The Design of the CADE-13 ATP System Competition

Technical Report 95/15[1] Technical Report AR-95-05[2]
TR Date 7.6.96

Geoff Sutcliffe
[1]Department of Computer Science
James Cook University, Australia
Phone: +61 77 814622 Fax: +61-77-814029
Email: geoff@cs.jcu.edu.au

Christian B. Suttner
[2]Institut für Informatik
TU München, Germany
Phone: +49-89-521098 Fax: +49-89-526502
Email: suttner@informatik.tu-muenchen.de

Abstract

Running a competition for Automated Theorem Proving (ATP) systems is a difficult and arguable venture. However, the potential benefits of such an event by far outweigh the controversial aspects. The motivations for running the CADE-13 ATP system competition are to contribute to the evaluation of ATP systems, to stimulate ATP research and system development, and to expose ATP systems to researchers both within and outside the ATP community. This paper identifies and discusses the issues that determine the nature of the competition. Choices and motivated decisions, with respect to the issues, are given.

Contents

1 Introduction 2

2 Competition Issues 3

3 ATP System Properties 4
- 3.1 Degree of Automation 4
- 3.2 System Soundness and Completeness 6
- 3.3 Proofs and Satisfiability Checking 7
- 3.4 Proof Objects 7
- 3.5 Monolithic and Compositional Systems 8

4 Problem Presentation 8
- 4.1 Scope of Logic 8
- 4.2 Built-in Theories 10
- 4.3 Input Language 10
- 4.4 Clause Type Information. 11
- 4.5 Clause and Literal Orderings. 12

5 Problem Selection 12
- 5.1 Problem Source 12
- 5.2 Problem Difficulty 12
- 5.3 Problem Axiomatization 13
- 5.4 Number of Problems 13

6 Resource Limits 14
- 6.1 Time Resource 14
- 6.2 Hardware and Software Resources 15

7 System Evaluation 16
- 7.1 Winner Assessment 16
- 7.2 Performance Metrics 16
- 7.3 Ranking Schemes 18

8 Conclusion 19

1 Introduction

Running a competition for Automated Theorem Proving (ATP) systems[1] is a difficult and arguable venture. The reasons for this are that existing ATP systems are based on different logics, are designed for different types of reasoning, require different amounts of user interaction, have different input formats, and may run on specialized hardware. Moreover, there is no clear work profile with respect to which ATP systems should be evaluated. However, a competition will make a significant contribution to the following important needs:

- to evaluate the relative capabilities of ATP systems,
- to stimulate ATP research in general,
- to stimulate ATP research towards autonomous systems,
- to provide motivation for implementing and fixing systems,
- to provide an inspiring environment for personal interaction between ATP researchers,
- to expose ATP systems to researchers both within and outside the ATP community.

In other disciplines competitions are regularly used to stimulate research and development. The annual ACM computer chess championship [New94] is very well known and attracts much interest. Participants in the chess championship are clearly motivated to improve their programs. Programming competitions are also popular [CRRP90, AKK93, Dem96], and encourage programmers to improve their skills. Competitions in machine learning [MMPS94], algorithm implementation [CIJ$^+$94], and other disciplines have all stimulated effort in their respective fields. In ATP there have already been competitions at the propositional [BKB92] and 1st order [Ove93] levels, which have encouraged researchers to improve their ATP systems. As an example, Otter's autonomous mode [McC94] resulted in part from the competition described in [Ove93].

As well as being of interest in its own right, an ATP system competition, or at least an examination of the issues involved in a competition, will provide insight into the more general notion of ATP system evaluation. For a long time the testing and evaluation of ATP systems has been very ad hoc. Results being published seldom provide an accurate reflection of the intrinsic power of the ATP system being considered. Inadequate evaluation has the potential to damage a field of research; bad ideas may appear to be good and are thus adopted and perpetuated[2], while good ideas may appear bad and are hence discarded. A first step towards improving ATP system evaluation has been the release of the TPTP problem library [SSY94, SS96]. A common library of problems is a necessary, but not sufficient, tool for meaningful system evaluation. There are also other issues that need careful consideration. This examination of the issues surrounding an ATP system competition goes a fair way towards capturing the requirements for meaningful ATP system evaluation. It should be noted that using benchmarks and test beds (such as the TPTP problem library) in system evaluation requires special care (see also [HPC93]).

In order to benefit fully from a competition, a thoroughly organized event, with unambiguous and motivated rules, is necessary. One approach to ATP system competitions, as was taken in

[1]We use the term "ATP system" to refer to the functional unity represented by the implementation of a reasoning calculus and its control, i.e., the implementation of a proof procedure.

[2]Geoff thinks that paramodulation is an example of this in ATP.

the competition at CADE-11 [Ove93], is to run very specialized competitions with specifically selected problems. While this allows a detailed analysis and comparison of the proofs obtained by ATP systems, it does not provide a realistic comparison of the ATP systems in terms of their general usefulness. In [Ove93] Overbeek notes that the ATP community had "never [been] able to formulate an acceptable mechanism for comparing different systems". In order for a comparison of different ATP systems to make sense, it is necessary that all the systems should be attempting to capture a common notion of "truth", as is described in the Realist viewpoint in [Pel91]. Given this commonality across all systems, we believe that it is possible to set out rules for a competition that can determine a winner, relative to some clearly specified constraints. For some issues relevant to an ATP competition, inevitable constraints emerge. For other issues there are several choices, and a decision has to be made for the competition. The arising issues, choices, and decisions are described in this paper.

Although this paper focuses on the issues relevant to experimental comparison of ATP systems, it is important to acknowledge the contribution of analytic approaches, e.g., [Let93, Dun94, Pla94]. These approaches investigate metrics such as search space duplication, search tree size, Herbrand multiplicity, number of variables to bind, and connectivity. At the propositional level analysis can be reasonably accurate, and is thus important for that limited situation. At the 1st order level the analysis is significantly more difficult, and results obtained so far are insufficient. Complete analysis of search guidance at the 1st order level is of course impossible (for otherwise 1st order logic would be decidable!). It is for these reasons that we have focused on experimental comparison of ATP systems; in this way meaningful judgements can be made.

2 Competition Issues

There are four types of issues (and resultant choices and decisions) relevant to the design of an ATP system competition:

1. Issues based on the fundamental nature of ATP.
2. Issues associated with common practice in ATP research.
3. Issues arising from the application of ATP systems.
4. Issues directly related to the competition.

An overview of the issues that are discussed in the following sections is given in Table 1. The table also shows the two types a particular issue most belongs to, according to the four types presented above.

Type	Issue
ATP System Properties	
2,3	Degree of Automatization : How much user interaction is allowed?
2,3	Soundness and Completeness: How do soundness and completeness affect evaluation?
2,3	Proofs and Satisfiability Checking: Should the systems be expected to find proofs or models?
2,3	Proof Objects : Are explicit proof objects required?
1,2	Monolithic and Compositional Systems: Does the system structure matter?
Problem Presentation	
1,2	Scope of Logic : Which logics (e.g., Classical, Modal) should be allowed?
1,2	Built-in Theories : Are built-in theories (e.g., equality) allowed?
2,3	Input Language : What input language (e.g., CNF, FOF) should be used?
3	Clause Type Information : Is usage of such information acceptible?
2	Clause and Literal Orderings : How can bias be avoided?
Problem Selection	
2,4	Problem Source : Where should the problems come from?
4	Problem Difficulty : How difficult should the problems be?
2,4	Problem Axiomatization : What about biases in problem axiomatizations?
4	Number of Problems : How many problems should be used?
Resource Limits	
3,4	Time resource : – Should a runtime limit apply per problem or for all problems altogether? – What runtime limit should be imposed?
4	Hardware and Software resources : What hardware and software should be available for the competition?
System Evaluation	
4	Winner Assessment : Should a distinguished "winner" be determined?
2,3	Performance Metrics : What metrics should be used for evaluation?
3,4	Ranking Scheme : How should performance metrics be combined for ranking?

Table 1: Relevant issues for an ATP system competition design.

3 ATP System Properties

3.1 Degree of Automation

From a user's viewpoint, ATP systems are of two types : fully automatic systems[3] and interactive systems. Currently, and presumably in the future, the most important application of fully automatic ATP systems is as embedded systems in more complex reasoning environments, where they serve as core inference engines. This setting is shown in Figure 1. The environment may call on the fully automatic ATP system to solve subtasks within the operation of the overall system. The subtasks are thus generated by the environment. Examples of fully au-

[3]Astrachan and Loveland [AL94] define three types of fully automated deduction (F.A.D.) systems ... "pure F.A.D. systems, where only the problem can be entered ..., the strong F.A.D. systems, where parameters can be set only once, and the weak F.A.D. system, where many experiments with parameters are permitted, but each adjustment requires a restart from the beginning of the problem.". For the purposes of this discussion, automatic ATP systems are the pure and the strong F.A.D. systems; weak F.A.D. systems are considered to be interactive.

tomatic systems are Otter [McC94], SETHEO [LSBB92], SATCHMO [SGar], the CLIN series [LP92, CP94, Ale95], and SPASS [WGRar].

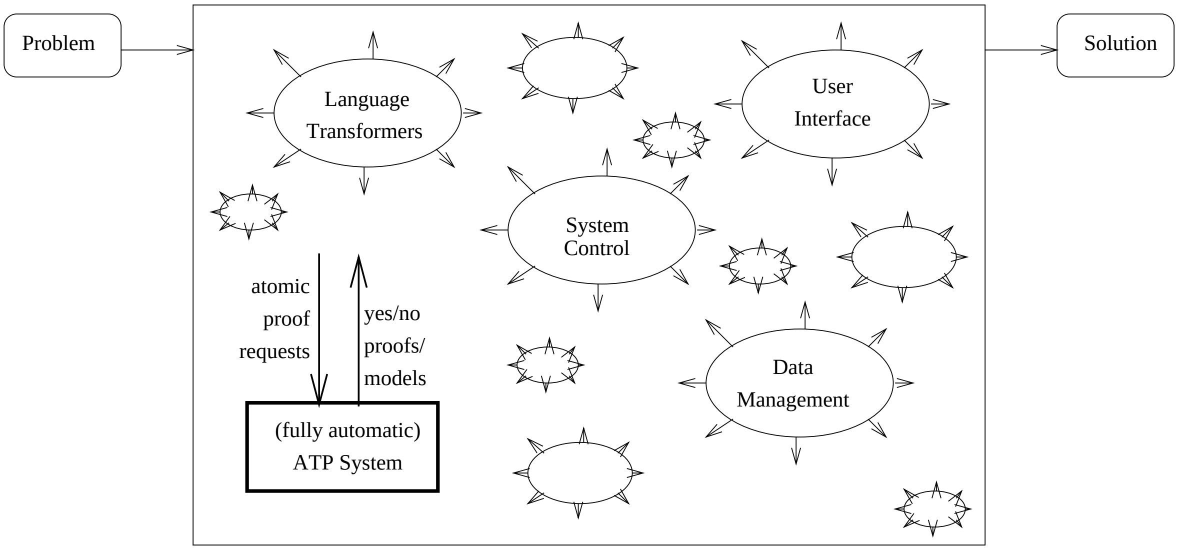

Figure 1: The embedding of an ATP system in some reasoning environment.

Interactive systems find application in hardware and software verification research [BM90, HRS90, ORS92, WG92], the formalization of informal mathematical proofs [FGT90, Pau90], the teaching of logic [Por94], and as tools of mathematical and meta-mathematical research [WWM+90, CAB+86]. Examples of interactive systems are Ω-MKRP [HKK+94], HOL [GM93], NQTHM [BM90], and NUPRL [CAB+86]. Interactive proof systems typically embed some form of a fully automatic ATP system as a subsystem[4], and are an example of the reasoning environments mentioned above. At any point the user of an interactive system may ask that some (sub)problem be submitted to the fully automatic component.

Both fully automatic and interactive theorem proving systems are important. It is clear, however, that the two types of system are very different. In terms of a competition, it would not make sense to compare fully automatic systems with interactive systems; the qualities that are required of the two types are largely distinct. In particular, an interactive system must be judged largely on its ability to assist the user in the organization and tracking of complex proof objects [PS94]. While for fully automatic systems the system runtime is the dominating issue. It may be the case that there are ATP system evaluation criteria that are independent of whether the systems are automatic or interactive. However, at this stage no such criteria are clear. Thus the ATP competition can compare either fully automatic systems or interactive systems, but the two types cannot be compared with each other.

Many of the assessment criteria for a competition between interactive theorem proving systems would be very subjective, as individual user abilities and tastes would necessarily factor into the comparison. There are some criteria, e.g., "user-friendliness", that do not appear to have any objective measure at all. In contrast, the criteria for comparing fully automatic ATP systems are mostly objective, as described in this paper. Also, recalling that interactive systems typically embed a fully automatic ATP system as a subsystem, a comparison of the fully automatic subsystems of interactive systems should be part of a comparison of interactive systems[5]. Decision : The ATP competition will be between fully automatic ATP systems. No

[4]At the current stage, most interactive systems contain a quite rudimentary and simple theorem proving part, while (justifiably so) most effort is spent on the development of adequate logics, proof tactics, and interfaces.

[5]Today, a direct comparison of the fully automatic subsystems would be difficult, because due to their

human interaction will be allowed.

In the rest of this paper attention is limited to fully automatic ATP systems, and they will be referred to simply as ATP systems. Note that the decision to focus on fully automatic systems does not exclude the possibility of a competition between semi-automated or interactive systems. Indeed, such a competition would be of interest, if the difficulties involved can be adequately resolved.

A consequence of the limitation to fully automatic systems is that the competition will not appreciate system properties which relate to human computer interaction, and therefore will not necessarily determine the best "human assistant". Rather, the competition will assess the pure reasoning capabilities, and nothing else[6]. The intention is to foster automation. In particular, the competition aims to identify powerful general purpose deduction procedures, and to encourage automatic problem analysis and system configuration. The development of deduction procedures has been a focus of ATP research since its inception, and is certainly a mature enough area to evaluate. On the other hand, automatic problem analysis and system configuration are underdeveloped. In various research groups there is much knowledge about how to analyze problems, and how to configure that group's ATP system. This knowledge is exploited when the ATP system is evaluated, by having the knowledgeable person manually configure the system for each problem. Due to this common practice of manual configuration, the effort to capture and implement problem analysis and configuration techniques is rarely taken. Implemented systems often feature strong reasoning components but have very poor automatic control. This makes it difficult for other users of such systems to get the best possible performance from the systems, and makes the systems unsuitable for use as embedded systems. Hopefully this ATP competition will encourage researchers to formalize and capture their analysis and configuration skills in their ATP systems.

3.2 System Soundness and Completeness

Traditionally, ATP systems are designed to be deductively sound and complete: every answer returned is deductively correct and if a solution exists it is eventually deduced. However, ATP systems are usually too complex to formally verify either property. Thus systems are tested experimentally to detect violations of these properties. While soundness can be assumed for extensively tested systems (at least if a large number of the produced solutions have been verified), completeness is more difficult to assess. The reason for this is that the inability to find a solution is not necessarily a sign of incompleteness - the problem could simply be too hard for the system. In fact, due to the finite amount of resources allocated to any particular ATP system run, every search for a solution has limited completeness, independent of the system's theoretical completeness. A special case of incompleteness is a bugged ATP system, which crashes on some problems. The question then arises, how should unsound and/or incomplete and/or bugged systems be treated in the ATP competition?

From a users point of view, there is little difference between not finding a solution due to system incompleteness, not finding a solution due to a resource limit being reached, and not finding a solution due to a system crash. In all cases the user learns nothing about the solution to the problem. There are also positive aspects to each form of failure. If a resource limit is reached then the user can optimistically increase the resources available (although this is futile

simplicity no modular system design is used and an isolated test is not easily possible.

[6]But note: the success of ATP systems as automated reasoning assistants will depend on their usability by non-experts. For high usability the dependence on user interaction, in terms of low level guidance, needs to be minimized. This argues that even in interactive systems less (low level) interaction is better.

if the resource limit was reached due to incompleteness, and further, the supply of resources typically cannot deal with the exponential growth in demand for them). If a system terminates quickly, due to incompleteness or a bug, then time is saved and there is hope of having the system repaired. In practice a bugged or incomplete system may solve more problems within some time limit than a bug free complete system, and therefore may be more useful. Decision: In the ATP competition, systems must be sound but may be incomplete and may be bugged. The soundness of competing systems will be assessed before the competition with specific test problems. Systems found to be unsound will be excluded from the competition.

A property associated with soundness and completeness is that of stability. If an ATP system finds a solution to a problem, then the system must be able to find that solution again if required. ATP systems that simply 'get lucky' are of little use. Repeatability of testing is also a standard requirement of the scientific method. Decision: For every solution found in the ATP competition, the solution process must be reproducible by running the system again. If randomness plays a role, reproducibility must still be ensured, e.g., by using pseudo-random numbers and recording the seeds.

3.3 Proofs and Satisfiability Checking

There are two distinct classes of problems that may be presented to ATP systems. Firstly there are problems that require the proof of a theorem, and secondly there are problems that require satisfiability to be established (via the generation of a model). Both types of problem are of interest. However, mixing proof performance and satisfiability assessment would blur the interpretation of competition results. One reason for this blurring is that most ATP systems are designed for one or the other purpose, but not both (exceptions are, for example, the RAMCEC algorithm [BCP94] and hyper-linking [LP92]). As a result, a comparison of specialized systems of the different types is arguable. In the future more dual purpose systems may be developed, in which case it may not be necessary to divide the work profile for a competition, as is now the case. Thus the work profile for a competition should not require both theorem proving and model generation.

Historically, more emphasis has been placed on the ability to find proofs, and therefore this is considered to be the more important issue (albeit this is not the case in some applications). Decision : The ATP competition will focus on theorem proving performance. In order to implement this decision, only theorems will be used (see Section 5.1).

3.4 Proof Objects

Depending on its generality and purpose, there are various responses that an ATP system may make when given a problem. A proof or a model may be returned, or only an assurance that a proof or model exists may be given.

There are some ATP systems, e.g., completion based systems, that conserve resources by not building a proof object during their search. Such systems merely retain enough information to build the proof object later, if required. Other ATP systems, e.g. model generation systems, are not able to build a proof object, and can only give an assurance that a proof exists.

There is no evidence that usage of an ATP system as an embedded system typically either requires or does not require delivery of proof objects or models, and it is desirable to make the ATP competition accessible to as many ATP systems as possible. Decision : The ATP competition will not require ATP systems to return proof objects. The soundness testing (see Section 3.2) will ensure that competing systems do not make invalid claims to have found a proof. This decision evidently gives some bias towards those systems that do not build a proof object. This bias is considered to be slight, and justified in terms of extending the scope of the competition. However, the added functionality of ATP systems that do produce proof objects should be acknowledged. Decision : In the presentation of the ATP competition results, it will be noted which of the systems do produce complete proof objects.

3.5 Monolithic and Compositional Systems

Today, a large number of different calculi, inference rules, and proof procedures exist. Ideally, an ATP system would have all known techniques available, and would, during a particular solution search, switch back and forth between the most suitable techniques. First steps in this direction are ATP systems which are formed from a collection of different proof procedures, and which analyze the given problem in order to decide which procedure or inference rules to use. A well known example of this type of system is Otter in its autonomous mode [McC94], which enables or disables inference rules according to a check whether the problem is propositional, whether it is Horn, and whether it contains equality and/or equality axioms.

An ATP system in which no components are chosen as alternatives to others, based on the given problem's characteristics, is called a *monolithic* ATP system. An ATP system that runs as one of several possible distinct monolithic systems, as determined by the given problem's characteristics, is called a *compositional* system. Combining several monolithic systems in a parallel manner also results in a compositional system. The definition does not limit monolithic systems to a single calculus or rule of inference, but requires that any switching back and forth between different calculi and/or rules is done in an integrated way, and occurs continuously during the proof process[7].

Compositional systems provide a valuable approach to building more versatile systems, and compositional systems can be expected to outperform monolithic systems. Conversely, monolithic systems can be improved by integrating more specialized components. Altogether, both the construction of compositional systems and of monolithic systems are valuable. Research into compositional systems will improve knowledge of which approach is best for what problems, while research into monolithic systems will push the performance of uniform approaches (which form the core of compositional systems) to the limit[8]. The results may one day lead to a type of monolithic system which integrates all available techniques. Altogether, the competition should not put either of these approaches at a disadvantage. Decision : In the ATP competition there will be two categories: Open, which includes both types of systems, and Monolithic, which includes only monolithic systems. Any controversy about which category a system belongs to will be resolved by the competition panel (see Section 7.1).

4 Problem Presentation

4.1 Scope of Logic

Many "hot logics" in ATP research, such as higher order logics, modal logics, and temporal logics, are very interesting. Sufficient research has been done to support a comparison of ATP

[7] Rigid definitions of monolithic and compositional systems seem hardly possible. For any formal definition made it seems that an ATP system that violates the intuitive notion can be contrived.

[8] This issue can also be viewed from the evaluation viewpoint: the goal of evaluating a system is not simply to see how well it performs, but also to learn why it achieves the observed performance. This is difficult for compositional systems, and therefore it is desirable to obtain evaluations of their individual components as well.

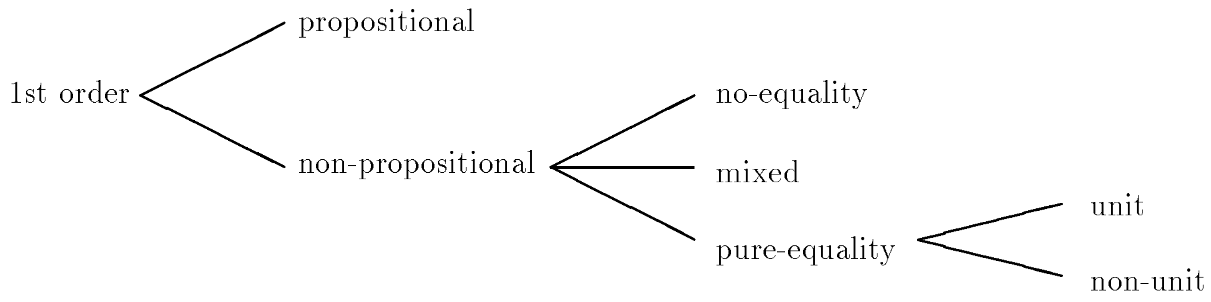

Figure 2: Classification of 1st order logic problems.

systems within many of these logics, as well as common classical logics. However, a comparison of ATP systems across logics would be unlikely to make sense, for several reasons: the types of output and expected performance vary from logic to logic, different logics may have different semantics, and specific logics may be particularly designed for specific applications (making a fair problem selection difficult). Thus the problems for the competition should include problems from any one (but only one) logic.

The wide-spread use of 1st order logic in ATP suggests it as a starting point; specialized competitions for other logics can still occur in the future. Furthermore, transformation techniques (e.g., [Ker91, Ohl91]) allow the use of 1st order logic ATP systems for solving problems in some other logics. Decision : The ATP competition will be restricted to theorems expressed in classical 1st order logic.

Classical 1st order logic can be subdivided as shown in Figure 2.

The leaves in Figure 2 correspond to syntactically identifiable problem classes, and it is trivial to determine which class a given problem belongs to. It would be possible to use problems from all classes together, or to have different competition categories for the classes.

For some of the problem classes specialist ATP systems, that employ the appropriate techniques, exist. Specialist systems can be expected to perform better on problems in their class than systems not specialized for that class. A best overall ATP system can be built easily by choosing the best system for each problem class, and using a *trivial switch* to invoke the appropriate system. Trivial switch systems are of high interest to users, because they provide better overall performance than any of their components. For the competition, however, there is little interest in evaluating the actual switches within such systems, due to their simplicity. Thus an overall performance evaluation of such systems is not desirable. Instead it is better to evaluate the component systems separately, and from that conclude the overall performance achievable by their combination. Decision : Specific problem classes of 1st order logic (according to the classification above), for which specialist ATP systems exist, should be treated in separate categories in the ATP competition.

Ideally the ATP competition would have separate categories for all of the problem classes. However, merging or omitting some classes may still be appropriate.

The development of specialized ATP systems for propositional logic is largely separated from ATP for non-propositional logic, and there are separate specialized competitions for evaluating propositional systems[9]. There is no need to duplicate those competitions. Decision : There will not be a category for propositional problems in the ATP competition.

In the original competition announcement the no-equality and mixed problem classes were merged, for the following reasons. Firstly, systems that include specialized equality treatment

[9]The next such event will be at the "International Competition and Symposium on Satisfiability Testing", to be held in Beijing, China.

are typically competitive on no-equality problems, and therefore can be evaluated on such problems. Secondly, a natural formulation of many interesting problems involves equality, and therefore most state-of-the-art ATP systems include specialized equality treatment. These motivations indicate that it is acceptable to keep these two classes merged, and it is desirable for the competition to maintain its announced format. Decision : The no-equality and mixed problem categories will be combined in the ATP competition.

Most pure equality problems known to the ATP community are written in Clause Normal Form (see Section 4.3) as unit equality problems. There are specialized unit equality ATP systems which cannot handle any other format of problem, so it is necessary to have a separate category for these systems. Decision : There will be a separate category for unit equality problems in the ATP competition. Non-unit pure equality systems are able to handle no-equality and mixed problems by translating such problems to pure equality, and are able to compete with no-equality and mixed systems (even if at a disadvantage, the extent of which is currently unknown). The small number of commonly known non-unit pure equality problems seems insufficient for a separate category. Decision : The no-equality, mixed, and non-unit pure equality problem categories will be combined in the ATP competition[10].

4.2 Built-in Theories

A theory can be built into an ATP system, rather than expressing it by a set of axioms (e.g., equality theory). This approach is chosen in order to provide specialized, and hopefully more efficient, treatment of that theory. Systems capable of such specialized action should not be put to a disadvantage by including axioms unnecessary for them. Decision : In the ATP competition, theory axioms that have been built into an ATP system can be removed from the input to that ATP system, using an automatic tool.

4.3 Input Language

The problems submitted to an ATP system may be expressed in full First Order Form (FOF) or possibly in some constrained subset of 1st order logic, such as clause normal form (CNF). Therefore the ATP system must, in general, be able to deal with FOF. An examination of current ATP systems reveals three possible approaches (see Figure 3):

- A non-specialized clausifier is used to convert the submitted problems to CNF, and the ATP system uses the CNF versions of the problems (CNF systems).
- A specialized clausifier is prepended to a CNF ATP system, and this combined system deals with the submitted problems directly (Clausifying systems).
- The ATP system does it's processing in FOF, and thus deals with the submitted problems directly (FOF systems).

A comparison of CNF systems and FOF systems is not sensible, as CNF systems are disadvantaged if FOF problems are used, and FOF systems are disadvantaged if CNF problems are used. A comparison of FOF systems and clausifying systems is possible using FOF problems, and a comparison of CNF systems and clausifying systems is possible using CNF problems.

A comparison of CNF systems and clausifying systems would constitute a comparison of the clausifiers as well as a comparison of the ATP systems' basic theorem proving capabilities.

[10] In future ATP competitions these categories should be separate.

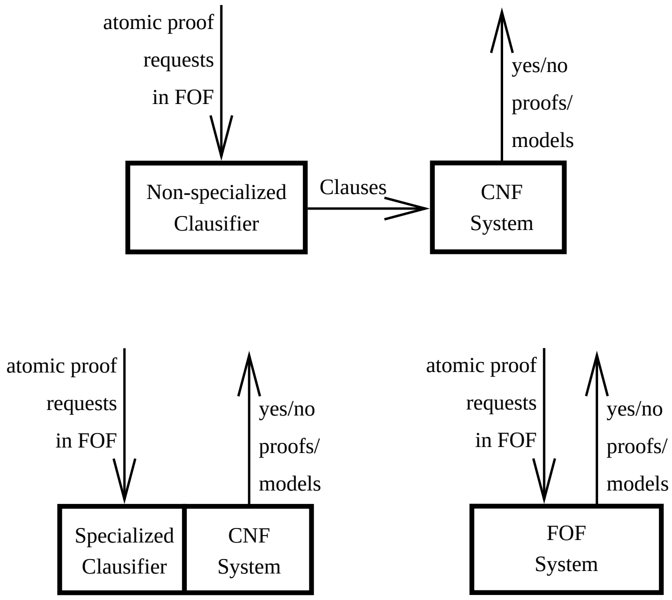

Figure 3: Alternative system designs regarding the input language.

This would blur the interpretation of the results. Also, current ATP research is dominated by CNF systems. There are applications that generate CNF problems directly (e.g., test pattern generation for electronic circuits). These factors indicate that a CNF competition is particularly viable. Decision : The ATP competition will use theorems expressed in Clause Normal Form. This constraint does not exclude the other two types of ATP systems from the competition. However, such systems will not be able to take advantage of their abilities to deal with FOF problems[11].

CNF problems can be either Horn or non-Horn, and there are particular techniques designed for solving each type. In the same way that the different classes of problems shown in Figure 2 are treated separately, ideally Horn and non-Horn problems should be treated separately.

It is not clear what techniques are best for which type of problem. ATP systems for non-Horn problems can perform well on Horn problems, and some non-Horn problems are handled well by systems designed for Horn problems. There is no corpus of ATP systems that are specialized for either type of problem. For switch systems, classifying problems as being Horn or non-Horn is easy, but selecting superior specialists for either type is not. Decision : Horn and non-Horn problems will be used together in the ATP competition.

4.4 Clause Type Information.

A clause of a CNF problem may be classified, depending on whether it is part of the conjecture, part of assumptions made, or part of the underlying theory. This clause type information may be useful to an ATP system.

In the application of ATP systems such information will typically be available, and therefore should be used. Decision : Systems may use clause type information in the ATP competition.

[11]A competition using FOF problems is planned for the future. Having established the quality of their CNF ATP systems in this competition, developers will be able to spend time building specialized clausifying programs ready for a subsequent FOF competition.

4.5 Clause and Literal Orderings.

A possible source of bias in the competition is the particular ordering of the clauses within each problem, and the literals within clauses.

For system evaluation it is necessary to ensure that pre-tuning (possibly accidental) to a particular clause or literal order is of no benefit. Decision : In the ATP competition, random clause and literal orderings will be used. The same clause and literal ordering will be used for all systems.

5 Problem Selection

5.1 Problem Source

There are several possible sources for the unsatisfiable sets of clauses to be used in the ATP competition. The competition could use problems supplied by the entrants, could use problems selected from those commonly used by the ATP community, or could use new problems designed especially for the competition (there are other options too).

If entrants were to supply the problems, they would almost inevitably supply (pathological) cases for which their ATP system is particularly effective and that are difficult for other competitors. The problems would not reflect any real work profile of ATP systems, and the ability of the entrant to make such selections would influence the results. Designing new problems for the competition would be acceptable, but requires additional effort and could still (unintentionally) be biased towards a particular system. Using problems from the ATP community will to some extent reflect common usage of ATP systems. The TPTP Problem Library [SSY94, SS96] contains a broad selection of such CNF theorems, as required. Decision : The unsatisfiable sets of clauses to be used in the ATP competition will be selected from the TPTP problem library[12].

The tptp2X utility, distributed with the TPTP problem library, will be used to remove unnecessary equality axioms (see Section 4.2) and to reorder the problem clauses and literals (see Section 4.5).

The tptp2X utility can be used to produce a pure equality representation of problems for non-unit pure equality systems (see Section 4.1).

TPTP problems provide clause type information (see Section 4.4). Use of the clause type information is possible either directly, by reading the TPTP problem, or indirectly, by using a tptp2X formatting option that takes clause type information into account (see [SS96] for details).

In principle, knowing the problem source allows problem-specific information to be precomputed and accessed during the competition (in the extreme, proof look-up). Such activity is contrary to the purpose of the competition, and is disallowed. Global optimization of an ATP system for the TPTP is acceptable, because such an optimization seems likely to work well as a general technique, due to the size and scope of the TPTP.

5.2 Problem Difficulty

The TPTP problem library contains a broad selection of problems in CNF, most of which are unsatisfiable sets of clauses, as required. Suitably difficult problems must be selected for the

[12] In a sense, entrants influence the problem selection by their submission of problems to the TPTP. However, due to the selection process this constitutes a very weak influence.

competition. The TPTP rating system categorizes problems as follows:

1	easy	solvable by all state-of-the-art ATP systems
2	difficult	solvable by some state-of-the-art ATP systems
3	unsolved	solvable by no state-of-the-art ATP systems
4	open	theorem-hood unknown

Here "state-of-the-art ATP systems" means specialist ATP systems for the class of the problem, according to an extended version of the classification given in Section 4.1.

Using easy problems would not differentiate sufficiently between systems performances. On the other hand, each theorem must be provable by at least some of the systems. Decision : The ATP competition will use TPTP problems with difficulty rating 2.

5.3 Problem Axiomatization

Many of the problems in the TPTP problem library have been taken from publications and existing problem libraries. Some of the theorems contain only those axioms necessary for a particular proof, or contain lemmas that assist a proof. Also, the axioms are often selected to suit a particular ATP system, and are therefore biased towards that system. For each problem using such a *non-standard* axiomatization, the TPTP provides a version of the problem using a *standard* axiomatization containing all the axioms and no lemmas.

The use of a standard axiomatization is desirable because it reflects the typical initial situation when formulating a new problem, and also avoids any bias towards a particular system. Decision : Only standard versions of TPTP problems will be used in the ATP competition.

5.4 Number of Problems

Ideally, each ATP system would be evaluated using all eligible TPTP problems. However, the limited time and hardware resources available determine a maximal number of problems[13]. At the same time, a minimal number of problems is necessary for meaningful system evaluation. The number of problems to be used needs to be determined.

The minimal number of problems will be determined to ensure sufficient confidence (say 85%) that the competition results are the same as would be obtained using all eligible problems. For an evaluation based on the number of problems solved, as is essentially the case for the ranking schemes to be used in the ATP competition (see Section 7.3), and assuming a worst case proportion of problems solved (50%), this is computed by first computing n_0, the minimal number assuming an infinite number of eligible problems:

$$n_0 = \frac{1.44^2 \times 0.5 \times (1 - 0.5)}{0.15^2} = 23.04$$

and then adjusting for the actual number of eligible problems as follows [Wad90]. Decision : The minimal number of problems to be used in the ATP competition will be:

[13] A complete evaluation pursued in a larger time frame is an interesting project by itself. Available TPTP result data for various systems have been collected in [SS95]. These results, however, are difficult to compare, because they have been obtained under differing constraints and on differing hardware platforms.

$$min\_number\_of\_problems = \left\lceil \frac{23.04 \times number\_of\_eligible\_problems}{23.04 + number\_of\_eligible\_problems} \right\rceil$$

The rating of TPTP problems is currently being completed. From that the number of eligible problems, and hence the minimal number of problems, will be determined.

The maximal number of problems is determined by the resources available for the competition (the number of workstations and the total time), the number of ATP systems competing, and the minimal time limit that can be imposed on each proof attempt (see Section 6.1). Decision : The maximal number of problems to be used in the ATP competition is:

$$max\_number\_of\_problems = \frac{number\_of\_workstations \times time\_for\_competition}{number\_of\_ATP\_systems \times minimal\_time\_limit}$$

6 Resource Limits

6.1 Time Resource

Problems may be presented to an ATP system one-at-a-time or in batches. In both cases the ATP system will have to deliver results within some time limit. Also, the time available for the competition is limited. It is thus necessary to impose a time limit in the competition, either on individual proof attempts or on the total time for all proof attempts.

If the problems are presented as a batch and an overall time limit is used, that will allow the ATP systems to examine all the problems and perform once-only tuning before starting the proof attempts. In terms of the competition, an overall time limit would simplify the winner assessment; the system that proves the most theorems within the time limit is the winner, and if more than one system proves the same number of theorems then the total time taken to prove those theorems is compared (this ranking scheme has been adapted in Ranking Scheme A in Section 7.3). In applications of ATP systems, problems are likely to be presented one-at-a-time, as and when the problems are generated. Even if an ATP system is presented with a batch of problems, they are likely to be treated individually, again requiring individual time limits to decide when an unsuccessful attempt to find a solution should be abandoned[14]. Also, an evaluation based on individual time limits still provides relevant performance information for the batch case, but not vice versa. Decision : In the ATP competition, a time limit will be imposed on individual proof attempts.

A minimal time limit is necessary so that the ATP systems spend most (say 95%) of their time on proof search. This ensures that their deductive capabilities are compared, rather than their initialization and termination efficiencies. The non-proof time of each ATP system will be estimated by measuring the time that it takes to prove a trivial theorem, e.g., $p \wedge \neg p$. The geometric average of the system's non-proof times will be taken to determine the *non_proof_time*. Then, to ensure that the systems spend 95% of their time searching for a proof, the $minimal\_time\_limit = non\_proof\_time \times 20$.

The time limit to be imposed on each proof attempt depends on the resources available for the competition, the number of ATP systems, and the number of problems (which in turn is determined by the minimal time limit; see Section 5.4). Decision : The time limit to be imposed on each proof attempt in the ATP competition will be:

[14] Processing all problems on a single processor via multiprogramming would not require imdividual time limits, but realistically allows only as many proving processes as can fit in memory.

$$time\_limit = \frac{number\_of\_workstations \times time\_for\_competition}{number\_of\_ATP\_systems \times number\_of\_problems}$$

The way that the *number_of_problems* is determined in Section 5.4 ensures that $time\_limit \geq minimal\_time\_limit$. In Section 7.2 it is explained that the granularity of timing will be one second. Thus the minimal *non_proof_time* is one second, so $time\_limit \geq minimal\_time\_limit \geq 20$ seconds.

It is important to note that the imposition of a time limit means that the competition results will have to be viewed as modulo that time limit. A different time limit may produce a different ranking of the systems. Results could be computed for a range of time limits, up to the time limit used, and this may give further insight into the relative abilities of the systems.

If the ranking remains stable for a reasonable range of time limits, this provides empirical evidence that the ranking reflects a time limit independent ranking of the ATP systems. Decision : The ATP competition results will be plotted as a function of the time limit.

6.2 Hardware and Software Resources

ATP systems have been developed using a wide range of hardware and software resources. The most commonly used computers in ATP research are UNIX based workstations. Many current ATP systems do, or can be made to, run on such workstations. Other hardware used includes PCs, LISP machines, parallel computers, and distributed networks of computers. The ATP competition could limit itself to the common workstation environment, or could allow the use of other hardware.

Running all ATP systems on the same hardware allows a direct comparison of the systems, and is thus the preferred environment for the ATP competition. The host institution of the competition is able to provide a homogeneous set of UNIX workstations, thus making this option available. However, some ATP systems have been specially designed for a particular computer, and cannot easily be ported. The host institution will be unable to supply the range of specialized hardware that the ATP systems could use. Although a direct comparison of ATP systems running on widely disparate hardware is not meaningful, performance data from such systems does allow a comparison of the overall performance from a user's perspective. Since it would be of interest to the ATP community to see the performance of all systems, ATP systems that run on specialized hardware need to be catered for in the ATP competition. Decision : Each of the ATP competition categories will be divided into two subcategories : General Hardware and Special Hardware. ATP systems in the General Hardware subcategories will all be executed on the same type of UNIX workstation, as supplied by the host institution. ATP systems in the Special Hardware subcategories will be allowed to use any hardware brought to the competition or accessible via the InterNet.

Common languages used to implement ATP systems are C, Prolog, and LISP. Other languages have also been used. For the ATP competition, the major difference between implementation languages is whether the code is interpreted, as is often the case for Prolog and LISP, or compiled to machine code, as for C. ATP systems that run in an interpreted environment are disadvantaged by the time taken to start up the interpreter, and by the relative slowness of interpreters compared to compiled code. It would be possible to provide separate categories for the two types of implementation.

There are some implementation environments that do not fall clearly into one or other of the two types. Many vendors of Prolog and LISP interpreters also provide compilers, thus alleviating the problems to at least some extent. Further, it is also important to acknowledge the effort taken to code an ATP system in a language such as C, and the benefit derived is earned. Decision : The ATP competition will not be concerned with the languages used to implement the ATP systems.

7 System Evaluation

7.1 Winner Assessment

It is arguable if an overall "winner" should be assessed. The reason for this is that potential entrants may be frightened off by the fact that some other system may be believed to perform better than theirs. The ATP competition could avoid this problem by simply reporting the performance data of the ATP systems and allowing observers to draw their own conclusions.

Not determining a winner would leave out much of the spice of the competition, and would remove much of the motivation for improving the capabilities of current systems[15]. It must be remembered that the assessment done in the competition will be with respect to the decisions made in this paper, i.e., the winner may not be the best ATP system in a general sense. Rather, the ranking will simply provide feedback to potential users about which system is currently the most adequate for their particular use. An useful byproduct of the competition will be the charting of the performance data; that may reveal interesting properties of systems that do not win. Decision : The ATP competition will both determine a specific ranking of the ATP systems and present a listing of the performance data.

Two possible ways of determining a ranking are to have a quantitative ranking scheme or to have a judging panel.

A quantitative ranking scheme can be implemented mechanically (as a computer program, such as the RoboJudge described in [AKK93]) and checked for bias, but has little flexibility. A judging panel is more flexible, can take into account more aspects, and can impose intuitive judgment on the results. The benefits of both approaches are desirable. Decision : The ranking of ATP systems in the ATP competition will be done by quantitatively evaluating system performance and having that evaluation vetted by a panel of ATP researchers.[16]

7.2 Performance Metrics

There are many criteria by which ATP systems can be evaluated. Issues such as completeness, soundness, correctness, and proof performance are all of direct interest to ATP researchers. Soundness (correctness) should be required in-so-far as falsely stating theoremhood of a non-theorem renders the system untrustworthy, and therefore useless. Completeness plays a role in-so-far as systems capable of proving more theorems should be ranked higher than systems proving less. Incomplete systems, systems that fail to find a proof due to resource constraints, and those that crash because they are bugged must be compared (see Section 3.2). Proof performance can be measured in many ways, including runtime, number of inference steps (proof size), and proof tree depth (if proofs are represented as trees). In the broader context of computing science, issues such as software quality (portability, modularity), user friendliness, execution environment, etc., may also be of interest. It is necessary to decide what criteria should be used to evaluate the ATP systems in the ATP competition.

[15]Would you train hard for the Olympics if you could not stand on a pedestal when you won?

[16]The panel will consist of Peter Andrews, Alan Bundy, and Jeff Pelletier.

The broader issues of computer science are of lesser interest to the ATP community, and factoring them into the evaluation would blur the focus on ATP. Also, no generally accepted metrics exist for the broad issues, and evaluation would become a matter of taste. Decision : The ATP competition will evaluate the ATP systems in terms of issues directly relevant to the ATP community.

The quantitative ranking scheme needs to observe performance metrics of the ATP systems, and combine those values to produce a ranking of the systems. Quantitative performance metrics available for evaluation are:

- Calculus-relative measures
  - number of inferences performed (ideally split according to the inference rules used)
  - number of unifications performed (ideally split into failed and succeeded unifications)
  - proof size (e.g., length, depth)
- Absolute measures
  - number of problems solved
  - runtime (possibly split into startup time and search time)
  - memory usage

For the purposes of the ATP competition the measures used must be independent of the ATP systems, so that values can be meaningfully compared. Measures such as the number of inference steps and proof length are not suitable because the units of measure can vary from calculus to calculus, and from implementation to implementation. System independent measures that are readily obtained are the number of problems solved, runtime, and memory usage.

The number of problems solved and the runtime are direct indicators of the quality of an ATP system. Memory usage is important in so far as it can affect runtime. If an ATP system requires less than the available memory, then the effect on runtime is negligible. If an ATP system requires more than the available memory, then either the system cannot handle the problem, or swapping increases the wall-clock time of the computation. Therefore the effect of memory usage can be subsumed in a proper definition of runtime. Decision : The number of problems solved and the runtime will be used for winner assessment in the ATP competition. The memory usage will also be recorded and presented.

The are two reasonable ways of measuring runtime: CPU time taken, and wall-clock time taken. The advantage of CPU time is that it is easy to measure and it is independent of system influences such as external load, daemon processes, memory size, disc performance. However, CPU time seems inappropriate if swapping occurs because it does not reflect the user's perception of the runtime. Wall-clock time takes swapping into account, but is dependent on system influences, and therefore can be difficult to measure in a reproducible manner. From the developer's viewpoint, CPU time is more interesting. From the user's viewpoint, wall-clock time is more relevant. Decision : For runtime, both CPU time and wall-clock time will be measured and reported in the ATP competition.

In the General Hardware category, the choice of which time measurement is used for winner assessment will depend on the computing environment. If no swapping occurs then: Decision : In the General Hardware category CPU time will be used for winner assessment. If swapping does occur, and the wall-clock time measurements are stable and representative of the time required for the computation (essentially CPU time plus time required for swapping), then: Decision : Wall-clock time will be used for winner assessment. Otherwise: Decision : CPU time plus an estimate of the swapping time will be used for winner assessment.

In the Special Hardware category CPU timings are typically incomparable. In contrast, wall-clock times can be compared, in the context of the hardware used. Decision : In the Special Hardware category wall-clock time will be used for winner assessment.

The precision of time measurement on a computer system is limited. In order to reduce the effect of this, and to emphasize significant rather than marginal differences, the competition timing will be discrete. Decision : In the ATP competition timing will be done in units of one second. In particular, the minimal time a system can take to find a proof is one second.

7.3 Ranking Schemes

The ATP competition must have a ranking scheme that combines the performance metrics to produce a ranking of the ATP systems. The ranking scheme must have certain properties:

Monotonicity requirements:

- The ranking improves as the number of problems solved increases.
- The ranking improves as the time taken to solve the problems decreases.

Required invariants:

- For systems that solve the same number of problems, the one that takes the least time to solve them obtains a better ranking.
- For systems that take the same amount of time, the one that solves the most problems obtains a better ranking.
- Systems which solve the same number of problems in the same total amount of time obtain the same ranking.

The issue of how to combine the performance metrics to obtain a ranking scheme is a contentious one. Two different quantitative schemes for determining a system ranking, representing different emphases between number of problems solved and time taken, have been developed. For both schemes higher scores are better. Both schemes use the $total_time_taken$as a parameter. This value is the sum of the time taken over all problems, including those for which no proof is found, in which case the time limit is used.

Ranking Scheme A. This ranking scheme focusses on the ability to find as many solutions as possible. The idea is to rank the systems according to the number of problems solved, and to further differentiate by considering the number of erroneous claims that no proof exists, the number of system errors, and the runtime. Since for erroneous claims and system errors the full time limit is used as system runtime, considering only the runtime accounts for all the latter issues. Thus, each system is given a score:

$$Score_A = number\_of\_problems\_solved - \frac{total\_time\_taken - best\_total\_time\_taken}{max\_time\_possible}$$

where $best\_total\_time\_taken$ = least $total\_time\_taken$ by a system that solves the same number of problems, and $max\_time\_possible = number\_of\_problems \times time\_limit$. Achievable scores range from 0 to the number of theorems.

Ranking Scheme B. This scheme measures solutions per unit time. Each system is given a score:

$$Score_B = \frac{number\_of\_problems\_solved}{total\_time\_taken}$$

Achievable scores range from 0 to 1.

The schemes place different emphasis on the two performance metrics: scheme A puts emphasis on the number of problems solved while scheme B balances the emphasis. This difference can lead to different rankings of ATP systems. Under scheme A a system which solves the most problems will win, while under scheme B this is not necessarily so. Each ranking scheme suits specific user requirements, and thus neither can be ignored. Decision : In the ATP competition, in each competition category a winner according to Scheme A and a winner according to Scheme B will be determined. As well as the scheme specific rankings, an overall assessment is desirable. However, any attempt to combine the scores from Schemes A and B requires a trade-off between the two metrics: number of problems solved and time taken. Any particular trade-off is unlikely to be acceptable to everyone. However, if a particular ATP system is the winner in both schemes, then it is superior in terms of both performance metrics. Decision : In the ATP competition, iff a single system is the winner of a category according to both Schemes A and B, it will be the overall winner of the category.

Note: If only one ATP system is registered for a particular category, no winner will be announced for that category, but the results for that system will still be presented.

Example. The example in Table 2 gives some system runtime values ("-" denotes an unsolved problem). The example is designed to illustrate the differences between the schemes, to the extent that each scheme produces a different ranking of the systems. It also shows the scores and rankings that are produced by the different schemes, assuming that $time\_limit = 20$.

System	Runtime for problem					Scores by scheme		Ranking by scheme	
	P1	P2	P3	P4	P5	RS A	RS B	RS A	RS B
S1	20	20	20	20	20	5	0.050	1.	3.
S2	1	5	10	10	-	4	0.087	2.	1.
S3	1	1	1	-	-	3	0.070	3.	2.

Table 2: Example showing the runtimes of three different systems on five different problems and the unscaled scores and rankings produced by the ranking schemes.

8 Conclusion

From the numerous issues that need to be resolved for organizing a useful competition, and the impossibility of making an indisputable decision in several cases, it becomes clear that alternative competition designs are possible. However, we believe that this rationally planned competition will provide most of the benefits hoped for in Section 1.

We see a clear potential for improved future competitions, by extending their scope; additional coverage of FOF systems (instead of just CNF systems) and model generation (instead of just theorem proving) are the most important issues. However, it seems preferable to start with a core design for a competition, and to add such extensions when more experience has been gained.

After more than 30 years of research, automated theorem proving abounds with techniques developed and ideas proposed. The future requires that these techniques and ideas be evaluated to determine which are viable, and to integrate them into systems which are far more flexible than the simple monolithic and compositional systems available today. The most important challenge ahead in theorem proving research will be to provide adequate control, a subject still in its infancy, since it is difficult to approach theoretically.

For all these goals, system evaluation is a crucial research tool, and competitions such as this one will provide stimulus and insight that can lay the basis for the development of future ATP systems.

Acknowledgments. We are indebted to many researchers who discussed and commented on our ideas; especially Alan Bundy, Bertram Fronhöfer, Reinhold Letz, Bill McCune, David Plaisted, and Christoph Weidenbach.

References

[AKK93] O. Astrachan, V. Khera, and D. Kotz. The Internet Programming Contest: A Report and Philosophy. In *Proceedings of the 24th SIGCSE Technical Conference (Indianapolis, USA, 1993)*, SIGCSE Bulletin, pages 48–52. ACM Press, 1993.

[AL94] O.L. Astrachan and D.W. Loveland. Measuring the Performance of Automated Theorem Provers. CADE-12 Workshop 2C (Evaluation of Automated Theorem Proving Systems), Nancy, 1994.

[Ale95] G. Alexander. *Proving First-Order Equality Theorems with Hyper-Linking.* PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, USA, 1995.

[BCP94] C. Bourely, R. Caferra, and N. Peltier. A Method for Building Models Automatically. Experiments with an extension of OTTER. In *Proceedings of the 12. International Conference on Automated Deduction (CADE-12)*, pages 72–86. Springer LNAI 814, 1994.

[BKB92] M. Buro and H. Kleine-Büning. Report on a SAT competition. Technical Report Reihe Informatik, Bericht Nr. 110, FB 17 - Mathematik/Informatik Universität Paderborn, November 1992.

[BM90] R.S. Boyer and J S. Moore. A Theorem Prover for a Computational Logic. In *Proceedings of the 10th International Conference on Automated Deduction* , number 449 in Lecture Note in Artificial Intelligence, pages 1–15, New York, USA, 1990. Springer-Verlag.

[CAB+86] R. Constable, S. Allen, H. Bromly, W. Cleaveland, J. Cremer, R. Harper, D. Howe, T. Knoblock, N. Mendler, P. Panangaden, J. Sasaki, and S. Smith. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, 1986.

[CIJ+94] E. Chen, S. Istrail, D. Johnson, J. Kececioglu, Messing J., J. Nadeau, P. Pevzner, P. Rice, M. Vingron, and M. Waterman. The 4th DIMACS Interantional Algorithm Implementation Challenge. URL http://dimacs.rutgers.edu/challenge.html, 1994.

[CP94] H. Chu and D. Plaisted. Semantically Guided First-order Theorem Proving using Hyper-linking. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, number 814 in Lecture Notes in Artificial Intelligence, pages 192–206. Springer-Verlag, 1994.

[CRRP90] J. Comer, R. Rinewalt, P. Ryan, and W.B. Poucher. The ACM Scholastic Programming Contest - 1977 to 1990. In *Proceedings of the 21st SIGCSE Technical Symposium on Computer Science Education*, volume 22(1) of *SIGCSE Bulletin*, page 256. ACM Press, 1990.

[Dem96] B. Demoen. The 2nd Annual Prolog Programming Contest. *The Newsletter for the Association for Logic Programming*, 9(2):3–4, 1996.

[Dun94] U. Dunker. Search Space and Proof Complexity of Theorem Proving Strategies, 1994. in: Proceedings of CADE-12 Workshop 2C.

[FGT90] W. Farmer, J. Guttman, and J. Thayer. IMPS: An Interactive Mathematical Proof System. In *Proceedings of the 10. International Conference on Automated Deduction (CADE-10)*, pages 653–654. Springer LNAI 449, 1990.

[GM93] M. Gordon and T. Melham. *Introduction to HOL, a Theorem Proving Environment for Higher Order Logic.* Cambridge University Press, 1993.

[HKK+94] X. Huang, M. Kerber, M. Kohlhase, E. Melis, D. Nesmith, J. Richts, and J. Siekmann. Ω-MKRP: A Proof Development Environment. In *Proceedings of the 12. International Conference on Automated Deduction (CADE-12)*, pages 788–792. Springer LNAI 814, 1994.

[HPC93] S. Hanks, M.E. Pollack, and P.R. Cohen. Benchmarks, Test Beds, Controlled Experimentation and the Design of Agent Architectures. *AI Magazine*, 14(4):17–42, 1993.

[HRS90] M. Heisel, W. Reif, and W. Stephan. Tactical Theorem Proving in Program Verification. In *Proceedings of the 10. International Conference on Automated Deduction (CADE-10)*, pages 117–131. Springer LNAI 449, 1990.

[Ker91] M. Kerber. How to Prove Higher Order Theorems in First Order Logic. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 137–142, 1991.

[Let93] R. Letz. On the Polynomial Transparency of Resolution. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 123–129, 1993.

[LP92] S-J. Lee and D.A. Plaisted. Eliminating Duplication with the Hyper-Linking Strategy. *Journal of Automated Reasoning*, 9(1):25–42, 1992.

[LSBB92] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.

[McC94] W.W. McCune. OTTER 3.0 Reference Manual and Guide. Technical Report ANL-94/6, Argonne National Laboratory, 1994.

[MMPS94] D. Michie, S. Muggleton, D. Page, and A. Srinivasan. 3 Inductive Learning Competitions. URL ftp://ftp.comlab.ox.ac.uk/pub/Packages/ILP, 1994.

[New94] M. Newborn. The 24th ACM International Computer Chess Championship. *Journal of the International Computer Chess Association*, 17(3):159–164, 1994.

[Ohl91] H.-J. Ohlbach. Semantics Based Translation Methods for Modal Logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[ORS92] S. Owre, J.M. Rushby, and N Shankar. PVS: A Prototype Verification System. In *Proceedings of the 11. International Conference on Automated Deduction (CADE-11)*, pages 748–752. Springer LNAI 607, 1992.

[Ove93] R. Overbeek. The CADE-11 Competitions: A Personal View. *Journal of Automated Reasoning*, 11:315–316, 1993.

[Pau90] L. Paulson. Isabelle: The Next 700 Theorem Provers. *Logic and Computer Science*, pages 361–386, 1990.

[Pel91] Pelletier, F.J. The Philosophy of Automated Theorem Proving. In *Proceedings of the 12. International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 1039–1045, 1991.

[Pla94] D.A. Plaisted. The Search Efficiency of Theorem Proving Strategies. In *Proceedings of the 12th International Conference on Automated Deduction (CADE-12)*, pages 57–71. Springer-Verlag LNAI 814, 1994.

[Por94] F.D. Portoraro. Symlog: Automated Advice in Fitch-style Proof Construction. In *Proceedings of the 12. International Conference on Automated Deduction (CADE-12)*, pages 802–806. Springer LNAI 814, 1994.

[PS94] C. Prehofer and K. Slind. Theorem Proving in Interactive Verification Systems, 1994. in: Proceedings of CADE-12 Workshop 2C.

[SGar] H. Schütz and T. Geisler. Efficient Model Generation through Compilation. In M. McRobbie and J. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence. Springer-Verlag, To appear.

[SS95] G. Sutcliffe and C.B. Suttner. ATP System Results for the TPTP Problem Library (upto TPTP v1.1.3), 1995. Technical Report 95/16, Department of Computer Science, James Cook University, Townsville, Australia; Technical Report AR-95-NN, Institut für Informatik, Technische Universität München, Munich, Germany.

[SS96] C.B. Suttner and G. Sutcliffe. The TPTP Problem Library (TPTP v1.2.1), 1996. Technical Report AR-95-03, Institut für Informatik, Technische Universität München, Munich, Germany; Technical Report 95/6, Department of Computer Science, James Cook University, Townsville, Australia.

[SSY94] G. Sutcliffe, C.B. Suttner, and T. Yemenis. The TPTP Problem Library. In *Proceedings of the 12. International Conference on Automated Deduction (CADE)*. Springer LNAI 814, 1994.

[Wad90] H.M. Wadsworth. *Handbook of Statistical Methods for Engineers and Scientists*. McGraw-Hill, 1990. p. 9.17 gives the relevant example for CADE13 paper.

[WG92] T.C. Wang and A. Goldberg. RVF: An Automated Formal Verification System. In *Proceedings of the 11. International Conference on Automated Deduction (CADE-11)*, pages 735–739. Springer LNAI 607, 1992.

[WGRar] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER. In M. McRobbie and J. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence. Springer-Verlag, To appear.

[WWM+90] L. Wos, S. Winker, W.W. McCune, R. Overbeek, E. Lusk, and R. Steven. Automated Reasoning Contributes to Mathematics and Logic. In *Proceedings of the 10. International Conference on Automated Deduction (CADE-10)*, pages 485–499. Springer LNAI 449, 1990.