HR - A System for Machine Discovery in Finite Algebras

Alan Bundy, Simon Colton[1] and **Toby Walsh**[2]

Abstract. We describe the ***HR*** concept formation program which invents mathematical definitions and conjectures in finite algebras such as group theory and ring theory. We give the methods behind and the reasons for the concept formation in ***HR***, an evaluation of its performance in its training domain, group theory, and a look at ***HR*** in domains other than group theory.

1 Introduction

We are trying to identify aspects of the concept formation processes performed by pure mathematicians. To understand these processes, we turn them into computational procedures undertaken by a computer program, ***HR***, acting as a machine discoverer in pure mathematics. Working primarily in group theory, we aim to:

1. Allow ***HR*** to invent a large subset of concepts found under the 'definition' headings of standard group theory texts.

2. Use procedures which can be applied to any finite algebra, and possibly other areas of mathematics.

3. Produce results understandable by mainstream mathematicians.

4. Keep the theory behind the concept formation clear and concise.

1.1 Background

Machine discovery in mathematics had an enthusiastic start and early programs, such as Lenat's AM, as described in [4], excited people because they re-invented classically interesting mathematical concepts. For example, AM looked at elementary number theory, and re-invented well known concepts such as prime numbers and square numbers, and well known conjectures such as Goldbach's conjecture (that every even number is the sum of two primes), and the unique prime factorisation theorem.

Unfortunately, the theory behind the concept formation performed by AM is very difficult to understand. In particular, the distinction between the space to be searched and the heuristics employed is very blurred. The methodology behind and reporting of AM's achievements has been widely criticised. Ritchie and Hanna, in [16] believe:

> ... that it would be extremely difficult to base further work in this area on AM, since the disparity between the written account and the actual program means that there is not in fact a tested theoretical basis from which to work.

There were, however, attempts to extend Lenat's work. Lenat himself, in [11], wrote Eurisko, an extension of AM which invented its own heuristics. Other efforts to improve Lenat's work include the Cyrano programs by Haase, [9], which identified problems in Eurisko and put them right, and the DC program from Morales, [15]. Shen, in [17], tidies up some theoretical aspects underlying AM by separating the space to be searched and the heuristics behind the search strategy. Shen concentrates on the first aspect, introducing functional transformations which derive one function from another. These functional transformations were implemented in the ARE system which was used in AM's domain, elementary mathematics with encouraging results. Among other improvements over AM were the elimination of many special-purpose operations and a more parsimonious and understandable theory of concept formation.

Breaking from the AM mould, Sims, as detailed in [18], chose a narrower range of concepts to be invented by his IL program. Given specifications for operators on number types such as Conway numbers, IL invents possible operators and uses the heuristic of building on those operators which satisfy the most specifications. IL follows the routine of generating a possible operator, testing empirically whether it satisfies the specifications, and if it does, using a first order theorem prover to prove or disprove that it works for all the numbers. Sims calls this the generate, prune and prove method.

Epstein's GT, as discussed in [6], works in graph theory, and generates and proves theorems of particular types, such as: a graph has property P iff it has property Q. Another graph theory program, Fajtlowicz's Graffiti, as described in [7] is probably the only machine discovery program which has inspired mathematicians to write papers about its work. Graffiti uses a database of around 200 graphs and heuristically checks for conjectures about graph invariants. Typically, for each graph, it calculates the number of occurrences of some phenomena, such as cliques, and compares it with the number of some other phenomena. If, for every graph in the database, the number of one is always less than the number of the other, this conjecture is made for the general case. Some of the conjectures made by Graffiti have inspired mathematicians to prove or disprove them, and this has sometimes lead to new mathematics, for instance, [5].

All the programs cited above have empirical aspects. Bagai et al in [1], break this mould. They invent theorems in plane geometry by starting with a blank plane and adding general points or general lines through points or relationships between lines and points, such as a point being on a line or two lines being parallel. Each addition is represented as either a function or a predicate which allows an efficient theorem prover to check for consistency whenever a new addition is made. A theorem is proved from every inconsistency reported.

For the ***HR*** project, we intend to learn from all of these programs. In particular, we want ***HR***, like AM, to explore entire domains, but, like Shen, we want it to do so using a clear theory of concept formation. Like IL, we want ***HR*** to have specific tasks for its concepts to perform, and like Graffiti, we want our program to produce and inspire new mathematics.

[1] Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN. Scotland
[2] APES Group, Department of Computer Science, University of Strathclyde, Glasgow G1 1XL. Scotland.

1.2 Search Space and Strategy

Consider the following mathematical objects:

- Equilateral triangles in geometry, which are those triangles with **all** their sides the same length.
- Cliques in graph theory, which are those subgraphs in which each node is connected to **all** the others in the subgraph.
- Central elements in group theory, which are those elements which commute with **all** the others.

Although from different areas of mathematics, we see that in each case a new concept arises as a specialisation of an old concept where some property is true in as many cases as possible. This technique allows the specialisation of triangles into equilateral triangles, subgraphs into cliques and elements (of groups) into central elements. Not only do techniques like this occur across domains, they repeatedly occur within the same domain. To see this, we only have to look at Abelian groups: these are groups where **all** the elements are central elements. We have identified six general production rules such as the one above, for inventing a new mathematical definition by basing it on an old one. The concepts which can be reached using these production rules make up the space ***HR*** searches when performing concept formation. The production rules are discussed in §2.

These objects also have something in common:

- The Alexander polynomial of a knot.
- The set of element orders of a group.
- The number of nodes in a graph.

Each of these is a certain kind of calculation, namely an invariant, ie. some value which does not differ however the object is represented. For example, if one draws a knot on paper, there are many possible outcomes depending on which direction the knot was viewed, and the same knot can easily be given two very different 2D representations. However, no matter how a knot is drawn, the Alexander polynomial, which can be calculated from the drawing, will always be the same. Similarly, no matter how one chooses to label the elements of a group, the set of element orders will always be the same, and obviously the number of nodes in a graph does not change if the nodes are labelled differently or the edges are moved. We see that, when addressing the task of identifying objects, calculations which are invariants have a desirable property.

This, and other tasks, have been identified for ***HR*** to complete, and measures with respect to the tasks have been derived by which we can assess the importance of any concept produced. This has enabled the use of a heuristic search strategy for the concept formation. This strategy is to (i) measure each concept, (ii) measure each production rule in terms of the value of the concepts it produces and (iii) build the new concept from the best old concept, using the best production rule. The tasks addressed by ***HR***, the measures of the concepts and the heuristics used are discussed in §3.

2 How *HR* Invents Concepts

Most concept formation in group theory occurs by basing a new concept on others which have already been defined. For example, the concepts of central elements, centres of groups, Abelian groups and central series are all based on the idea of commutativity. Each step from one concept to another involves a slight change in the semantics of the concept. For instance, from a pair of commutative elements of a group, G:

$$(a, b) \in G \times G \text{ s.t. } a * b = b * a,$$

we easily reach the concept of central elements:

$$a \in G \text{ s.t. } \forall\, b \in G, a * b = b * a.$$

The collection of central elements gives us the centre of the group and if the centre is the entire group, we have an Abelian group. This process of basing one concept on others is a good way to automate some of the concept formation process, and we have identified six general production rules to do this. Five of these production rules base a new concept on one old one, and a single production rule bases a new concept on two old ones.

To determine these production rules, we first narrow down the types of concept they deal with, to functions which output the results of a calculation made directly from the Cayley table (as described in §2.1) of any input group. To encompass concepts about element types, the collection of the types of elements is taken as the output to a function. So for instance, we can include the concept of central elements by representing them as a function taking a single group as input which outputs the set of central elements (ie. the centre). With these kinds of constructions, we can cover a good deal of the concepts formed in elementary group theory, and can reason about each concept as if it is a function taking a single group as input. This does not cover higher level concepts such as sequences of groups, but we introduce them later, as discussed in §2.4.

2.1 Representation of Concepts in *HR*

The next thing to address is how to represent concepts in ***HR***. In fact, concepts are represented by only two things. Firstly, the information about how they were constructed is stored, namely the old concept, production rule and parameters (as discussed in §2.2) which were used to construct the concepts. Each production rule performs a mathematically accountable operation, so, if we know the definition of the old concept, then given only the construction history of the new concept, we can express its definition. Secondly, the results of the calculation are stored as a representation of the concept, in a data table, and examples of these tables are given later. The data table is the main form of representation, and the construction information is not used in the concept formation process.

Noting that each concept produced may be used later as input to a production rule, we see that the input and output format of each production rule must be the same - a data table. This allows us to express the transformations of concepts as manipulations of one data table into another. To start the process, we give ***HR*** only one concept, the Cayley tables of the models of the algebra it is working with. All the concepts produced are based on this. Cayley tables of groups are pictorial ways of representing the product, $a * b$ of every pair of elements, a and b in the group, G. To be input to a production rule, this information must be in the form of a data table. So, these Cayley tables:

C1.1	1
1	1

C2.1	1	2
1	1	2
2	2	1

C2.2	1	2
1	2	1
2	1	2

are represented as the following data table:

Table 0

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	1	2	2
C2.1	2	1	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	1	2	1
C2.2	2	1	1
C2.2	2	2	2

This table is all ***HR*** needs to start its concept formation process. The functional definition of the calculation producing this table is:

$$f_0(g) = \{(a, b, c) \in G \times G \times G : a * b = c\},$$

The result of the calculation for a particular group can be lifted directly from the table, by taking all the rows where its name appears in the first column (but not taking the first column of the rows). It is also important to note that this calculation can be used to describe the group. So, for example, table 0 describes C2.1 with this list of triples: [[1,1,1],[1,2,2],[2,1,2],[2,2,1]]. These descriptions are used to categorise the groups, which is central to the assessment of the concepts, as discussed in §3.

2.2 The Production Rules

Each production rule simply takes in an old data table (or two old data tables), and manipulates them in some way to produce a new data table. For a given old data table, there are a number of ways a production rule could turn it into the new table, and exactly what will happen must be specified in a set of parameters given to the production rule. The action of the production rules, the parameters which control them and examples of them at work are given here in roughly ascending order of complexity.

The `exists` Production Rule

This simply removes columns from a data table. The parameters for this production rule tell it which columns to keep under the condition that column 1 is never removed, as this would remove the subject of the calculation. If this production rule is given table 0 above, and the parameters tell it to keep columns 1 and 2 only, we get:

Table 0

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	1	2	2
C2.1	2	1	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	1	2	1
C2.2	2	1	1
C2.2	2	2	2

`exists` $\Longrightarrow$ `[1, 2]`

Table 1

G	a
C1.1	1
C2.1	1
C2.1	1
C2.1	2
C2.1	2
C2.2	1
C2.2	1
C2.2	2
C2.2	2

and the functional definition of the calculation producing table 1 is:

$$f_1(G) = \{a \in G : \exists (b, c) \in G \times G \text{ s.t. } a * b = c\}.$$

The `permute` Production Rule

This permutes the columns of a data table. The parameters tell it which permutation to use under the condition that a column which previously contained objects of one type cannot contain objects of a different type after the permutation. If this production rule is given table 0 above, and the parameters tell it to use the permutation: $\left(\begin{smallmatrix}1&2&3&4\\1&3&2&4\end{smallmatrix}\right)$ we get:

Table 0

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	1	2	2
C2.1	2	1	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	1	2	1
C2.2	2	1	1
C2.2	2	2	2

`permute` $\Longrightarrow$ `[1, 3, 2, 4]`

Table 2

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	2	1	2
C2.1	1	2	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	2	1	1
C2.2	1	2	1
C2.2	2	2	2

and the functional definition of the calculation producing table 2 is:

$$f_2(G) = \{(a, b, c) \in G \times G \times G : b * a = c\}.$$

The `match` Production Rule

This extracts rows from a data table where certain columns have identical entries. The parameters tell it which columns must have identical entries under the condition that no two columns are matched where the type of object in the first is different to that in the second. If this production rule is given table 0 and the parameters tell it to extract rows where columns 2, 3 and 4 are the same, we get:

Table 0

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	1	2	2
C2.1	2	1	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	1	2	1
C2.2	2	1	1
C2.2	2	2	2

`match` $\Longrightarrow$ `[1, 2, 2, 2]`

Intermediate

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.2	2	2	2

$\Longrightarrow$

Table 3

G	a
C1.1	1
C2.1	1
C2.2	2

(Note that the [1,2,2,2] parameters say that column 1 should match column 1, and columns 2, 3 and 4 should all match column 2). Columns 3 and 4 are discarded in the above process, because they always contain the same elements as column 2, and are redundant. The functional definition for the calculation producing table 3 is:

$$f_3(G) = \{a \in G : a * a = a\}.$$

The `conjunct` Production Rule

This is the only production rule which takes two old data tables as input. It extracts those rows in the first table which somehow match a row in the second table, with the parameters telling it how to match the rows, under these conditions: (i) column one is always matched to column one, and (ii) no two columns are matched if they contain objects of different types. Note that if a row in the first table matches n rows in the second table, the row is repeated n times in the new table, as this information should not be lost. If this production rule is given tables 0 and 2 above, and the parameters tell it to find rows in table 0 for which there is an equal row in table 2, we get:

Table 0

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	1	2	2
C2.1	2	1	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	1	2	1
C2.2	2	1	1
C2.2	2	2	2

&

Table 2

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	2	1	2
C2.1	1	2	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	2	1	1
C2.2	1	2	1
C2.2	2	2	2

`conjunct` $\Longrightarrow$ `[1, 2, 3, 4]`

Table 4

G	a	b	$a * b$
C1.1	1	1	1
C2.1	1	1	1
C2.1	1	2	2
C2.1	2	1	2
C2.1	2	2	1
C2.2	1	1	2
C2.2	1	2	1
C2.2	2	1	1
C2.2	2	2	2

(Note that the [1,2,3,4] parameters say that column 1 of table 0 should match column 1 of table 2, column 2 of table 0 should match column 2 of table 2, and so on). Table 4 is exactly the same as table 1 because the functional definition of the calculation producing table 4 is the following:

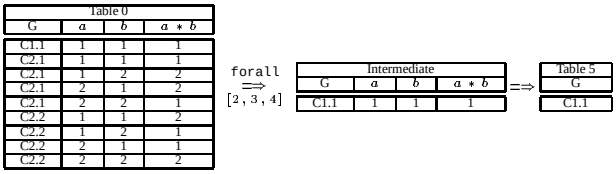
$$f_4(G) = \{(a, b, c) \in G \times G \times G : a * b = c \text{ and } b * a = c\},$$

and the groups C1.1, C2.1 and C2.2 are Abelian, so all triples (a, b, c) which satisfy $a * b = c$ also satisfy $b * a = c$.

The `forall` Production Rule

As we are working with finite tables, we can check that all the elements of a group are present in a particular column of a data table, or that all possible pairs of elements are found in two columns of a data table, etc. The parameters for this production rule tell it which columns to look for elements in, under the condition that the columns actually contain elements. Then, suppose row no. X contained data

about group G, ie. G was the name in the first column of row X. Then row X is kept only if there is a row for every element of G (or pair of elements, etc.), which matches row X. If this production rule is given table 0 above and the parameters tell it to find all groups where every triple $(a, b, c) \in G \times G \times G$ appears in columns 2,3 and 4, we get:



Again, columns 2,3 and 4 are discarded as we are only interested in which groups have all the triples, and if a group has all of them, we know what they are, and are not interested. The functional definition of the calculation producing table 5 is:

$$f(G) = \text{ true iff } \forall\,(a, b, c) \in G \times G \times G, a * b = c,$$

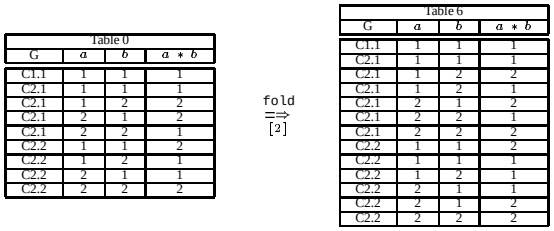
and the output in this case is boolean. Therefore, table 5 describes C1.1 as 'true' and all the other groups as 'false'.

The `fold` Production Rule

This production rule only works if the last column of the data table contains elements of the group. Given a data table with n columns, a function, g, is defined in the following way: $g(o_1, \ldots, o_{n-1}) = o_n$ iff the row $[o_1, \ldots, o_n]$ is found in the data table (and is the first row found which matches $o_1, \ldots, o_{n-1}$). The parameters then tell this production rule how to use g to fold the element in the last column into another column, under the condition that the other column actually contains elements. Suppose the data table has four columns, the last of which contains elements and the parameters tell the production rule to fold the last column into column 2, which has elements in. Then, for every row, $[G, a, b, c]$, an orbit of elements is constructed by left folding the output of g into column 2 in the following way:

$x_1 = g(G, a, b)$, $\quad x_2 = g(G, g(G, a, b), b)$,
$x_3 = g(G, g(G, g(G, a, b), b), b)$, $\quad$ etc.

This continues until the orbit makes a cycle (which must happen in a finite group). After all the rows have had their orbits collated, a new table is constructed with columns $[G, a, b, x_i]$ for each x_i found in the orbit of $[G, a, b]$. Note that the parameters are allowed to fold the last element into more than one column. If this production rule is given table 0 above and the parameters tell it to fold the last column into column 2, we get:



The functional definition of the calculation producing table 6 is:

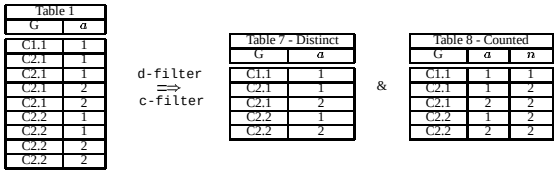
$f(G) = \{(a, b, c) \in G \times G \times G :$
$c = a * b,$ or $c = (a * b) * b,$ or $c = ((a * b) * b) * b,$ etc $\}$.

2.3 Negation and Counting

In group theory, positive examples of concepts are often more interesting than negative examples. For instance, central elements have a place in the theory, but non-central elements do not (probably because the central elements form a subgroup, but the non-central ones do not). However, these negative examples are genuine mathematical constructions and they should be included in the search space. In fact, they may give ***HR*** its best chance to find an important concept that has been overlooked classically. We decided that the negation of a concept should be formed at the same time as the concept itself, to cut down on computation time. There is no easy way to produce negative tables with the `exists`, `fold` or `permute` production rules. However, when the `conjunct`, `forall`, and `match` production rules are used, they produce two tables, one with positive examples of the concept, and one with negative examples. The negative tables are as you would expect: when using the `conjunct` production rule, the rows in the first table which do not match those in the second table constitute the negative table. Similarly, those rows which do not have the full complement of elements attached form the negative table for the `forall` production rule, and those rows where the columns do not match correctly form the negative table for the `match` production rule.

There is one further complication. You may have noticed that table 1 above has repeated rows. This is undesirable for two reasons. Firstly, every production rule works by going through the old data table row by row and manipulating each row into a new row in the new table. The manipulation would be repeated if there are repeated rows in the input table, and this would be a waste of time. Secondly, the data table produced does not quite match the output functional definitions of the calculation if there are repeated rows. In the functional definition for table 1, there is no mention that each element, a is repeated n times (where n is the number of pairs of elements (b, c) for which $a * b = c$).

For these reasons, any data table with repeated rows is passed through a filter which removes the duplication. However, in doing this, the information about how many times the row appeared in the table is lost, and this information could be useful. Therefore, alongside the filtered table, a second table is produced which calculates how many times the distinct row appeared in the original table. For example, the filtering process produces these tables from table 1:



Note that the n column in table 8 contains integers. The functional definition of the calculation producing table 7 is the same as that for table 1, but now the table actually fits the definition. The calculation producing table 8 has the following functional definition:

$$f_8(G) = \{(a, n) : a \in G, n = |\{(b, c) \in G \times G : a * b = c\}|\}.$$

The filtering process guarantees that any table produced has distinct rows. So, any table input to a production rule will have distinct rows. Because of this, most of the production rules produce a table which is also guaranteed to have distinct rows and the filtering process is only needed for tables produced by the `exists` production rule and the positive tables produced by the `conjunct` production rule.

2.4 Other Concept Types

Calculations made from the Cayley table of a group are the simplest type of concepts, and there are more interesting concept types which can be built from these which ***HR*** can find. Firstly, ***HR*** invents maps from one group to another. At the moment, this is done in only one way, but we intend to look into other methods. Presently, a map from

one group to another is formed by taking a calculation concept which outputs a set of elements, S, for every input group, G, and seeing if a new group can be derived from these elements. There are two ways ***HR*** tries to derive a new group from S: (i) it checks if S forms a group, that is, if the set of triples $(a, b, c) \in S \times S \times S$ satisfy the group axioms, and (ii) it checks if S generates a group, that is, if the set of triples $(a, b, a * b) \in S \times S \times G$ satisfy the group axioms, where $*$ is the operation from the parent group, G. If a group can be derived, and it is sometimes different from the original, then ***HR*** has invented a map concept.

From maps of groups, ***HR*** can form another type of concept, namely sequences of groups, where a subgroup of a group is itself checked for subgroups and so on. Unfortunately, in group theory, the order of a subgroup must divide the order of the parent group (Lagrange's theorem), and when only a handful of models of groups are in the database, the sequences ***HR*** produces are short and not very interesting. However, due to the natural ordering of the integers, the sequences produced in number theory, as described in §5.1, are much more interesting. ***HR*** also forms categorisations of groups using the descriptions they are given by the data tables. These categorisations are discussed more extensively in the next section.

3 Why *HR* Invents Concepts

In the previous section, we decided how ***HR*** would turn one concept into another, thus defining the space ***HR*** searches when performing concept formation. Using all the production rules except `conjunct`, and taking into account all the negative tables and all the counted tables, there are 44 possibly different tables which can be produced from table 0 alone. Each one of these can be input to a production rule, and any two can be used in the `conjunct` production rule, so the search ***HR*** undertakes is subject to the combinatorial explosion, and an exhaustive search is infeasible.

One way to improve the search is to cut down redundancy. As the search progresses only by basing one data table on another, if there are two equal data tables in the database, there will be duplication of work if one is not removed. For this reason, whenever ***HR*** constructs a new data table, it checks to see if it already has this table in the database and discards it if so. A more pro-active way of avoiding duplication of concepts is to stop particular paths. For instance, the `permute` production rule can be used to make any permutation of the columns, and so there is no point having two `permute` steps in a row, as this path is possible in just one permutation. ***HR*** has a set of forbidden paths which cut down the trivial duplication of concepts. Also, if a data table produced has no rows in it, then no progress will be made from this table, and it is discarded.

Another way to improve the search is to use a heuristic to decide what to do next. For each concept formation step, there is a choice of (a) old concept to use, (b) production rule to use and (c) parameterisation for the production rule, and we require sensible ways for ***HR*** to make these choices. At the moment, the parameters are not chosen by any heuristic, ***HR*** simply tries each possible parameterisation in a pre-determined order. Future heuristics may choose the parameterisation, but at the moment, they concentrate on choosing the best old concept and best production rule to use in the next step.

To choose which old concept to use next, ***HR*** measures each data table in certain ways, and takes a weighted sum of the measures. It then chooses the old concept with the highest total to base the next concept on. To choose which production rule to use next, ***HR*** uses the fact that each concept is produced by a single production rule. So, for a particular production rule, an average of the scores for each concept it produced can be used to measure the production rule itself. Therefore, we only need to derive sensible measures of the concepts produced to determine a heuristic search.

To derive measures, we wanted to give ***HR*** concrete tasks to achieve. These tasks, along with the relevant measures we have derived are discussed in §3.1 and §3.2. One measure which is independent of the task being addressed is the **parsimony** of the calculation. As previously stated, for an input group, the results of a calculation can be used to describe the group, and it is desirable to have these descriptions as succinct as possible. The amount of data produced in the calculation corresponds to the size of the descriptions and so the parsimony measure is based on the size of the data table: If a data table, T, has r rows of c columns then:

$$\text{Parsimony}(T) = \frac{1}{cr}.$$

Another independent measurement which is soon to be implemented is the **clarity** of a concept. This depends on the construction path of the concept, which is a hierarchy of the concepts which have been built upon in succession to arrive at the present concept. As this list gets larger, it becomes more difficult to understand what the concept actually is. For this reason, we propose that the clarity measure should be inversely proportional to the depth of the construction path. We may also introduce a bias towards construction paths containing concepts produced by easy to understand production rules. For example, a concept with lots of `conjunct` steps in its construction path is usually difficult to understand.

3.1 Classification of Groups

In the introduction, we mentioned certain calculations called invariants, and pointed out that the Alexander polynomial gives knot theorists the ability to tell when two knots are different (if the polynomials calculated for each are different). Unfortunately, the Alexander polynomial, and the other invariants they have invented so far, sometimes give the same result for two different knots. In fact, knot theorists are attempting to classify knots, that is, find some way of describing knots which is an invariant, but which differentiates between every pair of distinct knots.

Similarly, group theorists are trying to classify groups up to isomorphism. Two groups are isomorphic if there is a 1:1 mapping of the elements from the first to the elements of the second which makes the Cayley table of the first look identical to that of the second. Mathematically, the map ϕ, taking group G to group H should follow the rule that $\forall\, a, b \in G, \phi(a * b) = \phi(a) \bullet \phi(b)$, where $*$ is the operation in G, and $\bullet$ is the operation in H. To a group theorist, two groups are essentially the same if they are isomorphic, because they differ only in the choice of labels for the elements. Unfortunately, using Cayley tables to check if two groups are isomorphic is computationally expensive, although some good algorithms have been devised for this purpose, one of which is described in [14].

Historically, classification has been a high priority in group theory. In 1870, Kronecker devised a classification of Abelian groups up to isomorphism. This theorem states that each Abelian group can be decomposed into a cross product of cyclic groups, and that this decomposition is an invariant up to isomorphism, but differs between every pair of non-isomorphic Abelian groups. Therefore, given any two Abelian groups, a group theorist can find their decomposition into cyclic groups, and say for sure whether they are the same or not up to isomorphism. Also, in 1980, a landmark in human achievement was reached when the classification of finite simple groups was achieved, as described in [8].

So, classification of groups up to isomorphism is a worthy motivation for ***HR*** to form concepts. This can be stated as a task: given a set of groups, say [C1.1, C2.1, C2.2, C3.1, C3.2], and their classification up to isomorphism: [C1.1],[C2.1,C2.2],[C3.1, C3.2], (where groups in the same list are isomorphic), ***HR*** is asked to find a calculation which, when used to describe groups, gives the same categorisation of the groups as the isomorphic classification. This allows us to use the usual tactic in problem solving which is to generate possibilities, choose the one which best fits the solution criteria, and use this to generate more possibilities. We have derived the **invariance** and **discrimination** measures for a data table which show how good a concept is at telling whether two isomorphic groups are the same, and how good a concept is at telling whether two non-isomorphic groups are different.

Before stating these measures, we should appreciate that the classification of groups up to isomorphism is just one way to classify groups. This classification is very important, for obvious reasons, but a group theorist may also want to find a calculation which, say, classifies groups as cyclic and non-cyclic, (ie. they want a calculation which gives the same value for each cyclic group, but a different value for non-cyclic groups). For this reason, ***HR*** does not automatically try and classify groups up to isomorphism, but is given a 'gold standard' categorisation of groups by the user which it should aim for (of course, this could be the isomorphic classification). This gold standard can also be a partial categorisation if the user is only interested in one or more particular categories, and isn't worried about how the other categories look.

Now, for a data table, T, and a set of groups, S, the descriptions of each group can be lifted from T and used to categorise the groups in S, by saying that two groups are in the same category if they have the same description. We use the notation $G =_T H$ to show that group G is in the same category as group H in the categorisation of S given by T. We also use the notation $G =_{GS} H$ to show that G and H are in the same category in the gold standard categorisation. Then, the following measures are calculated for T:

$$\text{Invariance}(T) = \frac{|\{(G,H) : G =_{GS} H \text{ and } G =_T H\}|}{|\{(G,H) : G =_{GS} H\}|}$$

$$\text{Discrimination}(T) = \frac{|\{(G,H) : G \neq_{GS} H \text{ and } G \neq_T H\}|}{|\{(G,H) : G \neq_{GS} H\}|}$$

These give us an idea of how well the concept performs with respect to the task of classifying the groups.

3.2 Understanding the Domain

When ***HR*** forms concepts in order to classify groups up to isomorphism (or any other categorisation), it is looking for a particular calculation which completes the task, and when it has one, it can stop. This is a reactive process: a group theorist may be given a categorisation of groups and be asked to find a calculation which achieves this categorisation. One way for a group theorist to be pro-active is to try and make as many different categorisations of the group as possible, so they may have an answer at hand for future questions. This may sound unproductive, but would certainly help to understand the domain. The more categorisations the group theorist comes up with, the more ways they have of saying that two or more groups are/aren't the same, and this is an understanding of the domain at one level.

To encourage the production of many different categorisations, the **novelty** measure of a concept is used, which is based on how often the categorisation has been seen. If, when a new table, T, is formed, the categorisation of groups it produces has been seen m times, and there are n distinct categorisations of groups in the database, then the novelty of the concept is given by:

$$\text{Novelty}(T) = 1 - \frac{m-1}{n}$$

Note that every time a new concept is invented, the novelty value for every other concept is possibly affected, and they have to be updated.

To understand the domain at a higher level, it is necessary to conjecture and prove theorems about the objects in the domain. ***HR*** has no theorem proving abilities, although we intend to link it to a theorem prover, as discussed in §6. However, it can make conjectures based on empirical evidence. One type of conjecture which can be made is that two calculations are always the same, no matter what group they are performed on. For example, for any group, G, these calculations:

$$f_a(G) = \{a \in G : a * a = a\}$$

$$f_b(G) = \{b \in G : b \text{ is an identity of } G\}$$

always produce the same (singleton) set of elements. So the conjecture in this case, which is easy to prove, is:
For any group, G, a is an identity in G iff $a * a = a$.

If ***HR*** invented f_a then f_b above, it would notice that the results for both calculations is (empirically) always the same, decide whether or not to state this conjecture, and then discard f_b, to avoid duplication of work, as discussed in §3. A decision about whether the conjecture is worthy of keeping is required because many of these conjectures occur in a typical concept formation session. ***HR*** assesses the likely difficulty to prove such conjectures by looking at the construction paths of the two calculations. These paths have to pass two thresholds. Firstly, they must have a combined length greater than the **complexity threshold** (usually taken at 6 or 7). This ensures that at least one of the concepts being conjectured about is not trivial. Secondly, taking all the concepts which appear in either path, the percentage of them which appear in only one path must be above the **difference threshold** (usually taken at 70%). This ensures that the two calculations are not trivially related, and so the proof (or disproof) of the conjecture will probably be non-trivial.

When ***HR*** specialises the notion of group, eg. into cyclic groups, the conjectures it makes from these concepts are only about the special groups. A **plausibility** measure for conjectures is planned, which will ensure that the conjecture is about a large subset of the groups ***HR*** has, and not just a few. One of the reasons Graffiti's conjectures are so plausible is that they are true for a large number of graphs. Note that the measures of conjectures do not affect the search, as they are not measures of the concepts themselves. We plan to implement a measure of each concept which takes the average of the interestingness of the conjectures for the concept and thus helps to push the search towards concepts producing interesting conjectures. We believe that making conjectures is a more worthy task for ***HR*** than just inventing concepts, and we intend to improve ***HR***'s conjecture making abilities.

3.3 User Defined Parameters

In summary, the heuristic search works by (i) measuring each concept in various ways and taking a weighted sum of the results for a total indication of the importance of the concept, (ii) measuring each production rule in terms of the concepts it has produced, and (iii) choosing the best old concept to base the new concept on, using the best production rule to do this and running through every possible parameterisation of the production rule. The measures ***HR*** has

at the moment are: parsimony, invariance, discrimination and novelty, and the weights for the weighted sum of these can be set by the user. At present, there are 'modes' which *HR* runs under. One such mode is the classifying mode, where the weights for the parsimony and novelty are zero, so these measures do not affect the search, and *HR* spends all of its time looking for a classifying calculation. There is an equivalent mode for novelty, and an exhaustive mode, where all the weights are zero. The user can define and keep their own modes.

The other parameters which can be changed to alter the search are the initial ordering of the production rules, the forbidden paths and when to reorder the agenda. *HR* can measure and re-order the concepts and production rules at each step, but this produces a slower search. In practice, the reordering is only performed after some new concepts have been found. How many concepts *HR* must invent before reordering is chosen by the user. Values around 10 seem to produce the best results. Note that, once a concept and production rule pair have been used up (ie. all the possible parameterisations have been tried), *HR* keeps the present concept, but moves on to the next best production rule in the list. Only when all the production rules have been tried does *HR* move on to the next concept in the list.

A problem with this search method is that if a concept is not producing new concepts, the reordering will not happen, and the concept will continue to be looked at. For this reason, *HR* has a **productivity threshold** which the user can set. If a concept has been used in, say, ten steps in a row, none of which have produced a new concept, then the search moves to the next concept in the list. Of course, this might not be a good idea if the user is looking for conjectures. In any case, the productivity threshold can be set by the user, and this, along with the other extra parameters are supplements to the main heuristic intended to make the search quicker and more productive.

4 An Assessment Of *HR* in Finite Group Theory

We have chosen three out of many ways to assess *HR*. In §4.1, we assess whether *HR* is accomplishing the tasks we have set it, namely the classification of groups and gaining an understanding of the domain. In §4.2 we choose twenty concepts occurring classically in finite group theory and check whether *HR* re-invents them, giving derivations for those it finds and explanations for those it misses.

Before any more detailed assessment, we can assess *HR* by the four criteria laid down by Valdes-Perez in [21]. These criteria are about the knowledge produced by the programs, and not about the theory behind the concept formation. This is because they are meant to assess programs such as Graffiti, which collaborate with human researchers and so a means-end analysis is appropriate. *HR* aspires to be such a program and it is appropriate to assess it by these criteria. Valdes-Perez states that machine discoverers in science should produce knowledge which is (i) novel (ii) interesting (iii) plausible and (iv) intelligible, and we look at these in turn.

The concepts *HR* produces are likely to be novel because the search space it looks at is large, and it finds concepts which humans have overlooked. Group theory has been intensely studied, and the concepts *HR* produces in lesser known algebras are likely to be novel more often. The conjectures *HR* makes involve these novel concepts, and there is a good chance that they too will be novel. As detailed in §5.1 below, a sequence of integers *HR* found has been overlooked by humans, and is certainly novel.

Assessing whether a concept is interesting in pure mathematics is a very difficult task. It is unlikely that Diophantine equations would be so interesting if a correct proof to Fermat's last theorem had been found in his margin. It is possible that *HR* will define a novel concept that may prove useful in solving a problem or which appears in some data from the physical sciences, and then the concept will become interesting. It is exactly for this reason that we give *HR* specific tasks to do because, if it finds concepts which achieve the task, we can appeal on utilitarian grounds that they are interesting.

As stated previously, the conjectures *HR* makes are not as plausible as they could be, because they are based on a small amount of empirical evidence. *HR*'s concepts, however, are plausible because models of them exist. For example, if *HR* invented the concept of a group which has two distinct identities, it would find no examples of such a group, and discard the concept. They are also plausible because they are the product of a series of mathematically accountable operations.

This also adds to the intelligibility of the concepts. We have worked very hard to allow the user to understand *HR*'s inventions and there are four ways the output can be investigated. Firstly, there are command line queries available within the Prolog environment. Secondly, using the dot program [10], *HR* draws graphs showing the construction history of its concepts. Thirdly, *HR* writes and displays a marked up LaTeX script which gives a mathematical definition of each concept, the examples of it *HR* has, the categorisation of the groups given by it and any notes or conjectures *HR* has about the concept. Finally, there is a Java front end, a demonstration of which can be seen at [2]. This allows the user to navigate through the concepts and find ones with certain properties. We also hope to reduce the number of concepts which are difficult to understand, using the clarity measure described in §3.

4.1 Does *HR* do the Tasks it was Set?

All the sessions in group theory have started with the same set of data: the Cayley table of the trivial group and two isomorphic Cayley tables for each group up to order six. This comes to 15 Cayley tables which make a data table containing 285 quadruples.

Classification Up to Isomorphism

If asked to find a calculation which classifies the group up to order six, a group theorist would probably collate the orders of the elements for each group. In a series of sessions with varying parameters, *HR* has come up with over 50 ways to classify these groups. Of these, we've identified only three genuinely distinct core classifying concepts, with the others closely related to these. None of the core concepts were the set of element orders, as to get to them, *HR* had to pass through the concept of elements and their orders, which scored zero for invariance, as it depends on the naming of the elements. However, one of the core classifying concepts *HR* invented was very close to the idea of element orders. This, and another core classifying concept have fairly complicated definitions. However, we were genuinely surprised when *HR* invented the following concept, and told us that it classified the groups up to order 6:

$$f(G) = |\{(a,b,c) \in G \times G \times G : a * b = c \text{ and } b * c = a\}|.$$

We had thought that there wasn't a classifying concept with such a concise definition.

We should also assess the heuristic of building on concepts which produce a classification close to the gold standard. Unfortunately, the above classifying concept was not produced while *HR* was using this heuristic, but was found during a session when *HR* was exploring and understanding the domain. However, this is an exception, and almost all the other classifying concepts have been produced by using the classification heuristic at some stage. Interestingly, we have found

that using the heuristic straight away is a bad idea, as ***HR*** doggedly follows paths from concepts invented at the start of the session, which never lead to a classifying concept. So, in practice, we ask ***HR*** to find 50 or more categorisations of groups by exploring the domain before we ask it to look specifically for a classifying concept. It then has a broader range of good concepts to choose from, and can afford to abandon dead ends.

Using a combination of the novelty and classifying modes, ***HR*** has found a concept which categorises the groups in every way we have asked it. We have given it the following gold standards: classification up to isomorphism, classification into Abelian and non-Abelian groups, and into cyclic and non-cyclic groups, classification by identity element and others. In every case, ***HR*** has found a concept which classifies the groups correctly. It is not always true that the concept is the one we would expect (eg. the concept of the identity elements to classify the groups by their identity). However, in terms of achieving the classification, ***HR*** does its job well, and a comparison with other classification algorithms is planned.

Understanding the Domain

Here we can measure how well ***HR*** is doing by counting the number of categorisations of groups it comes up with. In a recent run, we asked ***HR*** to come up with 200 different ways of categorising the 15 groups in its database, using the the novelty mode previously described. It took about one hour to accomplish this task, and invented 381 distinct concepts to do this. Noting that the categorisations give us ways of comparing and contrasting groups, we measured the average number of reasons ***HR*** could give why any two groups are the same (ie. a calculation performed where the output is the same for both). There were 72 reasons ***HR*** could give. Also, ***HR*** had 38 reasons it could give, on average, why any three groups were the same, and 28 reasons why any four groups were the same.

We have found that the heuristic performs well in this task, and the novelty measure is a good way of getting the search to find new categorisations of groups. An exhaustive search for an hour produces less than 60 ways of categorising groups. Therefore, using the novelty measure increases the number of distinct categorisations formed by a factor of three. This has been borne out in many other similar comparisons of performance. Note that we want to improve ***HR*** before assessing its conjecture forming abilities.

4.2 Re-inventing Classical Definitions

Perhaps the most ambitious way to assess ***HR*** is to see if it re-invents any concepts which appear in standard group theory texts. Of course, as ***HR*** is trained in group theory, progress is made by identifying concepts it cannot find and adding production rules so that the search space expands to contain the desired concepts. In the development of ***HR***, each new production rule has enlarged the space to include multiple new classically interesting concepts. For instance, the `forall` production rule brought centres of groups, Abelian groups and identities of groups into reach, and the `fold` production rule brought element orders and cyclic groups within reach. Therefore, it should come as no surprise that ***HR*** finds these concepts.

However, ***HR*** does itself no favours by throwing away concepts with equal data tables to previous ones. For example, it might come up with one concept which outputs the centre of the group for all the groups it has, but the calculation is not actually defined by the centre of the group. When it does define the centre of the group, this will be thrown away because the data table is the same as the previous one. It may turn out that the two definitions are provably logically equivalent, and in some cases, we have to claim that as a partial success, until ***HR*** retains all the definitions it makes. In other cases, the correct definition is returned, and there is no arguing that ***HR*** has re-invented the concept. Sometimes, nothing like the concept is returned, and we give one of three excuses for these omissions: (i) the concept is in the search space, but is missed by the heuristic search, (ii) the concept is not in the search space, but we intend to expand ***HR***'s capabilities to find it and (iii) the concept is not in the search space, and it is unlikely that it will be, for some good reason.

The following twenty concepts have been picked as representative of those found in standard group theory texts. ***HR*** only starts with the group operation, ie. the set of triples (a, b, c), such that $a*b = c$, so it has to invent even the most elementary of concepts. It easily finds the **elements of a group** using `exists` with the group operation data. It finds the **order of the group** by using this elements concept in another `exists` step, and counting their number. In group theory, the left identities are the same as the right identities, so ***HR*** never retains both tables, and so cannot `conjunct` them together to get the concept of an **identity of a group**. ***HR*** does, however, come up with the logically equivalent definition of identities: those elements, a for which $a * a = a$. ***HR*** misses **inverses** of elements for the same reason it misses identities.

HR does re-invent the concept of **cyclic groups** - those groups for which there is an element which generates the entire group - by using `fold` to produce the orbits of pairs of elements under the group operation, then `match` to find the orbits $a, a^2, a^3, \ldots$, then `forall` to find an element with all the other elements in its orbit. The route to the concept of **elements and their orders** also uses `fold`, and ***HR*** reaches this concept. Unfortunately, as mentioned earlier, this concept scores zero for invariance, and paradoxically, only one step away is the invariant concept of **the set of element orders,** which is not reached because the heuristics steer it away from the badly scoring concept. The `forall` production rule was implemented for ***HR*** to reach, amongst other things, Abelian groups, but these were found in another way, using the negation of `conjunct` twice to define a group, G for which: $\nexists (a, b) \in G \times G$ for which $a * b \neq b * a$, which is very close to the classical way of defining Abelian groups: G for which $\forall (a, b) \in G \times G, a * b = b * a$. **Elementary Abelian groups** and **p-groups** are out of ***HR***'s reach because it does not know about prime numbers. As prime numbers are important in many domains, we could justify giving ***HR*** a primality test. Even though they have equivalent algebraic definitions, is unlikely that ***HR*** will re-invent the definition of **dihedral groups** as they have their origin in geometry, or **quarternion groups** as they originate from a number type.

It will take some extra production rules for ***HR*** to invent the important concept of **left cosets,** and we have to be sure that the production rules we introduce are indeed general. For this reason ***HR*** misses a large area of very important group theory, as left cosets are needed to define **quotient groups** from which we get **normal subgroups,** and **simple groups.** We intend to introduce general production rules which bring some of this theory within range. ***HR*** does re-invent the **centre of groups** using `permute` to look at triples (a, b, c) for which $b * a = c$, then using `conjunct` to look at triples for which $a * b = c$ and $b * a = c$. It then uses `exists` to look at pairs (a, b) for which $a * b = b * a$, then `forall` to look at those a for which $\forall b \in G, a*b = b*a$. These are the central elements, and their collection is the centre of the group. ***HR*** does not yet use subgroups broadly enough to invent concepts such as the **index of subgroups,** but we intend to add these aspects to the functionality. Finally, because ***HR*** does not find the quotient group, and series of groups are defined using this, ***HR*** finds none of the classically interesting series of groups,

such as the **central series** and **derived series**. (Derived series also need the concept of a **commutator,** which we hope to extend ***HR***'s production rules to find). ***HR***'s sequence making skill in group theory is an area we intend to improve upon.

A quick count of the twenty concepts shows that ***HR*** finds (a) 5 exactly, (b) logically equivalent definitions for 2 of them, and ***HR*** misses (c) 2 because of the heuristic search missing them, (d) 9 which we hope to include in future versions of ***HR***, and (e) 2 which we do not intend ***HR*** to find at the moment. We aim to move concepts from (b), (c) and (d) to (a) in future versions of ***HR***.

5 *HR* in Other Domains

Another way of evaluating ***HR*** is by looking at its performance in other domains. Because the input that ***HR*** takes is simply a data table, and the production rules it uses are very general manipulations of data tables, it is possible to look at any algebra and lots of other areas of mathematics. We concentrate here on number theory and graph theory. We also mention some other algebras, and explain why we have not assessed ***HR*** in these domains yet.

5.1 Number Theory

It recently occurred to us that information about integers, such as their factorisations, can be represented as a table, ie.

$$1 \times 1 = 1, 1 \times 2 = 2,$$
$$2 \times 1 = 2, 1 \times 3 = 3 \quad \Longrightarrow$$

Integer	Factor	Factor
1	1	1
2	1	2
2	2	1
3	1	3

Without making any modifications to HR, we ran it with the factorisations of the integers up from 1 to 100. As we weren't interested in any particular classification of the integers, we ran it in the novelty mode so that it produced as many classifications of the integers as possible. We were surprised at how many classically interesting number theory concepts were in the search space. Among the initial concepts ***HR*** produced were square numbers and prime numbers, and the tau function, which counts the number of factors of an integer.

At one stage, ***HR*** invented a type of number, which when listed in numerical order gave this sequence of numbers:

$$1, 2, 8, 9, 12, 18, 24, 36, 40, 56, 60, 72, 80, 84, 88, 96,$$

which looked fairly interesting. Before asking ***HR*** to tell us the definition of this sequence, we checked whether this was a known sequence of numbers, using the online encyclopedia of integer sequences [20]. This service informed us that the sequence was not among the 30,000 or so in its database. Because of this, we expected the sequence to have a very unintuitive and complicated definition, and be rather dull. We were amazed to find that they have a very succinct definition: they are those integers for which the number of factors is itself a factor of the integer. We have called these re-factorable numbers.

Encouraged by their novelty and succinct description, we have submitted them to the online encyclopedia of integer sequences, and they were accepted - their reference number is A033950. We have also proved that there are an infinite number of re-factorable numbers and that every odd re-factorable number is a square number. There are still two open conjectures concerning pairs of re-factorable numbers. The examples between 1 and 10,000 are (1,2), (8,9) and (1520,1521), and there are only thirteen such pairs between 1 and 1,000,000. The conjectures are: (i) there are an infinite number of such pairs and (ii) (1,2) is the only pair where the odd number comes first. If these conjectures are taken up by the mathematical community and take a lot of work to settle, re-factorable numbers may make it into mainstream mathematics. In any case, this is perhaps the first case of ***HR*** producing a mathematical concept of some interest.

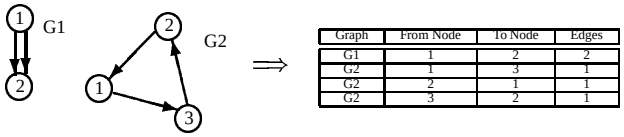
Note that ***HR*** did not make any of the conjectures or theorems just stated, but did make some, such as one concerning those re-factorable numbers, n, which are such that the number

$$|\{(a, b) \in N \times N : a \neq b \text{ and } a * b = n\}|$$

does **not** divide n. The conjecture ***HR*** made was that these numbers are always square numbers, and we have proved this also. One proposed extension to ***HR***'s functionality in number theory is to get it to accept a sequence of integers, and it give the next in the sequence, by forming a concept which explains the sequence. This is similar to the work done in [13]. ***HR*** will perform better in number theory once we give it the information about addition of integers, and then it should find concepts such as perfect numbers and triangle numbers.

5.2 Graph Theory

We have only very recently looked at graph theory, by turning directed graphs into tables, by listing the pairs of from-nodes and to-nodes on an edge, and how many edges are between the two nodes. eg.



Graph	From Node	To Node	Edges
G1	1	2	2
G2	1	3	1
G2	2	1	1
G2	3	2	1

While we have not looked closely at ***HR***'s abilities in graph theory, certain elementary concepts such as multigraphs, pseudographs and disconnected graphs have been found. At present, ***HR***'s fold production rule only gives the first orbit of a function, and this may not be unique. For this reason, ***HR*** can invent the concept of cycles, but doesn't calculate all the cycles within a graph, only the first one encountered for each edge. Other concepts such as cliques should be in ***HR***'s search space, but haven't been seen in the output yet. Also, there is an isomorphic classification in graph theory, so we can test whether ***HR*** can classify graphs as well as it classifies groups. Unfortunately we have not had time to do this yet.

We hope to compare and contrast the conjectures formed by Fajtlowicz's Graffiti with those formed by ***HR***. Certainly, Graffiti's conjectures will be more plausible because they have been tested on a larger number of graphs. However, ***HR***, unlike Graffiti, will be able to claim that not only did it invent the conjecture, it also invented the concepts about which the conjecture is stated.

5.3 Other Algebras

We have deliberately avoided evaluating ***HR***'s performance in other algebras such as ring theory, and will continue to do so until we near the end of the project. Once all the functionality we intend for ***HR*** has been implemented and trained on the domain of group theory, we will assess ***HR*** in other algebras, and use the assessment as part of an overall evaluation of the generality of the methods we employed. We hope to show that the theories formed in these algebras are influenced not only by the search space and the search strategies, but by the domain itself. This will be true because ***HR*** cannot do anything with an empty table, and so will discard any concept for which there are no examples. So, for instance, if a particular algebra we are looking at has no central elements, then this concept will be discarded. While

it may be dangerous to ignore a concept on empirical grounds, it will mean that the domain itself influences the concept formation.

HR has its own constraint solver enabling it to produce examples of algebras given only their axioms. Using this, we have models of groups, loops, quasigroups, semigroups, moufang loops and medial quasigroups, and intend to use ***HR*** on some of these lesser known algebras, and others, to (hopefully) invent something new and of use.

6 Conclusions and Future Development

We can now address the four aims we set down in the introduction. Firstly, we have allowed ***HR*** to invent a small subset of concepts found under the 'definition' headings of standard group theory texts, and have pointed out many ways we intend to improve the coverage. Secondly, our procedures are (i) production rules which work with data tables which could come from any algebra and (ii) heuristics based on measures for classification and understanding the domain, which work on any tables, given any gold standard categorisation. Hence our procedures can certainly be applied to any algebra, and we have shown some success applying them to other areas of mathematics, namely number theory and graph theory. ***HR***'s results are understandable by mainstream mathematicians, as they are given in a standard mathematical language, and are constructions they should be used to. Finally, by explaining the theory and implementation of ***HR*** in this short article, we hope to have shown that the theory behind the concept formation is clear and concise.

Along with all the other minor improvements to ***HR*** we have mentioned previously, such as more production rules and more measures, we have two major additions we wish to make. The first of these is to improve ***HR***'s abilities in understanding the domain. We wish to improve its conjecture making skill by letting ***HR*** make many types of conjectures, such as: the centre of a group is always a subgroup. We also wish to link ***HR*** to OTTER, a first order resolution theorem prover, [12], which has been used with some success in group theory. This will give two extensions to ***HR***'s abilities. Firstly, ***HR*** will be able to prove a subset of its conjectures - those which can be expressed in a language acceptable by OTTER. Secondly, ***HR*** may be able to prove that the only model for a particular concept is the trivial group, and so the concept is uninteresting. For example, in §2.2, the `forall` production rule invented the concept of groups, G, for which $\forall (a,b,c) \in G \times G \times G, a * b = c$, and it is easy to prove that there is only one such group, the trivial group.

The second major addition is to give ***HR*** another task to achieve, namely to find more models of the algebra it is working with. That is, we want ***HR*** to finish a session with more Cayley tables than it started with. As mentioned in §5.3, ***HR*** has its own constraint solver which allows it to find models of algebras given only the axioms. We first intend to link ***HR*** to Slaney's FINDER, [19], which is a much more efficient model generator. Then we intend to have ***HR*** output constraints to FINDER which make the search for models more efficient. Recently, we gave FINDER the axiomatisation of groups by the quasigroup and associativity axioms. We then gave it the extra information that $\exists! a$ for which $a * a = a$. When asked to find groups up to order six, FINDER performed better with the extra information, which could easily have been provided by ***HR***. We hope to take this further, and introduce measures such as the **efficiency** measure for concepts in [3], which will improve the yield of concepts which increase the number of models found by FINDER.

To conclude, we can list the mathematical tasks accomplished by ***HR***. Using all its functionality, given only the axioms of group theory (or any other algebra), ***HR*** can (i) find models of groups, (ii) categorise the groups up to isomorphism - it has an isomorphism checker, (iii) find group theory concepts such as calculations from the Cayley table, subgroups and series of groups (iv) find a calculation which classifies the groups it has up to isomorphism and (v) make conjectures about the concepts it has invented. Future versions of ***HR*** will (vi) prove some of its conjectures and (vii) invent constraints which help it to find more models of groups with which to work.

ACKNOWLEDGEMENTS

This project has been funded by EPSRC research grant GR/L 11724 and EPSRC studentship GR/K/65706. We would like to thank Jeremy Gow for his helpful comments on this paper.

REFERENCES

[1] R Bagai, Shanbhogue, J V, Zytkow, and S Chou, `Automatic theorem generation in plane geometry', in *Lecture Notes in Artificial Intelligence, 689*, pp. 415–424. Springer Verlag, (1993).

[2] S Colton, *HR - Automatic Concept Formation in Pure Mathematics*, http://dream.dai.ed.ac.uk/group/simonco/hr/hr.html.

[3] S Colton, S Cresswell, and A Bundy, `The use of classification in automated mathematical concept formation', in *Proceedings of SimCat 1997: An Interdisciplinary Workshop on Similarity and Categorisation*. Department of Artificial Intelligence, University of Edinburgh, (1997).

[4] R Davis and D Lenat, *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill Advanced Computer Science Series, 1982.

[5] E DeLaVina and S Fajtlowicz, `Ramseyian properties of graphs', *Electronic Journal of Combinatorics*, **3**, (1996).

[6] S Epstein, `On the discovery of mathematical theorems', in *Proceedings of the 10th International Joint Conference on Artificial Intelligence*. IOS Press, (1987).

[7] S Fajtlowicz, `On conjectures of graffiti', *Discrete Mathematics 72*, **23**, 113–118, (1988).

[8] D Gorenstein, *Finite Simple Groups: An Introduction to Their Classification*, Plenum Press, New York, 1982.

[9] K Haase, `Discovery systems', in *Proceedings of the 7th European Conference on Artificial Intelligence*, (1986).

[10] E Koutsofios and C North, *Dot User's Guide*, AT+T Bell Labs, Murray Hill, NJ.

[11] D Lenat, `Eurisko: A program which learns new heuristics and domain concepts', *Artificial Intelligence*, **21**, (1983).

[12] W. McCune, `The Otter user's guide', Technical Report ANL/90/9, Argonne National Laboratory, (1990).

[13] Marsha J. Ekstrom Meredith, *Seek-Whence: A Model of Pattern Perception*, Ph.D. dissertation, Department of Computer Science, Indiana University, 1987.

[14] G L Miller, `On the $n^{log_2 n}$ isomorphism technique', Technical Report TR17, The University of Rochester.

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

[16] G Ritchie and F Hanna, `AM: A case study in methodology', *Artificial Intelligence*, **23**, (1984).

[17] W Shen, `Functional transformations in AI discovery systems', Technical Report CMU-CS-87-117, Computer Science Department, Carnegie-Mellon University, (1987).

[18] M Sims, `IL: An artificial intelligence approach to theory formation in mathematics', Technical Report ML-TR-33, Department of Computer Science, Rutgers University, (1990).

[19] J Slaney, *FINDER 3.0 Manual*, Centre for Information Science Research, Australian National University.

[20] N Sloane, *The Online Encyclopedia of Integer Sequences*, http://www.research.att.com/ njas/sequences/eisonline.html.

[21] R Valdes-Perez, `Why some machines do science well', in *Proceedings of ICDC, The International Congress on Discovery and Creativity, Forthcoming*. Ghent University, (1998).