Progress in the Development of Automated Theorem Proving for Higher-order Logic*

Geoff Sutcliffe[1], Christoph Benzmüller[2], Chad E. Brown[3], and Frank Theiss[3]**

[1] University of Miami, USA***
[2] International University, Germany
[3] Saarland University, Germany

Abstract. The Thousands of Problems for Theorem Provers (TPTP) problem library is the basis of a well established infrastructure supporting research, development, and deployment of first-order Automated Theorem Proving (ATP) systems. Recently, the TPTP has been extended to include problems in higher-order logic, with corresponding infrastructure and resources. This paper describes the practical progress that has been made towards the goal of TPTP support for higher-order ATP systems.

1 Motivation and History

There is a well established infrastructure that supports research, development, and deployment of first-order Automated Theorem Proving (ATP) systems, stemming from the Thousands of Problems for Theorem Provers (TPTP) problem library [38]. This infrastructure includes the problem library itself, the TPTP language [36], the SZS ontologies [35], the Thousands of Solutions from Theorem Provers (TSTP) solution library, various tools associated with the libraries [34], and the CADE ATP System Competition (CASC) [37]. This infrastructure has been central to the progress that has been made in the development of high performance first-order ATP systems.

Until recently there has been no corresponding support in higher-order logic. In 2008, work commenced on extending the TPTP to include problems in higher-order logic, and developing the corresponding infrastructure and resources. These efforts aim to have an analogous impact on the development of higher-order ATP systems. The key steps have been:

- Development of the Typed Higher-order Form (THF) part of the TPTP language, compatible with the existing first-order forms (FOF and CNF).

* This research has received funding from the European Community's Seventh Framework Programme FP7/2007-2013, under grant agreement PIIF-GA-2008-219982.
** Supported by the German Federal Ministry of Education and Research (BMBF) in the framework of the Verisoft XT project under grant 01 IS 07 008.
*** Most of the work was done while hosted as a guest of the Automation of Logic group in the Max Planck Institut für Informatik.

– Collecting THF problems, for the TPTP.
– Building TPTP infrastructure for THF problems.
– Finding and implementing ATP systems for higher-order logic.
– Collecting ATP systems' solutions to the THF problems, for the TSTP.
– Planning for a THF division of CASC.

These topics are described in this paper.

2 The Higher-order TPTP

The development of the higher-order part of the TPTP involved three of the steps identified in the introduction: development of the Typed Higher-order Form (THF) part of the TPTP language, collecting THF problems for the TPTP, and building TPTP infrastructure for THF problems. These steps are described in this section.

2.1 The Typed Higher-order Form (THF) Language

The TPTP language is a human-readable, easily machine-parsable, flexible and extensible language, suitable for writing both ATP problems and solutions. A particular feature of the TPTP language, which has been maintained in the THF part, is Prolog compatibility. As a result the development of reasoning software (for TPTP data) in Prolog has a low entry barrier [36]. The top level building blocks of the TPTP language are *include directives* and *annotated formulae*. Include directives are used mainly in problem files to include the contents of axiom files. Annotated formulae have the form:

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

The *language*s supported are first-order form (`fof`), clause normal form (`cnf`), and now typed higher-order form (`thf`). The *role* gives the user semantics of the *formula*, e.g., `axiom`, `lemma`, `type`, `definition`, `conjecture`, and hence defines its use in an ATP system. The logical *formula* uses a consistent and easily understood notation, and uses only standard ASCII characters.

The THF language for logical formulae is a syntactically conservative extension of the existing first-order TPTP language, adding constructs for higher-order logic. Maintaining a consistent style between the first-order and higher-order languages allows easy adoption of the new higher-order language, through reuse or adaptation of existing infrastructure for processing TPTP format data, e.g., parsers, pretty-printing tools, system testing harnesses, and output processors. The THF language has been divided into three layers: THF0, THF, and THFX. The THF0 core language is based on Church's simple type theory, and provides the commonly used and accepted aspects of a higher-order logic language [13]. The full THF language drops the differentiation between terms and types, thus providing a significantly richer type system, and adds the ability to reason about types. It additionally offers more term constructs, more type constructs, and more connectives. The extended THFX language adds constructs

that are "syntactic sugar", but are usefully expressive. Initially the TPTP will contain higher-order problems in only THF0, to allow users to adopt the language without being swamped by the richness of the full THF language. The full THF language definition is available from the TPTP web site, www.tptp.org.

Figures 1 and 2 show an example of a TPTP problem file in THF0. The `thf` annotated formulae in Figure 1 illustrate common constructs in the THF0 language. The constructs that are not part of the first-order TPTP language are:

- The typing of constants and quantified variables. THF requires that all symbols be typed. Constants are globally typed in an annotated formula with role `type`, and variables are locally typed in their quantification (thus requiring all variables to be quantified).
- `$tType` for the collection of all types.
- `$i` and `$o` for the types of individuals and propositions. `$i` is non-empty, and may be finite or infinite.
- `>` for (right associative) function types.
- `^` as the lambda binder.
- `@` for (left associative) application.

Additional THF constructs, not shown in the example, are:

- The use of connectives as terms (THF0), e.g.,
  `(& @ $false) = (^ [P:$o] : $false)`
- `!!` and `??` for the Π (forall) and Σ (exists) operators (THF0), e.g.,
  `((!! (p)) & (!! (q))) = (! [X:$i] : (( p @ X) & (q @ X)))`
- `!>` and `?*` for Π (dependent product) and Σ (sum) types (THF), e.g.,
  `cons: !> [N:nat] : ($i > (list @ N) > (list @ (succ @ N)))`
- `[ ]` for tuples (THF), e.g.,
  `make_triple = ^ [X:$i,Y:$i:,Z:$i] : [X,Y,Z]`
- `*` and `+` for simple product and sum (disjoint union) types (THF), e.g.,
  `roots: quadratic > (($real * $real) + $real + undef)`
- `:=` as a connective for global definitions, and as a binder and separator for local definitions (ala `letrec`) (THFX), e.g.,
  `apply_twice:= ^ [F:$o > $o,X:$o] : (F @ (F @ X))`
  defines `apply_twice` with global scope, and
  `:= [NN:= (apply_twice @ ~)] : (NN = (apply_twice @ NN))`
  has `NN` defined with local (the formula) scope.
- `-->` as the sequent connective (THFX), e.g., `[p,q,r] --> [s,t]`

The choice of semantics for THF problems is of interest, as, unlike the first-order case, there are different options [6, 7]. For THF0 the default is Henkin semantics with extensionality (without choice or description). The default semantics of the higher THF layers have not been fixed yet, since this choice will be dependent on which systems adopt the higher-order TPTP.

The first section of each TPTP problem file is a header that contains information for the user. Figure 2 shows the header for the annotated formulae

```
%------------------------------------------------------------------------------
%----Include simple maths definitions and axioms
include('Axioms/LCL008^0.ax').
%------------------------------------------------------------------------------
thf(a,type,(
    a: $tType )).

thf(p,type,(
    p: ( a > $i > $o ) > $i > $o )).

thf(g,type,(
    g: a > $i > $o )).

thf(e,type,(
    e: ( a > $i > $o ) > a > $i > $o )).

thf(r,type,(
    r: $i > $i > $o )).

thf(mall_aio,type,(
    mall_aio: ( ( a > $i > $o ) > $i > $o ) > $i > $o )).

thf(mall_a,type,(
    mall_a: ( a > $i > $o ) > $i > $o )).

thf(mall_aio,definition,
    ( mall_aio
    = ( ^ [P: ( a > $i > $o ) > $i > $o,W: $i] :
        ! [X: a > $i > $o] :
          ( P @ X @ W ) ) )).

thf(mall_a,definition,
    ( mall_a
    = ( ^ [P: a > $i > $o,W: $i] :
        ! [X: a] :
          ( P @ X @ W ) ) )).

thf(positiveness,axiom,
    ( mvalid
    @ ( mall_aio
      @ ^ [X: a > $i > $o] :
          ( mimpl @ ( mnot @ ( p @ X ) )
          @ ( p
            @ ^ [Z: a] :
                ( mnot @ ( X @ Z ) ) ) ) ) )).

thf(g,definition,
    ( g
    = ( ^ [Z: a] :
          ( mall_aio
          @ ^ [X: a > $i > $o] :
              ( mimpl @ ( p @ X ) @ ( X @ Z ) ) ) ) )).

thf(e,definition,
    ( e
    = ( ^ [X: a > $i > $o,Z: a] :
          ( mall_aio
          @ ^ [Y: a > $i > $o] :
              ( mimpl @ ( Y @ Z )
              @ ( mbox @ r
                @ ( mall_a
                  @ ^ [W: a] :
                      ( mimpl @ ( X @ W ) @ ( Y @ W ) ) ) ) ) ) ) )).

thf(thm,conjecture,
    ( mvalid
    @ ( mall_a
      @ ^ [Z: a] :
          ( mimpl @ ( g @ Z ) @ ( e @ g @ Z ) ) ) )).

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

Fig. 1. `LCL634^1` formulae

```
%--------------------------------------------------------------------------
% File     : LCL634^1 : TPTP v3.7.0. Released v3.6.0.
% Domain   : Logical Calculi
% Problem  : Goedel's ontological argument on the existence of God
% Version  : [Ben08] axioms : Especial.
% English  :

% Refs     : [Fit00] Fitting (2000), Higher-Order Modal Logic - A Sketch
%          : [Ben08] Benzmueller (2008), Email to G. Sutcliffe
% Source   : [Ben08]
% Names    : Fitting-HOLML-Ex-God-alternative-b [Ben08]

% Status   : Theorem
% Rating   : 1.00 v3.7.0
% Syntax   : Number of formulae    :   48 (   3 unit;  27 type;  19 defn)
%            Number of atoms       :  323 (  19 equality;  60 variable)
%            Maximal formula depth :   13 (   5 average)
%            Number of connectives :   71 (   3   ~;   1   |;   2   &;  64   @)
%                                         (   0 <=>;   1  =>;   0  <=;   0 <~>)
%                                         (   0  ~|;   0  ~&;   0  !!;   0  ??)
%            Number of type conns  :  118 ( 118   >;   0   *;   0   +)
%            Number of symbols     :   28 (  27   :;   0  :=)
%            Number of variables   :   51 (   2 sgn;   6   !;   4   ?;  41   ^)
%                                         (  51   :;   0  :=;   0  !>;   0  ?*)

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

Fig. 2. `LCL634^1` header

of Figure 1. This information is not for use by ATP systems. It is divided into four parts. The first part identifies and describes the problem, the second part provides information about occurrences of the problem in the literature and elsewhere, the third part gives the problem's status as an SZS ontology value [35] and a table of syntactic measurements made on the problem, and the last part contains general comments about the problem. The status value is for the default semantics – Henkin semantics with extensionality. If the status is known to be different for other semantics, e.g., without functional/Boolean extensionality, or with addition of choice or description, this is provided on subsequent lines, with the modified semantics noted.

2.2 Collecting THF Problems, for the TPTP

The THF problems collected in the second half of 2008 and first quarter of 2009 were part of TPTP v3.7.0, which was released on 8th March 2009. This was a beta release of the THF part of the TPTP, and contained higher-order problems in only the THF0 language. There were 1275 THF problem versions, stemming from 852 abstract problems, in nine domains:

- `ALG` - 50 problems. These are problems concerning higher-order abstract syntax, encoded in higher-order logic [19].
- `GRA` - 93 problems. These are problems about Ramsey numbers, some of which are open in the mathematics community.

- `LCL` - 56 problems. These are of modal logic problems that have been encoded in higher-order logic.
- `NUM` - 221 problems. These are mostly theorems from Jutting's AUTOMATH formalization [40] of the well known Landau book [24]. These are also some Church numeral problems.
- `PUZ` - 5 problems. These are "knights and knaves" problems.
- `SET` and `SEU` - 749 problems. Many of these are "standard" problems in set theory that have TPTP versions in first-order logic. This allows for an evaluation of the relative benefits of the different encodings with respect to ATP systems for the logics [14]. There is also a significant group of problems in dependently typed set theory [17], and a group of interesting problems about binary relations.
- `SWV` - 37 problems. The two main groups of problems are (i) security problems in access control logic, initially encoded in modal logic, and subsequently encoded in higher-order logic [9], and (ii) problems about security in an authorization logic that can be converted via modal logic to higher-order logic [20].
- `SYN` - 59 problems. These are simple problems designed to test properties of higher-order ATP systems [6].

1038 of the problems (81%) contain equality. 1172 of the problems (92%) are known or believed to be theorems, 28 (2%) are known or believed to be non-theorems, and the remaining 75 problems (6%) have unknown status. Table 1 provides some further detailed statistics about the problems.

	Min	Max	Avg	Median
Number of formulae	1	749	118	16
% of unit formulae	0%	60%	24%	27%
Number of atoms	2	7624	1176	219
% of equality atoms	0%	33%	5%	6%
in problems with equality	1%	33%	7%	7%
% of variable atoms	0%	82%	33%	33%
Avg atoms per formula	1.8	998.0	50.0	8.0
Number of symbols	1	390	66	12
Number of variables	0	1189	182	31
^	0	175	22	5
!	0	1067	150	11
?	0	45	10	2
Number of connectives	0	4591	677	73
Number of type connectives	0	354	62	28
Maximal formula depth	2	351	67	14
Average formula depth	2	350	16	6

Table 1. Statistics for THF problems

2.3 TPTP Infrastructure for THF Problems

The first-order TPTP provides a range of resources to support use of the problem library [34]. Many of these resources are immediately applicable to the higher-order setting, while some have required changes for the new features of the THF language.

From a TPTP user perspective, the TPTP2X utility distributed with the TPTP will initially be most useful for manipulating THF problems. TPTP2X has been extended to read, manipulate, and output (pretty print) data in the full THF language. Additionally, format modules for outputting problems in the Tps [4], Twelf [28], OmDoc [23], Isabelle [27], and S-expression formats have been implemented. The TPTP4X tool has also been extended to read, manipulate, and output data in the THF0 language, and will be extended to the full THF language.

The SystemOnTPTP utility for running ATP systems and tools on TPTP problems and solutions has been updated to deal with THF data, including use of the new higher-order formats output by TPTP2X. The online interface to SystemOnTPTP (www.tptp.org/cgi-bin/SystemOnTPTP) has also been updated to deal with THF data, and includes ATP systems and tools for THF data.

Internally, an important resource is the Twelf-based type checking of THF problems, implemented by exporting a problem in Twelf format, and submitting the result to the Twelf tool - see [13] for details.

The BNF based parsers for the TPTP [41] naturally parse the full THF language, and the `lex`/`yacc` files used to build these parsers are freely available.

3 Collecting Solutions to THF Problems, for the TSTP

The Thousands of Solutions from Theorem Provers (TSTP) solution library, the "flip side" of the TPTP, is a corpus of contemporary ATP systems' solutions to the TPTP problems. A major use of the TSTP is for ATP system developers to examine solutions to problems, and thus understand how they can be solved. The TSTP is built using a harness that calls the SystemOnTPTP utility, and thus leverages many aspects of the TPTP infrastructure for THF data.

Four higher-order ATP systems, LEO-II 0.99a, Tps 3.0, and two automated versions of Isabelle 2008 (one - IsabelleP - trying to prove theorems, the other - IsabelleM - trying to find (counter-)models), have been run over the 1275 THF problems in TPTP v3.7.0, and their results added to the TSTP. The systems are described in Section 4. Table 2 tabulates the numbers of problems solved. The "Any", "All", and "None" rows are with respect to to the three theorem provers. All the runs were done on 2.80GHz computers with 1GB memory and running the Linux operating system, with a 600s CPU limit.

The results show that the `GRA` Ramsey number problems are very difficult - this was expected. For the remaining domains the problems pose interesting challenges for the ATP systems, and the differences between the systems lead to different problems being solved, including some that are solved uniquely by each of the systems.

	ALG	GRA	LCL	NUM	PUZ	SE?	SWV	SYN	Total	Unique
Problems	50	93	61	221	5	749	37	59	1275	
LEO-II 0.99a	34	0	48	181	3	401	19	42	725	127
IsabelleP 2008	0	0	0	197	5	361	1	30	594	74
TPS 3.0	10	0	40	150	3	285	9	35	532	6
Any	32	0	50	203	5	490	20	52	843	207
All	0	0	0	134	2	214	0	22	372	
None	18	93	12	18	0	259	17	15	432	
IsabelleM 2008	0	0	1	0	0	0	0	8	9	

Table 2. Results for THF problems

4 Higher-Order ATP for the TPTP

Research and development of computer-supported reasoning for higher-order logic has been in progress for as long as that for first-order logic. It is clear that the computational issues in the higher-order setting are significantly harder than those in first-order. Problems such as the undecidability of higher-order unification, the handling of equality and extensionality reasoning, and the instantiation of set variables, have hampered the development of effective higher-order automated reasoning. Thus, while there are many *interactive* proof assistants based on some form of higher-order logic [43], there are few *automated* systems for higher-order logic. This section describes the three (fully automatic) higher-order ATP systems that we know of.

4.1 LEO-II

LEO-II [12] is a resolution based higher-order ATP system. It is the successor of LEO [8], which was implemented in LISP and hardwired to the OMEGA proof assistant [32]. LEO-II is implemented in Objective Caml, and is freely available from http://www.ags.uni-sb.de/~leo/ under a BSD-like licence.

LEO-II is designed to cooperate with specialist systems for fragments of higher-order logic. The idea is to combine the strengths of the different systems: LEO-II predominantly addresses higher-order aspects in its reasoning process, with the aim of quickly removing higher-order clauses from the search space, and turning them into first-order clauses that can be refuted with a first-order ATP system. Currently, LEO-II is capable of cooperating with the first-order ATP systems E [31], SPASS [42], and Vampire [30].

In addition to a fully automatic mode, LEO-II provides an interactive mode [11]. This mode supports debugging and inspection of the search space, and also the tutoring of resolution based higher-order theorem proving to students. The interactive mode and the automatic mode can be interleaved.

LEO-II directly parses THF0 input. THF0 is the only input syntax supported by LEO-II. The THF problem collection has been a valuable testbed in this respect,

providing examples that exposed intricacies of the LEO-II parser. Some problems revealed differing precedences for logical connectives in THF0 and LEO-II. Instead of generating parsing errors these examples led to different semantic interpretations. An example is the tautologous axiom `! [X:$o]:(~(X) | X)`. Due to mistaken operator precedences, LEO-II used to (mis)read this axiom as `! [X:$o]:(~(X | X))`.

Communication between LEO-II and the cooperating first-order ATP system uses TPTP standards. LEO-II's clause set generally consists of higher-order clauses that are processed with LEO-II's calculus rules. Some of the clauses in LEO-II's search space additionally attain a special status: they are first-order clauses modulo the application of an appropriate transformation function. The default transformation is Hurd's fully typed translation [22]. LEO-II's extensional higher-order resolution approach enhances standard resolution proof search with specific extensionality rules that generate more and more essentially first-order clauses from higher-order ones. LEO-II is often too weak to find a refutation amongst the steadily growing set of essentially first-order clauses on its own. Therefore, LEO-II launches the cooperating first-order ATP system every n iterations of its (standard) resolution proof search loop (currently $n = 10$). The subproblem passed to the first-order ATP system is written in the TPTP language. If the first-order ATP system finds a refutation and communicates its success to LEO-II in the standard SZS format: `SZS status Unsatisfiable`. LEO-II analyzes this answer and recognizes the reference to unsatisfiablity in the SZS ontology. LEO-II stops the proof search and reports that the problem is a theorem, in the standard SZS format: `SZS status Theorem`.

Debugging of LEO-II benefits from the examples in the TPTP library. Several bugs in LEO-II, beyond the parsing bugs described above, have been detected through use of the TPTP library. These include problems in the translation from higher-order to first-order form, and accidentally omitted type checks in the higher-order unification algorithm. The library has thus provided an excellent basis for finding and curing various "Kinderkrankheiten" that a new ATP system inevitably experiences.

Future work includes further exploitation of TPTP infrastructure An important step will be the integration of TPTP format proofs output by the cooperating first-order ATP system into LEO-II's higher-order resolution proofs. The goal is to produce a single, coherent proof in the TPTP language.

4.2 TPS

TPS [4] is a higher-order theorem proving system that has been developed under the supervision of Peter B. Andrews since the 1980s. Theorems can be proven in TPS either interactively or automatically. Some of the key ingredients of the automated search procedures of TPS are mating search [2] (which is similar to Bibel's connection method [15]), Miller's expansion trees [25], and Huet's

higher-order pre-unification [21]. In essence, the goal of each search procedure is to find an appropriate set of connections (i.e., a complete mating), and to find appropriate instantiations for certain variables [3].

In TPS there are flags that can be set to affect the behavior of automated search. A collection of flag settings is called a *mode*. The mode determines which particular search procedure will be used, as well as how the exploration of the search space should be ordered. Over 500 modes are available in the TPS library. The two modes considered in this paper are `MS98-FO-MODE` and `BASIC-MS04-2-MODE`.

The mode `MS98-FO-MODE` uses a search procedure `MS98-1`, implemented by Matthew Bishop [16]. The procedure precomputes components (compatible sets of connections) and then attempts to combine the components to construct a complete mating. This approach enables TPS to solve a number of problems that were too hard for earlier search procedures. `MS98-1` and all earlier search procedures are based on Miller's expansion trees. Consequently, the procedures attempt to find proofs that do not use extensionality, i.e., these search procedures can prove theorems of only elementary type theory [1].

The mode `BASIC-MS04-2-MODE` uses a search procedure `MS04-2`, implemented by Chad Brown [18]. Unlike `MS98-1`, `MS04-2` can find proofs of theorems requiring extensionality. `MS04-2` is the only TPS search procedure that is complete relative to Henkin semantics [18]. The procedure is based on extensional expansion DAGs, a generalization of Miller's expansion trees. The trees become DAGs because connections can generate new nodes that are children of the two connected nodes. For theorems that do not require extensionality, this extra complication can expand the search space unnecessarily. Also, `MS04-2` relies on backtracking in a way `MS98-1` does not.

As the two TPS modes have quite different capabilities, and it is expected that any proofs found by either mode will be found quickly, running the two modes in competition parallel is a simple way of obtaining greater coverage. A simple `perl` script has ben used to do this, running two copies of TPS in parallel as separate UNIX processes, one for each of the modes. As soon as either process finds a proof, the script terminates the other. It was this competition parallel version of TPS that produced the 532 proofs noted in Table 2. Analysis of the system's outputs shows that the parallelism is effective, with the two modes each solving about half of the problems (first). This indicates that the TPTP has a good balance of problems with respect to these two TPS modes. A new strategy scheduling version of TPS is currently being developed, which will run many more modes, but in sequence to avoid memory contention.

4.3 IsabelleP and IsabelleM

Isabelle [27] is a well known proof assistant for higher-order logic. It is normally used interactively through the Proof General interface [5]. In this mode it is possible to apply various automated tactics that attempt to solve the current goal without further user interaction. Examples of these tactics are `blast`, `auto`, and `metis`. It is (a little known fact that it is) also possible to run Isabelle from

the command line, passing in a theory file with a `lemma` to solve. Finally, Isabelle theory files can include ML code to be executed when the file is processed. These three features have been combined to implement a fully automatic Isabelle, using the nine tactics `simp`, `blast`, `auto`, `metis`, `fast`, `fastsimp`, `best`, `force`, and `meson`. The TPTP2X Isabelle format module outputs a THF problem in Isabelle HOL syntax, augmented with ML code that (i) runs the nine tactics in sequence, each with a CPU time limit, until one succeeds or all fail, and (ii) reports the result and proof (if found) using the SZS standards. A `perl` script is used to insert the CPU time limit (equally divided over the nine tactics) into TPTP2X's Isabelle format output, and then run the command line `isabelle-process` on the resulting theory file. The complete system is named IsabelleP in Table 2.

While it was probably never intended to use Isabelle as a fully automatic system, this simple automation provides useful capability. It solves 74 problems that neither LEO-II nor Tps can solve. The strategy scheduling is effective, with eight of the modes contributing solutions. Over 400 of the 594 solutions are found by one of the first two tactics used - `simp` or `blast`, and more than another 100 by one of the next three tactics - `auto`, `metis`, or `fast`. Further research and development of this automated Isabelle will inevitably lead to improved performance.

The ability of Isabelle to find (counter-)models using the `refute` command has also been integrated into an automatic system, called IsabelleM in Table 2. This provides the TPTP with capability to confirm the satisfiability of axiom sets, and the countersatisfiability of non-theorems. It has been useful for exposing errors in some THF problem encodings. It is planned to extend IsabelleM to also use the (the newly developed) `nitpick` command for model finding.

5 Cunning Plans for the Future

CASC: The CADE ATP System Competition (CASC) [37] is held annually at each CADE (or IJCAR, of which CADE is a constituent) conference. CASC evaluates the performance of sound, fully automatic, ATP systems – it is the world championship for such systems. CASC has been a catalyst for impressive improvements in ATP, stimulating both theoretical and implementation advances [26]. The addition of a THF division to CASC is planned as a natural way to provide the same stimulation for the development of higher-order ATP systems. The first THF division of CASC will be part of CASC-22 at CADE-22. While the primary purpose of CASC is a public evaluation of the relative capabilities of ATP systems, it is important that the THF division should strongly focus on the other aims of CASC: to stimulate ATP research in general, to stimulate ATP research towards autonomous systems, to motivate implementation of robust ATP systems, to provide an inspiring environment for personal interaction between ATP researchers, and to expose ATP systems within and beyond the ATP community.

THF and THFX: Currently the TPTP contains problems in only the core THF0 fragment of the THF language. As ATP developers and users adopt the

language, it is anticipated that demand for the richer features of the full THF language and the extended THFX language will quickly emerge. In preparation for this demand the THF and THFX languages have already been defined, problems in these languages are being collected, and TPTP infrastructure for processing these problems is being developed. Thus the higher-order TPTP expects to be able to meet the expectations of the community, hence encouraging uptake of the THF language and use of the TPTP problems as a common basis for system evaluation.

6 Conclusion

This paper has described the significant practical progress that has been made towards developing the TPTP and associated infrastructure for automated reasoning in higher-order logic. An alpha-release of the TPTP (v3.6.0) with higher-order problems was made on 25th December 2008 (a Christmas present), a beta-release (v3.7.0) was made on 8th March 2009, and the first full release (v4.0.0) will be made in August 2009.

The core work of collecting THF problems is proceeding. Significant new contributions have come from the export of the TPS problem library [4] to THF0, and from a higher-order encoding [10] of problems from the Intuitionistic Logic Theorem Proving (ILTP) library [29]. TPTP v4.0.0 will have over 2500 THF problems.

Current work on the TPTP infrastructure is extending the Java parser for the TPTP language to read the THF language. This in turn will allow use of Java based tools, e.g., IDV [39], for manipulating THF data. The semantic derivation verifier GDV [33] is being updated to verify proofs that include formulae in the THF language.

A key goal of this work is to stimulate the development of ATP systems for higher-order logic - there are many potential applications for such systems. ATP systems that output proofs are particularly important, allowing proof verification. In the long term we hope to see burgeoning research and development of ATP for higher-order logic, with a richness similar to first-order ATP, with many ATP systems, common usage in applications, meta-systems, etc.

Acknowledgments: Thanks to Florian Rabe for help with the Twelf and OmDoc TPTP2X output formats, and to Lucas Dixon and Stefan Berghofer for help with the Isabelle format. Thanks to Florian Rabe for implementing the Twelf-based type checking. Thanks to Makarius Wenzel and Stefan Berghofer for writing the IsabelleP code. Thanks to Jasmin Blanchette for explaining how to implement IsabelleM. Thanks to Andrei Tchaltsev and Alexandre Riazanov for developing the first-order parts of the Java parser for the TPTP language.

References

1. P. B. Andrews. Resolution in Type Theory. *Journal of Symbolic Logic*, 36(3):414–432, 1971.

2. P. B. Andrews. Theorem Proving via General Matings. *Journal of the ACM*, 28(2):193–214, 1981.
3. P. B. Andrews. On Connections and Higher-Order Logic. *Journal of Automated Reasoning*, 5(3):257–291, 1989.
4. P. B. Andrews and C. E. Brown. TPS: A Hybrid Automatic-Interactive System for Developing Proofs. *Journal of Applied Logic*, 4(4):367–395, 2006.
5. D. Aspinall. Proof General: A Generic Tool for Proof Development. In S. Graf and M. Schwartzbach, editors, *Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, number 1785 in Lecture Notes in Computer Science, pages 38–42. Springer-Verlag, 2000.
6. C. Benzmüller and C. E. Brown. A Structured Set of Higher-Order Problems. In J. Hurd and T. Melham, editors, *Proceedings of the 18th International Conference on Theorem Proving in Higher Order Logics*, number 3606 in Lecture Notes in Artificial Intelligence, pages 66–81. Springer-Verlag, 2005.
7. C. Benzmüller, C. E. Brown, and M. Kohlhase. Higher-order Semantics and Extensionality. *Journal of Symbolic Logic*, 69(4):1027–1088, 2004.
8. C. Benzmüller and M. Kohlhase. LEO - A Higher-Order Theorem Prover. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction*, number 1421 in Lecture Notes in Artificial Intelligence, pages 139–143. Springer-Verlag, 1998.
9. C. Benzmüller and L. Paulson. Exploring Properties of Normal Multimodal Logics in Simple Type Theory with LEO-II. In C. Benzmüller, C. E. Brown, J. Siekmann, and R. Statman, editors, *Reasoning in Simple Type Theory: Festschrift in Honour of Peter B. Andrews on his 70th Birthday*, number 17 in Studies in Logic, Mathematical Logic and Foundations. College Publications, 2009.
10. C. Benzmüller and L. Paulson. Exploring Properties of Propositional Normal Multimodal Logics and Propositional Intuitionistic Logics in Simple Type Theory. *Journal of Symbolic Logic*, Submitted.
11. C. Benzmüller, L. Paulson, F. Theiss, and A. Fietzke. Progress Report on LEO-II - An Automatic Theorem Prover for Higher-Order Logic. In K. Schneider and J. Brandt, editors, *Proceedings of the 20th International Conference on Theorem Proving in Higher Order Logics*, pages 33–48, 2007.
12. C. Benzmüller, L. Paulson, F. Theiss, and A. Fietzke. LEO-II - A Cooperative Automatic Theorem Prover for Higher-Order Logic. In P. Baumgartner, A. Armando, and D. Gilles, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning*, number 5195 in Lecture Notes in Artificial Intelligence, pages 162–170. Springer-Verlag, 2008.
13. C. Benzmüller, F. Rabe, and G. Sutcliffe. THF0 - The Core TPTP Language for Classical Higher-Order Logic. In P. Baumgartner, A. Armando, and D. Gilles, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning*, number 5195 in Lecture Notes in Artificial Intelligence, pages 491–506. Springer-Verlag, 2008.
14. C. Benzmüller, V. Sorge, M. Jamnik, and M. Kerber. Combined Reasoning by Automated Cooperation. *Journal of Applied Logic*, 6(3):To appear, 2008.
15. W. Bibel. On Matrices with Connections. *Journal of the ACM*, 28(4):633–645, 1981.
16. M. Bishop. A Breadth-First Strategy for Mating Search. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 359–373. Springer-Verlag, 1999.

17. C. E. Brown. Dependently Typed Set Theory. Technical Report SWP-2006-03, Saarland University, 2006.
18. C. E. Brown. *Automated Reasoning in Higher-Order Logic: Set Comprehension and Extensionality in Church's Type Theory*. Number 10 in Studies in Logic: Logic and Cognitive Systems. College Publications, 2007.
19. C. E. Brown. M-Set Models. In C. Benzmüller, C. E. Brown, J. Siekmann, and R. Statman, editors, *Reasoning in Simple Type Theory: Festschrift in Honour of Peter B. Andrews on his 70th Birthday*, number 17 in Studies in Logic, Mathematical Logic and Foundations. College Publications, 2009.
20. D. Garg. Principal-Centric Reasoning in Constructive Authorization Logic. Technical Report CMU-CS-09-120, School of Computer Science, Carnegie Mellon University, 2009.
21. G. Huet. A Unification Algorithm for Typed Lambda-Calculus. *Theoretical Computer Science*, 1(1):27–57, 1975.
22. J. Hurd. First-Order Proof Tactics in Higher-Order Logic Theorem Provers. In M. Archer, B. Di Vito, and C. Munoz, editors, *Proceedings of the 1st International Workshop on Design and Application of Strategies/Tactics in Higher Order Logics*, number NASA/CP-2003-212448 in NASA Technical Reports, pages 56–68, 2003.
23. M. Kohlhase. *OMDoc - An Open Markup Format for Mathematical Documents [version 1.2]*. Number 4180 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 2006.
24. E. Landau. *Grundlagen der Analysis*. Akademische Verlagsgesellschaft M.B.H, 1930.
25. D. Miller. A Compact Representation of Proofs. *Studia Logica*, 46(4):347–370, 1987.
26. R. Nieuwenhuis. The Impact of CASC in the Development of Automated Deduction Systems. *AI Communications*, 15(2-3):77–78, 2002.
27. T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Number 2283 in Lecture Notes in Computer Science. Springer-Verlag, 2002.
28. F. Pfenning and C. Schürmann. System Description: Twelf - A Meta-Logical Framework for Deductive Systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 202–206. Springer-Verlag, 1999.
29. T. Raths, J. Otten, and C. Kreitz. The ILTP Problem Library for Intuitionistic Logic - Release v1.1. *Journal of Automated Reasoning*, 38(1-2):261–271, 2007.
30. A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.
31. S. Schulz. E: A Brainiac Theorem Prover. *AI Communications*, 15(2-3):111–126, 2002.
32. J. Siekman, C. Benzmüller, and S. Autexier. Computer Supported Mathematics with OMEGA. *Journal of Applied Logic*, 4(4):533–559, 2006.
33. G. Sutcliffe. Semantic Derivation Verification. *International Journal on Artificial Intelligence Tools*, 15(6):1053–1070, 2006.
34. G. Sutcliffe. TPTP, TSTP, CASC, etc. In V. Diekert, M. Volkov, and A. Voronkov, editors, *Proceedings of the 2nd International Computer Science Symposium in Russia*, number 4649 in Lecture Notes in Computer Science, pages 7–23. Springer-Verlag, 2007.
35. G. Sutcliffe. The SZS Ontologies for Automated Reasoning Software. In G. Sutcliffe, P. Rudnicki, R. Schmidt, B. Konev, and S. Schulz, editors, *Proceedings of the*

LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and The 7th International Workshop on the Implementation of Logics, number 418 in CEUR Workshop Proceedings, pages 38–49, 2008.
36. G. Sutcliffe, S. Schulz, K. Claessen, and A. Van Gelder. Using the TPTP Language for Writing Derivations and Finite Interpretations. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, number 4130 in Lecture Notes in Artificial Intelligence, pages 67–81, 2006.
37. G. Sutcliffe and C. Suttner. The State of CASC. *AI Communications*, 19(1):35–48, 2006.
38. G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.
39. S. Trac, Y. Puzis, and G. Sutcliffe. An Interactive Derivation Viewer. In S. Autexier and C. Benzmüller, editors, *Proceedings of the 7th Workshop on User Interfaces for Theorem Provers, 3rd International Joint Conference on Automated Reasoning*, volume 174 of *Electronic Notes in Theoretical Computer Science*, pages 109–123, 2006.
40. L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the AUTOMATH System*. PhD thesis, Eindhoven University, Eindhoven, The Netherlands, 1979.
41. A. Van Gelder and G. Sutcliffe. Extending the TPTP Language to Higher-Order Logic with Automated Parser Generation. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, number 4130 in Lecture Notes in Artificial Intelligence, pages 156–161. Springer-Verlag, 2006.
42. C. Weidenbach, R. Schmidt, T. Hillenbrand, R. Rusev, and D. Topic. SPASS Version 3.0. In F. Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction*, number 4603 in Lecture Notes in Artificial Intelligence, pages 514–520. Springer-Verlag, 2007.
43. F. Wiedijk. *The Seventeen Provers of the World*. Number 3600 in Lecture Notes in Computer Science. Springer-Verlag, 2006.