Automatic Generation of Benchmark Problems for Automated Theorem Proving Systems

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.miami.edu

Abstract

Automated Theorem Proving (ATP) researchers who always use the same problems for testing their systems, run the risk of producing systems that can solve only those problems, and are weak on new problems or applications. Furthermore, as the state-of-the-art in ATP progresses, existing test problems become too easy, and testing on them provides little useful information. It is thus important to regularly find new and harder problems for testing ATP systems. HR is a program that performs automated theory formation in mathematical domains, such as group theory, quasigroup theory, and ring theory. Given the axioms of the domain, HR invents concepts, finds examples of them using a model generator, and makes conjectures empirically about the concepts. HR has been used to discover new group theory theorems of sufficient difficulty to be included in the TPTP - the standard library of test problems for first order ATP systems. As HR produces tens of thousands of distinct theorems, there has also been an opportunity to determine some characteristics that can be used to identify hard problems.

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

Currently there are few "real" applications of first 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 problem library [SS98] 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 first order ATP systems.

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. There are, however, some concerns with developers' continual use of the TPTP. The most significant concern is that ATP researchers who always use the TPTP for testing their systems, run the risk of producing systems that can solve only TPTP problems, and are weak on new problems or applications. To counter this concern, it is important to regularly add new types of problems to the TPTP, including problems from emerging applications and sources. A lesser concern is that as the state-of-the-art in ATP progresses [SFS01], existing problems become too easy, and testing on them provides little useful information. Of the 5882 problems in v2.4.1 of the TPTP, 651 have had their difficulty rating dropped from "difficult" – not solvable by all state-of-the-art ATP systems, to "easy" – solvable by all state-of-the-art ATP systems, since they were added to the TPTP (see [SS01] for details of how TPTP problems are rated). It is thus necessary to also add progressively harder problems to the TPTP, and to avoid adding easy problems.

A new version of the TPTP is released at least once a year. Between releases many new problems are collected from ATP users and developers, and encoded for inclusion in the TPTP. These contributions are valuable, and the TPTP would be even more useful if an even larger number of problems could be added each year. The prospect of a "mechanical" source of problems, as described in the following sections of this paper, is therefore highly attractive.

2 The HR Program

The HR program (named after mathematicians Hardy and Ramanujan) has been developed to perform automated theory formation. It is now being applied to many different domains, but has so far been used to form theories primarily in areas of pure mathematics, such as number theory, graph theory, and finite algebras such as group theory and ring theory,

The information supplied to HR includes some initial concepts about a domain (e.g., multiplication and addition in number theory) and some examples of these concepts (e.g., triples of integers related by multiplication). In finite algebraic domains, HR can start with just the axioms of the theory and extract 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 10 production rules are described in detail in [CBW99], [CBW00a], and [Col00]. Only three are of concern here:

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

A concept 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 previously existing concept are equivalent. In the final case, the concept is simply added to the theory.

When HR makes conjectures in finite algebraic domains, it can invoke the Otter theorem prover [McC00b] to attempt to prove the conjecture. If Otter fails, HR then invokes the MACE model generator [McC00a] to attempt to find a counterexample. If neither Otter nor MACE are successful, then the conjecture remains open. In the case 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. That is, it takes ever-larger subsets of the premises from the implication theorem and tests if Otter can prove that they imply the goal. For instance, if HR produced the following theorem in group theory:

$$\forall XY((inv(X) = Y \mathbin{\&} X = id) \leftrightarrow (inv(Y) = X \mathbin{\&} Y = id))$$

it would break it into four implication theorems, e.g., one of them is:

$$\forall XY((inv(X) = Y \mathbin{\&} X = id) \rightarrow inv(Y) = X)$$

For each such implication, HR then attempts to find the smallest set of premises (predicates on the left hand side of the implication) which imply the goal, e.g., Otter can prove this prime implicate:

$$\forall XY(inv(X) = Y \rightarrow inv(Y) = X)$$

In this way, HR builds up a set of prime implicates which embody some of the fundamental facts in the domain.

3 Using HR to Generate Problems for the TPTP

There have been two attempts – the first unsuccessful, the second successful – to use HR to generate problems for the TPTP library. The aim was to generate theorems of sufficient difficulty that they provide a challenge for some (if not all) state-of-the-art ATP systems.

In the first attempt, HR was used with all it's functionality. It produced conjectures in various finite algebraic domains, including ring theory and group theory. All of the conjectures were proved to be theorems by Otter, and HR also extracted prime implicates as additional theorems. However, without exception, none of the hundreds of theorems produced in this way were sufficiently difficult for inclusion in the TPTP. This was for three reasons: First, the prime implicates were produced so that the user could understand the fundamental facts in the domain (e.g., in group theory: $\forall X(X * X = X \rightarrow X = id)$). These comprise some of the simplest theorems in the domain, and are therefore, in general, easy to prove. It appears that in extracting the prime implicates, between them, HR and Otter had reduced much of the difficulty. Second, although HR produces conjectures very quickly, and Otter can be very quick at proving them, the collaboration between HR and Otter was relatively slow, involving writing files and invoking programs, etc. Hence it was possible to produce only a few hundred theorems in a reasonable time. This meant that HR explored only a part of the domain, and did not form many concepts of sufficient complexity that conjectures involving them would be difficult to prove. Third, while some of the conjectures produced were of sufficient complexity to challenge Otter – requiring in some cases up to 10 seconds to prove – it turned out that other provers had little difficulty with them.

The second attempt concentrated on group theory and aimed for quantity, hoping that some fraction of the conjectures produced would turn out to be theorems of sufficient difficulty. HR was set to generate only equivalence conjectures, as the non-existence conjectures are usually easier to prove. HR's ability to extract prime implicates was not used, also because these are often easier to prove. To increase the complexity of the conjectures, a random search was used. Breadth first searches have been found to produce conjectures that are too simple, while depth first searches specialize the theory too much (in particular, with a depth first search in group theory, all the conjectures generated involved only the concept of inverse elements). In contrast, random searches tend to produce fairly complicated conjectures, without over-specializing the theory.

To increase the yield of conjectures, a new functionality was added to HR. Up until this stage, HR had stated an equivalence conjecture between an old concept C_0 and a new concept C_1, if C_0 and C_1 had the same examples. HR was extended to store all the (conjectured-to-be) equivalent definitions for each concept, so that if concept C_0 was conjectured to be equivalent to C_1, and C_0 was later conjectured to be equivalent to C_2, then C_1 would also be conjectured to be equivalent to C_2. HR was given the groups up to order eight as examples, and in previous usage of HR, it was found that the majority of HR's conjectures that were true for the groups up to order eight were true in the general case. Therefore, the conjectures were generated without resorting to Otter and MACE to test for theoremhood, thus removing that step of the process. Finally, HR can produce the same concept via different paths, for instance, a concept produced using the forall production rule (as described in [CBW00a]) can be produced using a negate step, followed by an exists step, followed by another negate step. To decrease the number of repeated conjectures, just the match, exists and compose rules were used.

The axiomatization of group theory was minimal, containing just:

- the left inverse axiom: $\forall X(inv(X) * X = id)$
- left identity axiom: $\forall(id * X = X)$
- the associativity axiom: $\forall XYZ((X * Y) * Z = X * (Y * Z))$

The axioms were encoded in first order logic using only the `equal`/2 predicate, i.e., resulting in pure equality problems. Based on the axioms, HR generated conjectures in first order logic. With the efficiency gains and yield increases, in just 1000 theory formation steps, taking around 10 minutes on a Pentium 500Mhz processor, HR produced 46186 syntactically unique equivalence conjectures.

Due to the optimized generation of the conjectures, there was no assurance that they were theorems, and in the cases that they were, there was no measure of their difficulty for ATP systems. Both these issues had to be checked before the problems could be considered for inclusion in the TPTP. Methodologies for evaluating ATP systems and problems are presented in [SS01]. Roughly, a problem is evaluated by testing state-of-the-art ATP systems on it, and then assigning a difficulty rating according to the fraction of the systems that fail to solve the problem within realistic resource limits.[1] The systems used in the testing are selected based on their performance on problems with the same syntactic characteristics. The group theory problems generated by HR are all pure equality problems, both Horn and non-Horn, and the systems selected for the testing were: E 0.62 [Sch01], Gandalf c-1.9c [Tam97], Otter 3.2 [McC00b], SPASS 1.03 [Wei99], and Vampire 2.0 [RV99]. The results of this testing, presented in Section 4, were used to identify problems that are sufficiently difficult for inclusion in the TPTP.

4 Testing Results

The five ATP systems were tested on the 46186 problems, using a SUN Ultra with 450MHz CPU, and a CPU time limit of 120 seconds. All of the systems prove theorems by finding a refutation of the clause normal form (CNF) of the axioms and negated conjecture. For this testing, Otter [McC00b] was used to do the conversion to CNF, and the `tptp2X` utility distributed with the TPTP was used to prepare the problems for submission to each of the systems.

Table 1 summarizes the results of the testing. Each problem was given a rating in the range 0 to 1, according to the fraction of systems that failed to solve it. 30232 of the 46186 problems are "easy", with a rating of 0.0, i.e., they were solved by all the systems. SPASS, E, and Vampire are clearly better at the problems than Gandalf and Otter, and the problems rated 0.20 and 0.40 are almost all solved by these three systems. The 184 problems rated 0.60 or 0.80 are acceptably difficult, and are suitable for inclusion in the TPTP. They have the required characteristic of being difficult enough to provide a challenge to ATP systems. They can also be used to contribute to a ranking of ATP systems, because they can be solved by only some

[1] A "realistic resource limit" is one beyond which a linear increase in resources does not result in the solution of significantly more problems.

systems.

Table 1: Results Summary

	Total	Rated 0.80	Rated 0.60	Rated 0.40	Rated 0.20
Problems	46186	40	144	6277	9493
SPASS	46186	40	144	6277	9493
E	46081	0	82	6274	9493
Vampire	45991	0	61	6243	9455
Gandalf	38262	0	0	14	8021
Otter	31766	0	1	23	1510

Analysis of the results shows strong correlations between the ratings and the number of clauses in the CNF of the problem, and the ratings and the number of non-Horn clauses in the CNF of the problem. Although such correlations may reasonably be expected (from an ATP perspective), they are not necessary correlations. This confirmation provides a way of roughly evaluating the difficulty of theorems generated by HR, without having to test ATP systems on them. Analysis of the results also shows a correlation between problem ratings and the numbers of universally and existentially quantified variables in the conjecture. Tables 2 and 3 show the average CPU times taken over problems solved (for the three strong ATP systems) and average ratings, for the problems whose conjectures have the indicated number of universally and existentially quantified variables. Table 2 shows an increase in average CPU time and rating up to 2 universally quantified variables, with a marked decrease thereafter. This was an unexpected result – the difficulty of the theorems was expected to increase in proportion to the number of universally quantified variables. This phenomenon is still being examined, but it does suggest that HR found some interesting theorems in the domain, because they are relatively easy to state, yet difficult to prove (many important theorems in pure mathematics have these two qualities, for instance the four color theorem and Fermat's Last theorem). Table 3 shows that for the large number of problems with three universally quantified variables, there is an increase in the average CPU time and rating as the number of existentially quantified variables increases (there is also a correlation for the smaller numbers of problems with less universally quantified variables). Note that the number of existentially quantified variables are counted in both the left hand and right hand concepts, and the larger of the two is the value taken, so that this measures the number of existentially quantified variables not in the conjecture as a whole, but in the most complicated concept. These correlations provide guidelines for generating more difficult problems – setting HR to produce conjectures with many variables, both universally and existentially quantified, is likely to produce more difficult problems.

In addition to looking at the statistics over all the theorems, it is worth examining some individual theorems. For example, the 460th theorem is the following:

$$\forall XY((\exists Z(inv(Z) = X \,\&\, Z * Y = X)\ \ \&\ \ \exists U(X * U = Y \,\&\, \exists V(V * X = U \,\&\, inv(V) = X))) \leftrightarrow$$

Table 2: Correlation with number of $\forall$ variables

$\forall$ variables	Problems	Average CPU Times E	Vampire	SPASS	Average Rating
0	28	0.00	0.20	0.02	0.000
1	301	0.13	0.90	0.12	0.031
2	356	13.37	21.38	0.51	0.233
3	45501	0.80	3.17	0.13	0.097

Table 3: Correlation with number of $\exists$ variables, for 3 $\forall$ variables

$\forall$ variables	$\exists$ variables	Problems	Average CPU Times E	Vampire	SPASS	Average Rating
3	0	5174	0.03	0.26	0.01	0.002
3	1	15707	0.13	0.91	0.07	0.333
3	2	18444	0.94	4.00	0.11	0.138
3	3	5208	2.37	7.23	0.18	0.203
3	4	639	2.58	15.38	0.32	0.264
3	5	32	8.67	36.12	5.44	0.175
3	6	297	9.66	20.89	3.65	0.405

$$(\exists A(inv(A) = X \,\&\, A * Y = X) \;\;\&\;\; \exists B(B * Y = X \,\&\, inv(B) = Y)))$$

Neither Vampire nor E could prove this in the 120 second time limit, but SPASS proved it in 0.2 seconds. To prove this theorem, both sides need to be simplified by removing the existential variables and using some known results in group theory. Firstly, it is known that the inverse of an element is unique, and if $inv(X) = Y$ then $inv(Y) = X$. Hence $\exists Z(inv(Z) = X \,\&\, Z * Y = X)$ can be replaced with just $inv(X) * Y = X$. Using this technique repeatedly throughout, the theorem simplifies to to:

$$\forall XY((inv(X) * Y = X \,\&\, X * inv(X) * X = Y) \leftrightarrow (inv(X) * Y = X \,\&\, inv(Y) * Y = X))$$

By definition, $inv(X) * X = id$, and $X * id = X$, which further simplifies the theorem to:

$$\forall XY((inv(X) * Y = X \,\&\, X = Y) \leftrightarrow (inv(X) * Y = X \,\&\, X = id))$$

Multiplying both sides of the equality: $inv(X) * Y = X$ by X gives $Y = X^2$ and the theorem statement becomes:

$$\forall XY(X = Y = X^2 \leftrightarrow X = Y = id)$$

Finally, the only idempotent element (such that $X = X^2$) in groups is the identity element, so the left hand side can be re-written as $X = Y = id$, which matches the right hand side. QED. On reflection, it is evident that HR has disguised the concept of a pair of elements, both of which are the identity element. The disguise on the left and right, and the conjecture about their equivalence, was of sufficient difficulty to beat the attempts of E and Vampire, but not SPASS.

The theorem that SPASS took the longest time to solve (27.3 seconds), is also worth examining:

$$\forall XYZ((Y * X = Z \,\&\, X * Y = Z \,\&\, inv(Y) = X) \leftrightarrow$$

$$(inv(X) = Y \;\;\&\;\; \exists U(X * U = Y \,\&\, \exists V(V * X = U \,\&\, inv(V) = X)) \;\;\&$$

$$\exists W(\exists A(A * W = Z \,\&\, inv(A) = W) \,\&\, W * Y = Z) \;\;\&\;\; \exists B(inv(B) = X \,\&\, B * Z = X)))$$

From the left hand side, it is evident that HR has again disguised the identity element, as Z must be the identity element. X and Y however, need only to commute.

5 Conclusion

HR has successfully been used to generate group theory problems that are sufficiently difficult to be a challenge to state-of-the-art ATP systems, and therefore sufficiently difficult for inclusion in the TPTP problem library. Success came through optimizations of HR, allowing it to produce very many conjectures very quickly, and subsequently using a selection of ATP systems to verify their theoremhood and difficulty. Analysis of the data has suggested problem characteristics that can be used to help ensure that conjectures generated in the future will be sufficiently difficult.

5.1 Future Work

Having shown that HR is capable of producing theorems that are sufficiently difficult, a goal is to increase the yield of such theorems. Based on the preliminary analysis, it is now possible to reasonably prune many of HR's conjectures before testing them with the ATP systems. In particular, problems with few universally and existentially quantified variables can be rejected. HR will also be used to generate problems in different domains, including domains with non-pure equality axiomatizations, and to produce types of problems other than equivalence conjectures. HR also has many measures of interestingness that can drive a best first search [CBW00b]. In particular, the novelty measure can be used to produce complicated concepts (and hence conjectures) without over-specializing the theory.

It should be noted that all 46186 conjectures produced by HR turned out to be theorems. This was due to the fact that (a) HR's conjectures were (relative to mainstream group theory) fairly simple, and (b) only conjectures true of the groups up to order 8 were made. It would also be desirable to generate hard (to demonstrate) non-theorems for the TPTP. In future runs of HR, generating many more conjectures (millions, rather than tens of thousands), the following protocol will be followed: If any system finds a proof, the conjecture is a theorem, and the rating will determine if it is suitable for inclusion in the TPTP. Otherwise, ATP systems will be used to try show that the conjecture is a non-theorem. If this can be done easily, the conjecture will be discarded. Otherwise, it is a hard theorem or non-theorem, and suitable for inclusion in the TPTP.

The results for the group theory problems show that one system, SPASS, was already well suited to problems using that axiomatization of group theory. Hence, another project of interest is the invention of new axiomatizations or domains that are difficult for all contemporary ATP systems. For this, HR will start in a core algebraic domain with no axioms, then invent operators and axioms on those operators. HR will then determine whether the domain it has invented has any models, and interesting concepts and conjectures. Hopefully, HR will invent a domain containing theorems that are sufficiently difficult for all ATP systems.

A practical goal of this project is to embed HR in the Mathweb architecture of mathematical services [FK99]. HR will be offered as a new service in the software bus, whereby the user can ask HR for a number of theorems of particular difficulty for a particular ATP system (or systems). HR will make use of Mathweb's access to a number of ATP systems, including the SystemOnTPTP service [Sut00], to check the quality of its theorems. In this way, ATP developers will be able to generate new problems and work out why they are difficult (or not) for their system to solve. This, like the TPTP library, will be a valuable service for ATP system developers.

Acknowledgments

Simon Colton is also affiliated with the Department of Computer Science, University of York. This work has been supported by EPSRC grant GR/M98012 and the European Union Calculemus project: (`http://www.eurice.de/calculemus`).

References

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

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

[FK99] A. Franke and M. Kohlhase. System Description: MathWeb, an Agent-Based Communication Layer for Distributed Automated Theorem Proving. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 217–221. Springer-Verlag, 1999.

[McC00a] W.W. McCune. MACE: Models and Counterexamples. http://www-unix.mcs.anl.gov/AR/mace/, 2000.

[McC00b] W.W. McCune. Otter: An Automated Deduction System. http://www-unix.mcs.anl.gov/AR/otter/, 2000.

[RV99] A. Riazanov and A. Voronkov. Vampire. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 292–296. Springer-Verlag, 1999.

[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.

[SFS01] G. Sutcliffe, M. Fuchs, and C. Suttner. Progress in Automated Theorem Proving, 1997-1999. In H. Hoos and T. Stützle, editors, *Proceedings of the IJCAI'01 Workshop on Empirical Methods in Artificial Intelligence*, pages 53–60, 2001.

[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.

[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.

[Tam97] T. Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199–204, 1997.

[Wei99] C. Weidenbach. SPASS: Combining Superposition, Sorts and Splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science, 1999.