ELSEVIER

Artificial Intelligence 131 (2001) 39–54

Artificial Intelligence

www.elsevier.com/locate/artint

Evaluating general purpose automated theorem proving systems

Geoff Sutcliffe [a,*], Christian Suttner [b]

[a] University of Miami, Department of Computer Science, P.O. Box 248154, Coral Gables, FL 33124, USA
[b] Schweppermannstr. 2, München, Germany

Received 12 October 1999; received in revised form 14 August 2000

Abstract

A key concern of ATP research is the development of more powerful systems, capable of solving more difficult problems within the same resource limits. In order to build more powerful systems, it is important to understand which systems, and hence which techniques, work well for what types of problems. This paper deals with the empirical evaluation of general purpose ATP systems, to determine which systems work well for what types of problems. This requires also dealing with the issues of assigning ATP problems into classes that are reasonably homogeneous with respect to the ATP systems that (attempt to) solve the problems, and assigning ratings to problems based on their difficulty. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: ATP system evaluation; ATP problem classification; ATP problem evaluation

1. Introduction

Automated Theorem Proving (ATP) is concerned with the development and use of systems that automate sound reasoning: the derivation of conclusions that follow inevitably from facts. This capability lies at the heart of many important computational tasks. Current ATP systems are capable of solving non-trivial problems, e.g., EQP [18] solved the Robbins problem [17]. In practice however, the search complexity of most interesting problems is enormous, and many problems cannot currently be solved within realistic resource limits. Therefore a key concern of ATP research is the development of more powerful systems, capable of solving more difficult problems within the same resource limits. In order to build

[*] Corresponding author.
E-mail addresses: geoff@cs.miami.edu (G. Sutcliffe), csuttner@antfactory.com (C. Suttner).

more powerful systems, it is important to understand which systems, and hence which techniques (understanding that a system can be understood as a collection and combination of techniques), work well for what types of problems. This knowledge is a key to further development, as it precedes any investigation into why the techniques and systems work well or badly. This knowledge is also crucial for users: given a specific problem, a user would like to know which systems are most likely to solve it. This paper deals with the empirical evaluation of general purpose ATP systems, to determine which systems work well for what types of problems. This requires also dealing with the issues of assigning ATP problems into classes that are reasonably homogeneous with respect to the ATP systems that (attempt to) solve the problems, and assigning ratings to problems based on their difficulty.

Analytic approaches to ATP system evaluation, such as presented in [7,15,20], provide insights into theoretical system capabilities. These approaches investigate metrics such as search space duplication, search tree size, Herbrand multiplicity, number of variables to bind, and connectivity. However, complete analysis of the search space at the 1st order level is, of course, impossible (for otherwise 1st order logic would be decidable). It is therefore necessary to make empirical evaluations of ATP systems. Empirical evaluation of individual ATP systems is a standard activity of system developers, and their results are frequently reported in the ATP literature, e.g., in the *Journal of Automated Reasoning*, and in the proceedings of the International Conference on Automated Deduction (CADE). As well as providing raw performance data, such reports often include comparisons against other known ATP systems. Despite mostly good intentions, these evaluations are often performed without consideration of the underlying issues, done in an ad hoc fashion, and are selective so as to highlight the particular strengths of the ATP system. Inappropriate evaluation can be seriously damaging to a field of research, because bad ideas may appear good and be perpetuated, while good ideas may be discarded. This leads to slow progress in the field, and a perception that the direction of research and development may be misguided. This paper provides the information required to perform meaningful ATP system evaluations, and in the long term to support and measure progress in ATP.

There are two main forms of empirical evaluation. The first form selects a few specially targeted problems from a problem domain of interest, and an ATP system is carefully tuned until it can solve those problems. This first form evaluates the extent to which the ATP system can be configured for a special purpose. The second form of evaluation selects a large representative sample of problems, and ATP systems are tested on the problems in a general purpose setting. This form of evaluation considers the general usefulness of the ATP systems for the type of problem. This paper deals with the second form of empirical evaluation of ATP systems. The paper carefully considers the issues underlying general purpose ATP system evaluation, and then provides two methodic schemes for the impartial empirical evaluation of ATP systems. The system evaluation schemes are linked by a scheme for evaluating the difficulty of ATP problems. General principles that should be considered in the empirical evaluation of search based systems (typically for artificial intelligence), such as described in [11–13], have contributed to the development of the schemes. Even more general work, e.g., [8], has also provided useful ideas.

The ATP system evaluation schemes described in this paper have been applied to current 1st order ATP systems, and have provided a way to substantiate realistic and meaningful claims made by ATP system developers. The evaluations have identified specialist and general capabilities of ATP systems, and have stimulated ATP research and development. Evaluating ATP systems within problem classes has made it possible to identify which systems are best suited to (the characteristics of the problems in) each class. This knowledge is used to provide users with system recommendations for attempting to solve a problem, based on the problem's characteristics. In addition to contributing to system evaluation, the evaluation of ATP problems simplifies problem selection according to a user's intentions, and over the years, changes in problem ratings provide a quantitative indicator of advancement in ATP.

Section 2 parameterises ATP systems according to various measures, and specifies the types of ATP systems the evaluation schemes have been developed for. Section 3 discusses criteria that can be used for evaluating ATP systems. The evaluation schemes require testing of the ATP systems on test problems. Section 4 discusses the appropriate selection and presentation of test problems, and Section 5 discusses the necessary classification of test problems. Sections 6 and 8 each describe a scheme for evaluating ATP systems according to their ability to solve problems, while Section 7 describes a scheme for evaluating ATP problems. Section 9 considers the evaluation of ATP systems according to their resource usage. The paper is concluded by Section 10.

2. Types of ATP systems

There are many types of ATP systems, designed for different types of problems, operating in different ways, and producing a range of different outputs. The evaluation of all types of ATP systems is interesting, but different evaluation processes are needed depending on the nature of the systems. This section considers some fundamental parameters by which ATP systems can be classified, and specifies the parameters relevant to the evaluation schemes described in this paper.

2.1. Language

Many different logics are used in ATP, such as classical logic, modal logics, and temporal logics. Further, many logics can be specified at various orders, such as propositional, 1st order, etc. The wide-spread development and use of ATP systems for 1st order classical logic suggests that their evaluation is useful and of interest. This paper deals with the evaluation of such systems.

2.2. Degree of automation

From a user's viewpoint, ATP systems are of two types: fully automatic systems and interactive systems.[1] Currently, and presumably in the future, the most important

[1] Astrachan and Loveland [1] 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. systems, 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.

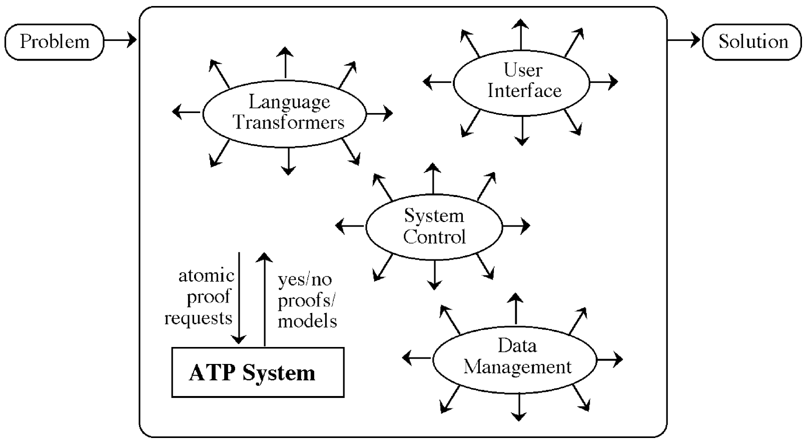

Fig. 1. Application of fully automatic ATP systems.

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 Fig. 1. The environment may call on the fully automatic ATP system to solve subproblems within the operation of the overall system. Examples of fully automatic ATP systems, for 1st order classical logic, are Bliksem [6], Gandalf [30], Otter [16], SETHEO [14], SPASS [32], Vampire [31], and E [22]. Interactive systems find application in hardware and software verification, formalization of informal mathematical proofs, teaching logic, and as tools of mathematical and meta-mathematical research. Examples of interactive systems are Analytica [2], ILF [5], ΩMEGA [4], KIV [21], Isabelle [19], HOL [10], and Coq [3]. Interactive systems typically embed some form of fully automatic ATP system as a subsystem, 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.

In this paper attention is limited to fully automatic ATP systems, and they are referred to simply as ATP systems. The decision to focus on fully automatic systems does not exclude the possibility of evaluating interactive systems. Indeed, such evaluation would be of interest, if the difficulties involved can be adequately resolved. Part of such an evaluation would necessarily include evaluation of the fully automatic subsystems.

2.3. Soundness

Sound ATP systems return only correct solutions to problems. Soundness can be assumed for extensively tested systems, at least if a large number of the solutions produced have been verified. Unsound systems are useless, and soundness is therefore required.

2.4. Completeness

Complete systems always return a solution if one exists. An ATP system may be known to be incomplete from a theoretical perspective, due to known weaknesses in the calculus or search strategy. An ATP system may also be incomplete due to bugs in the implementation, possibly causing it to crash on some problems. Even if an ATP system is theoretically complete and correctly implemented, the finite amount of resources allocated to any particular ATP system run means that every search for a solution has limited completeness.

From a user's point of view, there is little difference between not finding a solution due to theoretical incompleteness, not finding a solution due to a implementation bug, and not finding a solution due to a resource limit being reached. In all cases the user learns nothing about the solution to the problem. In practice, a theoretically incomplete or bugged system may solve more problems within some resource limit than a complete and bug free system, and therefore may be more useful. Systems may therefore be evaluated even if theoretically incomplete or bugged.

2.5. Solutions

Depending on its generality and purpose, there are various responses that an ATP system may make when given a problem. A solution (a proof or a model) may be returned, or the system may give only an assurance that a solution exists. There are some ATP systems that conserve resources by not building a solution during their search, e.g., completion based systems do not build a proof. Such systems retain only enough information to build a solution later if required. Other ATP systems are not able to build a solution, and can give only an assurance that a solution exists, e.g., systems that establish satisfiability through a failed complete search for a refutation cannot build a model.

There is no evidence that the use of ATP systems as embedded systems typically either requires or does not require delivery of proofs or models. Further, once the existence of a solution is assured, a user is then prepared to expend large amounts of resources to obtain the proof or model if it is required, i.e., the assurance of existence of a solution is the first important output. Thus in the evaluation schemes presented in this paper, an ATP system is considered to have solved a problem when it provides an assurance that a solution exists.

3. Criteria for evaluation

Section 2 explains that this paper deals with the evaluation of fully automatic, sound, possibly incomplete or bugged, ATP systems for 1st order classical logic, in terms of their ability to provide an assurance that a solution exists. Two intuitively acceptable criteria for the empirical evaluation of ATP systems are:

- What problems can they solve?
- What computational resources (CPU capability and CPU time) and memory resources do they need to find the solutions?

The first criterion, what problems the systems can solve, measures the completeness of the systems. If no resource limits are imposed then correctly implemented theoretically complete systems solve all problems, providing no differentiation between the systems. In reality however, as is explained in Section 2.4, issues that affect (practical) completeness are calculus, search control, implementation, and resource limits. The adequate supply of resources is not under the control of ATP systems, and needs to be factored out of system

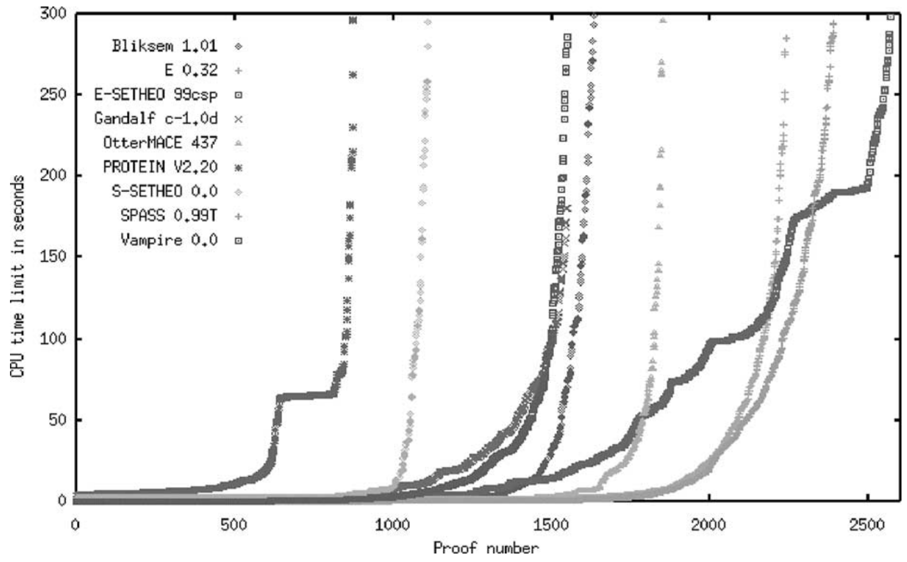

Fig. 2. Proof number vs. CPU time for ATP systems.

evaluation. The first criterion therefore apparently needs to be refined to "What problems can they solve, modulo realistic resource limits?" However, as is explained next, adequate evaluation can be achieved without this added qualification.

3.1. Resource limit independence

The TPTP (Thousands of Problems for Theorem Provers) problem library is a library of test problems for ATP systems [26]. Some researchers who have tested their ATP systems over the entire TPTP problem library have contributed their performance data to the TPTP results collection [25]. The performance data in the collection is provided by the individual system developers, which means that the systems have been tested using a range of computational and memory resource limits. Analysis shows that the differences in resource limits do not significantly affect which problems are solved by each ATP system. Fig. 2 illustrates this point.

Fig. 2 plots the CPU times taken by several contemporary ATP systems to solve TPTP problems, for each solution found, in increasing order of time taken.[2] The relevant feature of these plots is that each system has a point at which the time taken to find solutions starts to increase dramatically. This point is called the system's *Peter Principle Point* (PPP), as it is the point at which the system has reached its level of incompetence. Evidently a linear increase in the computational resources beyond the PPP would not lead to the solution of significantly more problems. The PPP thus defines a "realistic computational resource limit" for the system. From an ATP perspective, the PPP is the point at which

[2] The numbers of solutions found are not comparable, as the systems attempted the SPC in different TPTP versions.

the ATP system gets lost in its quickly growing search space. Even though there may be enough memory to represent the search space at the PPP, the system is largely unable to find a solution within the space. The point thus also defines a "realistic memory resource limit". Therefore, provided that enough CPU time and memory are allowed for the ATP system to pass its PPP, a usefully accurate measure of what problems it can solve within realistic resource limits is achieved.[3] For ATP system evaluation, this insight means that meaningful resource limited system evaluation is possible using the criterion "What problems can they solve?".

The performance data in the TPTP results collection is produced with adequate resource limits, making this data a suitable basis for system evaluation. Sample evaluations using the TPTP performance data are given in Sections 6, 7, and 8.

4. ATP problems

4.1. Source of problems

Currently there are few "real" applications of 1st order ATP, and therefore there is no corpus of application problems that can be used as a basis for empirical ATP system evaluation (current applications of ATP, such as software and hardware verification, largely use propositional techniques). Since the first release of the TPTP in 1993, many researchers have used the TPTP as an appropriate and convenient basis for testing their ATP systems. Although other test problems do exist and are sometimes used, the TPTP is now the de facto standard for testing 1st order ATP systems. The TPTP is large enough to obtain statistical significance, and spans a diversity of subject matters. The TPTP is the best available collection representing general purpose applications of ATP, and thus is the best source of problems for the evaluation of general purpose ATP systems. As the real applications of 1st order techniques grow, those types of problems are being added to the TPTP, so that the TPTP remains an appropriate source of problems for evaluating 1st order ATP systems. The TPTP also has an organizational structure designed for testing ATP systems.

4.2. Encoding of problems

For a fair evaluation of ATP systems, it is necessary to avoid problems that are explicitly suited or ill-suited to any of the ATP systems. Problems may be *biased* towards or against an ATP system through the style of the problem formulation, omission of axioms, or by the addition of useful lemmas. In the TPTP a problem is *standard* if it uses a complete axiomatization of an established theory, no lemmas have been added, and it is not biased. Standard TPTP problems are suitable for ATP system evaluation. In addition, the TPTP contains problems whose axioms do not capture any established theory, but whose axioms are not biased. These are called *especial* problems, and are also suitable for ATP system evaluation. The use of standard and especial problems reflects potential application, in which 'vanilla' axiomatizations would be used, at least initially.

[3] These system-defined resource boundaries are also important from a user perspective, as solutions are unlikely to be found beyond these boundaries.

4.3. Presentation of problems

The performance of decent ATP systems should not be very dependent on the specific presentation of the problems, in this case the presentation in the TPTP. The performance should be independent of the predicate and function symbols used, and of formula, clause, and literal ordering. In cases where a syntactically identifiable theory is built into an ATP system, e.g., equality, the axioms of that theory should be removed from the input in advance, as would be the case in an application. Conversely, if an established theory is known to be a logical consequence of the input, e.g., equality is a logical consequence of set theory, this theory may be added to the input or built into an ATP system. Finally, formula type information, such as provided by the TPTP, i.e., one of `axiom`, `hypothesis`, `conjecture`, or `theorem`, which would normally be available in any application of ATP, should be available to ATP systems when they are evaluated.

5. Specialist problem classes

Problems have easily identifiable logical, language, and syntactic characteristics. Various ATP systems and techniques have been observed to be particularly well suited to problems with certain characteristics (this specialization is sometimes by design, but often without intent). For example, everyone agrees that special techniques are deserved for problems with equality, and the CASC-15 results [23] showed that problems with true functions, i.e., with an infinite Herbrand universe, should be treated differently from those with only constants, i.e., essentially propositional problems. Due to this specialization, empirical evaluation of ATP systems must be done in the context of problems that are reasonably homogeneous with respect to the systems. These sets of problems are called *Specialist Problem Classes* (SPCs), and are based on problem characteristics. Evaluation of ATP systems within SPCs makes it possible to say which systems work well for what types of problems. This identifies specialist capabilities of ATP systems, while general capabilities can be inferred from the separate SPC capabilities. Also, ATP systems that are specialized out of an SPC, e.g., some ATP systems cannot deal with FOF, need not be evaluated within that SPC.

When defining SPCs it is necessary to decide on their granularity. The finest level of granularity is individual problems, which reduces system evaluation within an SPC to whether or not each system solves the problem. The coarsest level of granularity is the entire evaluation problem set, which defeats the notion. The appropriate level of SPC granularity for ATP system evaluation is that at which the next coarser granularity would merge SPCs for which the systems have distinguishable behaviour (i.e., the ATP systems are specialized to this level of granularity), and at which the next finer level of granularity would divide an SPC for which the systems have reasonably homogeneous behaviour (i.e., the ATP systems are not specialized below this level of granularity).

The choice of what problem characteristics are used to form the SPCs is based on community input and analysis of system performance data. The range of characteristics that have so far been identified as relevant are:

- Theoremhood: Theorems vs. Non-theorems;

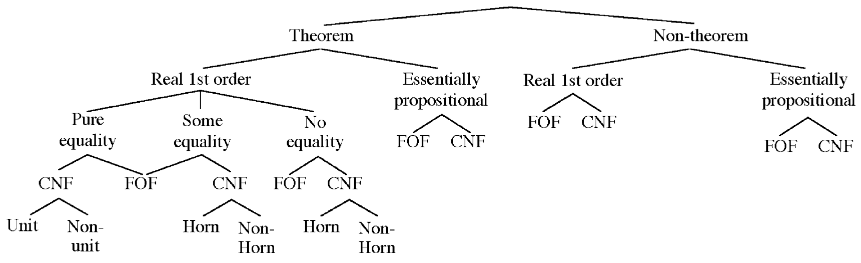

Fig. 3. Specialist problem classes.

- Order: Essentially propositional vs. Real 1st order;
- Equality: No equality vs. Some equality vs. Pure equality;
- Form: CNF (Clause Normal Form) vs. FOF (First Order Form);
- Horness: Horn vs. Non-Horn;
- Unit equality: Unit equality vs. Non-unit pure equality.

Based on these characteristics 14 SPCs have been defined, as indicated by the leaves of the tree in Fig. 3.

As is explained above, it is necessary for the SPCs to be reasonably homogeneous with respect to the ATP system being evaluated. This is ensured by examining the patterns of system performance across the problems in each SPC. If there are 'clumps' of problems that some system(s) solve while others do not, this suggests that the SPC may need to be split along some characteristic that separates out the 'clump'. For example, the separation of the "Essentially propositional" from others was motivated by observing that SPASS [32] performed differently on the ALC problems in the SYN domain of the TPTP.

Although the focus of this paper is on the evaluation of general purpose ATP systems, the underlying use of SPCs makes the methodology applicable to special purpose systems and domains. There are two types of special purpose systems. The first are systems that have a general purpose core, but which have been especially adapted for some special domain, e.g., EQP, which is a unit equality system especially adapted to deal with problems containing associative-commutative function symbols. The second are systems that have been custom built for some special type of problem, e.g., CoDe [9], which deals only with problems about condensed detachment. In the case of the first type of system, such systems are unlikely to be powerful in their specialized application if their general purpose core is weak. The general purpose core provides the underlying data structures and control mechanisms which the special purpose system must rely on. Thus general purpose evaluation of this type of system is meaningful, e.g., the general purpose core of EQP (without the features for dealing with AC functions) is rated quite highly in the unit equality SPC (the left most in the tree above) using the evaluation schemes described in this paper. For both types of specialization, the evaluation methodologies can be explicitly focussed, by defining appropriately narrow SPCs. For example, an SPC containing only condensed detachment problems would be a suitable for meaningful evaluation of CoDe (and other systems).

In the context of the application of fully automated ATP systems as subsystems of more complex environments, as described in Section 2.2, evaluation within the SPCs provides

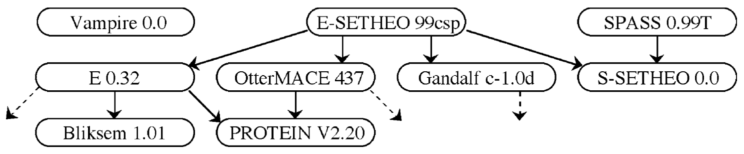

Fig. 4. System subsumptions for the SPC THM_RFO_NEQ_CNF_NHN.

useful information. For a particular application, the environment will generate problems within one or more of the SPCs. The systems that rank highly in those SPCs are candidates for further application specific testing, by defining SPCs containing precisely the types of problems generated by the environment. In this manner the evaluation methodologies described can be used to efficiently determine which systems are best suited to a specific application.

6. System ranking by subsumption

Two simple principles by which ATP systems can be compared are:

- Systems that solve exactly the same problems are ranked equal
- A system that solves a strict superset of the problems solved by another system is better than the other system. The first system is said to *subsume* the other.

Based on these principles, a partial ranking of ATP systems within an SPC is determined by the subsumption relationships.

Subsumption relationships are clearly represented in a directed acyclic graph. For example, Fig. 4 shows the top layers of the subsumption relationships for the SPC THM_RFO_NEQ_CNF_NHN,[4] based on the data in the TPTP results collection as at September 1999. An arrow downward from system A to system B indicates that system A subsumes system B. In Fig. 4, Vampire 0.0 subsumes no systems but is not subsumed either; E-SETHEO 99csp subsumes E 0.32, OtterMACE 437, Gandalf c-1.0d, and S-SETHEO 0.0; SPASS 0.99T subsumes S-SETHEO 0.0; E 0.32 subsumes Bliksem 1.01 and PROTEIN V2.20 (and some others not shown); OtterMACE 437 subsumes PROTEIN V2.20 (and some others not shown).

A *state-of-the-art (SOTA) system* is a system that solves every problem solved by any system, and would therefore subsume all the other systems. There is no SOTA system in the example shown in Fig. 4. A system is a *SOTA contributor* if it is not subsumed, e.g., Vampire 0.0, E-SETHEO 99csp, and SPASS 0.99T are the SOTA contributors in Fig. 4. A SOTA system can be built by running the SOTA contributors in parallel (if any systems are ranked equal, just one of them is needed for the SOTA system). The number of SOTA contributions over all SPCs also provides a form of system evaluation.

The strength of system ranking by subsumption is that, in terms of the numbers of problems that can be solved within realistic resource limits, it is indisputable. The weakness, however, is that it provides only a partial ranking of the systems. Section 8,

[4] Theorem, real 1st order, no equality, clause normal form, non-Horn.

based on results from Section 7, provides a rating of ATP systems within an SPC, and hence a total ranking.

7. SOTA problem rating

Like ATP system search spaces, ATP problem difficulty cannot be completely analysed in the 1st order case. Problem difficulty rating can be done only through empirical measurement, i.e., it must be based on ATP system performance. Problem rating is thus not only for ATP systems, it is also by ATP systems. ATP problems, like ATP systems, are therefore evaluated within SPCs.

The system evaluation scheme presented in Section 6 (and that to be presented in Section 8) provides only a ranking of ATP systems. The precise relative abilities of the ATP systems are unknown, and therefore cannot be used in determining problem ratings. Rather, problem ratings can be based on only whether or not systems solve the problems. A problem rating scheme based on whether or not systems solve problems needs to avoid the influence of weak systems, as it is possible that very many of these could exist; the existence of many weak systems that fail to solve a problem is not good evidence that the problem is hard. The SOTA problem rating scheme avoids the influence of weak systems by using performance data from only SOTA contributors, as defined in Section 6. The following principles are then reasonable:

- A problem's difficulty is proportional to the number of SOTA contributors that fail to solve it.
- A problem that is solved by all SOTA contributors is as easy as possible.
- A problem that is solved by no SOTA contributors, i.e., not solved by any system, is as hard as possible.

Based on these principles, the SOTA problem rating within an SPC is given by:

$$\frac{\text{Number of failing SOTA contributors}}{\text{Number of SOTA contributors}}$$

Problems that are solved by all SOTA contributors receive a rating of zero, and are called *easy* problems. Problems that are solved by some but not all SOTA contributors receive a rating between zero and one, proportional to the number of SOTA contributors that fail to solve the problem, and are called *difficult* problems. Problems that are solved by no systems receive a rating of one, and are called *unsolved* problems.

Using only SOTA contributors to rate problems is acceptable if there are several of them, but experience shows that it is possible for there to be only one or two SOTA contributors. In the extreme case of a single SOTA contributor, problems are rated either as easy or unsolved; there are no difficult problems. This situation does not provide adequate differentiation between problems, and in this case it is reasonable to allow non-SOTA contributors to affect the problem ratings. According to how many systems are required to rate a problem,[5] the non-SOTA contributors that solve the most problems are used.

[5] In the TPTP three systems are required.

```
ALG002-1         D:18 ED:15 S:14 F: 1 SC: 3 SCS: 3 SCF: 0 NSC: 0 Rating: 0.00
ANA001-1         D:18 ED:15 S: 0 F:15 SC: 3 SCS: 0 SCF: 3 NSC: 0 Rating: 1.00
ANA002-1         D:18 ED:15 S: 0 F:15 SC: 3 SCS: 0 SCF: 3 NSC: 0 Rating: 1.00
ANA002-2         D:18 ED:15 S: 4 F:11 SC: 3 SCS: 2 SCF: 1 NSC: 0 Rating: 0.33
ANA002-3         D:18 ED:15 S: 1 F:14 SC: 3 SCS: 1 SCF: 2 NSC: 0 Rating: 0.67
ANA002-4         D:18 ED:15 S: 4 F:11 SC: 3 SCS: 2 SCF: 1 NSC: 0 Rating: 0.33
ANA003-4         D:18 ED:15 S: 4 F:11 SC: 3 SCS: 3 SCF: 0 NSC: 0 Rating: 0.00
ANA004-4         D:18 ED:15 S: 3 F:12 SC: 3 SCS: 2 SCF: 1 NSC: 0 Rating: 0.33
ANA004-5         D:18 ED:15 S: 0 F:15 SC: 3 SCS: 0 SCF: 3 NSC: 0 Rating: 1.00
ANA005-4         D:18 ED:15 S: 0 F:15 SC: 3 SCS: 0 SCF: 3 NSC: 0 Rating: 1.00
ANA005-5         D:18 ED:15 S: 0 F:15 SC: 3 SCS: 0 SCF: 3 NSC: 0 Rating: 1.00
```

Fig. 5. Problem ratings for the SPC THM_RFO_NEQ_CNF_NHN.

These systems and the SOTA contributors are together called *rating contributors*, and the problem rating formula is modified to:

$$\frac{\text{Number of failing rating contributors}}{\text{Number of rating contributors}}$$

Using non-SOTA rating contributors has the effect of making some problems difficult rather than easy. If the non-SOTA rating contributors are reasonably capable systems then this effect is acceptable. If the non-SOTA rating contributors are weak systems then some problems that are really easy will be rated as difficult. The latter drawback occurs when too few decent systems provide performance data for the rating process, a situation that is easily avoided.

Fig. 5 shows some of the problem ratings for the SPC THM_RFO_NEQ_CNF_NHN, based on the data in the TPTP results collection as at September 1999 (the same SPC and data as used in the example in Section 6). The columns are:

D:	The number of systems with data in the TPTP results collection for this problem.
ED:	The number of systems with data that is eligible to be used (a system's data is eligible only if there is data from the system for all the problems in the SPC).
S: F:	The number of solving and failing systems out of the ED: systems.
SC:	The number of SOTA contributors.
SCS: SCF:	The number of solving and failing SOTA contributors.
NSC:	The number of non-SOTA rating contributors.
Rating:	The problem rating.

In the data shown in Fig. 5, three SOTA contributors (Vampire 0.0, E-SETHEO 99csp, and SPASS 0.99T, as described in Section 6) determine the rating of each problem.

The SOTA problem rating scheme provides intuitively acceptable and practically realistic ratings of ATP problems. It is used to provide difficulty ratings for the TPTP problems, thus allowing users to select appropriately difficult problems according to their needs. No anomalies in the ratings have been pointed out by TPTP users, suggesting that the ratings correspond to those human perceptions of the problems' difficulties. The scheme would be open to influence by many rating contributors that solve highly overlapping sets of problems. This possibility has not arisen in current evaluations, and would anyway be asymptotically decreasing with the number of such systems.

8. SOTA system rating

The system ranking by subsumption scheme provides a partial ranking of ATP systems, based on their abilities to solve problems. A scheme that rates the ATP systems would provide a complete ranking, and is therefore desirable. As before, the rating is done within SPCs.

The relative ability of ATP systems to solve easy problems is of little interest, as all decent systems should be able to solve easy problems. A system rating scheme needs to use problems that some systems can solve and others cannot, to provide differentiation between the systems. Furthermore, as ATP progresses, more and more problems will become easy, and if used in the rating of ATP systems they will have an excessive influence. The SOTA system rating scheme is therefore based on difficult problems, as defined in Section 7. The following principles are then reasonable:

- A system's quality is proportional to the number of difficult problems it can solve.
- A system that can solve all difficult problems is as strong as possible.
- A system that can solve no difficult problems is as weak as possible.

Based on these principles, the SOTA system rating within an SPC is given by:

$$\frac{\text{Number of difficult problems solved by the system}}{\text{Number of difficult problems solved by any SOTA contributor}}$$

The problem ratings are relative to a SOTA system. A system that cannot solve any difficult problems receives a rating of zero. Systems that solve some but not all difficult problems receive a rating between zero and one, proportional to the number of difficult problems solved. A SOTA system is able to solve all difficult problems and receives a rating of one. The rating scale is affected by the difficulty distribution of the problems, and is therefore typically non-linear. The scheme therefore provides a total ranking of the ATP systems (except between systems that solve the same number of difficult problems), but a ratio of ratings is usually not meaningful.

Fig. 6 shows some of the system ratings for the SPC THM_RFO_NEQ_CNF_NHN, based on the data in the TPTP results collection as at September 1999 (the same SPC and data as used in the example in Section 6). There are 107 problems in the SPC, of which the SOTA contributors, or equivalently, a SOTA system, solves 104. Ninety problems were solved by all of the SOTA contributors, i.e., there are 90 easy problems, 14 difficult problems, and 3 unsolved problems. For the individual systems, the "SC" flag indicates that the system is a SOTA contributor. In Fig. 6 E-SETHEO 99csp, SPASS 0.99T, and Vampire 0.0 are the SOTA contributors. The columns of numbers for the individual systems give:

- The number of problems solved.
- The fraction of problems solved by a SOTA system.
- The number of difficult problems solved.
- The fraction of the difficult problems solved, i.e., the SOTA system rating (SSR).

E-SETHEO 99csp, for example, solved 101 problems, which is 97% of the 104 problems solved by a SOTA system, and 11 of the 14 difficult problems, giving a SOTA system rating of 0.79.

```
SPC size                        107
SOTA contributors      solved   104
SOTA contributors all  solved    90 = 0.87 SOTA   14 = 1.00 SSR

E-SETHEO---99csp    SC solved   101 = 0.97 SOTA   11 = 0.79 SSR
SPASS---0.99T       SC solved   100 = 0.96 SOTA   10 = 0.71 SSR
Gandalf---c-1.0d       solved    95 = 0.91 SOTA    8 = 0.57 SSR
E---0.32               solved    95 = 0.91 SOTA    5 = 0.36 SSR
Vampire---0.0       SC solved    95 = 0.91 SOTA    5 = 0.36 SSR
S-SETHEO---0.0         solved    76 = 0.73 SOTA    4 = 0.29 SSR
OtterMACE---437        solved    85 = 0.82 SOTA    2 = 0.14 SSR
Bliksem---1.01         solved    84 = 0.81 SOTA    1 = 0.07 SSR
PROTEIN---V2.20        solved    64 = 0.62 SOTA    0 = 0.00 SSR
```

Fig. 6. System ratings for the SPC THM_RFO_NEQ_CNF_NHN.

Note how a non-SOTA contributor can attain a higher rating than a SOTA contributor, e.g., Gandalf c-1.0d gets a higher rating than Vampire 0.0. This comes about when the non-SOTA contributor is subsumed by a different SOTA contributor, in this case by E-SETHEO 99csp (see Fig. 4).

The SOTA system rating scheme provides an intuitively acceptable and practically realistic rating of ATP systems. A system's rating is not affected by its ability (or inability) to solve easy problems, thus the rating scheme focuses on the system's potential contribution to the state of the art.

9. Resource usage

There are three resources used by ATP systems that are of interest, and whose usage should be evaluated. They are CPU time, wall clock time, and memory.

From a user perspective, if there is limited CPU time available, it is of interest to know whether or not an ATP system on average finds its solutions within that CPU time limit. Average CPU time over problems solved is therefore an interesting measure for evaluation.[6]

Wall clock time taken is of direct interest to users, as this is the real time they have to wait for solutions. As for CPU time, the average must be taken over problems solved. It is relevant as a separate measure from CPU time for cases where swapping (caused by excessive memory usage) makes the wall clock time significantly greater than the CPU time.

Use of memory up to that available is of interest in situations where an ATP system has to share the machine with other processes. Average and maximal footprints provide interesting evaluation information. As indicated above, use of memory beyond that available is reflected by a wall clock time significantly higher than the CPU time. Thus evaluating ATP systems in terms of wall clock time also evaluates excess memory requirements.

[6] Average CPU time over all problems is typically not of interest: if, as is commonly the case, the CPU time limit is significantly greater than the average solution time and many problems are unsolved, the average CPU time over all problems corresponds directly to the number of problems unsolved, and no new evaluation information is provided.

10. Conclusion

Evaluating ATP systems is important, as it shows which systems, and hence which techniques, work well for what types of problem. It identifies specialist and general capabilities of the ATP systems, and provides information to potential users about systems' qualities. Evaluating ATP problems is important as it simplifies problem selection according to a user's intentions, and over the years, changes in problem ratings provide a quantitative indicator of advancement in ATP.

The paper carefully considers the issues underlying general purpose ATP system evaluation. The observation that an ATP system's ability to solve problems is largely independent of CPU and memory resource limits, and the definition of specialist problem classes, provide a basis for two methodic schemes for the impartial empirical evaluation of ATP systems: system ranking by subsumption and SOTA system rating. The two system evaluation schemes are linked by the SOTA problem rating scheme, which provides intuitively acceptable and practically realistic ratings of ATP problems. The SOTA system rating scheme is dependent on the identification of difficult problems, which is achieved by the SOTA problem rating scheme. The SOTA problem rating scheme is dependent on identification of the ATP systems that contribute to the state-of-the-art, which is achieved by the system ranking by subsumption scheme. A key observation is that evaluation of system quality and problem difficulty are thus inextricably intertwined.

The evaluation schemes are used in the annual CADE ATP system competitions [24, 27–29], for determining eligible TPTP problems to be used and for evaluating the ATP systems. The SOTA problem rating scheme is used to assign difficulty ratings in the TPTP problem library. Knowledge of which ATP systems are good for which SPCs is used in the SSCPA ATP system [23], which selects ATP systems to run in parallel based on the problem's SPC.

Future work includes investigating a finer grained system rating using problem ratings (which may differentiate between systems that solve the same number of difficult problems), and automatic identification of SPCs based on data analysis.

References

[1] O.L. Astrachan, D.W. Loveland, Measuring the performance of automated theorem provers, in: G. Sutcliffe, C. Suttner (Eds.), Proc. CADE-12 Workshop 2C—Evaluation of Automated Theorem Proving Systems, Nancy, France, 1994, pp. 37–41.

[2] A. Bauer, E. Clarke, Analytica—An experiment in combining theorem proving and symbolic computation, J. Automat. Reason. 21 (3) (1998) 295–325.

[3] B. Barras, S. Boutin, C. Cornes, J. Courant, J.C. Filliatre, E. Giménez, H. Herbelin, G. Huet, C. Muñoz, C. Murthy, C. Parent, C. Paulin, A. Saïbi, B. Werner, The COQ proof assistant reference manual—Version 6.1, Research Report 0203, INRIA, Nancy, France, 1997.

[4] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, A. Meier, E. Melis, W. Schaarschmidt, J. Siekmann, V. Sorge, OMEGA: Towards a mathematical assistant, in: W.W. McCune (Ed.), Proc. 14th International Conference on Automated Deduction, Townsville, Australia, Lecture Notes in Artificial Intelligence, Vol. 1249, Springer, Berlin, 1997, pp. 146–160.

[5] B.I. Dahn, J. Gehne, T. Honigmann, A. Wolf, Integration of automated and interactive theorem proving, in: W.W. McCune (Ed.), Proc. 14th International Conference on Automated Deduction, Townsville, Australia, Lecture Notes in Artificial Intelligence, Vol. 1249, Springer, Berlin, 1997, pp. 57–60.

[6] H. DeNivelle, Bliksem 1.00, in: G. Sutcliffe, C. Suttner (Eds.), Proc. CADE-15 ATP System Competition, Lindau, Germany, 1998, p. 9.
[7] U. Dunker, Search space and proof complexity of theorem proving strategies, in: G. Sutcliffe, C. Suttner (Eds.), Proc. CADE-12 Workshop 2C—Evaluation of Automated Theorem Proving Systems, Nancy, France, 1994.
[8] R. Feynmann, R. Leighton, Cargo cult science, in: E. Hutchings (Ed.), Surely You're Joking Mr Feynmann, Vintage, 1992, pp. 338–346.
[9] D. Fuchs, M. Fuchs, CoDe: A powerful prover for problems of condensed detachment, in: W.W. McCune (Ed.), Proc. 14th International Conference on Automated Deduction, Townsville, Australia, Lecture Notes in Artificial Intelligence, Vol. 1249, Springer, Berlin, 1997, pp. 260–263.
[10] M. Gordon, T. Melham, Introduction to HOL, a Theorem Proving Environment for Higher Order Logic, Cambridge University Press, 1993.
[11] S. Hanks, M.E. Pollack, P.R. Cohen, Benchmarks, test beds, controlled experimentation and the design of agent architectures, AI Magazine 14 (4) (1993) 17–42.
[12] J.N. Hooker, Testing heuristics: We have it all wrong, J. Heuristics 1 (1996) 33–42.
[13] D.S. Johnson, A theoretician's guide to the experimental analysis of algorithms, 1996. Draft available by ftp from: dimacs.rutgers.edu:pub/dsj/temp/exper.ps.
[14] R. Letz, J. Schumann, S. Bayerl, W. Bibel, SETHEO: A high-performance theorem prover, J. Automat. Reason. 8 (2) (1992) 183–212.
[15] R. Letz, On the polynomial transparency of resolution, in: R. Bajcsy (Ed.), Proc. IJCAI-93, Chambéry, France, 1993, pp. 123–129.
[16] W.W. McCune, Otter 3.0 reference manual and guide, Technical Report ANL-94/6, Argonne National Laboratory, Argonne, IL, 1994.
[17] W.W. McCune, Solution of the robbins problem, J. Automat. Reason. 19 (3) (1997) 263–276.
[18] W.W. McCune, EQP: Equational prover, 2000, http://www-unix.mcs.anl.gov/AR/eqp/.
[19] L.C. Paulson, T. Nipkow, Isabelle: A generic theorem prover, Lecture Notes in Computer Science, Vol. 828, Springer, Berlin, 1994.
[20] D.A. Plaisted, The search efficiency of theorem proving strategies, in: A. Bundy (Ed.), Proc. 12th International Conference on Automated Deduction, Nancy, France, Lecture Notes in Artificial Intelligence, Vol. 814, Springer, Berlin, 1994, pp. 57–71.
[21] W. Reif, The KIV-approach to software verification, in: M. Broy, S. Jähnichen (Eds.), KORSO: Methods, Languages, and Tools for the Construction of Correct Software—Final Report, Lecture Notes in Computer Science, Vol. 1009, Springer, Berlin, 1995.
[22] S. Schulz, System abstract: E 0.3, in: H. Ganzinger (Ed.), Proc. 16th International Conference on Automated Deduction, Trento, Italy, Lecture Notes in Artificial Intelligence, Vol. 1632, Springer, Berlin, 1999, pp. 297–301.
[23] G. Sutcliffe, D. Seyfang, Smart selective competition parallelism ATP, in: A. Kumar, I. Russell (Eds.), Proc. 12th Florida Artificial Intelligence Research Symposium, Orlando, FL, 1999, pp. 341–345.
[24] G. Sutcliffe, C. Suttner, Special issue: The CADE-13 ATP system competition, J. Automat. Reason. 18 (2) (1997).
[25] G. Sutcliffe, C. Suttner, ATP system results for the TPTP problem library, 1997, http://www.cs.jcu.edu.au/~tptp/TPTP/Results.html.
[26] G. Sutcliffe, C. Suttner, The TPTP problem library: CNF release v1.2.1, J. Automat. Reason. 21 (2) (1998) 177–203.
[27] G. Sutcliffe, C. Suttner, The CADE-15 ATP system competition, J. Automat. Reason. 23 (1) (1999) 1–23.
[28] G. Sutcliffe, The CADE-16 ATP system competition, J. Automat. Reason. 24 (3) (2000) 371–396.
[29] C. Suttner, G. Sutcliffe, The CADE-14 ATP system competition, J. Automat. Reason. 21 (1) (1998) 99–134.
[30] T. Tammet, Gandalf, J. Automat. Reason. 18 (2) (1997) 199–204.
[31] A. Voronkov, The anatomy of vampire, J. Automat. Reason. 15 (2) (1995) 237–265.
[32] C. Weidenbach, B. Afshordel, U. Brahm, C. Cohrs, T. Engel, E. Keen, C. Theobalt, D. Tpoic, System description: SPASS version 1.0.0, in: H. Ganzinger (Ed.), Proc. 16th International Conference on Automated Deduction, Trento, Italy, Lecture Notes in Artificial Intelligence, Vol. 1632, Springer, Berlin, 1999, pp. 378–382.