

Theory Formation Applied to Discovery, Learning and Problem Solving

Simon Colton

Division of Informatics, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, United Kingdom

`simonco@dai.ed.ac.uk`

Abstract

We discuss the HR program [1] which is designed to perform automated theory formation in domains of pure mathematics. We overview the recent application of theory formation to discovery, learning and problem solving tasks. We compare how theory formation is used for these tasks and discuss how to compare HR to the machine learning program Progol [9].

1 The HR Program

The HR program [1], named after mathematicians Hardy and Ramanujan, is designed to form theories in domains of mathematics such as group theory, graph theory and number theory. HR starts with background information such as the axioms of a finite algebra, or some concepts in number theory such as the divisors of integers, multiplication and addition. Each concept is supplied with a definition in both a Prolog format and a format acceptable as input to the Otter theorem prover [6], and a finite set of examples. HR uses one of seven general production rules to base a new concept on either one old concept (in which case we say the production rule is *unary*) or two old concepts (a binary production rule). This produces a set of concepts which form the core of the theory. Each production rule generates a definition in both formats and a set of examples for the new concept.

While producing concepts, HR makes conjectures supported by empirical evidence. If it notices that the examples of a new concept are exactly the same as an old concept (for the data available), it will conjecture that the definitions of the two concepts are logically equivalent — producing an ‘if and only if’ conjecture. In finite algebras such as finite group theory, HR invokes Otter to prove the conjectures it makes. Whenever Otter is unsuccessful, HR invokes the MACE model generator [7] to find a counterexample to disprove the conjecture. In this way, HR forms a theory which contains concepts, examples, open conjectures, theorems and proofs.

To improve the quality of the theories, HR uses

heuristic measures to estimate the worth of concepts. It then performs a best first search by using the more interesting concepts as the basis for new concepts before the less interesting ones. The user sets weights for a weighted sum of all the measures and this sum is taken as an estimate of the worth of each concept. The measures include intrinsic properties of the concept such as the number of examples it has, as well as relational measures such as the novelty of the categorisation it produces, as discussed in [1].

The quantity and quality of conjectures that a concept appears in is also assessed, with concepts appearing in interesting conjectures assessed as more interesting than those appearing in dull conjectures. The worth of a theorem is assessed by the length of the proof produced by Otter, with longer proofs indicating a more interesting conjecture statement. Conjectures are also assessed by their surprisingness: the syntactic difference between two definitions conjectured to be equivalent is taken as a measure of the surprisingness, with a greater difference indicating a more surprising conjecture. HR completes a cycle of mathematical activity where concept formation drives conjecture making which in turns improves concept formation.

HR improves on previous theory formation programs such as AM [4] and GT [5] by incorporating theorem proving (AM could not prove theorems) and by being able to work in different domains (GT could only work in graph theory). We have recently applied HR to discovery, learning and problem solving tasks. We provide enough detail of these tasks below to enable a comparison of the techniques involved.

2 Discovery Tasks

In less than an hour, HR can produce more than 2000 concepts in number theory and if the user sets different starting parameters, the yield of concepts differs each time HR is run. There is therefore the possibility of HR producing new and genuinely interesting concepts. We explored this possibility in [3] and we summarise this work below.

It is difficult to tell in general whether a concept is either new or interesting. In number theory, however, there is an Encyclopedia of Integer Sequences, [10] which contains over 55,000 sequences collected over 35 years by Neil Sloane, with contributions from many mathematicians. If a concept HR produces in number theory can be interpreted as an integer sequence which is missing from the Encyclopedia, this gives some indication — but by no means a guarantee — that the concept may be novel.

We also used the Encyclopedia to give some indication as to whether the new integer sequences HR produces are interesting. To do this for a chosen sequence S , we enabled HR to find sequences in the Encyclopedia which are empirically related to S . This relation is then stated as a conjecture about S . As a trivial example, given the sequence of prime numbers, HR makes the conjecture that they are never square numbers. It does this by noticing that none of the prime numbers in its sequence were in the Encyclopedia entry for square numbers. As well as finding disjoint sequences, HR is able to find subsequences and supersequences of the chosen sequences.

Due to the large number of sequences in the Encyclopedia, many sequences related to the chosen one are output and we had to implement pruning techniques to discard dull conjectures. For example, it is desirable that a sequence conjectured to be disjoint with the chosen sequence has its terms distributed over roughly the same part of the number line as the chosen sequence. If so, the two sequences occupy roughly the same part of the number line yet do not share any terms — which increases the possibility of the conjecture being true and/or interesting. Therefore, HR discards conjectures about disjoint sequences if the overlap of their ranges falls below a minimum specified by the user.

By finding conjectures relating the sequence HR has invented to the sequences already in the Encyclopedia, HR provides some evidence that the sequence is of interest. This invent and investigate approach has successfully led to 17 sequences invented by HR being added to the Encyclopedia, all supplied with interesting conjectures. A good example of this is the sequence of integers where the number of divisors is prime, which HR invented (in as much as it was produced by HR and not present in the Encyclopedia).

When asked to find subsequences of this sequence, the first answer produced was the sequence of integers where the *sum* of divisors is prime (submitted to the Encyclopedia by someone else). Interpreted as a conjecture, this result states that, given an integer, n , if the sum of divisors of n is prime, then the number of divisors of n will also be prime. We have subsequently proved this result, and while we do not know for certain whether it is a new result, it certainly adds interest to the sequence HR invented.

3 Learning Tasks

We have also explored the possibility of using HR as a machine learning program, where the user supplies examples and non-examples of a concept and HR finds a definition for the concept. So far, we have used HR to learn definitions for integer sequences, as discussed in [2]. We have also applied HR to Michalski-style train problems [8] where the program is asked to find a reason why a certain set of trains are going East, based on certain characteristics of the train, for example the shape of the carriages.

The naive way to use theory formation for learning tasks is to supply HR with background knowledge and ask it to form a theory, stopping when it has found a concept which matches the data supplied. To focus theory formation, we adapted HR's heuristic search to favour building on concepts which achieved a categorisation closer to the one achieved by the target concept. For reasons given in [2], we found that this approach often failed to learn integer sequences because it concentrated on the wrong concepts.

Instead of the heuristic search, we used an exhaustive search enhanced with a look ahead mechanism. As an example, given the sequence 2, 3, 5, 7, 11 (prime numbers) we have enabled HR to notice that, when it forms the concept of number of divisors, the prime numbers all have two divisors, a fact which is true of none of the other integers in HR's dataset. Each production rule has an algorithm for spotting a pattern which is true only for the positive examples, and when this happens a suitable theory formation step involving that production rule is added to the top of the agenda. Execution of the step will produce a concept which fits the data. The pattern spotting mechanism is faster than actually performing a theory formation step because (a) for the majority of the time, it is possible to quickly tell that there is no pattern and (b) there are overheads involved in performing a theory formation step.

We also implemented a scoring system for the look ahead mechanism whereby the largest number of positive examples for which a pattern could be found was output by the production rule. However, for the

examples we used — well known number theory concepts such as prime numbers, squarefree numbers and odious numbers — we found that the scoring system performed worse due to increased overheads and the suggestion of theory formation steps which do not lead to the desired concept. However, we did implement a ‘unary-first’ search by combining a depth first and breadth first search: the unary production rules are used exhaustively for each new concept before returning to the binary production rules with old concepts. In this way, each new concept receives some preliminary development, but is not combined with previous concepts until later.

The look ahead mechanism has been successful with both the problems about trains and integer sequences, and we supply results in [2]. It is particularly effective when the concept to be learned is a combination of two old concepts, for example odd prime numbers which combines the concepts of odd numbers and prime numbers. Depth first, breadth first and unary first searches do not find this concept quickly without the look ahead mechanism. However, with the look ahead mechanism, odd numbers are invented and as soon as prime numbers are introduced, HR notices that the positive examples are both odd and prime (and the negative examples are not). HR then combines these concepts and reaches the solution much quicker — in fact the time taken to learn the concept reduces from 384 to just 5 seconds. We are planning to compare HR and the Progol machine learning program [9] as discussed in §5 below.

4 Problem Solving

In his book on mathematical problem solving [11] Paul Zeitz suggests a ‘plug-and-chug’ method, whereby calculations are performed and the results analysed to see if any pattern emerges which might provide insight into the problem. Zeitz supplies the following problem — taken from a 1930s Hungarian mathematics contest — as an example where this approach leads to the solution:

Show that the product of four consecutive integers is never a square number.

Following the plug and chug method, Zeitz calculates examples of the product of four consecutive integers:

$$1 \times 2 \times 3 \times 4 = 24 \text{ and } 2 \times 3 \times 4 \times 5 = 120$$

The sequence of calculations continues: 24, 120, 360, 840 and a eureka moment occurs with the realisation that these are all one less than a square. Zeitz then makes the conjecture that all such numbers are

one less than a square and hence not square numbers. Zeitz states that:

Getting to the conjecture was the crux move. At this point the problem metamorphosed into an exercise!

To finish the problem, it is necessary to show that the product of four consecutive integers can be written as a square minus 1:

$$n(n+1)(n+2)(n+3) = (n^2 + 3n + 1)^2 - 1.$$

We have applied HR to plug-and-chug problems of this nature, by getting it to make suggestions which might lead to the eureka moment. To do this, HR is given a set of numbers which are related to the problem and asked to suggest properties in the hope that one of the suggestions will provide insight into the problem. This occurs through theory formation: for every new concept HR introduces, if the examples it is given have the property prescribed by the concept, then the definition is output. For example, when used for the Hungarian contest problem above, HR is given the numbers 24, 120, 360 and 840. As it forms a theory, it invents types of number and when the numbers 24, 120, 360 and 840 all satisfy the definition of a particular number type, the definition is output. Of course, some suggestions do not provide insight (for example that they are all even numbers). However, HR eventually invents the concept of squares-minus-one and so finds the conjecture which metamorphosed the problem. In this case, only 15 suggestions are made before the correct one, but we may have to introduce pruning measures if too many suggestions are produced.

The application of HR to problem solving is very recent and has not been reported anywhere other than here. We are presently experimenting with this application of theory formation, in particular we are compiling a corpus of problems where the plug-and-chug approach would help. We intend to attach this functionality to a computer algebra package such as Maple or Mathematica.

5 Conclusions and Further Work

It was not our intention here to discuss whether HR is better or worse than other programs at discovery, learning or problem solving tasks. Rather, we wanted to compare the use of theory formation in these three tasks. In particular, we note that all three problems are effectively learning tasks. For discovery, HR is asked to learn about the top level concept of number.

In doing so, it re-invents many well known concepts such as prime numbers, but also invents new concepts and supplies interesting conjectures about them. For problem solving, HR is asked to learn about a particular concept which the problem discusses. This highlights that the problem of learning a concept definition is a sub-task of the problem of learning *about* a concept, which involves forming a theory about a concept (whether it be the concept of number or a more specialised concept, such as the product of four consecutive integers).

In [2] we started the process of comparing HR with the machine learning program Progol [9]. We showed that various aspects of the inductive logic programming algorithm, including inverse resolution, the background knowledge supplied and the mode parameters, correspond directly with the production rules that HR uses, in terms of the types of concepts which Progol and HR can learn. We also pointed out the limitations in HR as a machine learning program. We intend to fully compare the programs by enabling Progol to work in a domain that HR works in (number theory) and by enabling HR to work in a domain that Progol works in (train theory). We have enabled HR to perform machine learning tasks such as those performed by Progol. Similarly, we intend to use Progol to form theories containing concepts and conjectures in a similar way to HR.

One way to do this in number theory would be to supply Progol with a set of categorisations of the numbers 1 to, say, 6 and ask it to learn a concept achieving the categorisation. There are 203 such categorisations, so the theory would contain at most 203 concepts — fewer if Progol could not learn a definition for some categorisations. Conjectures could also be formed by changing the settings and/or the background predicates in Progol and asking it to learn another definition for a particular concept. If it produced a different concept definition, a conjecture stating the equivalence of the two definitions could be made. Similarly, implication conjectures could be stated using two concepts expressing a type of number (which would be found for boolean categorisations). An implication conjecture could be stated if all the numbers in the positive category for one concept were in the positive category of the other concept (similarly for the negative category).

We have showed in principle that theory formation can be applied to learning, discovery and problem solving tasks. This has mainly been achieved in number theory and we need to test the application of theory formation in other domains, including non-mathematical ones — we have already started testing HR on train problems. We also hope to apply theory formation to constraint satisfaction problems — where some of the conjectures HR makes are turned

into constraints and added to the problem specification. We also hope to apply theory formation to the generation of puzzles, such as odd-one-out and next-in-sequence puzzles.

Acknowledgments

This work is supported by EPSRC grant GR/M98012.

References

- [1] S Colton, A Bundy, and T Walsh. HR: Automatic concept formation in pure mathematics. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1999.
- [2] S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proceedings of the 17th International Conference*, 2000.
- [3] S Colton, A Bundy, and T Walsh. Automatic invention of integer sequences. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2000.
- [4] R Davis and D Lenat. *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill Advanced Computer Science Series, 1982.
- [5] S Epstein. On the discovery of mathematical theorems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 1987.
- [6] W McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories, 1990.
- [7] W McCune. A Davis-Putnam program and its application to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories, 1994.
- [8] R Michalski and J Larson. Inductive inference of VL decision rules. In *Proceedings of the Workshop in Pattern-Directed Inference Systems (Published in SIGART Newsletter ACM, No. 63)*, 1977.
- [9] S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
- [10] N Sloane. The encyclopedia of integer sequences. <http://www.research.att.com/~njas/sequences>, 2000.
- [11] P Zeitz. *The Art and Craft of Problem Solving*. John Wiley and Sons, 1999.