Automatic Concept Formation in Pure Mathematics

Simon Colton and Alan Bundy
Division of Informatics,
University of Edinburgh,
80 South Bridge,
Edinburgh EH1 1HN, Scotland.
{simonco,bundy}@dai.ed.ac.uk

Toby Walsh
Department of Computer Science,
University of Strathclyde
Glasgow G1 1XH, Scotland.
tw@cs.strath.ac.uk

Abstract

The HR program forms concepts and makes conjectures in domains of pure mathematics and uses theorem prover OTTER and model generator MACE to prove or disprove the conjectures. HR measures properties of concepts and assesses the theorems and proofs involving them to estimate the interestingness of each concept and employ a best first search. This approach has led HR to the discovery of interesting new mathematics and enables it to build theories from just the axioms of finite algebras.

1 Introduction

The HR program invents definitions in finite algebras such as group and ring theory, and other areas of pure mathematics, such as graph and number theory. Using a set of production rules to derive a new concept from old ones and a set of measures for the interestingness of a concept, HR's best first search bases new concepts on the most interesting old ones. As it invents new definitions, HR uses empirical evidence to spot conjectures. Recently we have interfaced HR with the OTTER theorem prover, [McCune, 1990], to prove some of the conjectures HR makes. When OTTER fails, HR invokes the MACE model finder, [McCune, 1994], to find a counterexample. The proofs from OTTER help HR to assess the concepts involved in the conjectures, and the models given by MACE provide further empirical evidence for future conjectures. This closes a cycle of mathematical activity similar in nature to Buchberger's spiral of creativity, [Buchberger, 1993]. We detail how HR forms and assesses concepts and discuss how this has led to the introduction of new mathematics. We also show how a theory can be constructed from just the axioms of an algebra, and how the heuristic search improves the overall quality of the theory with respect to various measures.

1.1 Background

Lenat, in [Lenat, 1976] chose pure mathematics as the domain for his AM program to demonstrate the use of heuristic search in concept formation. AM re-invented classically interesting definitions and conjectures, such as highly composite numbers and Goldbach's conjecture. The conjectures were based on empirical evidence, but no attempt was made to prove them automatically. Thus AM performed concept formation, conjecture making and counterexample finding. Attempts to build on Lenat's work include the DC program, [Morales, 1985], the Cyrano programs, [Haase, 1986] and Lenat's own Eurisko program, [Lenat, 1983]. The ARE system, [Shen, 1987], used functional transformations to derive new functions from old ones. Shen eliminated many special-purpose operations required by AM and the functional transformations found more complex concepts than AM, such as logarithms.

The IL program, [Sims, 1990], used a generate, prune and prove method to invent operators for number types. The user specified some criteria for, say, multiplication of complex numbers and IL generated plausible operators, empirically checked whether they met the criteria, and if so, proved it. Conjectures were always that the operator met the user's requirements, so IL used a cycle of concept formation, counterexample finding and theorem proving. The concepts produced by the system described in [Bagai *et al.*, 1993] are situations in plane geometry involving properties of points and lines. Conjectures were about the existence of situations and an efficient theorem prover showed that certain situations were impossible. For example, the theorem prover showed that the situation of a parallelogram with parallel diagonals could not exist. Disproving a conjecture meant finding a situation in an infinite number of possibilities, so no counterexample finding was undertaken. The system therefore achieved an interplay of concept formation, conjecture making and theorem proving.

The GT program, [Epstein, 1987], generated and proved theorems in graph theory about properties of graphs. The representation of graphs employed enabled model generation, concept formation and some theorem proving, and GT demonstrated an efficient model for theory formation in graph theory. Another graph theory program, Graffiti, [Fajtlowicz, 1988] makes conjectures in graph theory that one summation of numerical invariants is always less than another. The invariants are given by the user and each conjecture is checked against a large database of graphs. If the conjecture provides a

stronger bound than all previous ones for at least one graph (the dalmation heuristic), and it is not seemingly implied by a previous one (the echo heuristic), the conjecture is posted to a mailing list for graph theorists. Over 60 papers have been written about Graffiti's conjectures, and this has led to important developments in graph theory. We see that Graffiti automates conjecture formation and counterexample finding aspects, but the concepts are provided by the user and the theorem proving is done by the graph theory community.

OTTER, [McCune, 1990], is a state of the art first order resolution theorem prover which uses the negation of a conjecture to find a contradiction, thus proving the conjecture. The model generator MACE, [McCune, 1994], uses the Davis-Putnam method to search for small finite models of first-order statements, and can be used to find example algebras, eg. finite groups or rings.

2 Concept Formation

Working, say, in group theory, HR starts off with a little initial information, namely some example groups and some core concepts. The initial information can be given by the user or generated using MACE and HR (see §3.1). In group theory, the core concepts are the group operation, the identity element and the inverse of elements. HR keeps a data-table of the models of each concept - for example, for the group operation concept, HR stores a data-table with rows $[G, a, b, c]$ where $a, b, c \in G$ and $a * b = c$. HR also keeps a definition of the concept, ie. a predicate which is satisfied by the entries in every row of the data-table. We use the notation:

1. $[G, a, b, c]$: `P(a,b,c)`

to indicate that concept 1 has a data-table with four columns, the first of which is the name of the group, and the last three are triples of elements from that group which satisfy predicate `P`.

2.1 Producing New Concepts

HR invents new concepts by using a production rule to manipulate the data from one (or two) old tables into a new table. Each of HR's 8 production rules performs a simple operation and HR is able to derive a definition for the new concept from a definition of the old concept. For a particular concept and production rule there are many possible manipulations, so a set of parameters give exact specifications. Below, we give a description of the manipulation each production rule (PR) performs and an example from group theory: an input and output concept. A more detailed description of the production rules is given in [Bundy *et al.*, 1998].

The *exists* PR: removes columns from the input data-table. The parameters tell the PR which columns to keep. Eg. using parameters `<1,2,3>`, we get:

$[G, a, b, c]$: `a*b=c` $\rightarrow$ $[G, a, b]$: `exists c (a*b=c).`

The *match* PR: finds rows where columns prescribed by the parameters are equal. Eg. parameters `<1,2,2,2>`, (col1=col1,col2=col2,col3=col2,col4=col2), gives:

$[G, a, b, c]$: `a*b=c` $\rightarrow$ $[G, a]$: `a*a=a.`

The *forall* PR: finds sets of rows with all elements of groups present in particular columns (the parameters specify which columns). With parameters `<4>`, we get:

$[G, a, b, c]$: `a*b=c` $\rightarrow$ $[G, a, b]$: `all c (a*b=c).`

The *negate* PR: constructs rows which do not appear in a data-table. There are no parameters here. Eg.

$[G, a, b, c]$: `a*b=c` $\rightarrow$ $[G, a, b, c]$: `a*b` $\neq$ `c.`

The *conjunct* PR: given two data-tables, X and Y, and parameters `<a,b,...>`, this finds rows in table X for which there is a row in table Y with entry `a` equal to entry 1 in X, with entry `b` equal to entry 2 in X and so on. As in the following example (using parameters `<1,3,2,4>`), the two input concepts can be the same:

$[G, a, b, c]$: `a*b=c` $\rightarrow$ $[G, a, b, c]$: `a*b=c & b*a=c.`
and $[G, a, b, c]$: `a*b=c`

The *size* PR: counts the number of tuples appearing in a data-table. The parameters specify which columns to look for elements in. Eg. with parameters `<3,4>` we get:

$[G, a, b, c]$: `a*b=c` $\rightarrow$ $[G, a, n]$: `n=|{(b,c):a*b=c}|.`

The *split* PR: finds rows where a column contains a particular number. The parameters specify which column to look in and what number to look for respectively. Eg. with parameters `<3,2>`, we get:

$[G, a, n]$ `:n=|{(b,c):a*b=c}|` $\rightarrow$ $[G, a]$ `:|{(b,c):a*b=c}|=2.`

The *compose* PR: certain predicates can be thought of as 1:1 functions. Given such a function, this PR replaces columns in a data-table by the output of the function on the entries in the column. The parameters specify which column to replace. Eg. using parameters `<2>`, with the second concept as the function, we get:

$[G, a, b, c]$: `a*b=c` $\rightarrow$ $[G, a, b, c]$: `inv(a)*b=c.`
and $[G, a, b]$: `b=inv(a)`

Note that *conjunct* and *compose* use 2 old concepts and *size* introduces a numerical value. We don't claim that all interesting concepts in a domain are covered by these 8 production rules, merely that it is possible to construct some interesting concepts using them. Figure 1 shows how HR constructs the concept of Abelian groups:

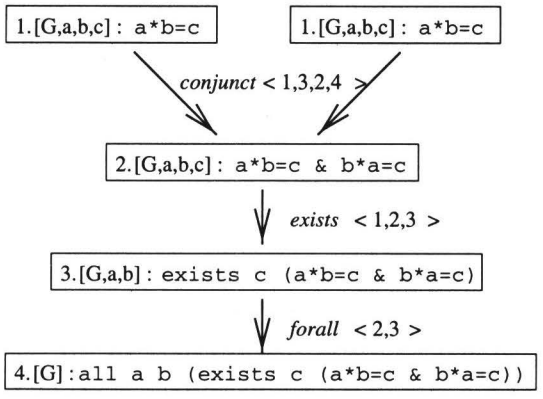

Figure 1: Construction path for Abelian groups.

2.2 Heuristic Search: Measuring Concepts

HR can perform an exhaustive search by using each concept with every production rule, exhausting all possible parameterisations and returning to used up concepts when it is possible to use them in a 2-concept production rule with a new concept. Some concepts will be more interesting than others, and we can improve the search by constructing new concepts from the best old ones. We discuss below five concrete measures HR uses to assess the interestingness of a concept.

Given a data-table, the set of tuples with a particular group name in the first column can be used to describe that group. Smaller descriptions are advantageous, so HR measures the **parsimony** of a concept, which is inversely proportional to the size of its data-table. Similarly, smaller, easier to understand definitions are advantageous, so HR measures the **complexity** of a concept, as inversely proportional to the number of production rule steps used in its construction (which gives a rough guide to the complexity of the concept's definition). HR's set of groups can be categorised by a concept if two groups are thought of as equal if they are given the same description by the concept's data-table. Concepts introducing new categorisations are interesting, so HR measures the **novelty** of a concept, which is inversely proportional to the number of other concepts giving the same categorisation. Every time a new concept is introduced, the novelty of all the old concepts is re-assessed.

Sometimes, the user may be looking for a concept which gives a particular categorisation, for example the classification of groups up to isomorphism (where all pairs of isomorphic groups are categorised as the same, but all pairs of non-isomorphic groups are categorised as different). If the user specifies such a gold standard classification, we can measure the **invariance** of a concept by taking all the pairs of groups which should be categorised as the same and finding the proportion of pairs which are categorised correctly by the concept. Similarly, the **discrimination** measure finds the proportion of pairs of groups which should be categorised as different that are categorised correctly by the concept.

To use the heuristic of basing new concepts on the most interesting previous ones, HR measures and orders its concepts after a given number of new concepts have been introduced, usually 10. To order the concepts, each of the above measures is normalised to a value between 0 and 1 (0 the worst score, 1 the best), and a weighted sum is taken. The user sets the weights, with the choice dependent on what type of theory they are looking for, eg. if they were looking for many different categorisations, the novelty measure would be heavily weighted. Often, to avoid a conflict, one or more weights are set to zero, eg. the novelty measure prefers concepts giving new categorisations, but the discrimination and invariance measures prefer concepts giving a particular categorisation, causing a conflict. The production rules can also be ordered because each concept is produced by a single rule, hence the average score of the concepts output by a production rule can be used to assess the rule.

3 Theory Formation

A set of concepts, however interesting, does not comprise a theory. A theory should also have, at minimum, a set of theorems and proofs. In §3.2 we discuss how HR spots and makes conjectures, and in §3.3 and §3.4, we discuss how HR uses OTTER and MACE respectively to try to prove or disprove the conjectures. At present, to form a theory, we cut back a little on HR's concept formation abilities, and only use the *conjunct, exists, forall, match* and *negate* production rules. This is because OTTER does not deal well with inductively defined concepts like the integers introduced by the *size* and *split* production rules. Also, the *compose* production rule usually adds a layer of complexity to conjectures which makes it difficult for OTTER to prove them in an acceptable time limit.

3.1 Starting a Theory

In algebraic domains such as finite group theory, HR can construct a theory from the bare minimum of information: the axioms. To do this, HR first passes the axioms to MACE, which is asked to construct a single example group. MACE must be told the size of the group it is to find, so HR first gives MACE 10 seconds to find a group with 1 element, then 10 seconds to find a group with 2 elements and so on until size 8, after which it is unlikely that MACE will succeed. In the unlikely event that no model of any order is found, HR must admit defeat as its concept formation process is model based. To describe a model it has found, MACE outputs a set of tables, each with a name. HR examines these tables and for each one, extracts the name and data contained as an initial (core) concept. In group theory, MACE finds the trivial group with one element and uses three tables to describe it. HR takes these tables and extracts the group operation, identity and inverse concepts - those core concepts which were implicit in the axioms.

3.2 Making Conjectures

Whenever HR invents a new concept, it immediately checks whether a previous concept has the same data-table. For example, one of the first concepts HR invents in group theory is: $[G, a]$: `a*a=a`. After finding the models for this concept, HR runs through the previous concepts, and finds that the only elements, `a`, for which `a*a=a` are the identity elements and vice versa. When HR finds a match like this, it makes the conjecture that the definitions are equivalent. ie.

```
all a (a=id <-> a*a=a).
```

There are other types of conjecture that HR can make. For example, if HR finds that the models of a newly formed concept are a subset of the models of a previous concept (or vice versa), HR can make the conjecture that one definition is a specialisation of the other. For example, HR makes the conjecture that the identity element is always a member of the centre of a group:

```
all a (a=id -> all b (exists c (a*b=c & b*a=c))).
```

For simplicity, conjectures hereafter are assumed to be of the first type: definition1 <-> definition2.

3.3 Proving Conjectures

Theorem proving is, in general, a difficult activity, so HR must give OTTER the best chance of proving the conjectures. Using the 5 production rules mentioned, each conjecture can be written in a succinct way acceptable to OTTER, and will be of the form:

```
all a b c
( P1(a,b,c) & P2(a,b,c) & ... <-> Q1(a,b,c) & ... ).
```

To give OTTER the best chance of proving this, HR splits it into easier problems (subgoals):

```
( P1(a,b,c) & P2(a,b,c) & ... -> Q1(a,b,c) ).
( P1(a,b,c) & P2(a,b,c) & ... -> Q2(a,b,c) ).
.           .           .           .
( Q1(a,b,c) & Q2(a,b,c) & ... -> P1(a,b,c) ).
( Q1(a,b,c) & Q2(a,b,c) & ... -> P2(a,b,c) ).
.           .           .           .
```

Of course, if the predicate on the right hand side coincides with a predicate on the left hand side, the subgoal is trivially true, so it is not passed to OTTER. Also, it is possible that the subgoal has been looked at in order to settle a previous conjecture, so HR stores all previous results, and uses them where possible. It also checks whether a subgoal which has the same right hand side, but a subset of the conditions on the left hand side has been proved - if so, the present subgoal follows as a corollary. Entirely novel subgoals are passed to OTTER with the axioms of the algebra. OTTER is then given a fixed time limit to prove the subgoal, usually 10 seconds. If all the subgoals are proved, the conjecture becomes a theorem and is added to HR's collection.

Regardless of the difficulty to prove a conjecture, if the concepts in it are too complex, OTTER has little chance of proving it in a short space of time. For this reason, when theory forming, we usually impose the condition that no concepts with a complexity greater than a given threshold are used to base new concepts on. If OTTER is only given 10 seconds to find proofs, it is a good idea to impose a complexity threshold of 5.

3.4 Disproving Conjectures

When HR has few example groups to provide empirical evidence, it often makes false conjectures such as:

```
all a b c (a*b=c <-> a*b=c & b*a=c).
```

which states that all groups are Abelian. When false conjectures are passed to OTTER, it either fails outright, or runs out of time. The unproved subgoals from the conjecture are then passed to MACE. HR gives MACE 10 seconds to find a counterexample of size 1, then 10 seconds to find one of size 2, and so on. For the untrue conjecture above, MACE finds the smallest non-Abelian group, $D(3)$ - with six elements. When a new group is found, HR reads MACE's output once again and adds to the data-tables of the core concepts. Then HR uses the new data to recalculate the data-tables for all its old concepts. By introducing new models only when needed to disprove a conjecture, HR guarantees that the models are also interesting - each model has a property which is true of no previous one.

3.5 Closing the Maths Cycle

Each new group found by MACE is different to the ones it already has, because it disproves a conjecture which was true of all the others. Once introduced, all further calculations will involve the new group, which will add more empirical plausibility to the conjecture making. Also, with the notion that concepts are more interesting if you can prove some interesting facts about them, we can use the conjectures, theorems and proofs to derive more measures for the interestingness of a concept.

HR has two ways to assess conjectures. Firstly note that each conjecture states that one definition is equivalent to another. Figure 2 shows a typical conjecture - two concepts have been constructed, with the dotted line indicating that those definitions are equivalent. The shaded nodes belong to the construction paths of both equivalent concepts and all nodes except X are concepts already found in the theory.

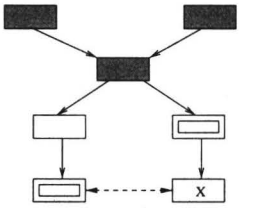

Figure 2: Construction path for a typical conjecture.

The more divergent the construction paths for the two definitions, the less obvious the fact that they are equivalent. Therefore, HR measures the **surprisingness** of a conjecture as the number of distinct concepts which appear in one, but not both, construction paths. In figure 2, noting that X is the same as a previous concept, the conjecture scores 3 for surprisingness.

When OTTER proves a subgoal, it outputs a proof length score, and an average of this over all the subgoals will roughly indicate the **difficulty** of the overall conjecture. Proof length is a measure of the interestingness of the proof (and hence the conjecture), but others, such as the number and length of clauses produced, have been suggested by OTTER's author, William McCune, as better alternatives, which we are currently implementing. A weighted sum of surprisingness and difficulty is used to assign a value between 0 and 1 to each theorem. Then, the average score for the conjectures a concept appears in can be used to order the concepts themselves. Note that, the conjecture in figure 2 is most concerned with two old concepts - those appearing with an extra border, and the interestingness of the conjecture is used to assess the interestingness of both these concepts.

HR has only limited ways to investigate and attempt to settle open conjectures (those which cannot be proved or disproved by OTTER or MACE in the given time). Thus we are presently more eager for HR to make provable conjectures. So, even though these are probably the most difficult and interesting, HR gives open conjectures a score of 0 for difficulty and the difficult concepts involved in the conjectures are discriminated against. We discuss how HR can deal with open conjectures in §5.

4 Results

4.1 Concept Formation

HR works in number theory by first generating a table of factorisations for integers 1 to 100. It can turn concepts into integer sequences by (i) listing integers of a particular type in numerical order, eg. prime numbers (ii) applying a function to the integers 1, 2, 3, etc. eg. the τ function (number of divisors) (iii) identifying integers setting a record for the highest value output by a function, eg. highly composite numbers (with more divisors than any previous integer), which was AM's major re-invention. HR finds all the number types found by AM and re-invents over 50 well known integer sequences. HR also outputs many integer sequences not found in the recognised repository, the online encyclopedia of integer sequences, [Sloane, 1999], which contains over 47,000 sequences. 8 sequences found by HR have been accepted into the encyclopedia, for example, this interesting sequence: $1, 2, 7, 38, 122, 2766, 64686, \ldots$ (those integers setting a record for $f(n) = |\{a < n : a + \tau(a) = n\}|$).

Presently, we believe that 7 of these 8 sequences are genuinely new inventions. Perhaps the most interesting sequence found by HR is the tau numbers (where the number of divisors is itself a divisor): $1, 2, 8, 9, 12, 18, 24, 36, \ldots$ This was missing from the encyclopedia, and a search of the relevant literature produced no reference, so we believed this to be a new invention, and we called these numbers 'refactorable'. However, we recently discovered that they were first defined in 1990, [Kennedy and Cooper, 1990]. HR made some interesting conjectures about tau numbers, eg. that they are congruent to 0, 1, 2 or 4 mod 8. Proving this and others from HR led us to some interesting results, [Colton, 1999], eg. odd tau numbers are squares. There are also some open conjectures, such as whether there are any triples of tau numbers (there are none $\leq 10^{53}$).

Asked to find concepts classifying groups up to order 6, (ie. the concept can be used to decide whether any two groups up to order 6 are isomorphic), HR finds many calculations unknown to the authors. The function: $f(G) = |\{(a, b, c) \in G \times G \times G : a * b = c \wedge b * c = a\}|$, classifies groups up to order 6 and was genuinely surprising as we hadn't thought such a simple function could perform the task. When given an hour to find as many categorisations as possible, HR finds around 200 categorisations of the groups up to order 6. Also, we chose 20 definitions from a group theory text, to see how many HR re-invented. We noted that HR found these 9:

- *Abelian, cyclic and exponent 2 groups, elements, identities, inverses and orders of elements, orders and centres of groups,*

but 7 required better handling of subgroups:

- *Normal subgroups, quotient groups, cosets, index of subgroups, simple groups, central series, derived subgroups*

and 4 involve concepts from other domains:

- *Elementary Abelian, dihedral, quarternion and p-groups.*

A hit rate of nearly 50% is encouraging, and in future, more production rules and working in 2 domains at once will increase the yield of classically interesting concepts.

4.2 Theory Formation

When forming theories, the complexity of concepts is capped and some production rules are removed, so the theories formed are more superficial. HR finds some conjectures of note, eg. groups of exponent 2 are Abelian:

```
all a (a*a=id) -> all a b (exists c (a*b=c & b*a=c))
```

In 9 experiments, we gave HR 2 hours to construct a theory. We used a complexity threshold of 5 and a weighting of 25% for the novelty of a concept, so that HR didn't specialise too much, and 75% for the interestingness of the theorems each concept was involved in. The interestingness of conjectures was measured with equal weights for surprisingness and difficulty. In (G)roup and (Q)uasigroup theory and (R)obbins algebra, we compared the (B)est first search choosing which concept and production rule to use next, against a (R)andom choice, and the (E)xhaustive search described in §2.2. In table 1, we recorded the content of the theories produced, ie. the number of concepts, theorems, open conjectures and models produced and two measures of the quality of the theories, namely the average proof length (difficulty) and surprisingness of all the proved theorems.

Algebra	Heuristic	Concepts	Theorems	Open Conjectures	Models	Av. Proof Length	Av. Surpris-ingness
G	B	143	495	19	6	13.6	1.7
G	E	91	665	9	5	12.2	1.6
G	R	190	465	12	6	8.8	2.1
Q	B	150	192	80	13	4.5	1.5
Q	E	243	184	87	15	3.6	1.0
Q	R	268	232	58	17	3.5	2.0
R	B	40	173	73	3	28.1	1.2
R	E	50	136	91	3	11.3	0.9
R	R	51	70	85	3	14.8	1.5

Table 1: Summaries of 9 theory formation sessions.

We draw two conclusions from these results:

- The nature of the theories produced is more dependent on the axioms than the choice of search method.
- The best first search improves the overall quality of the theory, with respect to the difficulty of theorems.

Conclusion 1 is clear because the rows in table 1 vary much more between algebras than between different searches with the same algebra. As expected, in quasigroup theory, with the least restrictive axioms, MACE finds most models and, as shown by the average proof length column, quasigroup conjectures are more easily proved by OTTER than those about the more complex algebras. From the significant increase in average proof length gained by using the best first search over the other searches, it is clear that the heuristic improves the quality of the theory with respect to the difficulty of the theorems produced. A similar increase is observed when HR uses the best first search to find concepts with classifying abilities. The best first search concentrates on

particular concepts, so the random search covers a larger space, and therefore spots more varied conjectures, which explains the increase in average surprisingness in the random searches over the best first searches.

5 Future Work and Conclusions

Presently, HR can make conjectures which OTTER has little chance of proving. To address this imbalance, we intend to interface HR with inductive theorem provers which can better handle conjectures with numerical content. Also, we wish to improve HR's treatment of open conjectures. We noted in §3.5 that HR gives open conjectures a score of 0 for difficulty, which is counter-intuitive, but sensible while HR has limited possibilities for settling such conjectures. Once HR has better tools to settle open conjectures, we will increase the interestingness of, and thus time spent investigating, open conjectures.

The obvious way to address open conjectures is to give OTTER and MACE more time to settle them. However, we find that the number of settled conjectures does not increase in line with the time given to the task, and more sophisticated techniques are required. HR already has one method to disprove previous open conjectures - when MACE finds a new model, HR checks whether this is a counterexample to any previous open conjecture. This approach can be effective - we have documented an example semigroup which disproved 14 of HR's open conjectures. HR can also use previously proved theorems as lemmas in open conjectures. However, lemma choice is a difficult problem and we hope to use techniques such as gazing, [Barker-Plummer, 1992], to improve matters. Our first attempt at lemma use - using the previous theorem with the most similar construction tree - was encouraging, as we found a conjecture where the lemma reduced OTTER's proving time from 75 to 8 seconds.

Concept formation in HR is (i) novel, because it is the first to work in domains as diverse as group theory, number theory and graph theory and (ii) important, as it has led to the introduction of new mathematics, most notably the integer sequences. Theory formation in HR is (i) novel, as it is the first to integrate with third party software to perform concept formation, conjecture making, theorem proving and counterexample finding and (ii) a bootstrapping system - a theory including models, definitions, theorems, proofs and open conjectures can be created from just the axioms of an algebra. HR improves on AM and Graffiti by using theorem proving when constructing theories, and improves on IL, GT and the Bagai et al system, by covering a wider range of concept types over more domains. Introducing models only when needed to disprove conjectures and driving a heuristic search with concrete measures of interestingness increases the quality of the theory produced and has brought HR success in automated mathematical concept formation. We hope to continue increasing the interestingness of the concepts and theories produced, building a system sophisticated enough for working mathematicians to explore new domains with.

6 Acknowledgments

This work is supported by EPSRC grants GR/M45030 and GR/K/65706. We would like to thank the anonymous reviewers for their useful comments.

References

[Bagai *et al.*, 1993] R Bagai, V Shanbhogue, J Żytkow, and S Chou. Automatic theorem generation in plane geometry. In *LNAI 689*. Springer Verlag, 1993.

[Barker-Plummer, 1992] D Barker-Plummer. Gazing: An approach to the problem of definition and lemma use. *Automated Reasoning*, 8(3):311–344, 1992.

[Buchberger, 1993] B Buchberger. Mathematica: A system for doing mathematics by computer? Technical Report 93-50, Risc-Linz Institute, Johannes Kepler University, 1993.

[Bundy *et al.*, 1998] A Bundy, S Colton, and T Walsh. HR - automatic concept formation in finite algebras. Technical Report 920, Department of Artificial Intelligence, University of Edinburgh, 1998.

[Colton, 1999] S Colton. Refactorable numbers - a machine invention. *Journal of Integer Sequences*, 2, 1999.

[Epstein, 1987] S Epstein. On the discovery of mathematical theorems. In *IJCAI Proceedings*, 1987.

[Fajtlowicz, 1988] S Fajtlowicz. On conjectures of Graffiti. *Discrete Mathematics 72*, 23:113–118, 1988.

[Haase, 1986] K Haase. Discovery systems. In *ECAI Proceedings*, 1986.

[Kennedy and Cooper, 1990] R Kennedy and C Cooper. Tau numbers, natural density and Hardy and Wright's theorem 437. *IJMMS*, 13, 1990.

[Lenat, 1976] D Lenat. *AM: An artificial intelligence approach to discovery in mathematics.* PhD thesis, Stanford University, 1976.

[Lenat, 1983] D Lenat. Eurisko: A program which learns new heuristics and domain concepts. *AI*, 21, 1983.

[McCune, 1990] W McCune. The OTTER user's guide. Technical Report ANL/90/9, ANL, 1990.

[McCune, 1994] W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, ANL, 1994.

[Morales, 1985] E Morales. DC: a system for the discovery of mathematical conjectures. Master's thesis, University of Edinburgh, 1985.

[Shen, 1987] W Shen. Functional transformations in AI discovery systems. Technical Report CMU-CS-87-117, Computer Science Department, CMU, 1987.

[Sims, 1990] M Sims. *IL: An artificial intelligence approach to theory formation in mathematics.* PhD thesis, Rutgers University, 1990.

[Sloane, 1999] N Sloane. *Online Encyclopedia of Integer Sequences.* www.research.att.com/~njas/sequences, 1999.