The TPTP World – Infrastructure for Automated Reasoning

Geoff Sutcliffe

University of Miami, USA

Abstract. The TPTP World is a well known and established infrastructure that supports research, development, and deployment of Automated Theorem Proving (ATP) systems for classical logics. The data, standards, and services provided by the TPTP World have made it increasingly easy to build, test, and apply ATP technology. This paper reviews the core features of the TPTP World, describes key service components of the TPTP World, presents some successful applications, and gives an overview of the most recent developments.

1 Introduction

Automated Theorem Proving (ATP) is concerned with the development and use of computer programs that automate sound reasoning: the derivation of conclusions that follow inevitably from facts. The dual discipline, automated model finding, develops computer programs that establish that a set of statements is consistent, and in this work we consider automated model finding to be part of ATP. These capabilities lie at the heart of many important computational tasks, e.g., formal methods for software and hardware design and verification, the analysis of network security protocols, solving hard problems in mathematics, and inference for the semantic web. High performance ATP systems (for logics supported in the TPTP World) include E/EP [16], LEO-II [1], Paradox [3], SPASS [33], Vampire [14], and Waldmeister [7].

The TPTP World is a well known and established infrastructure that supports research, development, and deployment of Automated Theorem Proving (ATP) systems for classical logics. The TPTP World includes the TPTP problem library, the TSTP solution library, standards for writing ATP problems and reporting ATP solutions, tools for processing ATP problems and solutions, and harnesses for controlling the execution of ATP systems and tools. The TPTP World infrastructure has been deployed in a range of applications, in both academia and industry. Section 2 of this paper reviews the core of the TPTP World, Section 3 describes key service components of the TPTP World, Section 4 presents some applications, and Section 5 gives an overview of the most recent developments in the TPTP World. Section 6 concludes.

E.M. Clarke and A. Voronkov (Eds.): LPAR-16, LNAI 6355, pp. 1–12, 2010.

2 The TPTP World Core Infrastructure

The Thousands of Problems for Theorem Provers (TPTP) problem library [20] is the de facto standard set of test problems for Automated Theorem Proving (ATP) systems for classical logics.[1] It is the original core component of the TPTP World, and is commonly referred to as "the TPTP". The TPTP problem library supplies the ATP community with a comprehensive library of the test problems that are available today, in order to provide an overview and a simple, unambiguous reference mechanism, to support the testing and evaluation of ATP systems, and to help ensure that performance results accurately reflect capabilities of the ATP systems being considered. The current TPTP (v4.0.1) has forty-one domains, in the fields of logic, mathematics, computer science, science and engineering, and social sciences. Each TPTP problem file has a header section that contains information for the user, and the logical formulae are wrapped with annotations that provide a unique name for each formula in the problem, a user role (axiom, conjecture, etc.), and auxiliary user information. The logical formulae are written in the TPTP language (described below), which has a consistent and easily understood notation. Since its first release in 1993, many researchers have used the TPTP as an appropriate and convenient basis for ATP system evaluation. Over the years the TPTP has also increasingly been used as a conduit for ATP users to provide samples of their problems to ATP system developers – users have found that contributing samples of their problems to the TPTP exposes the problems to the developers, who then improve their systems' performances on the problems, which completes a cycle to provide the users with more effective tools.

One of the keys to the success of the TPTP World is the consistent use of the TPTP language for writing problems and solutions [24], which enables convenient communication between different systems and researchers. The language shares many features with Prolog, a language that is widely known in the ATP community. Indeed, with a few operator definitions, units of TPTP data can be read in Prolog using a single `read`/1 call, and written with a single `writeq`/1 call. A principal goal of the development of the TPTP language grammar was to make it easy to translate the BNF into `lex`/`yacc`/`flex`/`bison` input, so that construction of parsers (in languages other than Prolog) can be a reasonably easy task [32]. The TPTP World services described in Section 3 naturally process data written in the TPTP language. Parsers written in C and Java, and the `lex`/`yacc`/`flex`/`bison` input files, are part of the TPTP World. Many ATP system developers and users have adopted the TPTP language.

In order to precisely specify what is known or has been established about a set of formulae, the TPTP World provides the SZS ontologies [19]. These ontologies provide status and dataform values to describe logical data. For example, a TPTP problem might be tagged as a `Theorem`, a model finder might report that a set of formulae is `Satisfiable`, and a parser might report that a formula contains a `SyntaxError`. The SZS standard also recommends the precise way

[1] Available at `http://www.tptp.org`

in which the ontology values should be presented, in order to facilitate easy processing.

The Thousands of Solutions from Theorem Provers (TSTP) solution library, the "flip side" of the TPTP, is a corpus of ATP systems' solutions to TPTP problems.[2] A major use of the TSTP is for ATP system developers to examine solutions to problems, and thus understand how they can be solved, leading to improvements to their own systems. At the time of writing this paper, the TSTP contained the results of running 54 ATP systems and system variants on all the problems in the TPTP that they can, in principle, attempt to solve (therefore, e.g., finite model finding systems are not run on problems that are known to be unsatisfiable). This has produced over 155000 files for solved problems, of which almost 100000 contain explicit proofs or models (rather than only an assurance of a solution). The first section of each TSTP solution file is a header that contains information about the TPTP problem, information about the ATP system, characteristics of the computer used, the SZS status and output dataform from the system, and statistics about the solution. The second section of each TSTP solution file contains the annotated formulae that make up the solution. A key feature of the TSTP is that solutions from many of the ATP systems are written in the TPTP language - the same language as used for TPTP problems.

An important feature of the TPTP is the problem ratings [28]. The ratings provide a well-defined measure of how difficult the problems are for ATP systems, and how effective the ATP systems are for different types of problems. For rating, the TPTP problems are divided into Specialist Problem Classes (SPCs), and the TSTP files for each SPC are analyzed. The performance of systems whose set of solved problems is not a subset of that of any other system is used to rate the problems. The fraction of such systems that fail on a problem is the difficulty rating for a problem: problems that are solved by all/some/none of the systems get ratings of 0.00/0.01-0.99/1.00, and are referred to as easy/difficult/hard problems respectively. Over time, decreasing ratings for individual problems provide an indication of progress in the field [23]. The analysis done for problem ratings also provides ratings for ATP systems, for each SPC: the rating of a system is the fraction of the difficult problems that it solves.

3 TPTP World Services

The TPTP World includes tools, programming libraries, and online services that are used to support the application and deployment of ATP systems.[3] This section describes a few of the components – there are many more!

SystemOnTPTP [17] is a TPTP World utility that allows an ATP problem or solution to be easily and quickly submitted in various ways to a range of ATP systems and tools. The utility uses a suite of currently available ATP systems and tools, whose properties (input format, reporting of result status, etc.) are stored in a simple text database. The implementation of SystemOnTPTP

[2] Available at `http://www.tptp.org/TSTP`

[3] The tools and libraries are available at `http://www.tptp.org/TPTPWorld.tgz`

uses several subsidiary tools to preprocess the input, control the execution of the ATP systems and tools, and postprocess the output. On the input side TPTP2X or TPTP4X (TPTP World tools for parsing and transforming TPTP format formulae) is used to prepare the input for processing. A program called `TreeLimitedRun` is used to monitor the execution of ATP systems and tools, and limit the CPU time and memory used – `TreeLimitedRun` monitors processes more tightly than is possible with standard operating system calls. Finally a program called `X2tptp` converts an ATP system's output to TPTP format, if requested by the user.

The TPTP World ATP system recommendation service uses the ATP system ratings to recommend ATP systems for solving a new problem. A new problem is analysed to determine its SPCs – one for establishing theoremhood or unsatisfiability, and one for establishing countersatisfiability or satisfiability. The systems that contributed to the problem ratings in those SPCs are recommended, in decreasing order of system rating for the SPCs.

GDV [18] is a TPTP World tool that uses structural and then semantic techniques to verify TPTP format derivations. Structural verification checks that inferences have been done correctly in the context of the derivation, e.g., checking that the derivation is acyclic, checking that assumptions have been discharged, and checking that introduced symbols (e.g., in Skolemization) are distinct. Semantic verification checks the expected semantic relationship between the parents and inferred formula of each inference step. This is done by encoding the expectation as a logical obligation in an ATP problem, and then discharging the obligation by solving the problem with trusted ATP systems. The expected semantic relationship between the parents and inferred formula of an inference step depends on the intent of the inference rule used. For example, deduction steps expect the inferred formula to be a theorem of its parent formulae. The expected relationship is recorded as an SZS value in each inferred formula of a derivation. GDV uses the SystemOnTPTP utility to execute the trusted ATP systems.

AGInTRater is a TPTP World tool that evaluates the interestingness of formulae in TPTP format derivations (AGInTRater is a component of the AGInT system that discovers interesting theorems of a given set of axioms [13]). AGInTRater has a filter that measures up to eight "interestingness" features of formulae (some features are inappropriate in some situations): preprocessing detects and discards obvious tautologies, obviousness estimates the difficulty of proving a formula, weight estimates the effort required to read a formula, complexity estimates the effort required to understand a formula, surprisingness measures new relationships between function and predicate symbols in a formula, intensity measures how much a formula summarizes information from its leaf ancestors, adaptivity measures how tightly the universally quantified variables of a formula are constrained, and focus measures the extent to which a formula is making a positive or negative statement. Formulae that pass the majority of the filters

are passed to a static ranker, which combines the measures from the filters with a measure of usefulness, which measures how much a formula has contributed to proofs of further formulae. The scores are then normalized and averaged to produce an interestingness score. AGInTRater is a unique feature of the IDV derivation visualizer, described next.

IDV [29] is a TPTP World tool for graphical rendering and analysis of TPTP format derivations. IDV provides an interactive interface that allows the user to quickly view features of the derivation, and access analysis facilities. The left hand side of Figure 1 shows the rendering of the derivation output by the EP ATP system, for the TPTP problem `PUZ001+1`. The IDV window is divided into three panes: the top pane contains control buttons and sliders, the middle pane shows the rendered DAG, and the bottom pane gives the text of the annotated formula for the node pointed to by the mouse. The rendering of the derivation DAG uses shapes, colors, and tags to provide information about the derivation. The user can interact with the rendering in various ways using mouse-over and mouse clicks. The buttons and sliders in the control pane provide a range of manipulations on the rendering – zooming, hiding and displaying parts of the DAG, and access to GDV for verification. A particularly novel feature of IDV is its ability to provide a synopsis of a derivation by using the AGInTRater to identify interesting lemmas, and hiding less interesting intermediate formulae. A synopsis is shown on the right hand side of Figure 1. IDV is easily appreciated by using it through the online interfaces described next.

All of the TPTP World services described in this section, and a few more besides, are accessible in web browser interfaces.[4] These interfaces execute most of the services using the SystemOnTPTP utility. The SystemB4TPTP interface provides access to problem pre-processing tools, including parsers, pretty-printers, first-order form to clause normal form conversion, type checking, and axiom relevance measures. It additionally provides a facility to convert a problem to TPTP format from some other formats. The SystemOnTPTP interface provides the core functionality of the SystemOnTPTP utility, to submit an ATP problem to ATP systems. It additionally provides system reports and recommendations for which systems to use on a given problem, based on the SPCs and system ratings. The SystemOnTPTP interface also has direct access to the SSCPA ATP meta-system [25], which runs multiple recommended ATP systems in competition parallel. Finally, the SystemOnTSTP interface provides access to derivation post-processing tools, including parsers, pretty-printers, answer extraction, GDV, AGInTRater, and IDV.

As an alternative to interactive access via a web browser, the TPTP World services can also be accessed programmatically using `http POST` multi-part form requests. Perl and Java code for doing this is available as part of the TPTP World. The benefits of using the TPTP World online service include not installing ATP systems and tools locally, having access to the latest version of ATP systems (e.g., versions from CASC and unpublished versions), and full access to all the TPTP World tools. You are encouraged to abuse my server!

[4] Available starting at `http://www.tptp.org/cgi-bin/SystemOnTPTP`

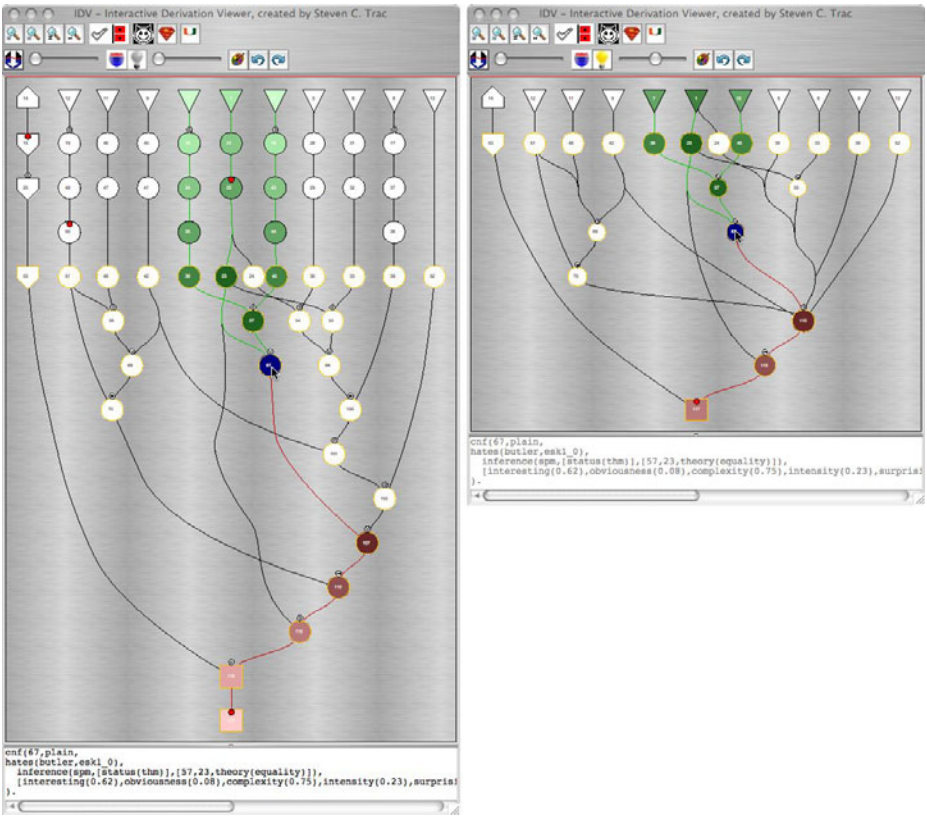

Fig. 1. EP's proof by refutation of PUZ001+1

4 TPTP World Applications

Research scientists in the Robust Software Engineering Group of the Intelligent Systems Division of NASA Ames have developed, implemented, and evaluated a certification approach that uses Hoare-style techniques to formally demonstrate the safety of aerospace programs that are automatically generated from high-level specifications [5]. A verification condition generator processes the automatically generated code, and produces a set of safety obligations in the form of TPTP format problems that are provable if and only if the code is safe. The obligation problems are discharged using the SSCPA ATP meta-system (mentioned in Section 3), to produce TPTP format proofs that are then verified by GDV. The proofs and verification logs serve as safety certificates for authorities like the FAA.

The MPTP project [31] aims to link-up the large formal Mizar Mathematical Library (MML) [15] with ATP technology, and to boost the development of AI-based ATP methods. The MPTP system converts Mizar format problems to an extended TPTP language that adds term-dependent sorts and abstract (Fraenkel) terms to the TPTP syntax. Problems in the extended language are transformed to standard TPTP format by relativization of sorts and deanonymization of abstract terms. Finding proofs for these problems provides cross verification of the underlying Mizar proofs. Mizar proofs are also exported as TPTP format derivations,[5] allowing a number of ATP experiments and use of TPTP tools, e.g., GDV and IDV.

Sledgehammer [12] is a linkup from the interactive theorem prover Isabelle/HOL [11] to first-order ATP systems. Sledgehammer is activated by the user, who wishes to prove a goal from the current background theory. A set of relevant facts is extracted from the background theory – this set is almost inevitably a superset of what is required for a proof of the goal. The goal and background facts are translated into a TPTP format first-order logic problem, which is given to one or more ATP systems. If one of the ATP systems finds a proof, the axioms used in the proof are extracted, and the goal is reproved using the Metis ATP system [8], which natively outputs an Isabelle/HOL proof. One of the options within Sledgehammer is to use the TPTP World service to run ATP systems; in particular, this is done for running the Vampire ATP system.

The Naproche (NAtural language PROof CHEcking) system [4] automatically checks mathematical texts (written in the Naproche controlled natural language) for logical correctness. Each statement of a text is required to follow from preceding information in the text. The text is first translated into a linguistic representation called a Proof Representation Structure, from which TPTP format proof obligations are created. The proof obligations are discharged by calling an ATP system, using the TPTP World service. The TPTP format proofs produced by the ATP system are used to create the Naproche output, and also analyzed (using another TPTP World tool) to help select what preceding information should be provided in subsequent proof obligations. IDV is also used to debug the Naproche system.

SPASS-XDB [26] is a modified version of the SPASS ATP system, with the ability to retrieve world knowledge axioms from a range of external sources (databases, internet, computations, etc.) asynchronously, on demand, during it's deduction. Figure 2 shows the system architecture, which is comprised of the SPASS-XDB ATP system, mediators, and external sources of world knowledge axioms. SPASS-XDB accepts a problem file containing (i) specifications for the external sources, (ii) internal axioms, and (iii) a conjecture to be proved. All the formulae are written in the TPTP language, and commonly axioms from a TPTP format export of the SUMO ontology [10] are provided as internal axioms that allow deep reasoning over the world knowledge. SPASS-XDB augments SPASS' classic CNF saturation algorithm with steps to request and accept world knowledge axioms. Requests are written as TPTP format "questions", and axioms

[5] Available at `http://www.tptp.org/MizarTPTP`

are delivered as TPTP format "answers", using SZS standards. The requests are made and the axioms delivered asynchronously, so that SPASS-XDB continues its deduction process while the axioms are being retrieved from the (comparatively slow) external sources. SPASS-XDB is available in the SystemOnTPTP interface.

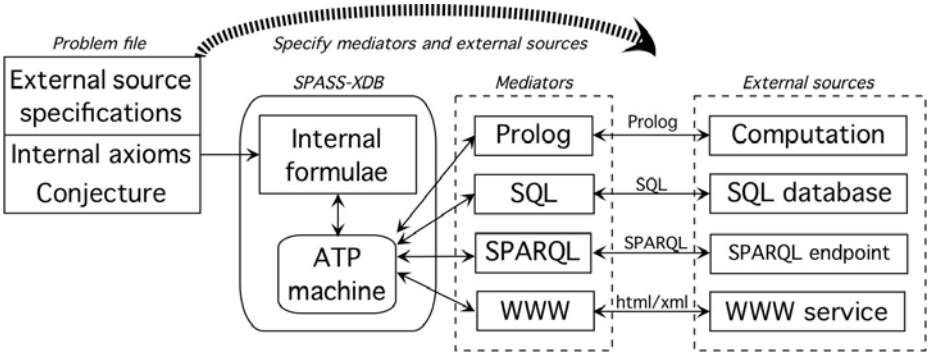

Fig. 2. System Architecture

The CADE ATP System Competition (CASC) [27] is held annually to evaluate the performance of sound, fully automatic ATP systems – it is the world championship for such systems. The design and implementation of CASC is closely linked to the TPTP World. The divisions and problem categories of CASC are similar to the SPCs used in the TPTP problem and system rating scheme. The problems used in CASC are taken from the TPTP problem library, and the TPTP problem ratings are used to select appropriately difficult problems that differentiate between the systems entered into CASC. CASC has been a catalyst for impressive improvements in ATP, stimulating both theoretical and implementation advances [9]. The positive effects of CASC on ATP system development have had reciprocal positive effects on the TPTP. Observers at the event have been encouraged to contribute their problems to the TPTP. The ATP systems entered into CASC are the most recent versions available, and after CASC they are added to the SystemOnTPTP suite. The systems are run over the TPTP problem library to update the TSTP solution library, which in turn provides updated problem and system ratings.

5 Current Developments in the TPTP World

The TPTP (problem library, and later World) was originally developed for untyped first-order logic. In 2008-9 the Typed Higher-order Form (THF) was added, and the first release of the TPTP problem library with THF problems was in 2009 [21]. The TPTP THF language is a syntactically conservative extension of the

untyped first-order language. It has been divided into three layers named THF0, THF, and THFX. THF0 [2] is a core subset based on Church's simply typed lambda calculus. THF provides a richer type system, the ability to reason about types, more term and type constructs, and more connectives. THFX provides "syntactic sugar" that is usefully expressive. In conjunction with the addition of THF problems to the TPTP problem library, other components of the TPTP World were extended to support THF. This includes parsers, pretty-printers, type checkers, export to existing higher-order ATP systems' formats, and proof presentation in IDV. Additionally, four theorem proving ATP systems and two model finders have been produced, and are available in SystemOnTPTP. The addition of THF to the TPTP World has had an immediate impact on progress in the development of automated reasoning in higher-order logic [22].

Following the addition of THF to the TPTP World, steps are now being taken to add support for a Typed First-order Form (TFF) in the TPTP World. The TPTP TFF language is (like the THF language) a syntactically conservative extension of the untyped first-order language. TFF problems are being collected,[6] and a TPTP release including TFF problems is expected in 2010. An ATP system has been built to solve these problems, by translating the problems to an equivalent first-order form and calling an existing first-order ATP system.

As well as being useful in it own right, TFF also provides a foundation for adding support for arithmetic in the TPTP World. For arithmetic, the TFF language includes atomic types for integer, rational and real numbers, with corresponding forms for constants. A core set of predicates and functions for arithmetic relations and operations has been defined. The extent to which ATP systems can do arithmetic is expected to vary, from a simple ability to evaluate ground terms, through an ability to instantiate variables in arithmetic expressions, to extensive algebraic manipulations. Five ATP systems have been configured for solving TFF arithmetic problems (although none are yet able to deal with all the TFF arithmetic constructs). ATP systems have been notorious for their lack of arithmetic capabilities, and it is expected that this development in the TPTP World will provide useful infrastructure that will help developers add arithmetic to their ATP systems, and consequently provide new functionality to ATP system users.

A long time challenge of artificial intelligence has been to provide a system that is capable of reasoning with world knowledge, with a natural language interface [30]. As a small step towards that goal in the TPTP World, SystemB4TPTP now supports input in Attempto Controlled English (ACE) [6]. ACE input is translated to first-order logic by an online service at the University of Zurich, and the TPTP format axioms and conjecture that are returned can be submitted to an ATP system. A modified version of the translation is being developed, which adds axioms that link the predicate and function symbols used in the translated formulae to those used in SUMO and the sources of external data available to SPASS-XDB. This will allow SPASS-XDB to be used, to provide deep reasoning with world knowledge.

[6] Available at `http://www.tptp.org/TPTP/Proposals/SampleProblems/TFF`

6 Conclusion

The TPTP World is a well known and established infrastructure that supports research, development, and deployment of Automated Theorem Proving (ATP) systems for classical logics. This paper has given an overview of the TPTP World, and shown how it can be successfully applied. Recent developments in the TPTP World are expected to broaden the user base of ATP system developers, and attract new ATP system users.

The TPTP problem library is a core component of the TPTP World, and users of the TPTP World are encouraged to contribute their problems to the library. The common adoption of TPTP standards, especially the TPTP language and SZS ontology, has made it easier to build complex reasoning systems. Developers are encouraged to adopt the TPTP standards, to provide compliant components that can be combined to meet users' needs.

References

1. Benzmüller, C., Paulson, L., Theiss, F., Fietzke, A.: LEO-II - A Cooperative Automatic Theorem Prover for Higher-Order Logic. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 162–170. Springer, Heidelberg (2008)
2. Benzmüller, C., Rabe, F., Sutcliffe, G.: THF0 - The Core TPTP Language for Classical Higher-Order Logic. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 491–506. Springer, Heidelberg (2008)
3. Claessen, K., Sörensson, N.: New Techniques that Improve MACE-style Finite Model Finding. In: Baumgartner, P., Fermueller, C. (eds.) Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications (2003)
4. Cramer, M., Fisseni, B., Koepke, P., Kühlwein, D., Schröder, B., Veldman, J.: The Naproche Project: Controlled Natural Language Proof Checking of Mathematical Texts. In: Fuchs, N.E. (ed.) CNL 2009 Workshop. LNCS, vol. 5972, pp. 170–186. Springer, Heidelberg (2010)
5. Denney, E., Fischer, B., Schumann, J.: Using Automated Theorem Provers to Certify Auto-generated Aerospace Software. In: Basin, D., Rusinowitch, M. (eds.) IJCAR 2004. LNCS (LNAI), vol. 3097, pp. 198–212. Springer, Heidelberg (2004)
6. Fuchs, N., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. LNCS, vol. 5224, pp. 104–124. Springer, Heidelberg (2008)
7. Hillenbrand, T.: Citius altius fortius: Lessons Learned from the Theorem Prover Waldmeister. In: Dahn, I., Vigneron, L. (eds.) Proceedings of the 4th International Workshop on First-Order Theorem Proving. Electronic Notes in Theoretical Computer Science, vol. 86.1, pp. 1–13 (2003)
8. Hurd, J.: First-Order Proof Tactics in Higher-Order Logic Theorem Provers. In: Archer, M., Di Vito, B., Munoz, C. (eds.) 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, pp. 56–68 (2003)

9. Nieuwenhuis, R.: The Impact of CASC in the Development of Automated Deduction Systems. AI Communications 15(2-3), 77–78 (2002)
10. Niles, I., Pease, A.: Towards A Standard Upper Ontology. In: Welty, C., Smith, B. (eds.) Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, pp. 2–9 (2001)
11. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. LNCS, vol. 2283. Springer, Heidelberg (2002)
12. Paulson, L., Susanto, K.: Source-level Proof Reconstruction for Interactive Theorem Proving. In: Schneider, K., Brandt, J. (eds.) TPHOLs 2007. LNCS, vol. 4732, pp. 232–245. Springer, Heidelberg (2007)
13. Puzis, Y., Gao, Y., Sutcliffe, G.: Automated Generation of Interesting Theorems. In: Sutcliffe, G., Goebel, R. (eds.) Proceedings of the 19th International FLAIRS Conference, pp. 49–54. AAAI Press, Menlo Park (2006)
14. Riazanov, A., Voronkov, A.: The Design and Implementation of Vampire. AI Communications 15(2-3), 91–110 (2002)
15. Rudnicki, P.: An Overview of the Mizar Project. In: Proceedings of the 1992 Workshop on Types for Proofs and Programs, pp. 311–332 (1992)
16. Schulz, S.: E: A Brainiac Theorem Prover. AI Communications 15(2-3), 111–126 (2002)
17. Sutcliffe, G.: SystemOnTPTP. In: McAllester, D. (ed.) CADE 2000. LNCS, vol. 1831, pp. 406–410. Springer, Heidelberg (2000)
18. Sutcliffe, G.: Semantic Derivation Verification. International Journal on Artificial Intelligence Tools 15(6), 1053–1070 (2006)
19. Sutcliffe, G.: The SZS Ontologies for Automated Reasoning Software. In: Sutcliffe, G., Rudnicki, P., Schmidt, R., Konev, B., Schulz, S. (eds.) Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and The 7th International Workshop on the Implementation of Logics. CEUR Workshop Proceedings, vol. 418, pp. 38–49 (2008)
20. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure. The FOF and CNF Parts, v3.5.0. Journal of Automated Reasoning 43(4), 337–362 (2009)
21. Sutcliffe, G., Benzmüller, C.: Automated Reasoning in Higher-Order Logic using the TPTP THF Infrastructure. Journal of Formalized Reasoning 3(1), 1–27 (2010)
22. Sutcliffe, G., Benzmüller, C., Brown, C.E., Theiss, F.: Progress in the Development of Automated Theorem Proving for Higher-order Logic. In: Schmidt, R.A. (ed.) Automated Deduction – CADE-22. LNCS, vol. 5663, pp. 116–130. Springer, Heidelberg (2009)
23. Sutcliffe, G., Fuchs, M., Suttner, C.: Progress in Automated Theorem Proving, 1997-1999. In: Hoos, H., Stützle, T. (eds.) Proceedings of the IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, pp. 53–60 (2001)
24. Sutcliffe, G., Schulz, S., Claessen, K., Van Gelder, A.: Using the TPTP Language for Writing Derivations and Finite Interpretations. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 67–81. Springer, Heidelberg (2006)
25. Sutcliffe, G., Seyfang, D.: Smart Selective Competition Parallelism ATP. In: Kumar, A., Russell, I. (eds.) Proceedings of the 12th International FLAIRS Conference, pp. 341–345. AAAI Press, Menlo Park (1999)
26. Sutcliffe, G., Suda, M., Teyssandier, A., Dellis, N., de Melo, G.: Progress Towards Effective Automated Reasoning with World Knowledge. In: Murray, C., Guesgen, H. (eds.) Proceedings of the 23rd International FLAIRS Conference. AAAI Press, Menlo Park (2010) (to appear)

27. Sutcliffe, G., Suttner, C.: The State of CASC. AI Communications 19(1), 35–48 (2006)
28. Sutcliffe, G., Suttner, C.B.: Evaluating General Purpose Automated Theorem Proving Systems. Artificial Intelligence 131(1-2), 39–54 (2001)
29. Trac, S., Puzis, Y., Sutcliffe, G.: An Interactive Derivation Viewer. In: Autexier, S., Benzmüller, C. (eds.) Proceedings of the 7th Workshop on User Interfaces for Theorem Provers, 3rd International Joint Conference on Automated Reasoning. Electronic Notes in Theoretical Computer Science, vol. 174, pp. 109–123 (2006)
30. Turing, A.: Computing Machinery and Intelligence. Mind 59(236), 433–460 (1950)
31. Urban, J.: MPTP 0.2: Design, Implementation, and Initial Experiments. Journal of Automated Reasoning 37(1-2), 21–43 (2006)
32. Van Gelder, A., Sutcliffe, G.: Extending the TPTP Language to Higher-Order Logic with Automated Parser Generation. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 156–161. Springer, Heidelberg (2006)
33. Weidenbach, C., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P., Dimova, D.: SPASS Version 3.5. In: Schmidt, R.A. (ed.) Automated Deduction – CADE-22. LNCS, vol. 5663, pp. 140–145. Springer, Heidelberg (2009)