The HR Program for Theorem Generation

Simon Colton*

Division of Informatics, University of Edinburgh, UK. simonco@dai.ed.ac.uk

1 Introduction

Automated theory formation involves the production of objects of interest, concepts about those objects, conjectures relating the concepts and proofs of the conjectures. In group theory, for example, the objects of interest are the groups themselves, the concepts include element types, subgroup types, etc., the conjectures include implication and if-and-only-if conjectures and these become theorems if they are proved, non-theorems if disproved. Similar to Zhang's MCS program [11], the HR system [1] – named after mathematicians Hardy and Ramanujan – performs theory formation in mathematical domains. It works by (i) using the MACE model generator [9] to generate objects of interest from axiom sets (ii) performing the concept formation and conjecture making itself and (iii) using the Otter theorem prover [8] to prove conjectures. In domains where Otter and MACE are effective, HR can produce large numbers of theorems for testing automated theorem provers (ATPs), or smaller numbers of prime implicates, which represent some of the fundamental facts in a domain. We explain how HR operates in §2 and give details of a representative session in §3. As discussed in §4, the applications of HR to automated reasoning include the generation of constraints for constraint satisfaction problems, the generation of lemmas for automated theorem proving, and the production of benchmark theorems for the TPTP library of test problems for ATP systems [10]. HR is a Java program available for download here: www.dai.ed.ac.uk/~simonco/research/hr.

2 Automated Theorem Generation

Initialising from Axioms

HR can be given background information in a number of different formats. In particular, it can form a theory from the bare minimum of a domain: the axiom set, which must be supplied in MACE input syntax. HR calls MACE in order to produce a single model satisfying the axioms, which becomes the first object of interest in the theory. Using the -N8 -m1 flags, HR asks MACE to look for a model of size 1, 2, 3, etc. until it finds a single one (stopping after size 8). MACE presents the model in terms of the concepts embedded in the axioms, e.g., in ring theory, MACE produces a single ring of size 1, described by four concepts: the addition and multiplication operations, the additive inverse of elements, and the additive identity. To initialise the theory, HR takes the concepts from MACE's output as the initial concepts in the domain and uses the model from MACE's output to calculate the first examples for these concepts. All future concepts and conjectures are built from the initial concepts.

* This work is supported by EPSRC Grant GR/M98012.

Inventing Concepts

HR invents concepts by passing old concepts through one of 10 production rules (PRs). This mechanism is described in detail in [2] and we give only a brief overview of 4 PRs here. The `match` PR equates variables in definitions, e.g., it could start with the multiplication operation ($a * b = c$) in group theory and equate $a = b = c$ to invent the concept of idempotent elements ($a * a = a$). The `compose` PR uses conjugation to compose two old concepts into a new one which describes objects satisfying both definitions. For example, it starts with the concept of self inverse elements in group theory ($a^{-1} = a$) and composes it with the squaring operation ($a * a = b$), to produce either the square of self inverse elements ($a * a = b \wedge a^{-1} = a$), or elements which square to give a self inverse element ($a * a = b \wedge b^{-1} = b$). The `negate` PR negates certain clauses in a definition, e.g., it produces the concept of non-self-inverse elements ($a^{-1} \neq a$). Finally, the `exists` production rule introduces existential quantification into a definition. For example, this PR invents the concept of elements appearing on the diagonal of a group's multiplication table ($\exists\ b$ s.t. $b * b = a$). Other production rules count set sizes and introduce constants, but produce concepts which are not easily expressible for first order provers, so we omit details here. Also, in future, we intend to enable HR to extract concepts from proofs.

Theory formation is driven by a heuristic search: more interesting concepts are used to build new concepts before less interesting ones. The heuristic measures include both properties of concepts, such as the complexity of the definition, and properties of the conjectures about the concepts, with details given in [3]. In previous versions of HR, in order to avoid repetitions, we specified forbidden paths, namely routes which HR was not allowed to take, so that certain concepts could only be produced in one way. However, we found that this interfered with the heuristic search – often hindering the development of the most interesting concepts. Hence, we removed the forbidden paths, and enabled HR to efficiently prune concepts at the definition stage. That is, if it invents a concept with the same definition as a previous one, the new concept is discarded. Also, if HR invents a concept with a negation conflict, such as ($a * a = a \wedge -(a * a = a)$) or a function conflict, such as ($a * a = b \wedge a * a = c \wedge b \neq c$), the concept is also discarded, because such concepts trivially have no examples.

Empirical Conjecture Making

If a new concept passes the pruning stage, HR calculates the examples for it, and one of 3 situations will occur. In the first case, the example set is empty and HR makes a *non-existence* conjecture stating that the concept's definition is inconsistent with the axioms. For example, when HR invents the concept of non-identity idempotent elements, it cannot find any, so makes the conjecture: $\nexists\ a$ s.t. $(-(a = id) \wedge a * a = a)$. Proved non-existence theorems are used later to prune concepts at the definition stage, i.e., they are used to show that a concept will have no examples before the examples are calculated.

In the second case, the example set is exactly the same as that for a concept already in the theory, and HR makes the *equivalence* conjecture that the new and old concepts have logically equivalent definitions. For example in group

theory, HR invents the concept of idempotent elements, notices that for all the groups it has, the idempotent elements are the identity elements, and makes the conjecture: $a * a = a \leftrightarrow a = identity$.

In the third case, the example set is non-empty and different to those of all previous concepts, and HR adds the concept to the theory. It then endeavours to find some conjectures about the new concept. To do this, HR looks for subsumption conjectures by finding old concepts where the examples are a subset of the examples for the new concept. For each it finds, it makes the implication conjecture that the definition of the old concept implies the definition of the new concept. Accordingly, HR makes an appropriate implication conjecture if an old concept has examples which are a superset of the examples for the new concept.

Extracting Prime Implicates

Given an implication conjecture of the form $A \wedge B \wedge C \rightarrow D \wedge E \wedge F$, HR extracts the implicates $A \wedge B \wedge C \rightarrow D$, $A \wedge B \wedge C \rightarrow E$ and so on. The left-hand and right-hand implication conjectures which make up an equivalence conjecture are dealt with similarly. Given a non-existence conjecture of the form $\nexists(A \wedge B \wedge C)$, HR extracts implicates of the form $A \wedge B \rightarrow \neg C$ and so on. HR discards any implicate if it has already proved a stronger conjecture, i.e., with the same goal but a subset of the premises. From each implicate which survives, HR employs Otter to try to prove it, with a time limit set by the user (usually 10 seconds). If the implicate is true, HR then tries to extract prime implicates, which are implicate theorems such that no subset of the premises implies the goal. While there are more sophisticated methods for finding prime implicates [7], HR employs a naive search for prime implicates: using Otter to try to prove that ever larger subsets of the premises imply the goal, stopping when such a subset is found or the subset becomes the entire set of premises.

Introducing Counterexamples

HR employs MACE to find a counterexample to any implicate extracted from the empirical conjectures that Otter fails to prove. Each counterexample is added to the theory and examples for all the concepts are re-calculated so that similar non-theorems are avoided later. Prime implicates which are extracted from the implicates are assumed to be false if Otter does not prove them, and MACE is not used to find a counterexample. This is because the extracted prime implicates are not supported by empirical evidence, and in practice we have found that, as they are simply stated, if Otter cannot prove them quickly, then they are false in the vast majority of cases (we have yet to see an exception to this rule).

3 A Representative Session

We ran HR with the RNG-004 ring theory axioms from the TPTP library [10] for 1000 theory formation steps, which took 6481 seconds on a Pentium 1Ghz processor. We used the compose, match, exists and negate PRs in a breadth first search, as we were interested in finding prime implicates, rather than interesting concepts. 340 steps resulted in concepts which were discarded due to

their definitions, 255 resulted in new concepts, 401 resulted in equivalence conjectures and 4 resulted in non-existence conjectures. This highlights that there is a fair amount of redundancy in the theory formation, as 340 steps resulted in discarded concepts. Fortunately, such steps are carried out very quickly. The high number of equivalence conjectures is indicative of algebras constrained with many axioms, as many concepts can be proved equivalent. The low number of non-existence conjectures is because rings have an additive identity: it is rare for a concept to not be satisfied by the identity element or some derivative of it.

From the non-existence and equivalence conjectures, HR extracted 275 proved prime implicates, of which 39 had a proof length of 10 or more (indicating the complexity of the theorem). Hence the prime implicates were mostly easy for Otter to prove, but a few managed to tax it. This theorem had a proof length of 40, which was the longest: $\forall\ b, c\ (b*c = c \wedge c*b = b \wedge c+c = b \rightarrow b+b = c)$, and we found that, like this one, the most difficult to prove conjectures involved both the addition and multiplication operations. MACE introduced counterexamples to 30 non-theorems, with 2 models of size 2 and of size 3, 25 models of size 4 and 1 model of size 7, given below along with the conjecture which it disproved.

$\forall\ b, c \in G$
$(b*b = c \wedge b+b =$
$c \wedge c*c = b \rightarrow$
$b^{-1} = c)$

*	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	2	5	4	6	1	3
2	0	5	1	6	3	2	4
3	0	4	6	5	1	3	2
4	0	6	3	1	2	4	5
5	0	1	2	3	4	5	6
6	0	3	4	2	5	6	1

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	5	0	6	4
2	2	3	5	6	1	4	0
3	3	5	6	4	2	0	1
4	4	0	1	2	6	3	5
5	5	6	4	0	3	1	2
6	6	4	0	1	5	2	3

id = 0

inv	0	1	2	3	4	5	6
	0	4	6	5	1	3	2

4 Applications

Automated theory formation has much potential for automated reasoning. Firstly, we have used HR to improve efficiency in solving constraint satisfaction problems (CSPs) [4]. HR extracts concepts and axioms from the basic CSP specification and forms a theory. Then, using various measures of interestingness in HR, we identify concepts and theorems to use as additional constraints for the CSP. When interpreted as constraints, the theorems can be added as implied constraints with no loss of generality. In contrast, the concepts suggest induced constraints, which act as case splits for the problem. Adding both implied and induced constraints can have a dramatic effect on the time taken to solve a CSP. For example, for QG3-quasigroups (where every element appears in each row and column, and additionally: $\forall\ a, b, (a*b)*(b*a) = a$), HR identified (i) an all-different constraint on the diagonal of the multiplication table (ii) a symmetry constraint on the diagonal: $a*a = b \rightarrow b*b = a$ and (iii) that QG3-quasigroups are anti-Abelian, i.e., $a*b = b*a \rightarrow a = b$. These additional constraints greatly improved efficiency, producing a 10-times speedup for some solution sizes.

We have also used HR to generate benchmark problems for the TPTP library of theorems for ATP systems, the de facto standard for assessing ATP systems [10]. Our challenge was to use HR to find theorems that some state of the art provers were able to prove, but others were not. Working in group theory, our first attempt involved a brute force search for theorems. We gave HR the non-isomorphic groups up to order 8 and turned off the theorem proving and

counterexample finding abilities. As discussed in [5], HR produced 46,000 distinct equivalence conjectures in a ten minute session. These were passed to Geoff Sutcliffe (who maintains the TPTP library), who tested them on the Spass, E, Vampire, Gandalf and Otter provers. Spass was able to prove all of them, but 40 were rated 0.8 and 144 were rated 0.6 (provable by only 1 out of 5 and 2 out of 5 provers respectively), and these 184 were accepted into the TPTP library. The next challenge was to find theorems which E could prove, but not Spass. Working again in group theory, we added HR to the MathWeb Software Bus [6], which gave HR direct access to E and Spass (taking Sutcliffe out of the loop). We allowed both provers 120 seconds to prove theorems and ran HR until it had produced 12,000 equivalence conjectures. This managed to produce just four theorems which E proved, but Spass could not prove in 120 seconds, including: $\forall\ b,c,d\ (b*c = d \wedge b^{-1} = c \wedge b*d = c \wedge \exists\ e,f\ (e^{-1} = f \wedge e*b = f) \wedge \exists\ g\ (g \neq id) \wedge d^{-1} = d \Leftrightarrow b^{-1} = c \wedge c*b = d \wedge b*d = c \wedge \exists\ e,g\ (e^{-1} = f \wedge e*b = f) \wedge \exists\ g\ (g \neq id) \wedge d*b = c)$.

Our third application is to automated theorem proving itself. Given the axioms for a theorem, we use HR to form a theory containing many prime implicates, such as the theory described above. Using their proof lengths, we select certain prime implicates to be added to the original problem as lemmas. We aim to show that the time using HR can be compensated for by the decrease in time taken to prove the theorem (or, indeed, sets of theorems) when the lemmas are added. We have positive results with the group theory TPTP theorems supplied by HR itself, but an initial application to TPTP ring theorems (not supplied by HR) has been less successful. Working with ATP designers, we hope to make HR generate and choose lemmas more intelligently to improve the results.

References

1. S Colton. *Automated Theory Formation in Pure Mathematics*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 2000.
2. S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proceedings of the 17th International Conference*, 2000.
3. S Colton, A Bundy, and T Walsh. On the notion of interestingness in automated mathematical discovery. *IJHCS*, 53(3):351–375, 2000.
4. S. Colton and I Miguel. Constraint generation via automated theory formation. In *Proceedings of CP-01*, 2001.
5. S. Colton and G Sutcliffe. Automatic Generation of Benchmark Problems for ATP Systems. In *Proceedings of the 7th Symposium on AI and Mathematics*, 2002.
6. Andreas Franke and Michael Kohlhase. Mathweb, an agent-based communication layer for distributed automated theorem proving. In *CADE 16*, 1999.
7. P Jackson. Computing prime implicates incrementally. In *CADE 11*, 1992.
8. W McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories, 1990.
9. W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories, 1994.
10. G. Sutcliffe and C. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998. `www.cs.miami.edu/~tptp/.`
11. J Zhang. MCS: Model-based conjecture searching. In *CADE-16*, 1999.