a simple, unambiguous source and reference mechanism for ATP problems. It is comprehensive and up to date and thus provides an overview of the current application of ATP. The TPTP problems are stored in a specifically designed, easy-to-understand format. A utility is provided for manipulating the problems and for converting the problems to other known ATP formats. Since the TPTP's first release in 1993, many researchers have used the library as an appropriate and convenient basis for ATP system evaluation.

This paper describes the part of the TPTP that uses the clause normal form (CNF) of first-order logic. Release v1.2.1 of the TPTP was the last release of the TPTP containing only CNF problems and thus serves as the exemplar (TPTP v1.2.1 is also interesting because it was used as the source of problems for the CADE-13 ATP System Competition [109]). This paper explains the motivations and reasoning behind the development of the TPTP (thus implicitly explaining the design decisions made) and describes the TPTP contents and organization. It also provides guidelines for obtaining and using the library, summary statistics about release v1.2.1, and an overview of the tptp2X utility program. References for all the sources of TPTP problems are provided.

1.1. OBTAINING AND USING THE TPTP

The TPTP may be obtained by FTP or via the Web from Australia:

```
ftp.cs.jcu.edu.au:pub/research/tptp-library/*
http://www.cs.jcu.edu.au/~tptp/
```

or Germany:

```
flop.informatik.tu-muenchen.de:pub/tptp-library/*
http://wwwjessen.informatik.tu-muenchen.de/~tptp/.
```

By providing this library of ATP problems and a specification of how these problems should be presented to ATP systems, we intend to place the testing, evaluation, and comparison of ATP systems on a firm footing. For this reason, the technical report that accompanies each TPTP release contains specific conditions for using the TPTP and presenting results based on that release of the TPTP. For TPTP v1.2.1 these are conditions as follows:

- The TPTP release number must be stated.
- Each problem must be referenced by its unambiguous syntactic name.
- No clauses/literals may be changed, added, or removed without explicit notice (this condition 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 reordering is done by using the tptp2X utility (see Section 3), the reordering 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.
- All system switches and default settings must be recorded.

Abiding by the stated conditions will allow unambiguous identification of the problem, the arrangement of clauses, and further input to the ATP system. Anyone following the rules is asked to make that clear, in any presentation of results, by the following explicit statement: “These results were obtained in compliance with the guidelines for use of TPTP ⟨Release number⟩.” By making this clear statement, ATP researchers are assured of one’s awareness of our guidelines. Conversely, it will become clear when the guidelines may have been ignored.

1.2. PREVIOUS PROBLEM COLLECTIONS

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 collected problems in order to obtain a basis for experimentation. The first major publication[⋆] in this regard was [61], which provides an explicit listing of clauses for 63 problems, many of which are still relevant today. In the same year, Wilson and Minker documented 86 problems, which have since commonly been used for ATP testing. The problem clauses are not supplied in [125], however. A second major thrust was provided by [77], which lists 75 problems. Other more recent papers are [17, 85, 63], and [65], to name a few. The *Journal of Automated Reasoning*'s Problem Corner also provided interesting challenge problems. Problems published in hardcopy form are, however, 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 small problems. A problem library in electronic form was made publicly available by Argonne National Laboratory (in Otter format [66]) in 1988 [5]. 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 [102]). 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, several factors limit the usefulness of such collections as a common basis for research. In particular, previously existing problem collections

- are often hard to discover and obtain.
  System development and system evaluations typically rely on a small set of test problems, depending on the collections of problems available to the researcher.
- need to be transformed to the syntax of the ATP system being considered.
  The problem 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).
- are often limited in scope and size.
  The problems used are often homogeneous and thus cannot be used for a broad

[⋆] To our knowledge, the first circulation of problems for testing ATP systems was due to Larry Wos in the late sixties.

test of the capabilities of the ATP system under consideration. If there are too few problems, statistically significant testing is not possible.

- may be outdated.
  The problems may insufficiently reflect the state of the art in ATP research.
- are sometimes designed and tuned (regarding clause selection, clause ordering, and literal ordering) for a particular ATP system.
  Using a collection designed and tuned for a particular ATP system may lead to biases in results.
- provide no indication of the difficulty or significance of the problems.
  The significance and difficulty of a problem, with respect to the state of the art in ATP systems, are hard to assess by newcomers to the field. Existing test problems are often not adequate anymore (e.g., Schubert's steamroller [106]), while others may be solvable only with specialized techniques (e.g., LIM+ [13]) and therefore are much too hard to start with.
- are inconsistent in their presentation of equally named problems.
  Many copies and variants of the same "original" problem may exist in different collections. Hence, unambiguous identification of problems, and therefore a clear interpretation of performance figures for given problems, has become difficult.
- are usually undocumented.
  It is hard to obtain information on problem semantics, the original problem source, and the particular style of axiomatization. This situation also contributes to the difficulty of unambiguous problem identification.
- are almost always unserviced.
  Such collections do not provide a mechanism for adding new problems or correcting errors in existing problems and cannot be used to electronically distribute new and corrected problems to the ATP community. This situation, in turn, perpetuates the use of old and erroneous problems.
- provide no guidelines for their use.
  Quite often, inadequate system evaluations are performed. As a consequence, results that provide little indication of the system properties are reported.

The problem of meaningfully interpreting results can be even worse than indicated. A few problems may be selected and hand-tuned (clauses and literals arranged in a special order, irrelevant clauses omitted, lemmas added in, etc.) specifically for the ATP system being tested. 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 approach 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. The TPTP is such a library.

1.3. WHAT IS REQUIRED?

The goal for building the TPTP has been to overcome previous 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 easy to discover and obtain.
  Awareness of the TPTP is assured by extensive formal and informal announcements. The TPTP is available via the Internet (FTP and WWW) and thus is easily available to the research community.
- is easy to use.
  Problems are presented in a specifically designed, easy-to-understand format. Automatic conversion to other known formats is also provided, thus eliminating the necessity for any other transcription.
- spans a diversity of subject matters.
  This diversity reduces biases in the development and testing of ATP systems, which arise from the use of a limited scope of problems. It also provides an overview of the domains in which ATP systems are used.
- is large enough for statistically significant testing.
  In contrast to common practice, 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.
  The TPTP contains most problems known to the community. There is no longer a need to look elsewhere.
- is up to date.
  As new problems appear in the literature and elsewhere (see Section 2), they are added to the TPTP as soon as possible.
- is independent of any particular ATP system.
  The problem clauses are arranged so as to be modular and human-readable, rather than arranged for a particular ATP system.
- contains problems varying in difficulty.
  The difficulty of problems in the TPTP ranges from very simple problems to open problems. This range allows all interested researchers, from newcomers to experts, to rely on the same problem library.
- will provide a rating for the difficulty of each problem.
  This rating 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.

- provides statistics for each problem and the library as a whole.
  Such statistics provide information about the syntactic nature of the problems.

- has an unambiguous naming scheme.
  This scheme provides unambiguous problem reference, and makes the comparison of results meaningful. See Section 2.4 for details.

- is well structured and documented.
  This feature allows effective and efficient use of library. Useful background information, such as an overview of ATP application domains, is provided.

- documents each problem.
  This feature contributes to the unambiguous identification of each problem.

- provides a mechanism for adding new problems.
  The TPTP contains standard axiomatizations that can be used in new problems. This simplifies the construction of new problems. A template is provided for submission of new problems. The TPTP is thus a channel for making new problems available to the community, in a simple and effective way.

- provides a mechanism for correcting errors in existing problems.
  All errors, noticed by the developers or reported by users, are corrected. Patched TPTP releases are made regularly.

- provides guidelines for its use in evaluating ATP systems.
  A standard library of problems together with evaluation guidelines makes reported results meaningful and reproducible by others. This will in turn simplify and improve system comparisons and will allow ATP researchers to accurately gauge their progress.

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

1.4. TPTP HISTORY UP TO V1.2.1

The development of the TPTP, by Geoff Sutcliffe at James Cook University (initially at The University of Western Australia) and Christian Suttner at the Technische Universität München (with initial support from Theodor Yemenis), started in mid-1992. A beta release, v0.5.0, of the TPTP was made to selected researchers on 15 April 1993. The beta release contained 1211 abstract problems, which resulted in 1695 ATP problems (because of alternative presentations); see Section 2.2. The first public release of the TPTP was v1.0.0, made on Friday, 12 November 1993. This release contained 2295 problems based on 1577 abstract problems, in 23 domains. Subsequent public releases were as follows:

- v1.1.0, made on 8 April 1994, containing 2652 problems based on 1897 abstract problems, in 25 domains. This release included 736 bugfixes. The tptp2X utility had also been substantially overhauled.
- v1.1.1, made on 5 July 1994. This release was not much different from v1.1.0, containing only a few bugfixes.
- v1.1.3, made on 25 August 1994. This release was made to distribute some more bugfixes and some repairs to the tptp2X utility.
- v1.2.0, made on 30 August 1995, containing 2752 problems based on 2044 abstract problems, in 25 domains. This release included 49 bugfixes. Problem generators (see Section 2.3) were introduced in this release, and the tptp2X utility had been extended and improved in various ways. The TPTP technical report had also been substantially revised.
- v1.2.1, made on 12 June 1996. This release was the one used in the CADE-13 ATP System Competition [109]. This release contained 233 bugfixes; the aim was to distribute the most bugfree TPTP release possible for the competition. An important repair made in this release was the renaming of the `equal`/2 predicate in those problems where the equality axiomatization is incomplete. The tptp2X utility in this release had been extended and improved; in particular, the equality transformations had been upgraded to detect incomplete equality axiomatizations. Finally, the clause type information had been reviewed, and throughout the TPTP the `theorem` type had been replaced by `conjecture`, to reflect the true status of those clauses when the problems are submitted to an ATP system. This is the release documented in this paper.

2. Inside the TPTP

Scope. The technical report that accompanies each TPTP release contains tables of statistics on the release. The statistics for TPTP v1.2.1, extracted from [110], are given here. They are provided as an example to illustrate the scope of the TPTP.

Release v1.2.1 of the TPTP contains 2044 abstract problems, which result in 2752 ATP problems (because of alternative presentations; see Section 2.2). Tables I, II, and III provide some statistics about the release.

The problems in the TPTP are syntactically diverse, as is indicated by the ranges of the values in Tables II and III. 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 25 domains, covering topics in the fields of logic, mathematics, computer science, engineering, and others. The domains are presented and discussed in Section 2.1.

Sources. The problems have been collected from various sources. The two principal sources have been existing electronic problem collections and the ATP literature. Other sources include logic programming, mathematics, puzzles, and

Table I. Statistics on the TPTP

Number of problem domains	25	
Number of abstract problems	2044	
Number of generic problems	85	
Number of problems	2752	
Number of non-Horn problems	1486	(54%)
Number of range-restricted problems	105	(4%)
Number of problems with equality	1920	(70%)
Number of pure equality problems	493	(18%)
Number of satisfiable problems	$\geq$ 59	(2%)
Number of propositional problems	42	(2%)
. . . being non-Horn	30	(1%)
. . . being satisfiable	6	(0%)
Total number of clauses	323,074	
Total number of literals	743,104	

Table II. Statistics for nonpropositional TPTP problems

Measure	Min.	Max.	Ave.	Median
Number of clauses	2	504	118	54
Percentage of non-Horn clauses	0%	99%	5%	4%
. . . in non-Horn problems	2%	99%	9%	5%
Percentage of unit clauses	0%	100%	34%	22%
Percentage of range-rest. clauses	0%	100%	60%	63%
Number of literals	2	1512	273	129
Percentage of equality literals	0%	100%	43%	46%
. . . in equality problems	19%	100%	61%	47%
Maximal clause size	1	25	4	5
Number of predicate symbols	1	48	9	3
Percentage of propositions	0%	67%	0%	0%
Minimal predicate arities	0	5	1	1
Maximal predicate arities	1	10	2	3
Number of functors	1	93	23	9
Percentage of constants	0%	100%	50%	50%
Minimal functor arities	0	2	0	0
Maximal functor arities	0	8	2	2
Number of variables	0	1094	269	130
Percentage of singletons	0%	100%	8%	7%
Maximal term depth	1	14	4	4

Table III. Statistics for propositional TPTP problems

Measure	Min.	Max.	Ave.	Median
Number of clauses	3	82	24	18
Percentage of non-Horn clauses	0%	81%	25%	25%
. . . in non-Horn problems	8%	81%	35%	33%
Percentage of unit clauses	0%	100%	19%	13%
Number of literals	4	232	64	40
Maximal clause size	1	11	3	3
Number of predicate symbols	2	63	13	11

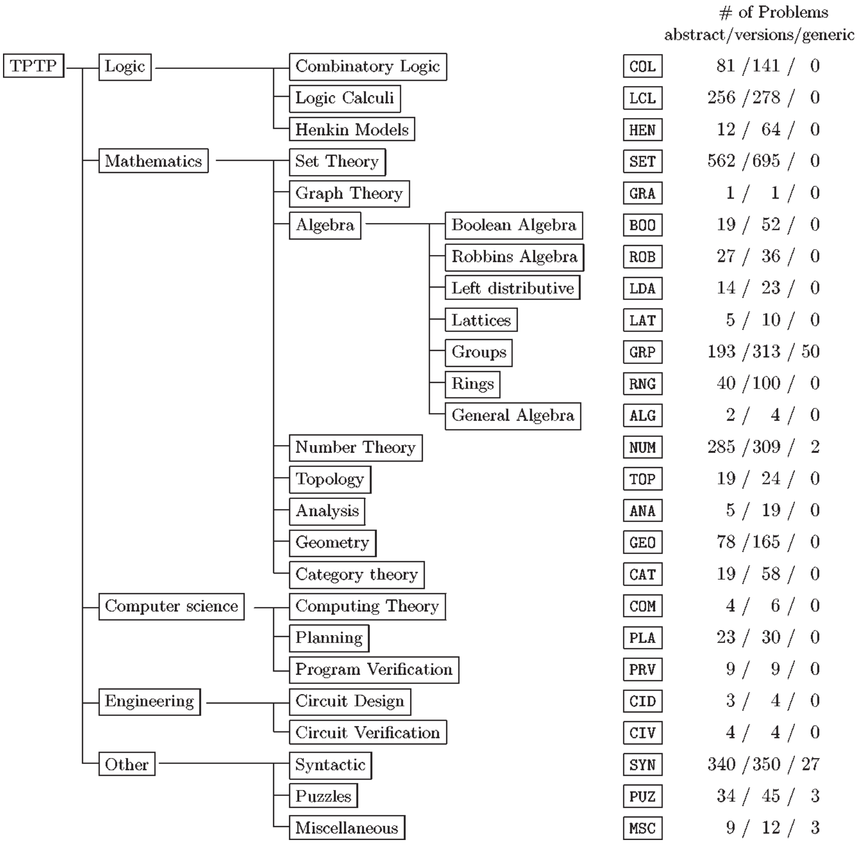
correspondence with ATP researchers. Many people and organizations have contributed to the TPTP. In particular, the foundations of the TPTP were laid with David Plaisted's SPRFN collection; many problems have been taken from Argonne National Laboratory's ATP problem library (special thanks to Bill McCune here); Art Quaife provided several hundred problems in set theory and algebra; the *Journal of Automated Reasoning*, CADE proceedings, and Association for Automated Reasoning newsletters have provided a wealth of material; and smaller numbers of problems have been provided by a number of further contributors. The books, papers, reports, and so forth from which problems have been sourced are marked with a ⊕ in the References.

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>.<Patch level>. 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 nontrivial changes are recorded in a history file, as well as in the file for an affected problem.

2.1. THE TPTP DOMAIN STRUCTURE

This section provides the structure according to which the problems are grouped into domains. Some information about the domains is also given.

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 (DDC) [25] and the Mathematics Subject Classification (MSC) [1] used for the Mathematical Reviews by the American Mathematical Society. Five main fields are defined: logic, mathematics, computer science, engineering, and other. Each field contains further subdivisions, called *domains*. Each domain is identified by a three-letter mnemonic. These mnemonics are also part of the problem naming scheme (see Section 2.4). The TPTP domains constitute the basic units of the classification. The full classification scheme is shown in Figure 1.



# of Problems
abstract/versions/generic
TPTP
Logic
Combinatory Logic COL 81 /141 / 0
Logic Calculi LCL 256 /278 / 0
Henkin Models HEN 12 / 64 / 0
Mathematics
Set Theory SET 562 /695 / 0
Graph Theory GRA 1 / 1 / 0
Algebra
Boolean Algebra BOO 19 / 52 / 0
Robbins Algebra ROB 27 / 36 / 0
Left distributive LDA 14 / 23 / 0
Lattices LAT 5 / 10 / 0
Groups GRP 193 /313 / 50
Rings RNG 40 /100 / 0
General Algebra ALG 2 / 4 / 0
Number Theory NUM 285 /309 / 2
Topology TOP 19 / 24 / 0
Analysis ANA 5 / 19 / 0
Geometry GEO 78 /165 / 0
Category theory CAT 19 / 58 / 0
Computer science
Computing Theory COM 4 / 6 / 0
Planning PLA 23 / 30 / 0
Program Verification PRV 9 / 9 / 0
Engineering
Circuit Design CID 3 / 4 / 0
Circuit Verification CIV 4 / 4 / 0
Other
Syntactic SYN 340 /350 / 27
Puzzles PUZ 34 / 45 / 3
Miscellaneous MSC 9 / 12 / 3

Figure 1. The domain structure of the TPTP.

Given below is a brief description of the domains, with a non-ATP reference for a general introduction and a generic ATP reference. For each domain, appropriate DDC and MSC numbers are also given:

ALG *Algebra.*
An algebra is a set with a system of operations defined on it.
Indices : DDC 512; MSC 06XX, 20XX.
References : General [15, 11, 10], ATP –.

ANA *Analysis.*
Analysis is a branch of mathematics concerned with functions and limits. The main parts of analysis are differential calculus, integral calculus, and the theory of functions.
Indices : DDC 515; MSC 26XX.
References : General [91], ATP [13].

BOO *Boolean Algebra.*

A Boolean algebra is a set of elements with two binary operations that are idempotent, commutative, and associative. These operations are mutually distributive; there exist universal bounds $0,\ 1$; and there is a unary operation of complementation.

Indices : DDC 511.324, 512.89; MSC 06EXX.
References : General [122, 11, 10], ATP –.

CAT *Category Theory.*

A category is a mathematical structure together with the morphisms that preserve this structure.

Indices : DDC 512.55; MSC 18XX.
References : General [59], ATP [61].

CID *Circuit Design.*

Circuits are formed by interconnecting logic gates. Circuit design is used to form a circuit that will transform given input patterns to required output patterns.

Indices : DDC 621.395; MSC 94CXX.
References : General [36], ATP [128].

CIV *Circuit Verification.*

Circuit verification is used to ensure that a previously designed circuit performs the desired transformation of input patterns to required output patterns. One approach is to check the performance of the circuit for every possible combination of given inputs. Other techniques are also used.

Indices : DDC 621.395; MSC 94CXX.
References : General [36], ATP [129].

COL *Combinatory Logic.*

Combinatory logic is about applying one function to another. It can be viewed as an alternative foundation of mathematics (or, because of its Turing completeness, as a programming language). More formally, it is a system satisfying two combinators and satisfying reflexivity, symmetry, transitivity, and two equality substitution axioms for the function that exists implicitly for applying one combinator to another.

Indices : DDC 510.101; MSC 03B40.
References : General [23, 24, 7], ATP [135].

COM *Computing Theory.*

Computing theory is a subfield of computer science dealing with theoretical issues such as decidability (does a given problem admit an algorithmic solution?), completeness (does an algorithm always find a solution if one exists?), correctness (are only solutions produced?), and computational complexity (what are the resource requirements of algorithms?).

Indices : DDC 004-006; MSC 68XX.
References : General [39], ATP –.

GEO *Geometry.*
Geometry is a branch of mathematics that deals with the measurement, properties, and relationships of points, lines, angles, surfaces, and solids. In the TPTP the Geometry domain deals mainly with plane geometry, based on Tarski's axiom system for Euclidean geometry.
Indices : DDC 516; MSC 51.
References : General [112, 113], ATP [86].

GRA *Graph Theory.*
A graph consists of a finite nonempty set of vertices together with a prescribed set of edges, each edge connecting a pair of vertices.
Indices : DDC 510.09; MSC 05CXX, 68R10.
References : General [34, 10], ATP –.

GRP *Group Theory.*
A group is a set G and a binary operation +:GxG→G that is associative, for which there is an identity element in G, and for which each element of G has an inverse in G.
Indices : DDC 512.2; MSC 20
References : General [15, 11], ATP [61].

HEN *Henkin Models.*
Henkin models provide a generalized semantics for higher-order logics. This leads to a larger class of models and, as a consequence, fewer true sentences. However, in contrast to standard semantics, complete and correct calculi can be found.
Indices : DDC 160; MSC 03CXX.
References : General [37, 46], ATP –.

LAT *Lattice Theory.*
A lattice is a set of elements with two binary operations that are idempotent, commutative, and associative and that satisfy the absorption law.
Indices : DDC 512.865; MSC 06BXX.
References : General [11], ATP [62].

LCL *Logic Calculi.*
A logic calculus defines axioms and rules of inference that can be used to prove theorems. This domain currently contains the following logical calculi:

- Implication/Negation 2-valued modal
- Implication/Negation 2-valued sentential (CN-calculus)
- Implicational propositional (IC-calculus, negation-free part of sentential calculus)
- Implication/Falsehood 2-valued sentential (C0-calculus)
- Disjunction/Negation 2-valued sentential (AN-calculus)
- Many-valued sentential (MV-calculus)
- Equivalential (EC-calculus; theorems represent group identity in Boolean groups)

- R (under some constraint, theorems represent identity in Abelian groups)
- Left group (LG-calculus; under some constraint, theorems represent formulas equal to identity in groups)
- Right group (RG-calculus; theorems are related to identity in groups)
- Wajsberg algebra.

Indices : DDC 511.3; MSC 03XX.
References : General [53], ATP [63].

LDA *Left Distributive Algebra.*
LD-algebras are related to large cardinals. Under a very strong large cardinal assumption, the free-monogenic LD-algebra can be represented by an algebra of elementary embeddings. Theorems about this algebra can be proved from a small number of properties, suggesting the definition of an embedding algebra.
Indices : DDC 512; MSC 20N02, 03E55, 08B20
References : General –, ATP [41].

MSC *Miscellaneous.*
A collection of problems that do not fit well into any other domain.

NUM *Number Theory.*
Number theory is the study of integers and their properties.
Indices : DDC 512.7; MSC 11YXX.
References : General [35], ATP [86].

PLA *Planning.*
Planning is the process of determining the sequence of actions to be performed by an agent to reach a desired state. The initial state and the desired state are provided.
Indices : DDC 006.3; MSC 68T99.
References : General [2], ATP [79, 80].

PRV *Program Verification.*
Program verification formally establishes that a computer program does the task it is designed for.
Indices : DDC 005.14; MSC 68Q60.
References : General –, ATP [136, 61].

PUZ *Puzzles.*
Puzzles are designed to test the ingenuity of humans.
Indices : DDC 510, 793.73; MSC –.
References : General [20, 101, 100], ATP –.

RNG *Ring Theory.*
A ring is a group (see above) in which the binary operation is commutative, with an associative and distributive operation $*:GxG\rightarrow G$ for which there is an identity element in G.
Indices : DDC 512.4; MSC 13XX, 16XX.
References : General [15, 10], ATP [61].

ROB *Robbins Algebra.*
The Robbins algebra domain revolves around the question “Is every Robbins algebra Boolean?” Most of the problems in this domain identify conditions that make a near-Boolean algebra Boolean.
Indices : DDC 512; MSC 03G15.
References : General [38], ATP [127].

SET *Set Theory.*
Classically, a set is a totality of certain definite, distinguishable objects of our intuition or thought, called the elements of the set. Because paradoxes arise from such a naive definition, mathematicians now regard the notion of a set as an undefined, primitive concept [40]. In this domain, naive and Neumann–Bernays–Gödel axiom sets are used.
Indices : DDC 511.322, 512.817; MSC 03EXX, 04XX.
References : General [71, 88], ATP [86].

SYN *Syntactic.*
Syntactic problems have no obvious semantic interpretation.
Indices : DDC –; MSC –.
References : General [22], ATP [77].

TOP *Topology.*
Topology is the study of properties of geometric configurations (e.g., point sets) that are unaltered by elastic deformations (homeomorphisms, i.e., functions that are 1-1 mappings between sets such that both the function and its inverse are continuous).
Indices : DDC 514; MSC 46AXX.
References : General [43, 70], ATP [123].

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*. Since 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 as such. 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 different formulations of the theorem to be proved.

Different Axiomatizations. Commonly, many different axiomatizations of a theory exist. In the TPTP an axiomatization is *standard* if it is a complete axiomatization of an established theory and it has not had any lemmas added. (Note: A standard axiomatization may be redundant, because some axioms are dependent on others. In general, it is not known whether or not an axiomatization is minimal, and thus the possibility of redundancy in standard axiomatizations must be tolerated.) In the TPTP, standard axiomatizations are kept in separate axiom files and are included in problems as appropriate. By using different standard axiomatizations of a particular theory, one can form different versions of a problem. Sets of *specialization* axioms may be used to extend or constrain an axiomatization. Specialization axioms are also kept in separate axiom files. 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. A side effect of separating out the axioms into axiom files is that the clause order in the TPTP presentation of problems is typically different from that of any original presentation. This reordering is acceptable because the performance of a decent ATP system should not be very dependent on a particular clause ordering.

Within the ATP community, some problems have been created with *nonstandard* axiomatizations. A nonstandard axiomatization is formed by modifying a standard axiomatization. The standard axiomatization may be *reduced* (i.e., axioms are removed) and the result is an *incomplete* axiomatization, or it may be *augmented* (i.e., lemmas are added) and the result is a *redundant* axiomatization. Nonstandard 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 nonstandard axiomatization, a new version of the problem is created with a standard axiomatization.

The axioms in some TPTP problems do not capture any established theory; in other words, a standard axiomatization cannot exist. These axiomatizations are called *special* axiomatizations. Typically, such axiomatizations are used in only one problem.

Equality Axiomatization. In the TPTP equality is represented using the `equal`/2 predicate, written in prefix notation like all other predicates. The `equal`/2 predicate is used only if the equality axiomatization in the problem is complete, that is, including the axioms of reflexivity, symmetry, transitivity, function substitution for all functors, and predicate substitution for all predicate symbols. If the equality axiomatization is not complete, but the “intention” is to represent equality, the `equalish`/2 predicate is used. The TPTP problems containing the `equal`/2 predicate do contain a complete equality axiomatization.

Many ATP systems have built in mechanisms (e.g., paramodulation) that make some or all of the equality and substitution axioms unnecessary. If any of these axioms is removed when the problems are submitted to an ATP system, the removal must be explicitly noted in reported results (see Section 1.1). The tptp2X utility (see Section 3) can be used to remove equality axioms.

2.3. PROBLEM GENERATORS

Some abstract problems have a generic nature, and particular instances of the abstract problem are formed according to some size parameter(s). An example of a generic problem is the N-queens problem: place N queens on a $N \times N$ chess board such that no queen attacks another. The clauses for any size of this problem can be generated automatically, for any size of $N \geq 2$. Note that satisfiability may depend on the problem size.

Up to TPTP v1.1.3, the TPTP simply contained problem files for particular sizes of generic problems. This, however, was undesirable. First, only a finite number of different problem sizes could be included, and therefore a desired size may have been missing. Second, even a small number of different problem sizes for each generic problem could consume a considerable amount of disk space. To overcome these problems the TPTP now contains *generator* files. Generator files are used to generate instances of generic problems, according to user-supplied size parameters. The generators are used in conjunction with the tptp2X utility (see Section 3).

For convenience, the TPTP still contains a particular instance of each generic problem. The size chosen for each such instance is a compromise between being large enough to be nontrivial and small enough to avoid using too much disk space. An unsatisfiable size is chosen wherever one exists. The statistics in Tables I, II, and III take into account these instances of generic problems.

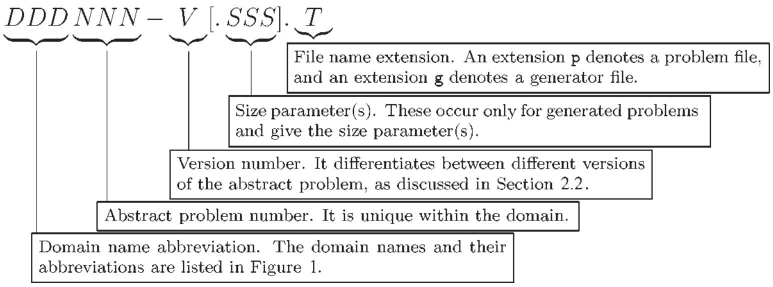
2.4. PROBLEM, GENERATOR, AND AXIOMATIZATION NAMING

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, problems, generators, and axiomatizations. Files are assigned names according to the schemes depicted in Figures 2 and 3. The `DDDNNN` combination provides an unambiguous name for an abstract problem or axiomatization. The `DDDNNN-V[.SSS]` combination provides an unambiguous name for a problem or generator, and the `DDDNNN-E` combination provides an unambiguous name for a set of axioms. The complete file names are unique within the TPTP. For example, the file `GRP135-1.002.p` contains the 135th group theory problem, version number `1`, generated size `2`. Similarly, the file `CAT001-0.ax` contains the first basic category theory axiomatization.

Wherever possible the same version number is assigned to all problems that come from the same source, within each domain.

If a file is ever removed from or renamed in the TPTP, the extension is changed to `.rm`. The file is not physically removed, and a comment is added to the file to explain what has happened. This mechanism maintains the unique identification of problems and axiomatizations.

Semantic Names. Abstract problems and axiomatizations were also allocated semantic names. The semantic names could be used to augment file names, so as to provide an indication of the file contents. While these names were provided for



$\underbrace{DDD}\,\underbrace{NNN}-\underbrace{V}\,[.\underbrace{SSS}].\underbrace{T}$

File name extension. An extension p denotes a problem file, and an extension g denotes a generator file.

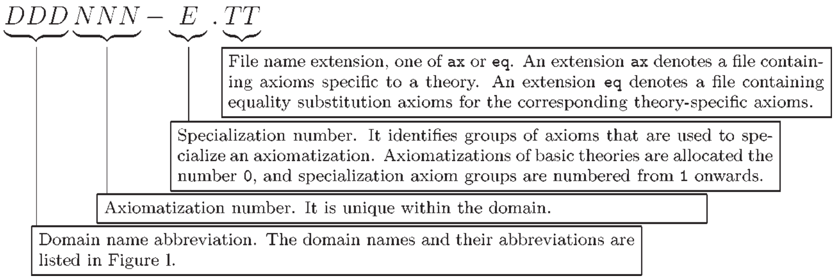
Size parameter(s). These occur only for generated problems and give the size parameter(s).

Version number. It differentiates between different versions of the abstract problem, as discussed in Section 2.2.

Abstract problem number. It is unique within the domain.

Domain name abbreviation. The domain names and their abbreviations are listed in Figure 1.

Figure 2. Problem file naming scheme.



$\underbrace{DDD}\,\underbrace{NNN}-\underbrace{E}\,.\underbrace{TT}$

File name extension, one of `ax` or `eq`. An extension `ax` denotes a file containing axioms specific to a theory. An extension `eq` denotes a file containing equality substitution axioms for the corresponding theory-specific axioms.

Specialization number. It identifies groups of axioms that are used to specialize an axiomatization. Axiomatizations of basic theories are allocated the number `0`, and specialization axiom groups are numbered from `1` onwards.

Axiomatization number. It is unique within the domain.

Domain name abbreviation. The domain names and their abbreviations are listed in Figure 1.

Figure 3. Axiom file naming scheme.

users who like to work with mnemonic names, only the standard syntactic names are guaranteed to provide unambiguous reference. Because of their rare usage, the semantic names will not be supported in future TPTP releases.

2.5. PROBLEM PRESENTATION

The physical presentation of the TPTP problem library is such that ATP researchers can easily use the problems. The syntax of all 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 `%`. The clauses are formatted as Prolog facts. The tptp2X utility can be used to convert TPTP files to other known ATP system formats (see Section 3).

Figure 4 shows an example of a TPTP problem file. The first section is a header section that contains information for the user. This information is not for use by ATP systems. It is divided into four parts separated by blank lines. The first part identifies and describes the problem:

- The `% File` field gives the problem's name, the current TPTP release number, and the TPTP release in which the problem was released or last bugfixed.
- The `% Domain` field identifies the domain, and possibly a subdomain, from which the problem is drawn (see Section 2.1).
- The `% Problem` field provides a one-line, high-level description of the abstract problem.

```
%--------------------------------------------------------------------------
% File     : GRP039=SubGI2Norm-7 : TPTP v1.2.1. Bugfixed v1.0.1.
% 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    :   25 (   2 non-Horn;  13 unit;  12 RR)
%            Number of literals   :   43 (  28 equality)
%            Maximal clause size  :    4
%            Number of predicates :    2 (   0 propositional; 1-2 arity)
%            Number of functors   :    8 (   5 constant; 0-2 arity)
%            Number of variables  :   38 (   0 singleton)
%            Maximal term depth   :    3

% Comments : Used to define a subgroup of index two is a theorem which
%            says that {for all x, for all y, there exists a z such that
%            ... (some lines removed here for brevity)
% Bugfixes : v1.0.1 - Duplicate axioms multiply_inverse_left and
%            multiply_inverse_right removed.
%--------------------------------------------------------------------------
%----Include the axioms of equality
include('Axioms/EQU001-0.ax').
%----Include the axioms for group theory in equality form
include('Axioms/GRP004-0.ax').
%----Include the subgroup axioms in equality formulation
include('Axioms/GRP004-1.ax').
%--------------------------------------------------------------------------
%----Redundant two axioms
input_clause(right_identity,axiom,
    [++equal(multiply(X,identity),X)]).

input_clause(right_inverse,axiom,
    [++equal(multiply(X,inverse(X)),identity)]).
    <some clauses omitted>

%----Denial of theorem
input_clause(b_in_O2,hypothesis,
    [++subgroup_member(b)]).

input_clause(b_times_a_inverse_is_c,hypothesis,
    [++equal(multiply(b,inverse(a)),c)]).

input_clause(a_times_c_is_d,hypothesis,
    [++equal(multiply(a,c),d)]).

input_clause(prove_d_in_O2,conjecture,
    [--subgroup_member(d)]).
%--------------------------------------------------------------------------
```

Figure 4. Example of a problem file (`GRP039-7.p`).

- 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 in the `% Problem` field is too terse.

The second part provides information about occurrences of the problem in the literature and elsewhere:

- The `% Refs` field provides a list of references in which the problem has been presented.
- The `% Source` field acknowledges the source of the problem, usually as a citation.
- The `% Names` field lists names by which this problem is known in the ATP community.

The third part gives the problem's ATP status and a table of syntactic measurements made on the problem:

- The `% Status` field gives the ATP status of the problem, one of `satisfiable`, `unsatisfiable`, `open`, or `unknown`.
- The `% Syntax` field lists various syntactic measures of the problem's clauses.

The last part contains general information about the problem:

- The `% Comments` field contains free-format comments about the problem.
- The `% Bugfixes` field describes any bugfixes that have been done to the clauses of the problem.

The second section of a problem file contains `include` instructions for axiom files. 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. If required, full versions of TPTP problems (without `include` instructions) can be created by using the tptp2X utility (see Section 3).

The last section of a problem or axiom file contains the clauses that are specific to the problem or axiomatization. The literals that make up a clause are presented as a Prolog list of terms (i.e., in `[]`). Each literal is a unary term whose functor is either `++` or `--`, indicating a positive or negative literal, respectively. The argument of the term is the atom of the literal, in the form of a Prolog term. Each clause also has a name, in the form of a Prolog atom, and a type, one of `axiom`, `hypothesis`, or `conjecture`. The `hypothesis` and `conjecture` clauses are those that are derived from the negation of the conjecture to be proved. The name, type, and literal list of each clause are bundled as the three arguments of a Prolog fact, whose predicate symbol is `input_clause`.

TPTP generator files have three sections, different from the problem and axiom files. The header section of generator files is similar to that of problem and axiom files, but with place-holders for information that is filled in based on the size of the problem and the clauses generated. Following that comes Prolog source code to generate the clauses, and finally data describing the permissible sizes and the chosen TPTP size for the problem.

2.6. PHYSICAL ORGANIZATION

The TPTP is physically organized into six subdirectories, as follows:

- `Axioms` - The axiom files directory.
- `Problems` - The problem files directory with subdirectories for each domain. The domain name abbreviations, as described in Section 2.1, are used as subdirectory names. The subdirectories contain the problem files.
- `Generators` - The generator files directory.
- `Documents` - A directory containing documents that relate to the TPTP. The files contain comprehensive online information about the TPTP. Their format provides quick access to the data, using standard system tools (e.g., `grep`, `awk`).
- `Scripts` - A directory containing scripts that may be used with the TPTP.
- `TPTP2X` - The directory containing the tptp2X utility, described in Section 3.

3. The tptp2X Utility

The tptp2X utility is a multifunctional utility for reformatting, transforming, and generating TPTP problem files. In particular, it

– Converts from the TPTP format to formats used by existing ATP systems. The system formats available are KIF [31], leanTAP [8], 3TAP [33], METEOR [6], MGTP [29], Otter [66], PTTP [105], SETHEO [93], SPASS [120, 119], SPRFN [81], and TPTP (substituting `include` instructions with the actual clauses).

– Applies various transformations to the clauses of TPTP problems. The transformations available are to reverse the order of the literals or clauses, to randomly reorder the clauses and literals, to remove selected equality axioms, to add missing equality axioms, to do Mark Stickel's magic set transformation [107], and to replace all the predicate, function, and variable symbols by short, meaningless symbols.

– Controls the generation of TPTP problem files from TPTP generator files.

The tptp2X utility is written in Prolog and should run on most Prolog systems.[⋆] It is simple to add new output formats, transformations, and generators to the TPTP.

[⋆] In particular, the tptp2X code will run on BinProlog. BinProlog is written by Paul Tarau of the University of Moncton (Canada) and is freely available by anonymous ftp, from `clement.info.umoncton.ca:BinProlog`.

4. Conclusion

This paper describes the CNF part of the TPTP Problem Library. The motivation and reasoning behind the development of the TPTP are explained, thus implicitly explaining the design decisions made. The structure and use of the TPTP are also described.

Since its first release in 1993, the TPTP has been used extensively by a large number of researchers. This has resulted in visible improvements to ATP system evaluation. Most important, direct comparisons of systems have become easily possible, through the common use of the TPTP. In particular, a growing number of publications now contain evaluations that can be meaningfully interpreted and compared with other results.

There are also some less visible, but important, effects on ATP system development. System devlopers now commonly make extensive evaluations of their own systems using the TPTP. Because of its large size and problem variety, testing based on the TPTP has revealed bugs in a number of ATP systems. The development of heuristics has also been influenced by the large number of problems in the TPTP, which provides a significantly broader basis for evaluating a general-purpose strategy. Finally, qualitative judgments about ATP systems, based on comparatively narrow evaluations, are no longer necessary.

A collection of performance data from ATP systems evaluated over the whole TPTP is being made (see, e.g., [108], but more recent data is available). The collection provides a basis for quantitative assessments of ATP systems. TPTP users are invited to submit performance data for their ATP systems. Please see the results WWW page:

`http://www.cs.jcu.edu.au/~tptp/TPTP/Results.html`

or contact us for details for contributing.

Future versions of the TPTP will contain some important new features:

- Problems in First Order Format (FOF), that is, including quantifiers, will be integrated into the TPTP. This will extend the TPTP user community to researchers working on non-normal form systems. ATP systems with automatic conversion to clause normal form will be able to derive additional information regarding a given problem, such as which functors are Skolem functors. For systems unable to deal with FOF, a clause normal form transformation will be provided in the tptp2X utility.

- Each problem will obtain an individual difficulty rating. The collection of performance data is used as the basis for this. As advances in automated theorem proving are made, the problem ratings should decrease. The long-term development of the individual problem ratings will therefore provide an objective measure of progress in the field.

- A BibTeX file will be added to the TPTP, containing entries for all the references found in the TPTP. To complement this, a WWW page of references to the TPTP is also being created. This will give an overview on the usage of the TPTP, and also will allow users to compare published results obtained for different systems.

There are also several short- and long-term plans for further development of the TPTP:

- A benchmark suite (the BSTP) will be selected from the TPTP. The BSTP will be 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. The problems will be a subset of those eligible for the CADE ATP system competitions [109].
- Translators between various logical forms will be provided, for example, from non-Horn to Horn form, and from first-order to propositional form.
- The TPTP may be extended to include problems expressed in nonclassical and higher-order logics.

References

The books, papers, reports, and so on from which problems have been sourced are marked with a ⊕.

1. *Mathematical Subject Classification*, American Mathematical Society, 1992.
2. Allen, J. F., Kautz, H. A., Pelavin, R. N., and Tenenberg, J. D.: *Reasoning about Plans*, Morgan Kaufmann, 1991.
3. ⊕ Anantharaman, S. and Hsiang, J.: Automated proofs of the Moufang identities in alternative rings, *Journal of Automated Reasoning* **6**(1) (1990), 79–110.
4. ⊕ Angshuman, G. and Zhang, H.: Andrews' challenge problem: Clause conversion and solutions, *AAR Newsletter* **14** (1989), 5–8.
5. ⊕ ANL. Argonne National Laboratory Problem Library. Available by anonymous ftp from info.msc.anl.gov, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois.
6. Astrachan, O. L.: METEOR: Exploring model elimination theorem proving, *Journal of Automated Reasoning* **13**(3) (1994), 283–296.
7. Barendregt, H. P.: *The Lambda Calculus: Its Syntax and Semantics*, North-Holland, 1981.
8. Beckert, B. and Posegga, J.: leanTAP: Lean, tableau-based deduction, *Journal of Automated Reasoning* **15**(3) (1995), 339–358.
9. ⊕ Benanav, D.: Recognising unnecessary clauses in resolution based systems, *Journal of Automated Reasoning* **9**(1) (1992), 43–76.
10. Birkhoff, G. and Bartee, T.: *Modern Applied Algebra*, McGraw-Hill, 1970.
11. Birkhoff, G. and MacLane, S.: *A Survey of Modern Algebra*, Macmillan, 1965.
12. ⊕ Bledsoe, W. W.: Non-Resolution Theorem Proving, *Artificial Intelligence* **9** (1977), 1–35.
13. ⊕ Bledsoe, W. W.: Challenge problems in elementary calculus, *Journal of Automated Reasoning* **6** (1990), 341–359.
14. ⊕ Bonacina, M. P.: Problems in Lukasiewicz logic, *AAR Newsletter* **18** (1991), 5–12.
15. Bourbaki, N.: *Algebra I – Chapters 1–3*, Springer-Verlag, 1989.
16. ⊕ Bourely, C., Caferra, R., and Peltier, N.: A method for building models automatically. Experiments with an extension of Otter, in A. Bundy (ed.), *Proceedings of the 12th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence 814, Springer-Verlag, 1994, pp. 72–86.

17. ⊕ Boyer, R., Lusk, E., McCune, W. W., Overbeek, R., Stickel, M., and Wos, L.: Set theory in first-order logic: Clauses for Godel's axioms, *Journal of Automated Reasoning* **2**(3) (1986), 287–327.
18. ⊕ Brushi, M.: The halting problem, *AAR Newsletter* **17** (1991), 7–12.
19. ⊕ Burkholder, L.: A 76th automated theorem proving problem, *AAR Newsletter* **8** (1987), 6–7.
20. Carroll, L.: *Lewis Carroll's Symbolic Logic*, C. N. Potter, 1986.
21. ⊕ Chang, C.-L.: The unit proof and the input proof in theorem proving, *Journal of the ACM* **17**(4) (1970), 698–707.
22. Church, A.: *Introduction to Mathematical Logic I*, Princeton University Press, 1956.
23. Curry, H. B. and Feys, R.: *Combinatory Logic I*, North-Holland, 1958.
24. Curry, H. B., Hindley, J. R., and Seldin, J. P.: *Combinatory Logic II*, North-Holland, 1972.
25. Dewey, M.: *Dewey Decimal Classification and Relative Index*, 20th edition, Forest Press, 1989.
26. ⊕ Fermuller, C.: E-mail to G. Sutcliffe, 1994.
27. ⊕ Fermuller, C., Leitsch, A., Tammet, T., and Zamov, N.: *Resolution Methods for the Decision Problem*, Lecture Notes in Computer Science 679, Springer-Verlag, 1993.
28. ⊕ Fleisig, S., Loveland, D. W., Smiley, A. K., and Yarmush, D. L.: An implementation of the model elimination proof procedure, *Journal of the ACM* **21**(1) (1974), 124–139.
29. Fujita, M., Hasegawa, R., Koshimura, M., and Fujita, H.: Model generation theorem provers on a parallel inference machine, in *Proceedings of the International Conference on Fifth Generation Computer Systems*, 1992, pp. 357–375.
30. ⊕ Fujita, M., Slaney, J., and Bennett, F.: Automatic generation of some results in finite algebra, in R. Bajcsy (ed.), *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1993, pp. 52–57.
31. Genesereth, M. R. and Fikes, R. E.: Knowledge Interchange Format, Version 3.0 Reference Manual, Technical Report Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.
32. ⊕ Glickfield, B. and Overbeek, R.: A foray into combinatory logic, *Journal of Automated Reasoning* **2**(4) (1986), 419–431.
33. Hähnle, R., Beckert, B., and Gerberding, S.: The many-valued tableau-based theorem prover 3TAP, Technical Report TR 30/94, Fakultät für Informatik, Universät Karlsruhe, Karlsruhe, Germany, 1994.
34. Harary, F.: *Graph Theory*, Addison-Wesley, 1969.
35. Hardy, G. F. and Wright, E. M.: *An Introduction to the Theory of Numbers*, 5th edition, Oxford University Press, 1992.
36. Hayes, J. P.: *Introduction to Digital Logic Circuit Design*, Addison-Wesley, 1993.
37. Henkin, L.: Completeness in the theory of types, *Journal of Symbolic Logic* **15** (1950), 81–91.
38. Henkin, L., Monk, J., and Tarski, A.: *Cylindrical Algebras*, volume Part 1, North-Holland, 1971.
39. Hopcroft, J. and Ullman, J.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.
40. Howson, A. G.: *A Handbook of Terms used in Algebra and Analysis*, Cambridge University Press, 1972.
41. ⊕ Jech, T.: LD-algebras, *Association for Automated Reasoning Newsletter* **22** (1993), 9–12.
42. ⊕ Jech, T.: Otter experiments in a system of combinatory logic, *Journal of Automated Reasoning* **14**(3) (1995), 413–426.
43. Kelley, J. L.: *General Topology*, D. Van Nostrand, 1955.
44. ⊕ Kunen, K.: Single axioms for groups, *Journal of Automated Reasoning* **9**(3) (1992), 291–308.

45. ⊕ Lawrence, J. D. and Starkey, J. D.: Experimental tests of resolution based theorem-proving strategies, Technical Report, Computer Science Department, Washington State University, Pullman, USA, 1974.

46. Leblanc, H.: Alternatives to standard first-oder semantics, in D. Gabbay and F. Guenther (eds), *Handbook of Philosophical Logic*, volume I, chapter I.3, D. Reidel, 1983, pp. 189–274.

47. ⊕ Lee, S.-J. and Plaisted, D. A.: Eliminating duplication with the hyper-linking strategy, *Journal of Automated Reasoning* **9**(1) (1992), 25–42.

48. ⊕ Letz, R., Mayr, K., and Goller, C.: Controlled integration of the cut rule into connection tableau calculi. *Journal of Automated Reasoning* **13**(3) (1994), 297–337.

49. Letz, R., Schumann, J., Bayerl, S., and Bibel, W.: SETHEO: A high-performance theorem prover, *Journal of Automated Reasoning* **8**(2) (1992), 183–212.

50. ⊕ Loveland, D. W.: Mechanical theorem proving by model elimination, *Journal of the ACM* **15**(2) (1968), 236–251.

51. ⊕ Loveland, D. W.: Theorem-provers combining model elimination and resolution, *Machine Intelligence* **4** (1969), 73–86.

52. ⊕ Luckham, D.: Some tree-paring strategies for theorem proving, *Machine Intelligence* **3** (1968), 95–112.

53. Lukasiewicz, J.: *Elements of Mathematical Logic*, Pergamon Press, 1963.

54. ⊕ Lusk, E. and Overbeek, R.: Non-Horn problems, *Journal of Automated Reasoning* **1**(1) (1985), 103–114.

55. ⊕ Lusk, E. and Wos, L.: Benchmark problems in which equality plays the major role, Technical Report MCS-P275-1191, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, USA, 1991.

56. ⊕ Lusk, E. and Wos, L.: Benchmark problems in which equality plays the major role, in: D. Kapur (ed.), *Proceedings of the 11th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence 607, Springer-Verlag, 1992, pp. 781–785.

57. ⊕ Lusk, E. L. and McCune, W. W.: Experiments with ROO, a parallel automated deduction system, in B. Fronhofer and G. Wrightson (eds), *Parallelization in Inference Systems*, volume 590, 1992, pp. 139–162.

58. ⊕ Lusk, E. L. and McCune, W. W.: Uniform strategies: The CADE-11 theorem proving contest, *Journal of Automated Reasoning* **11**(3) (1993), 317–331.

59. MacLane, S.: *Categories for the Working Mathematician*, Springer-Verlag, 1971.

60. ⊕ Manthey, R. and Bry, F.: SATCHMO: A theorem prover implemented in prolog, in R. Overbeek and E. Lusk (ed.), *Proceedings of the 9th International Conference on Automated Deduction*, Lecture Notes in Computer Science 310, Springer-Verlag, 1988, pp. 415–434.

61. ⊕ McCharen, J. D., Overbeek, R. A., and Wos, L. A.: Problems and experiments for and with automated theorem-proving programs, *IEEE Transactions on Computers* **25**(8) (1976), 773–782.

62. ⊕ McCune, W. W.: Challenge equality problems in lattice theory, in *Proceedings of the 9th International Conference on Automated Deduction – Argonne, Illinois, USA, May 1988*, Springer-Verlag, 1988, pp. 704–709.

63. ⊕ McCune, W. W. and Wos, L.: Experiments in automated deduction with condensed detachment, in D. Kapur (ed.), *Proceedings of the 11th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence 607, Springer-Verlag, 1992, pp. 209–223.

64. ⊕ McCune, W. W.: Automated discovery of new axiomatizations of the left group and right group calculi, *Journal of Automated Reasoning* **9**(1) (1992), 1–24.

65. ⊕ McCune, W. W.: Single axioms for groups and Abelian groups with various operations, *Journal of Automated Reasoning* **10**(1) (1993), 1–13.

66. McCune, W. W.: Otter 3.0 reference manual and guide, Technical Report ANL-94/6, Argonne National Laboratory, Argonne, USA, 1994.

67. ⊕ McCune, W. W. and Lusk, E.: A challenging theorem of Levi, *AAR Newsletter* **21** (1992), 8.

68. ⊕ McCune, W. W. and Wos, L.: Some fixed point problems in combinatory logic, *AAR Newsletter* **10** (1988), 7–8.

69. ⊕ Michie, D., Ross, R., and Shannan, G. J.: G-deduction, *Machine Intelligence* **7** (1972), 141–165.

70. Munkres, J. R.: *Topology: A First Course*, Prentice-Hall, 1975.

71. von Neumann, J.: Eine Axiomatisierung der Mengenlehre, *Journal für die Reine und Angewandte Mathematik* **154** (1925), 219–240.

72. ⊕ Ohlbach, H. J.: Predicate logic hacker tricks, *Journal of Automated Reasoning* **1**(4) (1985), 435–440.

73. ⊕ Ohlbach, H. J. and Schmidt-Schauss, M.: The lion and the unicorn, *Journal of Automated Reasoning* **1**(3) (1985), 327–332.

74. ⊕ Otter, The problem collection distributed with the Otter ATP system [McC90].

75. ⊕ Overbeek, R.: ATP competition announced at CADE-10, 1990.

76. ⊕ Overbeek, R., McCharen, J., and Wos, L.: Complexity and related enhancements for automated theorem-proving programs, *Computers and Mathematics with Applications* **2** (1976), 1–16.

77. ⊕ Pelletier, F. J.: Seventy-five problems for testing automatic theorem provers, *Journal of Automated Reasoning* **2**(2) (1986), 191–216.

78. ⊕ Pfenning, F.: Single axioms in the implicational propositional calculus, in: R. Overbeek and E. Lusk (eds), *Proceedings of the 9th International Conference on Automated Deduction*, Lecture Notes in Computer Science 310, Springer-Verlag, 1988, pp. 710–713.

79. ⊕ Plaisted, D. A.: Theorem proving with abstraction, *Artificial Intelligence* **16** (1981), 47–108.

80. ⊕ Plaisted, D. A.: A simplified problem reduction format, *Artificial Intelligence* **18** (1982), 227–261.

81. Plaisted, D. A.: Non-Horn clause logic programming without contrapositives, *Journal of Automated Reasoning* **4**(3) (1988), 287–325.

82. ⊕ Plaisted, D. A.: The search efficiency of theorem proving strategies, in A. Bundy (ed.), *Proceedings of the 12th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence 814, Springer-Verlag, 1994, pp. 57–71.

83. ⊕ Quaife, A.: Automated development of Tarski's geometry, *Journal of Automated Reasoning* **5**(1) (1989), 97–118.

84. ⊕ Quaife, A.: Andrews' challenge problem revisited, *AAR Newsletter* **15** (1990), 3–7.

85. ⊕ Quaife, A.: Automated deduction in von Neumann–Bernays–Godel set theory, *Journal of Automated Reasoning* **8**(1) (1992), 91–147.

86. ⊕ Quaife, A.: *Automated Development of Fundamental Mathematical Theories*, Kluwer Academic Publishers, 1992.

87. ⊕ Quaife, A.: E-mail to G. Sutcliffe, 1992.

88. Quine, W. V.: *Set Theory and its Logic*, Harvard University Press, 1969.

89. ⊕ Reboh, R., Raphael, B., Yates, R. A., Kling, R. E., and Velarde, C.: Study of Automatic Theorem Proving Programs. Technical Report Technical Note 72, Artificial Intelligence Center, SRI International, Menlo Park, CA, 1972.

90. ⊕ Robinson, J. A.: Theorem proving on the computer, *Jounal of the ACM* **10**(2) (1963), 163–174.

91. Ross, K. A.: *Elementary Analysis: The Theory of Calculus*, 2nd edition, Springer-Verlag, 1990.

92. ⊕ Schulz, S.: Explanation based learning for distributed equational deduction, Master's thesis, Universität Kaiserslautern, 1995.

93. Schumann, J., Trapp, N., and van der Koelen, M.: SETHEO/PARTHEO Users Manual, Technical Report SFB Bericht 342/7/90 A, Institut für Informatik, Technische Universität München, Munich, Germany, 1990.

94. ⊕ Segre, A. and Elkan, C.: A high-performance explanation-based learning algorithm, *Artificial Intelligence* **69**(NEED) (1994), 1–50.

95. ⊕ SETHEO, AR Research Group, Technische Universität München, Problem Library, No longer available.

96. ⊕ Shostak, R. E.: Refutation graphs, *Artificial Intelligence* **7** (1976), 51–64.

97. ⊕ Slagle, J. R.: Automatic theorem proving with renamable and semantic resolution, *Journal of the ACM* **14**(4) (1967), 687–697.

98. ⊕ Slaney, J.: E-mail to G. Sutcliffe, 1993.

99. ⊕ Slaney, J., Fujita, M., and Stickel, M.: Automated reasoning and exhaustive search: Quasigroup existence problems, *Computers and Mathematics with Applications* **29**(2) (1995), 115–132.

100. Smullyan, R. M.: *To Mock a Mocking Bird and Other Logic Puzzles*, Knopf, 1978.

101. ⊕ Smullyan, R. M.: *What Is the Name of This Book? The Riddle of Dracula and Other Logical Puzzles*, Prentice-Hall, 1978.

102. ⊕ SPRFN, The problem collection distributed with the SPRFN ATP system.

103. ⊕ Stevens, R. L.: Some experiments in nonassociative ring theory with an automated theorem prover, *Journal of Automated Reasoning* **3**(2) (1987), 211–221.

104. ⊕ Stickel, M.: E-mail to G. Sutcliffe, 1993.

105. Stickel, M. E.: A Prolog technology theorem prover, *New Generation Computing* **2**(4) (1984), 371–383.

106. Stickel, M. E.: Schubert's steamroller problem: Formulations and solutions, *Journal of Automated Reasoning* **2**(1) (1986), 89–101.

107. Stickel, M. E.: Upside-down meta-interpretation of the model elimination theorem-proving procedure for deduction and abduction, *Journal of Automated Reasoning* **13**(2) (1994), 189–210.

108. Sutcliffe, G. and Suttner, C. B.: ATP system results for the TPTP problem library (upto TPTP v1.1.3), 1995, Technical Report 95/16, Department of Computer Science, James Cook University, Townsville, Australia; Technical Report AR-95-06, Institut für Informatik, Technische Universität München, Munich, Germany.

109. Sutcliffe, G. and Suttner, C. B.: Special issue: The CADE-13 ATP system competition, *Journal of Automated Reasoning* **18**(2) (1997).

110. Suttner, C. B. and Sutcliffe, G.: The TPTP problem library (TPTP v1.2.1), 1996, Technical Report AR-96-02, Institut für Informatik, Technische Universität München, Munich, Germany; Technical Report 96/09, Department of Computer Science, James Cook University, Townsville, Australia.

111. ⊕ Tammet, T.: E-mail to Geoff Sutcliffe, 1994.

112. Tarski, A.: *A Decision Method for Elementary Algebra and Geometry*, 2nd edition, University of California Press, 1951.

113. Tarski, A.: What is Elementary Geometry? in L. Henkin (ed.), *Proceedings of an International Symposium. The Axiomatic Method with Special Reference to Geometry and Physics*, North-Holland, 1959.

114. ⊕ Veroff, R.: Group Theory Problems, 1992.

115. ⊕ Veroff, R.: Problem set, *AAR Newsletter* **24** (1994), 8.

116. ⊕ Wang, H.: Formalization and automatic theorem-proving, in W. A. Kalenich (ed.), *Proceedings of the IFIP Congress*, Spartan Books, 1965, pp. 51–58.

117. ⊕ Wang, T.-C.: Designing examples for semantically guided hierarchical deduction, in A. Joshi (ed.), *Proceedings of the 9th International Joint Conference on Artificial Intelli-*

gence, Los Altos, International Joint Conferences on Artificial Intelligence, Inc., 1985, pp. 1201–1207

118. ⊕ Wang, T.-C. and Bledsoe, W. W.: Hierarchical deduction, *Journal of Automated Reasoning* **3**(1) (1987), 35–77.
119. Weidenbach, C.: SPASS – a first-order theorem prover and a CNF-translator, URL http://www.mpi-sb.mpg.de/guide/software/spass.html, 1996.
120. Weidenbach, C., Gaede, B., and Rock, G.: SPASS & FLOTTER, in M. McRobbie and J. Slaney (eds), *Proceedings of the 13th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence 1104, Springer-Verlag, 1996, pp. 141–145.
121. ⊕ Whitehead, A. N. and Russell, B.: *Principia Mathematica*, Vol. 1, 2nd edition, Cambridge Univ. Press, 1927.
122. Whitesitt, J. E.: *Boolean Algebra and Its Applications*, Addison-Wesley, 1961.
123. ⊕ Wick, C. and McCune, W.: Automated reasoning about elementary point-set topology, *Journal of Automated Reasoning* **5** (1989), 239–255.
124. ⊕ Wilson, D. S. and Loveland, D. W.: Incorporating relevancy testing in SATCHMO, Technical Report CS-1989-24, Department of Computer Science, Duke University, Durham, USA, 1989.
125. ⊕ Wilson, G. A. and Minker, J.: Resolution strategies: A comparative study, *IEEE Transactions on Computers* **C-25**(8) (1976), 782–801.
126. ⊕ Winker, S.: Generation and verification of finite models and counterexamples using an automated theorem prover answering two open questions, *Journal of the ACM* **29**(2) (1982), 273–284.
127. ⊕ Winker, S.: Robbins algebra: Conditions that make a near-Boolean algebra Boolean, *Journal of Automated Reasoning* **6**(4) (1990), 465–489.
128. Wojciechowski, W. S. and Wojcik, A. S.: Automated design of multi-valued logic circuits by automated theorem proving techniques, *IEEE Transactions on Computers* **C-32**(8) (1983), 785–798.
129. Wojcik, A. S.: Formal design verification of digital systems, in *Proceedings of the 20th Design Automation Conference*, 1983.
130. ⊕ Wos, L.: Unpublished note, Argonne National Laboratory, Argonne, USA, 1965.
131. ⊕ Wos, L.: *Automated Reasoning – 33 Basic Research Problems*, Prentice-Hall, 1988.
132. ⊕ Wos, L.: A challenge problem and a recent workshop, *AAR Newsletter* **13** (1989), 2–8.
133. ⊕ Wos, L.: The kernel strategy and its use for the study of combinatory logic, *Journal of Automated Reasoning* **10**(3) (1993), 287–344.
134. ⊕ Wos, L.: Challenge in group theory, *AAR Newsletter* **26** (1994), 3–5.
135. ⊕ Wos, L. and McCune, W.: Challenge problems focusing on equality and combinatory logic: Evaluating automated theorem-proving programs, in R. Overbeek and E. Lusk (eds), *Proceedings of the 9th International Conference on Automated Deduction*, Lecture Notes in Computer Science 310, Springer-Verlag, 1988, pp. 714–729.
136. ⊕ Wos, L., Overbeek, R., Lusk, E., and Boyle, J.: *Automated Reasoning – Introduction and Applications*, 2nd edition, McGraw-Hill, 1992.
137. ⊕ Wos, L., Winker, S., McCune, W. W., Overbeek, R., Lusk, E., and Stevens, R.: Automated reasoning contributes to mathematics and logic, in M. Stickel (ed.), *Proceedings of the 10th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence 449, Springer-Verlag, 1990, pp. 485–499.
138. ⊕ Zhang, Z.: Automated proofs of equality problems in Overbeek's competition, *Journal of Automated Reasoning* **11**(3) (1993), 333–351.