Integrating HR and tptp2X into MathWeb to Compare Automated Theorem Provers

Jürgen Zimmer and Andreas Franke
Universität des Saarlandes
D-66041 Saarbrücken, Germany
Email: {jzimmer|afranke}@mathweb.org

Simon Colton
Division of Informatics
University of Edinburgh, UK
Email: simonco@dai.ed.ac.uk

Geoff Sutcliffe
Department of Computer Science
University of Miami, USA
Email: geoff@cs.jcu.edu.au

Abstract

The assessment and comparison of automated theorem proving systems (ATPs) is important for the advancement of the field. At present, the de facto assessment method is to test provers on the TPTP library of nearly 6000 theorems. We describe here a project which aims to complement the TPTP service by automatically generating theorems of sufficient difficulty to provide a significant test for first order provers. This has been achieved by integrating the HR automated theory formation program into the MathWeb Software Bus. HR generates first order conjectures in TPTP format and passes them to a concurrent ATP service in MathWeb. MathWeb then uses the tptp2X utility to translate the conjectures into the input format of a set of provers. In this way, various ATP systems can be compared on their performance over sets of thousands of theorems they have not been previously exposed to. Our purpose here is to describe the integration of various new programs into the MathWeb architecture, rather than to present a full analysis of the performance of theorem provers. However, to demonstrate the potential of the combination of the systems, we describe some preliminary results from experiments in group theory.

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. 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 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. For classical first order ATP, empirical evaluation is a necessary tool for obtaining this knowledge.

One way to evaluate and compare ATP systems is to use the TPTP problem library [SS98], as discussed in §2.1 below. There are, however, some concerns with developers' continual use of the TPTP. The most significant concern is that ATP researchers who always use this library for testing their systems run the risk of producing systems that can solve only TPTP problems, and are weak on new problems or applications. One way to counter this concern is to keep adding new and increasingly more difficult theorems to the TPTP library. New problems are continually being added to the TPTP library, and in [CSar] we discuss how the HR program [CBW99] — as discussed in §3.1 below — has been used to generate novel theorems for this library.

We discuss here another way to counter the problem of researchers fine-tuning their systems to perform well on a library of problems. We intend to compliment the TPTP library with a new service available for ATP researchers. This service will generate a set of theorems over a number of domains, which are difficult enough to differentiate (in terms of efficiency) a set of provers. The analysis of why particular provers perform well over sets of (or individual) theorems will give insight into the strengths and weaknesses of the provers, which will drive their development. To achieve this, we have integrated the HR program [CBW99] and the tptp2X utility (see §3.2 below) into the MathWeb Software Bus [FK99]. In short, the integration is as follows: (i) HR generates a conjecture in TPTP format which is empirically true over a certain number of examples (ii) MathWeb employs the tptp2X utility to translate the theorems into the input formats of a number of ATPs and (iii) MathWeb invokes the provers to prove the theorems.

This project has three stages:

1. Integrate HR and tptp2X into MathWeb and demonstrate the system.
2. Make HR available as a service within MathWeb for ATP researchers.
3. Analyse the performance of a set of provers on sets of theorems.

We describe here the first stage of the project: how we have integrated HR and tptp2X into MathWeb in order to generate and prove thousands of conjectures using a variety of provers. To describe the integration, in §2 we discuss the TPTP library and MathWeb. Following this, in §3 we discuss the integration of the new additions to MathWeb, namely the E ATP system, HR and tptp2X. To demonstrate the potential of this system for ATP researchers, in §4 we describe some preliminary experiments using the provers Bliksem, E, Otter and Spass to prove thousands of theorems in group theory, with the results given in §5. We discuss the next stages of this project in §6.

2 Background

2.1 The TPTP Library

The TPTP (Thousands of Problems for Theorem Provers) Problem Library [SS98] is a library of test problems for ATP systems. It was developed in order to move the testing and evaluation of ATP systems from the previously ad hoc situation onto a firm footing. Since the first release in 1993, many researchers have used the TPTP library for testing their ATP systems and this is now the de facto standard for testing first order ATP systems.

Problems in the TPTP library are in full first-order form (FOF problems) or clause normal form (CNF problems). Each problem contains header information that identifies and describes the problem, provides information about occurrences of the problem in the literature and elsewhere, and gives the problem's ATP status and a table of syntactic characteristics of the problem. A problem may `include` standard axiom sets, and all problems contain their specific formulae. An important item of status information in each problem's header is its *difficulty rating*. This rating is computed using performance data from state-of-the-art ATP systems [SS01], and is a value in the range 0 to 1. Problems with a rating of 0 are *easy*, and can be solved by all state-of-the-art ATP systems. Problems with a rating between 0 and 1 are *difficult*. The rating value is a measure of the fraction of state-of-the-art ATP systems that fail to solve the problem within realistic resource limits. Problems with a rating of 1 are *unsolved* by any ATP system, in normal testing. The ratings are important, as they allow users to select problems according to their intentions.

The syntax of problems files is that of Prolog, which makes it trivial to manipulate the files using Prolog. In particular, the TPTP comes with the tptp2X utility (written in Prolog) that can convert problems from TPTP syntax to the syntax used by existing ATP systems, as discussed in §3.2. Access to the TPTP and related software is available at `www.cs.miami.edu/~tptp/`. In particular, the SystemOnTPTP interface [Sut00] allows a TPTP problem to be submitted in various ways to a range of ATP systems. In addition, the interface will recommend which ATP systems are most likely to be able to solve the problem. For example, for problems expressed in non-Horn pure equality, E-SETHEO csp01, Gandalf c-1.9c, SCOTT 6.0.0, and Spass 1.03 are currently recommended by SystemOnTPTP.

2.2 The MathWeb Software Bus

The MathWeb Software Bus (`www.mathweb.org/mathweb/`) is a platform for distributed automated reasoning that supports the connection of a wide range of mathematical services by a common software bus [FK99]. MathWeb provides the functionality to turn existing theorem proving systems, computer algebra systems (CAS), and other reasoning systems into mathematical services that are homogeneously integrated into a networked proof development environment. The environment thus gains the services from these particular modules, but each

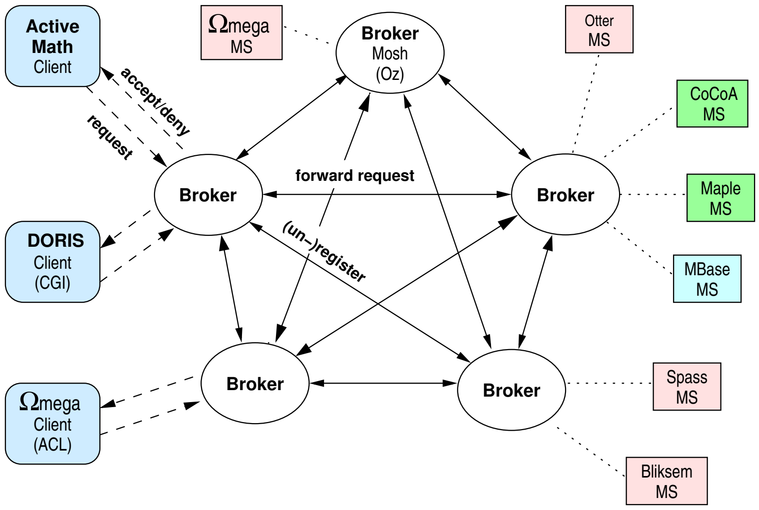

Figure 1: Current state of the MathWeb system

module in turn gains from using the features of other components in MathWeb. MathWeb is implemented in the multi-paradigm programming language Mozart (`www.mozart-oz.org`) which enables easy distribution of applications over a LAN or the Internet [Smo95]. This enabled us to create a stable network of mathematical services which is in every day use. Client applications can access 23 different reasoning and computation systems: computer algebra systems such as Maple, Magma and Cocoa, constraint solvers, mediators, model generators such as Mace and Satchmo. Moreover, MathWeb integrates nine automated theorem provers, such as Otter, Spass and Bliksem.

Mediators are mathematical services which transform mathematical knowledge from one format to another. Currently, MathWeb integrates mediators to translate (i) OpenMath formulae [CC98] into a variety of formats (ii) OmDoc documents [Koh00] into the logic of the Omega system [BCF+97], and (iii) ATP problems in DFG syntax into Otter input syntax. In addition to first order automated theorem provers, MathWeb also offers a concurrent ATP service which calls a selection of ATPs concurrently on a given problem. An application using this service can choose to see only the first result (i.e. the result of the fastest prover) or all the results. The latter enables a runtime comparison and/or several independent judgements about a given conjecture to be obtained.

The current architecture of the MathWeb system is depicted in figure 1. We see that local *brokers* provide routing and authentication information to the mathematical services (see [FK99, SHS98] for details). So called *meta-services*, offer the mathematical services (e.g., an ATP, CAS or a model checker) to their local broker. MathWeb brokers register with each other thus building a dynamic web of brokers. Client applications such as the Omega system, LOUI (a GUI for Omega), or CGI-scripts connect to one of the MathWeb brokers and request services. If the requested service is not offered by a local meta-service, the broker forwards the request to all known brokers until the service is found. If the requested service is found, the client application receives a reference to a newly created *service object* and can send messages directly to the object. Some service objects such as Omega also act as clients, and request other services themselves.

3 Integration of New Systems into MathWeb

For this project, we have integrated three new systems into MathWeb:

- The automated theory formation system HR to generate theorems.
- The tptp2X utility to translate the theorems.
- The E automated theorem prover to prove the theorems.

HR was implemented as a new client and tptp2X as a mediator in order to test the ATP systems in MathWeb with conjectures supplied by HR, and we discuss these systems in this section. However, E was added as a new system to test, and we discuss this in §4, along with the other provers employed in the experiments. We also had to extend the existing MathWeb ATPs which were used in the experiments — namely Bliksem, Otter and Spass — to accept TPTP problem descriptions and to use the new tptp2X service. We discuss the technical problems which arose in the integration of the new systems and the adaptation of the current systems in §3.3.

3.1 The HR System

The HR program (named after mathematicians Hardy and Ramanujan) performs automated theory formation in domains of pure mathematics such as number theory, graph theory, and finite algebras, such as group theory and ring theory. The initial information about a domain supplied to HR include the axioms of the domain and optionally some initial concepts (e.g., multiplication and addition in number theory). The concepts are supplied with both a definition and some examples (e.g., triples of integers related by multiplication). In finite algebraic domains, HR can start with just the axioms of the theory, as it extracts initial concepts from these, e.g., given the identity axiom in group theory, HR extracts the concept of identity elements. HR operates by performing a theory formation step that attempts to invent a new concept from one (or two) old ones. Concept formation is facilitated using one of a set of general production rules that generate both a definition and set of examples for the new concept, from the definition and examples of the old concept(s). The *complexity* of a concept is measured as the number of production rule steps which were used to construct the concept. The 10 production rules are described in detail in [CBW99], [CBW00a] and [Col00], and include the following four:

- Compose rule: uses conjunction to join the definitions of two previous concepts
- Exists rule: introduces existential quantification
- Match rule: equates variables in a concept's definition
- Negate rule: negates predicates within a definition

A theory formation step will lead to either: (a) a concept that has no examples, (b) a concept that has exactly the same examples as a previous concept, or (c) a concept that has non-trivial examples that differ from those of all previously existing concepts. In the first case, there may be no examples for the concept because of the lack of data given to HR, or it may be because the definition of the concept is inconsistent with the axioms of the domain. Hence, HR makes a conjecture that no examples of the concept exist. In the second case, HR makes an if-and-only-if conjecture, stating that the definitions of the new concept and the previous one are equivalent. In the last case, the concept is simply added to the theory.

When HR makes conjectures in finite algebraic domains, it can invoke the Otter theorem prover [McC94] to attempt to prove the conjecture. If Otter fails, HR invokes the Mace model generator [McC01] to attempt to find a counterexample. If neither Otter nor Mace are successful, then the conjecture remains open. In cases where Otter proves an equivalence theorem, HR breaks this into a set of implication theorems, where a set of premise predicates imply a single goal predicate. Furthermore, HR extracts prime implicates from each implication theorem, i.e. it takes ever-larger subsets of the premises from the implication theorem and sees whether Otter can prove that they imply the goal.

3.2 The tptp2X Utility

The tptp2X utility is a multi-functional utility for reformatting, transforming, and generating TPTP problem files. In particular, tptp2X can be used to:

- Control the generation of TPTP problem files from TPTP generator files.
- Apply various transformations to TPTP problems.
- Convert problems in TPTP format to formats used by existing ATP systems.

The transformations currently available in tptp2X include conversion of FOF problems to CNF, random reordering of formulae and literals (to test sensitivity to the particular presentation in the TPTP), addition and removal of equality axioms from problems (as required by many ATP systems), and to apply Stickel's magic set transformation [Sti94]. The output format currently available from tptp2X include the Bliksem, Dedam, DFG, DIMACS, KIF, Otter, Protein, PTTP, Setheo, and Waldmeister formats. The core of tptp2X is written in Prolog, thus it can easily read and manipulate the Prolog format syntax of TPTP problems. The core has a modular construction, and it is easy for users to add new transformations and formats. The most common use of tptp2X is from a terminal command prompt, and a shell script interface is provided for this purpose. Direct use via Prolog is also possible, and various features of tptp2X have been optimized for this style of use.

3.3 Integration Details

We have integrated HR into MathWeb with the intention that HR will be both a client (utilising other services in MathWeb) and a service (utilised by other clients in MathWeb), with this latter aspect discussed in §6 below. Hence we wrote a MathWeb wrapper that communicates with HR via two sockets (one to handle service requests and the other to issue client commands). Due to the high number of conjectures HR typically produces in a session, we preferred the socket interface to the less efficient XML-RPC interface, which is also available in MathWeb. The high numbers of conjectures being passed around MathWeb caused many other problems, and we had to improve many aspects of MathWeb, for instance the handling of temporary files. HR was also improved. In particular, we wrote two new classes (HR is a Java program) to handle the interaction with MathWeb. The first class (MathWeb.class) abstracts the socket communication details and offers an easy way for HR to access MathWeb services, and for the user to choose which services to access. The second class (MathWebProver.class) is able to translate HR's conjectures into TPTP first order format, call MathWeb.class to employ tptp2X and the concurrent ATP service, and read the results, which are returned to HR as Java objects.

The integration of the E ATP system was straight-forward, as there is a standard MathWeb wrapper called ShellProver for automated theorem provers, which only has to be adjusted for the particular output of the prover at hand. First-order problems are written to (temporary) files and the ATPs are called as shell commands on the respective input file. Finally, the ShellProver wrapper analyses the output of the prover. The analysis of the output is the most difficult part of the integration of an ATP because automated theorem provers are typically not designed to produce machine readable output, but rather to inform a human user about their results. Hence we had to write bespoke algorithms within the prover-calling wrappers to determine the status of a theorem, extract a proof object and so on.

The integration of tptp2X was also a non-trivial task, because the standard tptp2X shell command was not efficient enough for our purposes (it took between 3 and 4 seconds to translate each theorem, as the Prolog interpreter was loaded, and tptp2X compiled each time). We therefore had to design and implement a tptp2X servlet based on a permanently running Prolog process with the pre-compiled tptp2X code that evaluates incoming transformation requests immediately. With this new service, we reduced the transformation time to 100-200 ms (for an average sized problem). This speedup was crucial, as HR produces thousands of conjectures in a short time, and we need n tptp2X transformations for running n provers in parallel on a single TPTP problem. We also upgraded how the runtime of the provers is recorded. All provers are started directly in the shell without any intermediate scripts and we use the Unix `time` command to record the CPU time of the prover process. We also recorded the CPU time that was used for operating system calls and added the two values. This gives an indication of the time required to prove a theorem, rather than how fast the prover is after intermediate processes have been undertaken.

4 Experiments

Our experiments were designed to show that a stable integration has been achieved and to highlight the potential usage of HR within MathWeb. Our aim here was to differentiate four first order theorem provers in terms of their efficiency proving a set of theorems generated by HR. The provers we employed are described in §4.1, and we describe the sessions in §4.2.

4.1 ATP Systems Employed

We used the latest publicly available versions of four first-order theorem provers, namely Bliksem 1.12, E 0.62, Otter 3.2, and Spass 1.03. These systems were used for the CASC system competition in 2001 [Sut01].

• <u>Bliksem 1.12</u>

Bliksem [DeN] implements the ordered resolution + superposition calculus. It supports many different orders, including reduction orders, A-orders, and non-liftable orders. Special attention has been given to resolution strategies that provide decision procedures for certain subsets of first order logic. Bliksem is able to transform first order formulae into clausal normal form, using different structural or non-structural clause transformations. The more recently added features of Bliksem include a posteriori orders, equality factoring and tautology elimination, optimized normal form transformations, non-unit demodulators, and equality subsumption (which takes into account commutativity of equality). High priority has been given to portability. The data structures have been carefully chosen after benchmark tests on the basic operations.

• <u>E 0.62</u>

E 0.62 [Sch01] is a purely equational theorem prover. The calculus used by E combines superposition (with selection of negative literals) and rewriting. No special rules for non-equational literals have been implemented, i.e., resolution is simulated via paramodulation and equality resolution. E 0.62 includes AC redundancy elimination and AC simplification for dynamically recognized associative and commutative equational theories, as well as simulated clause splitting. E is based on the DISCOUNT-loop variant of the given-clause algorithm, i.e., a strict separation of active and passive facts. Proof search in E is primarily controlled by a literal selection strategy, a clause evaluation heuristic, and a simplification ordering. Supported term orderings are several parameterized instances of Knuth-Bendix-Ordering (KBO) and Lexicographic Path Ordering (LPO). The most unique feature of the implementation is the maximally shared term representation. This includes parallel rewriting for all instances of a particular subterm. A second important feature is the use of perfect discrimination trees with age and size constraints for rewriting and unit-subsumption.

• <u>Otter 3.2</u>

Otter [McC94] is designed to prove theorems stated in first-order logic with equality. Otter's inference rules are based on resolution and paramodulation, and it includes facilities for term rewriting, term orderings, Knuth-Bendix completion, weighting, and strategies for directing and restricting searches for proofs. Otter can also be used as a symbolic calculator and has an embedded equational programming system. Otter is a fourth-generation Argonne National Laboratory deduction system whose ancestors (dating from the early 1960s) include the TP series, NIUTP, AURA, and ITP.

- Spass 1.03

Spass [WAB+99] is an automated theorem prover for first-order logic with equality. It is a saturation based prover employing superposition, sorts and splitting. In contrast to many approaches to order-sorted clausal reasoning, the calculus enables sort predicates and equations to occur arbitrarily within clauses. Therefore, the sort theory is not separated from the problem clauses, but automatically and dynamically extracted. Spass also offers a variety of further inference and reduction rules including hyper resolution, unit resulting resolution, various variants of paramodulation and a terminator. Spass relies on an internal library supporting specific data structures and algorithms like, for example, indexing or orderings (KBO, RPOS).

4.2 Sessions with MathWeb

We ran HR for two sessions within MathWeb. In the first session, we supplied conjectures to all four provers. In the second session, we employed only Bliksem, E and Spass. The reason for the omission of Otter is that we wanted to perform a much more extensive test of the system in the second session, and had found that Otter was increasingly unable to prove the theorems produced. Otter repeatedly timing out at 120 seconds meant that the session was very slow, and so we removed Otter in order to increase the yield of theorems proved in a reasonable time. Both sessions were undertaken in the domain of group theory, and all conjectures HR produced were true of the groups up to order 8, as these were supplied to HR. The provers were given two sets of axioms from the TPTP library, namely `GRP004+0.ax` (group theory axioms) and `EQU001+0.ax` (axioms of equality). With these axioms, all the theorems produced were expressed in non-Horn pure equality. Each prover was run with the default settings and given a 120 second time limit for each theorem.

In the first session, we ran HR until it produced 1500 equivalence conjectures (with the four provers having attempted proofs). In the second session, we ran HR until it produced 12000 equivalence conjectures (again with proof attempts from the three provers). In both sessions, we ran HR with a random search, but in the first session, we employed a complexity limit of 6, with a complexity limit of 15 for the second session. Breadth first searches have been found to produce conjectures that are too simple, while depth first searches specialize the theory too much. In contrast, random searches — where, at each step, both concept and production rule are randomly chosen — tend to produce fairly complicated conjectures, without over-specializing the theory. We used only the exists, compose, negate and match production rules in these sessions.

5 Results

The first result to note is that the integration of HR and tptp2X into MathWeb is clearly stable, as we were able to complete a session where 12000 theorems were generated and proved. Moreover, there has been an increase in efficiency, with the four provers being called in roughly the same time as HR used to call just Otter. This is due in part to the new socket interface rather than the previous file interface, and in part to a slow implementation of HR reading the output files from Otter. For the two sessions, we recorded the number of theorems proved by each prover, the average time taken to prove those theorems which were proved, and we identified some theorems of interest, i.e. those which differentiate the provers in a significant way.

- Session 1

Prover	No. Proved	Av. Time to Prove (ms)
Bliksem 1.12	1500	188.94
E 0.62	1500	95.27
Otter 3.2	1445	633.90
Spass 1.02	1500	103.88

Table 1: Number proved and average time to prove - session 1

To produce 1500 theorems which were proved (or at least attempted) by all four provers took around 90 minutes on a Sun Ultra 10 workstation. The results from this session are given in table 1, and we see that Otter failed to prove 55 theorems. Of these, Otter timed out on 16, but returned 'no solution' to the other 39, and we are still investigating why Otter cannot find a solution to these. The 16 theorems which Otter timed-out on took around a third of the overall session time, which justifies our reason to remove Otter for the second session. For example, figure 2 shows a theorem (number 493) which Otter 3.2 could not prove in the 120 second limit, but all the other provers proved relatively quickly. For completeness, we present the theorem in the TPTP format which was passed to tptp2X, in addition to a more mainstream mathematical format.

```
include('Axioms/EQU001+0.ax').
include('Axioms/GRP004+0.ax').
input_formula(conjecture493,conjecture,(! [B,C,D] :
  ((equal(inverse(B),C) & equal(multiply(B,D),C) & equal(inverse(D),B)
  & ? [E,F] : (equal(inverse(E),F) & equal(multiply(E,B),F))
  & equal(multiply(D,C),B)) <=>
  (equal(inverse(D),C) & equal(inverse(B),D) & equal(multiply(C,D),B))))).
```

$$\forall\, b,c,d\ (b^{-1} = c\ \&\ b{*}d = c\ \&\ d^{-1} = b\ \&\ \exists\, e,f\ (e^{-1} = f\ \&\ e{*}b = f)\ \&\ d{*}c = b$$
$$\iff d^{-1} = c\ \&\ b^{-1} = d\ \&\ c * d = b)$$

Figure 2: Theorem 493 in session 1, which Otter could not prove in 120 seconds

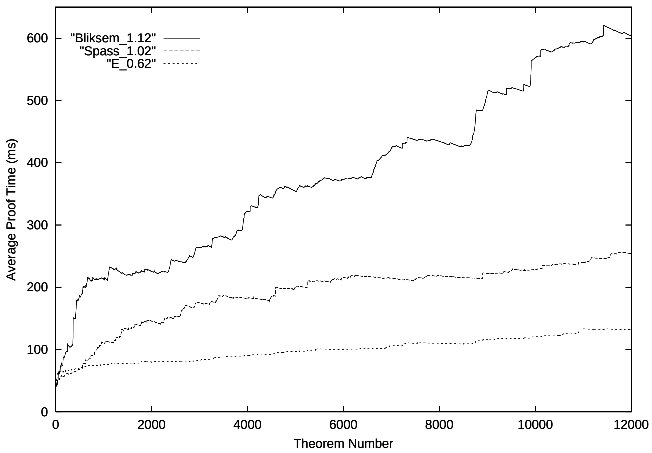

Figure 3: Average times to prove theorems

- Session 2

This session took around 14 hours on a Sun Ultra 10 to produce the required 12000 theorems. All were proved by at least one prover, with 70 exceptions. We have yet to determine which, if any, of the 70 were actually non-theorems. More interesting from the perspective of comparing and contrasting the provers, for each prover, there was at least one conjecture which it could not prove in 120 seconds, but both the others could. Specifically, there were 19 which were proved by E and Spass, but not by Bliksem, 4 which were proved by E and Bliksem, but not Spass and only 1 which Spass and Bliksem proved, but E did not. In appendix A, we provide examples of these theorems.

The average time taken to prove the theorems by each prover are given in table 2. These suggest that (with the default settings) E is most suited to problems of this type, followed by Spass and then Bliksem. Also, Bliksem failed to prove more theorems than E and Spass, although the differences are not great. These results correlate with SystemOnTPTP, which — as mentioned in §2.1 — recommends E (actually E-Setheo, a hybrid) and Spass, but not Bliksem, for non-Horn pure equality theorems such as those HR produced. We also recorded how the average time to prove a theorem changed as the session progressed (see figure 3). For all the provers, the average time to prove a theorem increased in general as HR's theory progressed. This suggests that, given enough time, HR can get to a stage where a theorem will, on average, take an arbitrarily long time to prove by all provers. This is important, as discussed in §6 below, although we need more experimentation to confirm this observation.

Prover	No. Proved	Av. Time to Prove (ms)
Bliksem 1.12	11908	607.01
E 0.62	11931	132.32
Spass 1.02	11927	254.97

Table 2: Number proved and average time to prove - session 2

6 Conclusions and Future Work

In previous work, we have used the HR system to produce many thousands of group theory conjectures [CSar]. Human intervention was then required to employ automated theorem provers to identify those conjectures which were (a) theorems and (b) difficult for some of the provers. This has led to 184 theorems produced by HR being added to the TPTP library. Hence we have shown that HR can be used to improve the TPTP library and we are currently undertaking a project using HR to discover theorems in more complicated algebraic domains, in particular Zariski spaces, [MMS98]. We hope that results from this application will also find their way into the TPTP library.

In addition to improving the TPTP library, we have concentrated here on the potential of using HR to compliment the library. We have described the integration of HR, the tptp2X utility and automated theorem provers into the MathWeb software bus in such a way that HR generates, tptp2X translates and the ATP systems prove conjectures. The implementation is stable enough to process thousands of theorems, and initial experiments have demonstrated that HR can produce theorems which are hard to prove for some of the ATPs but not for others, which we suggest is a potential tool for differentiation the provers. Furthermore, we have shown that the system as a whole (i.e. the integration of HR, tptp2X and the provers in MathWeb) can identify the theorems which only one prover finds difficult. We have argued that the automated generation of such theorems can be used to identify strengths and weaknesses in particular provers, in much the same way as the TPTP library does. This in turn can be used to drive the development of provers.

It is important to stress that we draw no conclusions here about the efficiencies of the provers on the theorems HR generated. For each prover, there are many settings which can drastically improve settings, and we need to perform much more extensive testing before we can make any detailed comparisons of the provers. We have been more concerned here with detailing the integration of the various systems and to demonstrate that the integration is stable and has a potential application to the comparison of provers. By presenting the results from a session where 12000 theorems were stated and proved, we have shown that the system is indeed stable. Perhaps the most compelling demonstration of the potential to compare provers are the theorems given in appendix A. While these are syntactically fairly similar, the performance of the provers on them is strikingly different.

The next stage of this project is to offer HR as a service within MathWeb, both for users and for other applications. To allow other MathWeb applications to use HR as a service, we will define an interface which allows them to call HR's conjecture generation mechanism on a given set of axioms in a specific theory. We plan to offer a service based on some standard encoding of mathematical knowledge, such as OpenMath or OmDoc. We envisage users asking HR via MathWeb to provide them with a certain number of theorems with a certain average difficulty (in terms of proof time and/or number of provers which are successful) over a set of ATP systems. The fact that the average time to prove the theorems generally increased for each prover as HR's theory progressed (as depicted in figure 3) is encouraging. This means that we can run HR for a certain amount of time before assessing the difficulty of the theorems it produces using the ATP systems. As discussed in [CSar], in a previous session, HR produced 46,000 syntactically different group theory conjectures in just 10 minutes on a 500Mhz Pentium processor. Hence it is plausible that we could run HR for a chosen amount of time, then start the provers and reasonably expect the average proof time to be similar to the one requested by the user.

We also intend to add more intelligent ways for HR to choose the theorems that it will pass to the provers for assessment of difficulty. In [CSar], we showed some corelation between the number of existential and universally quantified variables in a theorem and the difficulty of that theorem. HR also has certain measures of interestingness of both concepts and conjectures, and we will experiment to see if there is any corelation between the value of the measures of concepts in a theorem and/or the value of the measures of the overall theorem and the time to prove it for particular provers. In particular — as discussed in [CBW00b] — for equivalence conjectures, HR estimates the 'surprisingness' by looking at how different the two concepts conjectured to be equivalence are (in terms of how they were constructed).

The final stage of the project will be to systematically test a range of provers with different settings over large sets of conjectures produced in a variety of domains by HR within MathWeb. By offering HR as a MathWeb service, we hope to do this in conjunction with the developers of the ATP systems tested. Such large-scale testing of the ATP systems will compliment the TPTP library, providing a new tool for developers of ATP systems, which will hopefully contribute positively to the development of automated theorem proving in general.

Acknowledgments

This work has been supported by EPSRC grant GR/M98012 and the Calculemus network (`www.eurice.de/calculemus`). Simon Colton is also affiliated with the Department of Computer Science at the University of York. We are grateful to the reviewers for their useful comments on this paper.

References

[BCF+97] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant. In William McCune, editor, *Proc. of the 14th Conference on Automated Deduction*, number 1249 in LNAI, pages 252–255, Townsville, Australia, 1997. Springer Verlag.

[CBW99] S. Colton, A. Bundy, and T. Walsh. Automatic Concept Formation in Pure Mathematics. In T. Dean, editor, *Proceedings of the 16th Interna-*

tional Joint Conference on Artificial Intelligence , pages 183–190. Morgan Kaufmann, 1999.

[CBW00a] S. Colton, A. Bundy, and T. Walsh. Automatic Identification of Mathematical Concepts. In *Proceedings of the 17th International Conference on Machine Learning* , 2000.

[CBW00b] S. Colton, A. Bundy, and T. Walsh. On the Notion of Interestingness in Automated Mathematical Discovery. *International Journal of Human Computer Studies*, 53(3):351–375, 2000.

[CC98] Olga Caprotti and Arjeh M. Cohen. Draft of the Open Math standard. The Open Math Society, `www.nag.co.uk/projects/OpenMath/omstd/`, 1998.

[Col00] S. Colton. *Automated Theory Formation in Pure Mathematics*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, 2000.

[CSar] S. Colton and G Sutcliffe. Automatic Generation of Benchmark Problems for Automated Theorem Proving Systems. In *Proceedings of the 7th Symposium on Artificial Intelligence and Mathematics* , 2002 (to appear).

[DeN] H. DeNivelle. Bliksem Resolution Prover. www.mpi-sb.mpg.de/ nivelle.

[FK99] Andreas Franke and Michael Kohlhase. System description: MathWeb, an agent-based communication layer for distributed automated theorem proving. In Harald Ganzinger, editor, *Proceedings of the 16th Conference on Automated Deduction*, number 1632 in LNAI, pages 217–221. Springer Verlag, 1999.

[Koh00] Michael Kohlhase. OMDoc: An open markup format for mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. `http://www.mathweb.org/omdoc`.

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

[McC01] W.W. McCune. MACE 2.0 Reference Manual and Guide. Technical Report ANL/MCS-TM-249, Argonne National Laboratory, Argonne, USA, 2001.

[MMS98] R. McCasland, M. Moore, and P. Smith. An introduction to Zariski spaces over Zariski topologies. *Rocky Mountain Journal of Mathematics*, 28:1357–1369, 1998.

[Sch01] S. Schulz. System Abstract: E 0.61. In R. Gore, A. Leitsch, and T. Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning*, number 2083 in Lecture Notes in Artificial Intelligence, pages 370–375. Springer-Verlag, 2001.

[SHS98] M. Kohlhase S. Hess, Ch. Jung and V. Sorge. An implementation of distributed mathematical services. In Arjeh Cohen and Henk Barendregt, editors, *6th CALCULEMUS and TYPES Workshop*, Eindhoven, The Netherlands, July 13–15 1998.

[Smo95] G. Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, volume 1000 of *LNCS*, pages 324–343. Springer-Verlag, Berlin, 1995.

[SS98] G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[SS01] G. Sutcliffe and C.B. Suttner. Evaluating General Purpose Automated Theorem Proving Systems. *Artificial Intelligence*, 131(1-2):39–54, 2001.

[Sti94] M.E. Stickel. Upside-Down Meta-Interpretation of the Model Elimination Theorem-Proving Procedure for Deduction and Abduction. *Journal of Automated Reasoning*, 13(2):189–210, 1994.

[Sut00] G. Sutcliffe. SystemOnTPTP. In D. McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction*, number 1831 in Lecture Notes in Artificial Intelligence, pages 406–410. Springer-Verlag, 2000.

[Sut01] G Sutcliffe. The CADE-17 ATP System Competition. *Journal of Automated Reasoning*, 27(3):227–250, 2001.

[WAB+99] C. Weidenbach, B. Afshordel, U. Brahm, C. Cohrs, T. Engel, E. Keen, C. Theobalt, and D. Tpoic. System Description: SPASS Version 1.0.0. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 378–382. Springer-Verlag, 1999.

Appendix A - Example Theorems

The following examples are in TPTP format and require TPTP axiom files `EQU001+0.ax` and `GRP004+0.ax`.

- Proved by E in 0.9 seconds, Spass in 0.2 seconds, but not by Bliksem:

```
! [B,C,D] : ((equal(multiply(C,B),D) & equal(multiply(C,D),B) & ? [E,F] :
(equal(inverse(E),F) & equal(multiply(E,B),F)) & equal(inverse(B),D) &
equal(multiply(D,B),C)) <=> (equal(multiply(B,C),D) &
equal(multiply(C,B),D) & equal(multiply(C,D),B) & equal(multiply(B,D),C) &
? [E,F] : (equal(inverse(E),F) & equal(multiply(E,B),F)) &
equal(inverse(B),D) & ? [G,H] : (equal(inverse(G),H) &
equal(multiply(G,D),H)) & equal(multiply(D,C),B))).
```

- Proved by E in 1.4 seconds, by Bliksem in 5.0 seconds, but not by Spass:

```
! [B,C,D] :
((equal(multiply(B,C),D) & equal(inverse(B),C) & equal(multiply(B,D),C) &
? [E,F] : (equal(inverse(E),F) & equal(multiply(E,B),F)) &
? [G] : (~(equal(G,identity))) & equal(inverse(D),D) ) <=>
(equal(inverse(B),C) & equal(multiply(C,B),D) & equal(multiply(B,D),C) &
? [E,F] : (equal(inverse(E),F) & equal(multiply(E,B),F)) &
? [G] : (~(equal(G,identity))) & equal(multiply(D,B),C))).
```

- Proved by Spass in 0.9 seconds, by Bliksem in 16.1 seconds, but not by E:

```
! [B,C,D] : ((equal(inverse(B),C) & equal(multiply(B,D),C) &
? [E,F] : (equal(inverse(E),F) & equal(multiply(E,B),F)) &
? [G] : (~(equal(G,identity)))) <=> (equal(inverse(B),C) &
equal(multiply(B,D),C) & ? [E,F] : (equal(inverse(E),F) &
equal(multiply(E,B),F)) & ? [G] : (~(equal(G,identity))) &
equal(multiply(D,B),C) & ? [H,I] : (equal(inverse(H),I) &
equal(multiply(H,C),I)))).
```