The Use of Classification in Automated Mathematical Concept Formation

Simon Colton, Stephen Cresswell and Alan Bundy
Department of Artificial Intelligence
University of Edinburgh
(simonco@dai.ed.ac.uk, stephenc@dai.ed.ac.uk, bundy@dai.ed.ac.uk)

Abstract

Concept formation programs aim to produce a high yield of concepts which are considered interesting. One intelligent way to do this is to base a new concept on one or more concepts which are already known to be interesting. This requires a concrete notion of the 'interestingness' of a particular concept. Restricting the concepts formed to mathematical definitions in finite group theory, we derive three measures of the interestingness of a concept. These measures are based on how much the concept improves a classification of finite groups.

Introduction

Concept formation, broadly speaking, involves analysing what is already known and coming up with a new idea based on the original knowledge. How and why particular concepts are formed can be investigated by writing computer programs designed to automatically create new concepts based on a corpus of knowledge. Such programs are implementations of some theory about concept formation, and the practicalities of writing computer programs require the theory to be concrete. Computers allow a great deal of experimentation and the results obtained when running the programs can be used to suggest improvements to the underlying theory. A good way to test the programs is to perform concept formation in a well known domain and to compare the output obtained with the concepts which are classically found in the domain.

Once concept formation programs have reached a certain standard, they can be used in a more exploratory fashion, assisting in the construction of theories by searching for new concepts in less well known domains. In these cases, the computer program will need a concrete and calculable notion of 'interestingness' to help it decide which concepts to keep and which to discard. This notion can also be used in a more pro-active way by employing heuristics designed to increase the average level of interestingness. One such heuristic is to base new concepts on old concepts which have been shown to be interesting, hoping that interesting concepts lead to further interesting concepts. It is therefore clear that a concrete notion of interestingness should be central to any automated concept formation program.

The most cited work in automated mathematical concept formation is Lenat's work on his AM[1] computer program which invented new definitions and made conjectures based on empirical evidence. This was an exploratory program designed to work in elementary set theory which actually delved into elementary number theory. A notion of interestingness was used to maintain an agenda of tasks to do next, calculated using values for how interesting old concepts were and values for how worthwhile a task involving those concepts might be. The calculations are complex and based on many heuristic notions of why a concept might be interesting, and there was no overall goal to the concept formation.

Other attempts to formalise the notion of interestingness concentrated on reducing the search space so that exploration only occurred within a narrow band of concepts. With each of the newly formed concepts having a similar nature, it is easier to make a comparison of two concepts, and interestingness can be measured more easily. For instance, in the BACON[2] programs written by Langley et al, the concepts formed were polynomial relations between measurements taken during physics experiments. A concept was interesting if the polynomial relation was observable in the data, and uninteresting if not. In this case, they were able to guarantee some level of interestingness by looking at the data first, spotting patterns and trends and forming the new relations in this data driven manner.

Another example of narrowing the search space is Sims' IL[3] program, which accepted specifications for an operator (for instance the multiplication operator on Conway numbers), and formed operators designed to fit the specifications. The number of specifications satisfied by the invented operator gives an indication of its importance, and the process can stop once one has been invented which satisfies all the specifications.

Therefore, finding an effective measure of the interestingness of concepts is a two step process; (i) imposing a similar format on all the concepts produced, (ii) defining why one concept in this format is better than another, preferably in terms of numeric values, or any partial order which can be used to grade the concepts.

To begin to impose a similar format between the concepts, we first restricted the mathematical domain in which the concepts were formed to finite group theory. This choice was due to our mathematical background and the fact that many other algebras, such as ring theory, Galois theory and infinite group theory depend on the concepts from finite group theory. We further restricted the type of concept formed to definitions, loosely speaking, the sentences which appear under the 'definition' heading of finite group theory texts. As detailed later, further uniformity is achieved by thinking of each definition as a function mapping a group to some output.

[1] See [Davis & Lenat 82].

[2] See [Langley *et al* 87].

[3] See [Sims 90].

A major goal in the development of finite group theory is the classification of finite groups. For instance, there is a classification of finite Abelian groups which can be found in most elementary group theory texts. More remarkably, in 1980, a complete classification of finite simple groups was achieved, which must surely be regarded as one of the major intellectual achievements of all time.[4] Finite simple groups are finite groups with no non-trivial normal subgroups and are the atoms from which the other finite groups are constructed, which is similar to the way prime numbers are the building blocks for integers. Hence the classification of finite simple groups goes a long way to classifying finite groups.

Taking the classification of finite groups as the reason to form concepts, we can judge how important a particular concept is by how much it improves a classification. A classification of an object describes the object, and we concentrate on three intrinsic properties of a description:

- Parsimony: How succinctly the description can be stated.
- Acuity: How well the description differentiates between two similar objects.
- Efficiency: How efficiently the object can be explicitly constructed, given only the description.

The work here develops calculations which can be performed to measure these aspects of a classification, and how much of an improvement a particular concept makes. This allows us to measure three values for each concept, its *parsimony*, its *acuity*, and its *efficiency*, which can be used to determine the importance of any concept in the theory. Note that, from this point, we abbreviate 'finite group' to just 'group'.

Classifications

The first step to introducing a formal notion of interestingness is to add some uniformity to the concepts so that the similar structure makes it easier to compare two concepts. As previously stated, the concepts being formed here are definitions in group theory. The three most common ways of introducing new definitions in group theory books are as follows:

(A) By specialising the concept of group using a test on the group. For instance:
"A group, G, is Abelian iff $\forall\ a, b \in G, a * b = b * a$."

(B) By specifying a calculation which can be performed on the group. For instance:
"The centre of a group, G, is given by
$Z(G) = \{a \in G : \forall\ b \in G, a * b = b * a\}$."

(C) By detailing a construction which is possible from a group. For instance:
"Given a group, G, then H is a subgroup of G iff $H \subseteq G$ and H forms a group itself under the group operation from G."

It is possible to think of formats (A) and (C) in terms of format (B). Firstly, the tests on groups can be written as boolean functions, which take a group as input, and output 'true' or 'false'. For example, we could write the Abelian property of groups given above as:

$$\text{IsAbelian}(G) = \begin{cases} \text{true} & \text{if } \forall\ a, b \in G, a * b = b * a \\ \text{false} & \text{otherwise} \end{cases}$$

Secondly, to use constructions as a function on a group we can make a function which outputs all the possible examples of the construction for a given input group. Eg.

$$\text{SubgroupCollection}(G) = \{H : H \text{ is a subgroup of } G\}.$$

Then, as soon as a construction definition is given, we can make this second, functional, definition which can be used instead of the construction definition.

Note that in practice there are many ways to turn the construction definition into a function accepting a group as input. For instance, we could make a function which tests whether a given group has any examples of a particular construction. Eg.

$$\text{HasSubgroup}(G) = \begin{cases} \text{true} & \text{if } \exists\ H \text{ such that} \\ & H \text{ is a subgroup of } G \\ \text{false} & \text{otherwise} \end{cases}$$

Hence all the sentences given in the definitions can be written as functions taking a single group as input, and outputting something based on the group. To add further uniformity to the format of the concepts, and to make later calculations possible, we impose the restriction that the output from the functions is a nested vector of atoms. This is true anyway of the functions made from definitions of type A and C above, and we only have to worry about those of type B, which are the functions occurring naturally in group theory. As group theory is built heavily on set theory, most of the outputs are elements, sets or groups, all of which can be written as nested vectors, and in practice this imposition is not too restrictive.

We now have a starting point for a formalisation, and can continue by noting that the output of a function can be used to describe any input group. For instance, given a particular group, G, instead of saying "G is Abelian", we can say: "IsAbelian$(G) =$ true". Further, a set of such functions can be used in conjunction for a more complete description of groups. We can then use these descriptions to build a classification of a set of groups. We formalise this as follows:

Definition 1

- A **classifying function** in group theory is a unary function which takes a single group as input, and outputs a nested vector. [Note that in the examples, the outputs will be single atoms, either truth values or integers, which can be thought of as vectors with a single entry].

- A **classifying theory** of groups, C, is a vector of (the names of) classifying functions, $C = \langle f_1, \ldots, f_k \rangle$.

- Given a group, G, the **description of G by C,** written $desc_C(G)$, is the vector of output values given when G is used as input to all the functions in C. ie.

$$desc_C(G) = \langle f_1(G), \ldots, f_k(G) \rangle.$$

- Given a set of groups, S, **the classification of S by C,** written $class_C(S)$, is the set of descriptions of the members of S. ie.

$$class_C(S) = \{desc_C(G) : G \in S\}.$$

[4] To quote John Humphreys in [Humphreys 96]. For details of the classification of finite simple groups, see [Gorenstein 82].

Example 1

The following are classifying functions in group theory:

$Order(G) = |G|$ *[The size of the underlying set].*

$$IsAbelian(G) = \begin{cases} \textit{true} & \textit{if } \forall\ a, b \in G, a * b = b * a \\ \textit{false} & \textit{otherwise} \end{cases}$$

$NSIE(G) = |\{a \in G : a * a = 1\}|$

($NSIE$ stands for the Number of Self Inverse Elements).

---

and using $S = \{G_1, G_2, G_3, G_4\}$, with group tables:

G_1	1	2	3
1	1	2	3
2	2	3	1
3	3	1	2

G_2	1	2	3	4
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2
4	4	1	2	3

G_3	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	3	4	5	6	1
3	3	4	5	6	1	2
4	4	5	6	1	2	3
5	5	6	1	2	3	4
6	6	1	2	3	4	5

G_4	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	1	5	6	3	4
3	3	6	1	5	4	2
4	4	5	6	1	2	3
5	5	4	2	3	6	1
6	6	3	4	2	1	5

it is easy to calculate:

$Order(G_1) = 3$	$IsAbelian(G_1) =$ true	$NSIE(G_1) = 1$
$Order(G_2) = 4$	$IsAbelian(G_2) =$ true	$NSIE(G_2) = 2$
$Order(G_3) = 6$	$IsAbelian(G_3) =$ true	$NSIE(G_3) = 2$
$Order(G_4) = 6$	$IsAbelian(G_4) =$ false	$NSIE(G_4) = 4$

So, if we let $C = \langle Order, IsAbelian, NSIE\rangle$, we get:

$desc_C(G_1) = \langle 3, \textit{true}, 1\rangle,\ desc_C(G_2) = \langle 4, \textit{true}, 2\rangle,$
$desc_C(G_3) = \langle 6, \textit{true}, 2\rangle,\ desc_C(G_4) = \langle 6, \textit{false}, 4\rangle,$

and this gives us:

$$class_C(S) = \{\langle 3, \textit{true}, 1\rangle, \langle 4, \textit{true}, 2\rangle, \langle 6, \textit{true}, 2\rangle, \langle 6, \textit{false}, 4\rangle\}.$$

Measures of Importance

Having defined a classification of a set of groups, we can now find some measurable properties of it, and compare the classification given by a single function against that given by a set of functions, to gauge the importance of the single function. It is therefore necessary to identify what is desirable in a classification. To help with this, there are two special classifications which represent the worst and best cases. Firstly, there is the **trivial classification,** describing each group with the same sentence, "G is a group". ie. The trivial classification is given by the single function: $IsGroup(G) = true$ iff G is a group. Secondly, there is the **explicit classification** which describes each group by giving its group table.

Note that each of the three measures introduced are calculated for a classifying theory. To assess each individual classifying function, the classification given using only the single function is measured. Note also that each of the measures is normalised to give a value between 0 and 1, with 0 representing the worst case and 1 representing the best case.

Parsimony of a Classification

If you wanted to tell a colleague something about a particular group, you would first have to establish that they knew exactly which group you were talking about. This could be achieved either by writing down the group table, or describing the group in sufficient detail to ensure there is no ambiguity. The second method has the advantage of being more parsimonious than the first, as the group table is a very cumbersome way to represent the group. For instance, if the group under discussion was of order 20, a group table would have 400 entries, and it is clearly more sensible to write this down once in a reference book, but refer to the group by descriptions in day to day use. To illustrate this point, group theorists would describe the groups G_1, G_2, G_3 and G_4 in example 1 as C_3, C_4, C_6 and $S(3)$ respectively, with no ambiguity. We see that they waste as little ink as possible.

One of the reasons to pursue a classification is to enable us to describe a complex object in a compact manner, with the description being used to represent the object, rather than a more cumbersome, explicit representation. We can then try to measure how parsimonious a particular classification is, and use this to assess the worth of the classifying theory, and the individual functions in that theory. Remembering firstly that the output of the functions will be used to describe the groups, and secondly that we restricted the outputs to be nested vectors only, if we flatten these vectors into a single list of atoms, the size of the list will give an indication of the parsimony. We can formalise this with the following definitions:

Definition 2

• Given a classifying function, f, and a group, G, then $f(G)$ will be a nested vector. If we flatten this vector completely into a list of atoms, and call the flattened vector L, then the **storage space required to describe** G **with** f**,** written $stor_f(G)$, is the size of L. ie.

$$stor_f(G) = |L|.$$

• Given a classifying theory, C, the **storage space required to describe** G **using** C**,** written $stor_C(G)$, is a measure of the space required to describe G using all the functions available in C, and is given by:

$$stor_C(G) = \sum_{g \in C} stor_g(G).$$

• Given a set of m groups, S, the **parsimony of** C**, approximated using** S**,** written $S\_parsimony(C)$, measures the compactness of the outputs of the functions when C is used to describe the members of S. It is normalised to give a value between 0 and 1 and is given by:

$$S\_parsimony(C) = \frac{1}{m} \sum_{G \in S} \frac{1}{stor_C(G)}.$$

• Given a classifying function, $f \in C$, the **parsimony of** f **approximated using** S**,** written $S\_function\_parsimony(f)$, is a measure of the parsimony of the classification given by the function alone. It is given by:

$$S\_function\_parsimony(f) = S\_parsimony(\langle f\rangle).$$

Example 2

The $Order$, $IsAbelian$ and $NSIE$ functions all output single atoms, either the word true, or the word false, or an integer. Hence the nested vector outputs in these cases contain only one element and so we find that, with S and C as before:

$stor_C(G_i) = \sum_{f \in C} stor_f(G_i) = 1 + 1 + 1 = 3$

for $i = 1, 2, 3, 4$.

So, $S\_parsimony(C) = \frac{1}{4}\left(\frac{1}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3}\right) = \frac{1}{3}$

and it is easy to check that:

$S\_function\_parsimony(Order) = 1,$
$S\_function\_parsimony(IsAbelian) = 1,$
$S\_function\_parsimony(NSIE) = 1.$

Therefore, in terms of parsimony, these functions are perfect. They were chosen for this reason, to reduce the space needed for the examples. To see a less parsimonious example, look at the $Centre$ function given by:

$$Centre(G) = \{x \in G : \forall\ y \in G, x * y = y * x\}.$$

Here, the output is a set, which can be written as a vector. It is not difficult to calculate:

$Centre(G_1) = \langle 1, 2, 3\rangle,\ \ Centre(G_2) = \langle 1, 2, 3, 4\rangle.$
$Centre(G_3) = \langle 1, 2, 3, 4, 5, 6\rangle,\ \ Centre(G_4) = \langle 1\rangle.$

Hence $stor_{Centre}(G_1) = 3,\ \ stor_{Centre}(G_2) = 4,$
$stor_{Centre}(G_3) = 6,\ \ stor_{Centre}(G_4) = 1.$

From these, using S as before, we can calculate:

$$\begin{aligned} S\_function\_parsimony(Center) &= \tfrac{1}{4}\left(\tfrac{1}{3} + \tfrac{1}{4} + \tfrac{1}{6} + \tfrac{1}{1}\right) \\ &= \tfrac{7}{16}. \end{aligned}$$

Acuity of a Classification

When representing a group with a description rather than its group table there are drawbacks, because, to increase the parsimony, the amount of information on display must be reduced. The first drawback is a loss in the power to discriminate between two groups. If we look at the trivial classification, we see that it describes every group in the same way, which is clearly a defect. It is desirable to have a classification which describes each group differently. We can approximate how well the descriptions differentiate between any two groups by looking at a set of groups, and judging how many have different descriptions. This leads to the following definitions:

Definition 3

• Given a set of m groups, S, and a classifying theory, C, the **acuity of C, approximated using S,** written $S\_acuity(C)$, is a measure of the number of groups which are given distinct descriptions by C. It is normalised to give a value between 0 and 1 and is given by:

$$S\_acuity(C) = \begin{cases} 1 & \text{if } m = 1 \\ \frac{|class_C(S)| - 1}{m - 1} & \text{otherwise} \end{cases}$$

[Note that if two groups have the same description using C, then as $class_C(S)$ is a set, the description only appears once in the set, making $|class_C(S)| < m$, and $S\_acuity(C) < 1$.]

• Given a classifying function, $f \in C$, the **acuity of f approximated using S,** written $S\_function\_acuity(f)$, is a measure of the acuity of the classification given by the function alone. It is given by:

$$S\_function\_acuity(f) = S\_acuity(\langle f\rangle).$$

Example 3

Using S and C from the previous examples, we see that each G_i has a different description using C, and so they are all uniquely identified by C. As this is the ideal case, we should find that the acuity of C on S is 1. We can check this:

There are 4 groups in S, so $m = 4$, and remembering that

$$\begin{aligned} class_C(S) = \{&\langle 3, \textit{true}, 1\rangle, \langle 4, \textit{true}, 2\rangle, \\ &\langle 6, \textit{true}, 2\rangle, \langle 6, \textit{false}, 4\rangle\}, \end{aligned}$$

we see that

$S\_acuity(C) = \frac{|class_C(S)| - 1}{m - 1} = \frac{4-1}{4-1} = 1.$

So C forms a classification of S which is perfect in terms of acuity. We can also calculate the acuities of the three classifying functions in C. Noting that

$$class_{\langle f\rangle}(S) = \{f(G) : G \in S\},$$

we see that

$class_{\langle Order\rangle}(S) = \{3, 4, 6\},$
$class_{\langle IsAbelian\rangle}(S) = \{\textit{true, false}\},$
$class_{\langle NSIE\rangle}(S) = \{1, 2, 4\}.$

Hence

$$\begin{aligned} S\_function\_acuity(Order) &= \frac{|class_{<Order>}(S)| - 1}{4 - 1} \\ &= \frac{3-1}{3} = \frac{2}{3}, \end{aligned}$$

and similar calculations show that

$S\_function\_acuity(IsAbelian) = \frac{1}{3}$ *and*
$S\_function\_acuity(NSIE) = \frac{2}{3}.$

These figures indicate that the $IsAbelian$ function is below par in terms of acuity. This is because it has only two possible outputs, true or false, and so splits groups into only two categories. This will be a problem for all boolean functions.

Before leaving the subject of acuity, it is worth noting that the acuity measure does not address the problem of a function describing two equivalent groups differently. In group theory, two groups are regarded as the same if they are isomorphic, that is, there is a 1:1 map between the elements of the two groups which preserves the group operation. Isomorphic groups can have different group tables, and so it is possible for a classifying function to describe isomorphic groups differently. Given a set of isomorphic group tables, this deficiency can also be measured, but such calculations have been omitted for the sake of brevity.

Calculations of acuity and those determining the problem caused by isomorphisms are both addressing the same problem: whether we can tell if two objects are truly different given only descriptions of them. This would appear to be a goal for classifications in any domain.

Efficiency of a Classification

The third value we can calculate to assess the quality of a classification measures the second drawback to having more parsimonious descriptions, and takes the most explaining. The group table tells us $a*b$ for every pair of elements in the group, and this information is enough for us to perform any calculation involving the group. In general, any description of a group other than the group table will not give us enough information to perform an arbitrary calculation, which is clearly a drawback. Therefore, if group theorists are going to work with the descriptions, they have two alternatives. They can either conjecture and prove theorems about the results of certain calculations when performed upon groups with given descriptions (for instance, that all cyclic groups are Abelian), or they can derive methods by which the group table can be recovered from the description. Of course, once the group table has been recovered, it can be used to perform any calculation.

We concentrate on the second method for working with group descriptions, ie. we try to assess how difficult it is to recover a group table given only a description of it. Of course, if we are given an explicit description of the group, that is, one from which we can read $a*b$ for each $a, b \in G$, then it takes no effort to write down the group table. Other descriptions will require more effort to reconstruct the group table, involving a search of some kind wherein various possibilities for the group table (or parts of it) are tested to see if they fit the description, with backtracking occurring if not.

Note that if we know the order of the group, a search for the group table of the group will be finite, and, while losing generality slightly, things are made much easier if we assume that the order of the group is a constraint which is always given in the description. Given the assumption that the order of the group is known to be n, one implausible way that a mathematician might construct a group table for a group is to write down all the possible multiplication tables for a set of n elements, and check each one to see if it fits the description given. There are n^{n^2} possible multiplication tables, and noting that $4^{4^2} \simeq 4.3 \times 10^9$, we see that this method is impractical for groups of any moderate size.

A much more plausible method is to fill in the group table one entry at a time, checking that the incomplete multiplication table satisfies the description. This means that the description of the group has to be interpreted as ways of ruling out incomplete multiplication tables. This interpretation is not always straightforward but is usually possible. The advantage to this method is that once an incomplete multiplication table has been rejected by the description, any larger table built by adding more elements to this table will also be rejected, and so there is no point trying those possibilities. This has the potential to drastically cut down the search space.

Stated in this manner, we see that the problem of finding the group table given a non-explicit description of the group is a Constraint Satisfaction Problem (CSP).[5] A CSP is defined by three things, (i) variables which we want values for, (ii) domains from which these values can be assigned (one domain for each variable), and (iii) constraints which rule out certain combinations of assignments. Each description of a group of order n given by our classifying functions will produce a separate CSP with n^2 variables (one for each entry of the group table). These variables will be assigned values from the group elements, that is the integers $1, \ldots, n$, and the constraints will be provided by the descriptions of the group.

Having formulated the retrieval of the group table as a CSP, we have been able to use the large body of knowledge in this area to define a concrete calculation which estimates how difficult the search will be when constructing the group table given only a description of the group. This estimate is an adaption of the cost to find all solutions measure introduced in [Williams & Hogg 94], and draws heavily on that paper, specifically sections 3.3, 3.3.2 and 6.1.4. We urge consultation of this work to put the following definitions in context.

Definition 4

• An **incomplete multiplication table of order** n [Shorthand: $imt(n)$] is a multiplication table for a closed binary operation on the first n integers, with 1 or 2 or ... or n^2 entries filled in.

• Given a group of order n, G, and a classifying theory, C, then an $imt(n)$, T, is a **no-good for** G **in** C if it contains enough entries to decide that it violates the description of G given by C. ie. Any group, H, for which T is a subtable of the group table for H will be such that $desc_C(G) \neq desc_C(H)$.

• T is a **minimised no-good for** G **in** C if there is no proper subtable of T which is also a no-good for G in C.

• Letting m_i be the number of minimised no-goods for G in C with i entries filled in, the **expected cost to construct** G **given** C**,** written $\langle Cost \rangle_C(G)$, is given by:

$$\langle Cost \rangle_C(G) = \sum_{j=1}^{n^2} \bar{p_j} n^j,$$

where

$$\bar{p_j} = \prod_{i=1}^{j} \left(\begin{pmatrix} n^2 \\ i \end{pmatrix} n^i - \begin{pmatrix} j \\ i \end{pmatrix} \atop m_i \right) \left(\begin{pmatrix} n^2 \\ i \end{pmatrix} n^i \atop m_i \right)^{-1}$$

[with $\binom{a}{b} = {}^aC_b = \frac{a!}{b!(a-b)!}$ if $a \geq b$, but $\binom{a}{b} = 0$ if $a < b$].

• We can scale the expected cost to give us a value between 0 and 1 by dividing by the cost of doing the search without any constraints. We get the **scaled expected cost to construct** G **given** C**,** written $SC_C(G)$, which is given by:

$$SC_C(G) = 1 - \frac{\langle Cost \rangle_C(G)}{\sum_{j=1}^{n^2} n^j}.$$

• Given a set of m groups, S, the **efficiency of** C **approximated using** S**,** written $S\_efficiency(C)$, is given by:

$$S\_efficiency(C) = \frac{1}{m} \sum_{G \in S} SC_C(G).$$

• Given a classifying function, $f \in C$, the **efficiency of** f **approximated using** S**,** written $S\_function\_efficiency(f)$, is a measure of the efficiency of the classification given by the function alone. It is given by:

$$S\_function\_efficiency(f) = S\_efficiency(\langle f \rangle).$$

[5] For a comprehensive account of CSPs, see [Tsang 93].

Example 4

To find a true value for the efficiency of a classifying theory, it is necessary to find the minimised no-goods for each group given by each classifying function, and also those given by the group axioms. It is then necessary to determine the extent of the overlap between the sets of no-goods, and account for this in the calculation. This is too much work for this introductory example, so we shall concentrate on the easier problem of finding the efficiency of the IsAbelian and NSIE functions.

If we first look at the efficiency of the $IsAbelian$ *function on* G_1*, then we see that* $IsAbelian(G_1) = true$*,*

ie. $\forall\ a, b \in G_1,\ \ a * b = b * a$*.*

Hence, a no-good will occur if the table has two entries filled in, one for $a * b$ *and one for* $b * a$*, and the values in the entries are different.*

For instance, this imt(3) must be no-good for G_1*, because we can see that* $1 * 2 = 3$ *but* $2 * 1 = 2$*, so it cannot be part of any Abelian group table.*

T	1	2	3
1			3
2	2		
3			

It is clear that $imts$ *of this type are minimised, as the only smaller alternative is to have 1 entry filled in, and these will not break the Abelian rule. Any larger no-good must break the rule with at least one pair of entries, and that pair will form a subtable which is no-good, and so the larger no-good will not be minimised. Therefore, these are the only types of minimised no-goods which violate the 'IsAbelian(*G_1*) = true' description.*

In constructing a no-good of this type, we can first choose any position in the group table above the diagonal, and there are $(n^2 - n)/2$ *ways to do this. [For an arbitrary group of order* n*]. We can then put any value into that entry, so there are* n *ways to do that. The entry above the diagonal fixes the position of the entry below the diagonal, but we can choose any value for that entry, as long as it isn't the same as the value above the diagonal. There are* $n - 1$ *ways to do this. Therefore for a group,* G*, of order* n*, there are*

$$n(n^2 - n)(n - 1)/2 = n^2(n - 1)^2/2$$

possible minimised no-goods with 2 entries filled in which violate the 'IsAbelian(G) = true' description. As these are the only minimised no-goods, if we let m_i *be the number of minimised no-goods with* i *entries filled in, for a group,* G*, of order* n *described as 'IsAbelian(G) = true,' then*

$$m_i = \begin{cases} n^2(n-1)^2/2 & \text{if } i = 2, \\ 0 & \text{otherwise} \end{cases}$$

Using the Maple Computer Algebra System,[6] *the above values for* m_i *were used to calculate these scaled expected costs:*

$$SC_{\langle IsAbelian\rangle}(G_1) = 0.8439073\ \ (7d.p)$$
$$SC_{\langle IsAbelian\rangle}(G_2) = 0.9881643\ \ (7d.p)$$
$$SC_{\langle IsAbelian\rangle}(G_3) = 0.9999965\ \ (7d.p)$$

Now, G_4 *is described as 'IsAbelian(*G_4*) = false,' so we cannot use the above values of* m_i*. To have a minimised no-good in this case, we must be able to tell that it is not non-Abelian. To do this, we must know it is Abelian, and so the minimum we require is that all the elements above the diagonal are filled in and the corresponding entries below the diagonal are filled in with the same values. There are* $n^{(n^2-n)/2}$ *ways to do this, and so we see that in this case,*

$$m_i = \begin{cases} n^{(n^2-n)/2} & \text{if } i = n^2 - n, \\ 0 & \text{otherwise} \end{cases}$$

Using the approximation $x! \approx e^{ln(x)-x}$ *for large* x*, these values for* m_i *were used to calculate:*

$$SC_{\langle IsAbelian\rangle}(G_4) = 0.0000000\ \ (7d.p)$$

The reason for this very poor score is that during a search for a non-Abelian group which fills in elements of the group table one at a time, if we reject an incomplete table on the grounds that it is not non-Abelian, we must have filled in all the non-diagonal entries. Hence most of the search will have been carried out before the constraining power of the description can be used, so the 'IsAbelian(G) = false' description is inefficient as a constraint for finding group tables.

With this last scaled cost value, we can finally calculate:

$$S\_function\_efficiency(IsAbelian) =$$
$$\frac{0.8439073+0.9881643+0.9999965+0.0000000}{4} = 0.7080170\ \ (7d.p)$$

An analysis of the minimised no-goods for the NSIE classifying function revealed that, for a group, G*, of order* n*, if* $NSIE(G) = q$*, the number of minimised no-goods are:*

$$m_i = \begin{cases} {}^nC_{n-q} & \text{if } i = q + 1 \\ {}^nC_{n-q+1}(n-1)^{n-q+1} & \text{if } i = n - q + 1 \\ 0 & \text{otherwise} \end{cases}$$

[Note that if $i = q + 1 = n - q + 1$ *then* $m_i = {}^nC_{n-q} + {}^nC_{n-q+1}(n-1)^{n-q+1}$*].*

Using the above approximation for $x!$*, we can calculate:*

$$SC_{\langle NSIE\rangle}(G_1) = 0.4316053\ \ (7d.p)$$
$$SC_{\langle NSIE\rangle}(G_2) = 0.8053208\ \ (7d.p)$$
$$SC_{\langle NSIE\rangle}(G_3) = 0.9097097\ \ (7d.p)$$
$$SC_{\langle NSIE\rangle}(G_4) = 0.9999870\ \ (7d.p)$$

and so $S\_function\_efficiency(NSIE) =$

$$\frac{0.4316053+0.8053208+0.9097097+0.9999870}{4} = 0.7866557\ \ (7d.p)$$

Examining these results, we see that both the choice of classifying function used to describe a group and the output from that function determine how efficient a search for the group table will be. For example, if a group is described as 'IsAbelian(G) = true', this is a fairly efficient description, but if described as 'IsAbelian(G) = false', the search for its group table is hardly improved at all by this information. Similarly, the more elements which are self inversing, ie. the larger the value of q *when* $NSIE(G) = q$*, the more efficient the search. Note that because the* $Order$ *function adds no more constraints to the search, it scores zero with this efficiency measure, which does not reflect the fact that a constrained search is only possible knowing the order of the group.*

[6] See [Wat] for details about Maple.

Conclusions and Further Work

With the aim of automating the formation of definitions in finite group theory, we set out to derive concrete calculations to measure the interestingness, or importance, of a definition, as this is a central requirement of any concept formation program. Noting that a major driving force in group theory is the classification of groups and that classifications provide descriptions of groups, we wrote the group theory definitions as descriptions of groups. We then identified the parsimony, acuity and efficiency properties of descriptions, which can be evaluated to measure how good the descriptions are.

Intuitively, these properties are desirable of descriptions in any domain: Given an object, and asked to describe it, you would hope that your description was (a) concise and to the point, (b) enough to tell the difference between the object and a similar one and (c) some help towards constructing the object. For example, if you were asked to describe a person, you might perhaps say they were tall, or equally you might say that they lived at number 17. While both of these are concise, the first would be of use if you had to pick the person out of an identity parade, but the second would be of use if you had to track the person down.

In an arbitrary domain it is unlikely that concrete measures could be specified to approximate the worth of descriptions with respect to these three properties. However, due to the formal nature of mathematics and by imposing certain restrictions, we have been able to derive calculations and produce some values for example definitions. Although the example calculations are fairly poor approximations, due to the small sample of groups used, it is possible to compare the classifications given by the $Order$, $IsAbelian$, $NSIE$ and $Centre$ functions against an explicit classification and the trivial classification. Instead of the explicit classification which gives the group table for each group, we will use the function $Explicit(G) = \{< a, b, a * b >: a, b \in G\}$, as this has output which can be written as a nested vector, a restriction needed to calculate its parsimony. The values for parsimony, acuity and efficiency are approximately:

Function	Parsimony	Acuity	Efficiency
Trivial	1	0	0
Order	1	0.67	0
NSIE	1	0.67	0.79
IsAbelian	1	0.33	0.71
Centre	0.44	1^*	0.95^*
Explicit	0.02^*	1	1

$*$ - Calculation of this value not included in the examples.

Remembering that these values have 0 as the worst case and 1 as the best case, we see that the more parsimonious descriptions have a lower acuity and efficiency than the less parsimonious descriptions. Hence the figures calculated highlight the drawbacks of working with descriptions of objects rather than the objects themselves.

Now, as we can measure how good a particular function is, it is possible to write heuristic methods which produce new functions, with the heuristics designed to increase the average parsimony, acuity or efficiency of the functions. For instance, if the acuity of the classification given by all the functions in the classifying theory is lower than we require, the program could use a production rule to base a new concept on an old one which has a high acuity itself. The heuristic employed here is that interesting concepts lead to further interesting concepts. Such a heuristic could be similarly employed if the parsimony or efficiency of the classification was below par, and more analytical heuristics, based on the failings of the classification for particular groups, are planned.

The writing and testing of the heuristic production rules as a computer program[7] is an ongoing project. Further work will include relating these definitions to the more general work of information theory and constraint satisfaction problems. The link with information theory is possible because by representing a group with a description, we are writing it in a code. Then the average codeword size and our parsimony measure are analogous, as are the entropy of the code and our acuity measure and the decoding time and our efficiency measure. The method chosen to decode the descriptions is to solve a constraint satisfaction problem using the group axioms and the description of the group as constraints on a search over the space of incomplete multiplication tables.

Acknowledgements

We would like to thank Ian Gent and Toby Walsh for their comments on constraint satisfaction problems. This project has been funded by EPSRC research grant GR/L 11724.

References

[Colton 97] S Colton. Classification driven theory formation in mathematics. Available by FTP from ftp://ftp.dai.ed.ac.uk /pub/user/simonco/proposal.ps , 1997.

[Davis & Lenat 82] R Davis and D Lenat. *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill Advanced Computer Science Series, 1982.

[Gorenstein 82] D Gorenstein. *Finite Simple Groups: An Introduction to Their Classification*. Plenum Press, New York, 1982.

[Humphreys 96] J Humphreys. *A Course in Group Theory*. OUP, 1996.

[Langley *et al* 87] P Langley, H A Simon, G L Bradshaw, and J M Zytkow. *Scientific Discovery - Computational Explorations of the Creative Processes*. MIT Press, 1987.

[Sims 90] M Sims. IL: An artificial intelligence approach to theory formation in mathematics. Technical Report ML-TR-33, Department of Computer Science, Rutgers University, 1990.

[Tsang 93] E Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego, 1993.

[Wat] Waterloo Maple. *Maple Manual at http://www.maplesoft.on.ca*.

[Williams & Hogg 94] C Williams and T Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70, 1994.

[7] Named HR after Hardy and Ramanujan - see [Colton 97].