Communication Formalisms for Automated Theorem Proving Tools

Geoff Sutcliffe
Department of Computer Science
University of Miami
geoff@cs.miami.edu

Jürgen Zimmer
Fachbereich Informatik
Universität des Saarlandes
jzimmer@ags.uni-sb.de

Stephan Schulz
Institüt für Informatik
Technische Universität München
schulz@informatik.tu-muenchen.de

Abstract

This paper describes two communication formalisms for Automated Theorem Proving (ATP) tools. First, a problem and solution language has been designed. The language will be used for writing problems to be input to ATP systems, and for writing solutions output by ATP systems. Second, a hierarchy of result statuses, which adequately express the range of results output by ATP systems, has been established. These formalisms will support application and research in ATP, and will facilitate direct communication between ATP tools when they are used as embedded components in larger systems.

1 Introduction

Automated Theorem Proving (ATP) deals with the development of computer programs that show that some statement (the conjecture) is a logical consequence of a set of statements (the axioms). ATP systems are used in a wide variety of domains: problems in mathematics have been solved, e.g., [Slaney *et al.*, 1995; McCune, 1997], software and hardware have been designed and verified, e.g., [Whalen *et al.*, 2002; Claessen *et al.*, 2002], and applications to the WWW seem possible [Horrocks and Sattler, 2001]. ATP has been highly successful when the problem is expressed in classical 1st order logic, so that a proof by refutation of the clause normal form of the problem can be obtained. There are some well known high performance ATP systems that search for a refutation of a set of clauses, e.g., Gandalf [Tammet, 1997], SPASS [Weidenbach *et al.*, 2002], E [Schulz, 2002], Vampire [Riazanov and Voronkov, 2002]. This paper presents two communication formalisms in the context of ATP systems for classical 1st order logic. However, the design principles used are suitable for other logics, and it would be desirable to adapt or extend these formalisms to provide for communication between 1st order ATP systems and systems for other logics, e.g., Coq [Coq, 2003], HOL [HOL, 2003], ACL2 [ACL, 2003].

The success of ATP systems is in large part attributable to progress in ATP research, four important aspects of which are outlined in Figure 1. The rest of Figure 1 shows lines of dependence between some other issues, indicating that ATP applications are dependent on ATP research, and that ATP research is dependent on many interlinked topics. The subpaths in the diagram that particularly motivate the work described here are:

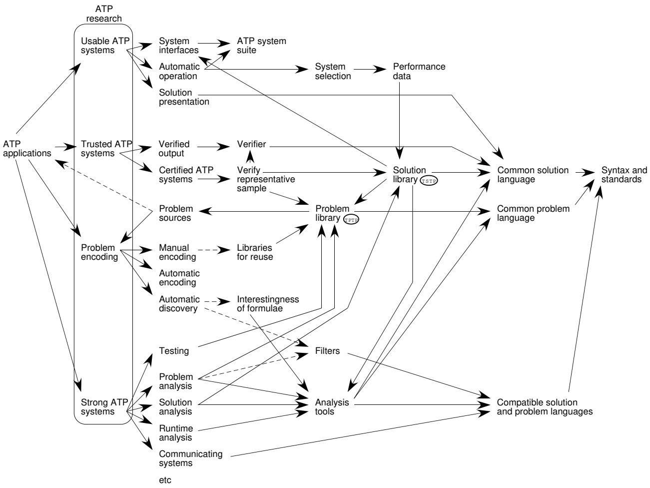
- From the need for a *Problem library* for testing and evaluation, to the need for a *Common problem language*, to the need for *Syntax and standards*.
- From the need for a *Solution library* for analysis and development, to the need for a *Common solution language*, to the need for *Syntax and standards*.
- From the demand for series of *Filters*, common *Analysis tools*, and *Communicating systems*, to the need for *Compatible solution and problem languages*, to the need for *Syntax and standards*.

These paths motivated the creation of two formalisms for ATP. First, a problem and solution language has been designed. The language will be used for writing problems to be input to ATP systems, and for writing solutions output by ATP systems. Second, a hierarchy of result statuses, which adequately express the range of results output by ATP systems, has been established. Details of these formalisms are given in Sections 2 and 3 respectively.

2 The TSTP Problem and Solution Language

Several different syntaxes exist for writing the problems that are input to ATP systems. The Knowledge Interchange Format (KIF) [Genesereth and Fikes, 1992] is a logically comprehensive language for the for the representation of knowledge, and has declarative semantics. KIF is LISP like, and provides expressive power beyond that required for ATP. The "DFG" syntax [Hähnle *et al.*, 1996] was designed as a common exchange format for logic problems used by members of the German DFG-Schwerpunktprogramm Deduction. DFG has a prefix-style grammar that is neither particularly easy for programs to parse nor for humans to read and write. Moreover, despite its aim, it is not very widely used. The Common Logic (CL) syntax [CL-, 2003] is a framework for a family of logic-based languages. It does not specify any concrete syntax, but rather specifies an abstract syntax that can be specialized to a concrete language. CL grew out of work on the KIF language. The OmDoc [Kohlhase, 2000], OpenMath [Caprotti and Carlisle, 1999], and MathML [Caprotti

Figure 1: Motivation Paths for Communication Standards



and Carlisle, 1999] languages specify XML based syntaxes for writing mathematical notions. These languages are quite expressive, but require a large amount of mark-up for quite simple content. As a result, reading and writing problems written in these languages is difficult without specialized software. The syntax used for problems in the TPTP problem library [Sutcliffe and Suttner, 1998] is widely used in the ATP community. It has a simple syntax, but has the weakness that the Clause Normal Form (CNF) syntax is not a subsyntax of the First Order Form (FOF) syntax (while CNF formulae are a subset of FOF formulae).

All of the above syntaxes were designed for expressing the *input* to logical reasoning tools. There appear to be no consistently used formalisms for *output* from reasoning tools. Although many of the above could be reasonably used for writing the logical formulae output by reasoning tools, none were designed or contain features specific for comprehensively capturing output information. To provide seamless communication between reasoning tools it is necessary that the output from one tool should immediately be suitable as input for other tools.

The goal of designing a language for communication between reasoning tools cannot ignore the requirement of human readability. Humans are often the source of initial input and the destination of final output. It is thus necessary that the input and output be human readable. In an ideal world it would not be necessary for humans to look at intermediate results being passed between tools, but in reality close examination of intermediate data is necessary for debugging, understanding of system behaviour, and development of ideas [Wos and Pieper, 1999]. Different users have different needs for reading and writing the input, intermediate, and output data. It may not be possible to design a language that perfectly suits all the needs, but certainly a syntax that is designed for machine processing and ignores human readability is unlikely meet any of those needs (a classic case, in one author's opinion, is XML based syntaxes). One way around this issue is to provide two languages, one for human interfaces and one for machine interfaces. This approach, however, requires extra software support, and the necessary translations may hide relevant details.

The TSTP syntax is an outgrowth of the TPTP syntax. It is a comprehensive syntax, suitable for writing the input and output of ATP tools. The TSTP syntax was designed with the following aims and constraints:

- It must be able to completely express the problems that are input to ATP systems.
- It must be able to capture sufficient details of ATP systems' outputs to allow presentation and other forms of postprocessing, e.g., various styles of proof verification.
- The same syntax must be used for both input and output.
- It must be possible to annotate formulae with arbitrary information.
- It must be easy for humans to read and write.
  - It must be easy to write using a plain text editor.
  - It must be compact.
- It must be easy for programs to parse.
- It should be backward compatible with the TPTP syntax, but a single syntax must be used for CNF and FOF formulae.
- It must be extensible, to allow for expression of new types of formulae.
- It need have only local context and semantics, i.e., the syntax need not support universal denotation, as in, e.g., the semantic web[1].

An *annotated TSTP formula* has the following structure (where items in <>s are placeholders for specific values, and items in [ ]s are optional):[2]

```
<language>(<name>,<type>[-<subtype>],
    <formula>[,
    <source>[,
    <useful info>]]).
```

An example of a FOF input formula in TSTP syntax is:

```
fof(subclass_defn,axiom,
    ! [X,Y] :
      ( subclass(X,Y)
    <=> ! [U] :
          ( member(U,X)
         => member(U,Y) ) ),
    file('SET005+0.ax',subclass_defn),
    [description('Definition of subclass'),
relevance(0.9)]).
```

[1]Such worldly goals seem doomed to failure in the absence of worldly cooperation.

[2]The full BNF of the TSTP syntax is available from the TSTP WWW site [Sutcliffe, 2003]. The BNF is reasonably stable, but not final.

An example of a CNF output formula in TSTP syntax is:

```
cnf(140,derived,
    ( equal_sets(a,aUa)
    | member(member_of(a,aUa),a) ),
    inference(unit_del,[status(thm)],[
        inference(hyper,[status(thm)],[
            5,16]),11]),
    [iquote('hyper,5,16.1,unit_del,11')]).
```

The `<language>` field makes the syntax extensible. New types of formulae are possible by specifying the new `<language>` name and defining the `<formula>` syntax. The `<type>` indicates the semantics of input formulae, with values such as `axiom`, `definition`, `assumption`, `conjecture`. For output the `<type>` is either `initial` or `derived`, and is optionally followed by a subtype that indicates the semantics.

The `<formula>` uses the FOF syntax of the TPTP, with CNF formulae expressed as FOF disjunctions with the universal quantifiers omitted. This syntax was very carefully designed, following a survey of notation used for 1st order logic. It has features that make it easy for humans to read and write, e.g., it uses only characters available on a qwerty keyboard, and uses short notations for connectives, e.g., ! rather than a word such as `forall` for universal quantification. It is easy for programs to parse, and a parser is easily constructed using a parser generated (such as `bison`).

The `<source>` may be an external source such as a file or human creator, or for derived formulae may be an inference term. An inference term has the form:

```
inference(<rule name>,<useful info>,
    [<parent info>,<parent info>, ...])
```

Each `<parent info>` is either the name of another annotated formula, or another inference term. All `<useful info>` fields are lists of terms, and are used for annotations. For inferred formulae the `<useful info>` in the inference term is used to capture the status of the formula, as described in Section 3. In the future a selection of defined `<rule name>`s will be explicitly supported in the TSTP syntax, and the `<parent info>` will be annotated sufficiently for deterministic reproduction of inference steps.

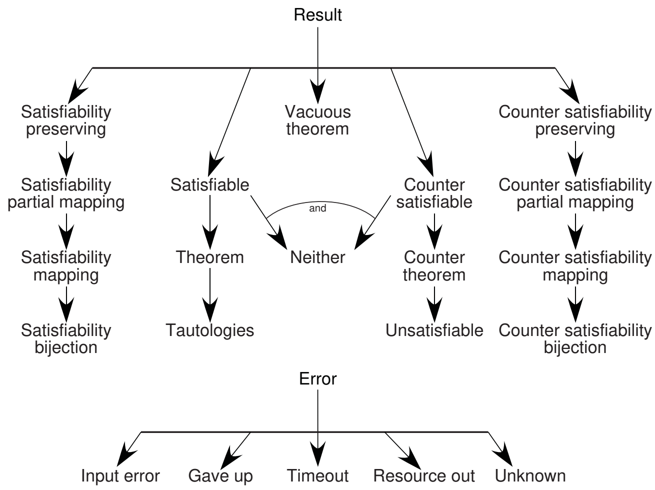
The TSTP syntax is already in use in the TSTP project [Sutcliffe, 2003], where it is being used to capture the outputs from contemporary ATP systems on problems in the TPTP. Parsing tools, written in C, are available, and the `tptp2X` utility distributed with the TPTP is now compatible with the TSTP syntax.

3 ATP System Result Statuses

The output from current ATP systems varies widely in quantity, quality, and meaning. At the low end of the scale, systems that search for a refutation of a set of clauses may output only an assurance that a refutation exists (the wonderful "`yes`" output). At the high end of the scale a system may output a natural deduction proof of a problem expressed in FOF, e.g., [Meier, 2000]. In some cases the output is misleading, e.g., when a CNF based system claims that a FOF input problem is "unsatisfiable" it typically means that the negated CNF of the problem is unsatisfiable.

In order to use ATP systems' results, e.g., as input to other tools, it is necessary that the ATP systems correctly and precisely specify what has been established. To this end a hierarchy of result status values has been established. The hierarchy was based on initial work [Armando *et al.*, 2000] done to establish communication protocols for systems on the MathWeb Software Bus [Zimmer and Kohlhase, 2002]. The hierarchy is shown in Figure 2

Figure 2: Status Hierarchy for ATP Systems' Outputs



The hierarchy assumes that the input F to the ATP system is of the form $Ax \Rightarrow C$. If Ax is empty, i.e., F is a monolithic formula (a particular example is a set of clauses), that's the same as Ax being $true$. If C is empty, e.g., testing the satisfiability of a set of axioms, that's the same as C being $true$. By showing that F is valid, an ATP system shows that the conjecture C is a theorem (a logical consequence) of the axioms Ax, i.e., $Ax \models C$, where $\models$ is the standard 1st order entailment. If F is not valid there are several other possible relationships between Ax and C, as shown in the hierarchy and enumerated below. Associated with each possible status are the possible outputs from the ATP system. The status values and possible outputs are ordered as follows:

1. Every interpretation is a model of Ax and a model of C
   - F is valid; $\sim F$ is unsatisfiable; C is a tautology
   - TSTP status: Tautologies
   - Outputs: Assurance; Proof of F; Refutation of $\sim F$
2. Every model of Ax (and there are some) is a model of C, but not case Tautologies
   - F is valid; C is a theorem of Ax
   - TSTP status: Theorem
   - Outputs: Assurance; Proof of C from Ax; Refutation of $Ax \cup \{\sim C\}$; Refutation of $CNF(Ax \cup \{\sim C\})$
3. Some models of Ax (and there are some) are models of C

- F is satisfiable; $\sim F$ is not valid; C is not a theorem of Ax
- TSTP status: Satisfiable
- Outputs: Assurance; Model; Saturation

4. There is a bijection between the models of Ax (and there are some) and models of C
   - Example: Skolemization, splitting by new predicates
   - TSTP status: Satisfiability bijection
   - Outputs: Assurance
5. There is a mapping from the models of Ax (and there are some) to models of C
   - TSTP status: Satisfiability mapping
   - Outputs: Assurance
6. There is a partial mapping from the models of Ax (and there are some) to models of C
   - Example: $Ax = p|q, C = p\&r$
   - TSTP status: Satisfiability partial mapping
   - Outputs: Assurance; Pairs of models; Pairs of saturations
7. If there exists a model of Ax then there exists a model of C
   - TSTP status: Satisfiability preserving
   - Outputs: Assurance
8. There are no models of Ax
   - F is valid; Anything is a theorem of Ax
   - TSTP status: Vacuous theorem
   - Outputs: Assurance; Refutation of Ax; Refutation of $CNF(Ax)$
9. Some models of Ax (and there are some) are models of C, and some are models of $\sim C$.
   - F is not valid; F is satisfiable; $\sim F$ is not valid; $\sim F$ is satisfiable; C is not a theorem of Ax
   - TSTP status: Neither
   - Outputs: Assurance; Pair of models; Pair of saturations
10. If there exists a model of Ax then there exists a model of $\sim C$
    - TSTP status: Counter satisfiability preserving
    - Outputs: Assurance
11. There is a partial mapping from the models of Ax (and there are some) to models of $\sim C$
    - TSTP status: Counter satisfiability partial mapping
    - Outputs: Assurance; Pairs of models
12. There is a mapping from the models of Ax (and there are some) to models of $\sim C$
    - TSTP status: Counter satisfiability mapping
    - Outputs: Assurance
13. There is a bijection between the models of Ax (and there are some) and models of $\sim C$
    - TSTP status: Counter satisfiability bijection
    - Outputs: Assurance
14. Some models of Ax (and there are some) are models of $\sim C$
    - F is not valid; $\sim F$ is satisfiable; C is not a theorem of Ax
    - TSTP status: Counter satisfiable
    - Outputs: Assurance; Model; Saturation
15. Every model of Ax (and there are some) is a model of $\sim C$, but not Unsatisfiable
    - F is not valid; $\sim C$ is a theorem of Ax C cannot be made into a theorem by extending Ax;
    - TSTP status: Counter theorem
    - Outputs: Assurance; Proof of $\sim C$ from Ax; Refutation of $Ax \cup C$; Refutation of $CNF(Ax \cup C)$
16. Every interpretation is a model of Ax and a model of $\sim C$
    - F is unsatisfiable; $\sim F$ is valid; $\sim C$ is a tautology
    - TSTP status: Unsatisfiable
    - Outputs: Assurance; Refutation of F; Proof of $\sim F$

It is hoped that future releases of ATP systems will use this hierarchy of result statuses. It is already expected that the next release of the E prover [Schulz, 2002] will do so.

4 Conclusion

Two communication formalisms for ATP tools have been presented. In their common role as embedded components in larger systems, the ability of ATP tools to interface directly with other components has an important influence on usability and uptake. These formalisms facilitate direct and correct communication between ATP tools.

The formalisms presented in this paper are finding immediate application in various projects (that the authors are involved with). Most directly these include the various ARTists' projects at the University of Miami [ARTists, 2003], MathWeb, the MPTP project [Urban, 2003], and the E system. A reason why these formalisms can be put to use immediately is that they are highly pragmatic. The formalisms are sufficient to be immediately useful, and have not become embroiled in any attempt to encompass highly abstract and global ambitions - such goals seem to be difficult to achieve, as is evidenced in some other larger projects[3]. It will be important now to use these formalisms in more systems, to reveal the strengths and any weaknesses. As with the TPTP syntax, which in some sense is now a standard in the ATP community, these formalisms are most likely to succeed if there is sufficient successful real usage. If they are successfully adopted, they may too become standards.

In order to assist the ATP community to use the TSTP syntax for input to ATP systems, TPTP v3.0.0 will be distributed

[3] Names may be named only at the workshop.

in both the TPTP syntax and in the TSTP syntax. To encourage adoption of the TSTP syntax and the output statuses, the TSTP library will be distributed in the TSTP syntax, and a suite of TSTP postprocessing tools (which necessarily use the TSTP syntax and status values) will be made freely available. Future versions of the MathWeb Software Bus will provide full support for the TSTP language.

Future work includes designing formalisms for the output from model generation programs such as MACE [McCune, 2001] and FINDER [Slaney, 1994], and building a hierarchy of possible outputs (outputs such as "assurance", "refutation", "proof") from ATP systems.

References

[ACL, 2003] ACL2. http://www.cs.utexas.edu/users/moore/acl2/.

[Armando *et al.*, 2000] A. Armando, M. Kohlhase, and S. Ranise. Communication Protocols for Mathematical Services based on KQML and OMRS. In M. Kerber and M. Kohlhase, editors, *Proceedings of the Calculemus Symposium 2000*, 2000.

[ARTists, 2003] The ARTists. Automated Reasoning Tools at the University of Miami. http://www.cs.miami.edu/ geoff/ResearchProjects/ART/.

[Caprotti and Carlisle, 1999] O. Caprotti and D. Carlisle. OpenMath and MathML: Semantic Mark Up for Mathematics. *ACM Crossroads*, 6(2), 1999.

[CL-, 2003] Common Logic Standard. http://cl.tamu.edu/.

[Claessen *et al.*, 2002] K. Claessen, R. Hähnle, and J. Mårtensson. Verification of Hardware Systems with First-Order Logic. In G. Sutcliffe, J. Pelletier, and C. Suttner, editors, *Proceedings of the CADE-18 Workshop - Problem and Problem Sets for ATP*, number 02/10 in Department of Computer Science, University of Copenhagen, Technical Report, 2002.

[Coq, 2003] The Coq Proof Assistant. http://pauillac.inria.fr/coq.

[Genesereth and Fikes, 1992] M.R. Genesereth and R.E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[Hähnle *et al.*, 1996] R. Hähnle, M. Kerber, and C. Weidenbach. Common Syntax of the DFG-Schwerpunktprogramm Deduction. Technical Report TR 10/96, Fakultät für Informatik, Universät Karlsruhe, Karlsruhe, Germany, 1996.

[HOL, 2003] Automated Reasoning Group HOL Page. http://www.cl.cam.ac.uk/Research/HVG/HOL/.

[Horrocks and Sattler, 2001] I. Horrocks and U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence* , pages 199–204. Morgan Kaufmann, 2001.

[Kohlhase, 2000] M. Kohlhase. OMDOC: Towards an Internet Standard for the Administration, Distribution, and Teaching of Mathematical Knowledge. In J.A. Campbell and E. Roanes-Lozano, editors, *Proceedings of the Artificial Intelligence and Symbolic Computation Conference, 2000*, number 1930 in Lecture Notes in Computer Science, pages 32–52. Springer-Verlag, 2000.

[McCune, 1997] W.W. McCune. Solution of the Robbins Problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[McCune, 2001] W.W. McCune. MACE 2.0 Reference Manual and Guide. Technical Report ANL/MCS-TM-249, Argonne National Laboratory, Argonne, USA, 2001.

[Meier, 2000] A. Meier. System Description: TRAMP - Transformation of Machine-Found Proofs into Natural Deduction Proofs at the Assertion Level. In D. McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction*, number 1831 in Lecture Notes in Artificial Intelligence, pages 460–464. Springer-Verlag, 2000.

[Riazanov and Voronkov, 2002] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[Schulz, 2002] S. Schulz. E: A Brainiac Theorem Prover. *AI Communications*, 15(2-3):111–126, 2002.

[Slaney *et al.*, 1995] J.K. Slaney, M. Fujita, and M.E. Stickel. Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems. *Computers and Mathematics with Applications*, 29(2):115–132, 1995.

[Slaney, 1994] J.K. Slaney. FINDER: Finite Domain Enumerator, System Description. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, number 814 in Lecture Notes in Artificial Intelligence, pages 798–801. Springer-Verlag, 1994.

[Sutcliffe, 2003] G. Sutcliffe. The TSTP Solution Library. http://www.TPTP.org/TSTP.

[Sutcliffe and Suttner, 1998] G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[Tammet, 1997] T. Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199–204, 1997.

[Urban, 2003] J. Urban. Automated Reasoning Contributes to Mathemetics and Logic. In A. Asperti, B. Buchberger, and J.H. Davenport, editors, *Proceedings of the 2nd International Conference on Mathematical Knowledge Management*, number 2594 in Lecture Notes in Computer Science, pages 203–215. Springer-Verlag, 2003.

[Weidenbach *et al.*, 2002] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobald, and D. Topic. SPASS Version 2.0. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in Lecture Notes in Artificial Intelligence, pages 275–279. Springer-Verlag, 2002.

[Whalen *et al.*, 2002] M. Whalen, J. Schumann, and B. Fischer. Synthesizing Certified Code. In L.H. Eriksson and P Lindsay, editors, *Proceedings of the International*

Symposium of Formal Methods Europe (FME 2002: Formal Methods - Getting IT Right), number 2391 in Lecture Notes in Computer Science, pages 431–450. Springer-Verlag, 2002.

[Wos and Pieper, 1999] L. Wos and G. Pieper. *A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning*. World Scientific, 1999.

[Zimmer and Kohlhase, 2002] J. Zimmer and M. Kohlhase. System Description: The MathWeb Software Bus for Distributed Mathematical Reasoning. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, number 2392 in Lecture Notes in Artificial Intelligence, pages 139–143. Springer-Verlag, 2002.