Automated Puzzle Generation

Simon Colton
Division of Informatics
University of Edinburgh
Edinburgh EH1 1HN
United Kingdom
simonco@dai.ed.ac.uk

Abstract

We give a characterisation of certain types of puzzle in terms of the structure of the question posed and the nature of the answer to the puzzle. Using this characterisation, we have extended the HR theory formation system (2) to enable it to automatically generate puzzles given background information about a set of objects of interest. The main technical difficulty to overcome was to ensure that the puzzles generated by HR had a single solution (up to a level of plausibility). We give details of the implementation and some results from its application.

1 Introduction

Broadly characterised as the generation of novel information about a domain given some background information, the specific tasks in machine learning include finding a concept given some positive and negative examples of the concept, and generating a scheme for predicting the nature of a given object. We can add to this list of tasks the generation of problems and puzzles about a given set of objects in a domain. While much effort has been expended on determining and implementing general problem solving techniques (8; 5), there has been little research on the question of automatically generating puzzles. This is in spite of the observations that scientific advance sometimes occurs through specifying exactly the right question to ask (and finding the answer), and that setting exercises is an important educational tool.

Our primary aim here is to determine the nature of certain types of puzzle and to show how the production of them can be automated. This is similar to how Ritchie specified the internal linguistic structure of a joke, separately to the notion of what makes a joke funny (7). However, a secondary aim is to determine the nature of a "good puzzle". While a full study of the notion of a good puzzle would require a field test of the puzzles produced automatically, we make some preliminary observations and some plausible suggestions about increasing the difficulty of a puzzle. We begin in §2 with an analysis of some puzzles from which we make a characterisation of certain types of puzzle. In §3, we discuss ways of increasing the difficulty of the puzzles. In §4 we discuss the HR theory formation system (2) and in §5, we describe the extension to it which has enabled puzzle generation. We present some results in §6 and conclude in §7 with a brief discussion of puzzles in the context of creativity research.

2 A Characterisation of Puzzles

We take as examples for study eight puzzles taken verbatim from the web pages at queendom.com, a popular site for people interested in solving puzzles.

---

1. Q. Which is the odd one out:
coconuts, oysters, clams, eggs, walnuts, haddock?
A. Haddock: all the others have shells.

2. Q. Which is the odd one out:
hair, triangles, squares, plants, words, trees?
A. Triangles: all the others have roots.

3. Q. Which is the odd one out:
soft ice cream cone, salsa, landscape, raw vegetable, sales, chips?
A. Salsa: all the others can be dipped.

4. Q. Jingle is to corporation as ___ is to politician:
(a) campaign (b) platform (c) slogan (d) promises?
A. Slogan.

5. Q. Hair is to stubble as potatoes are to ___:
(a) french fries (b) sweet potatoes (c) potato skins (d) vegetable?
A. French fries.

6. Q. What is next in the sequence: 3, 8, 15, 24, 35?
A. 48: Starting from 2, square each consecutive integer then subtract 1.

7. Q. What is next in the sequence: 2, 7, 4, 14, 6?
A. 21: These are the multiples of 2, alternatively interspaced with the multiples of 7.

8. Q. What is the next in the sequence: 15, 210, 115, 220, 125? A. 230. These are the multiples of five with the digit 1 or 2 alternatively placed in front of each number.

The first thing to notice about these puzzles is that they fall into three classes:

- Odd one out puzzles (puzzles 1, 2 and 3)
- Analogy puzzles (puzzles 4 and 5)
- Next in sequence puzzles (puzzles 6, 7 and 8)

Indeed, the puzzles at queendom.com are arranged into classes, and there are many others, such as "hidden word" and "word associations".

We impose the following characterisation on these puzzles: each puzzle consists of (i) a question (ii) a set of choices, one of which is the answer (iii) the answer (iv) an explanation which consists of a single, positively stated concept. For example, in puzzle 1, the question is: "Which is the odd one out", the set of choices is {coconuts, oysters, clams, eggs, walnuts, haddock}, the answer is haddock and the concept is the notion of having a shell.

We note that the puzzles stated above do not all fit this characterisation, and some flexibility is required. Firstly, the next in sequence puzzles do not have a set of possible answers to choose from. However, the implicit assumption is that the answer is an integer, hence we can say that the set of choices in this case is the set of natural numbers. Secondly, the analogy puzzles do not have an explanation. This is perhaps because, in the examples given above, the analogy is not exact, but rather the solution has a closer analogy than the other possible answers. However, it is reasonable to assume that puzzles where the analogy is via a single concept are acceptable. For instance, if the concept was "doubling" and the domain was numbers, then the analogy puzzle:

Q. 10 is to 20 as 30 is to 60 as 40 is to ___:
70, 80, 90?
A. 80 because 20 is 10 times 2, 60 is 30 times 2 and 80 is 40 times 2.

is valid. Hence to impose our characterisation, we assume that there is a single concept which explains the solution to the puzzle.

We call this concept the *embedded* concept, and by saying that it must be positively stated, we mean that the concept is not obviously the negation of another concept. For instance, in odd one out puzzles, it is usual for the odd one out to be lacking a property that the others all have. Puzzles where the opposite is the case are to some extent unsatisfying, for instance, consider the puzzle:

Q. Which is the odd one out: 2, 3, 9, 20?
A. 9 because it is a square and the others are not.

We see that the standard format has been violated: it is expected that the solution lies in finding a concept which 2, 3 and 20 satisfy and 9 doesn't, not in finding a property unique to 9. We also note that the embedded concepts are, in general, fairly simple and are rarely formed through disjunction. For example, if an explanation was something like: "because the others are either a fruit or have a shell", this too would be slightly unsatisfying.

Given this characterisation, we note that the three puzzle types differ only in the presentation of the question and explanation. In odd one out puzzles, the problem is to find the choice which does not have a property that the others have. In analogy puzzles, the problem is to find the choice which is most analogous with the given object. In next in sequence puzzles, the problem lies in extrapolating a sequence.

Another important observation is that the puzzles must have only one plausible solution. Consider, for example the puzzle:

Q. Which is the odd one out: 4, 9, 18, 36?

Here, there are (at least) two simple answers: (a) 18 is the odd one out because the other are all square numbers and (b) 9 is the odd one out because the others are all even numbers. Hence this puzzle is ineffective as there is a possibility that the solver will get the "wrong" answer, but not accept that his or her solution was worse than the supposed "right" answer. We see that the puzzle generator must take into account all simple concepts and ensure that the examples for the embedded concept do not form a puzzle for another embedded concept of similar or lesser complexity.

3 Increasing the Difficulty of Puzzles

We have shown that – with a little flexibility – the eight puzzles provided fit into a characterisation in terms of a question statement, a set of objects from a domain to choose from, an answer, and an embedded concept which explains the solution. This characterisation is useful in implementing puzzle generation, as discussed in §5 below. We have also noted that for each puzzle, the set of objects of interest do not embed a puzzle of similar or lesser complexity. This will help to ensure (but by no means guarantee) that the solver will be satisfied with the solution.

We must also concern ourselves with the difficulty of the puzzle. In particular, a puzzle which is too easy will be of little interest to the solver, yet a puzzle which is so difficult that no-one can find the solution is also ineffective. A balance needs to be found so that the intended puzzle solvers have some difficulty finding the answer, but in general, they eventually prevail. We describe below three properties of a puzzle which may help us to assess the difficulty of the puzzle. We make an implicit assumption that the time taken to solve a puzzle gives an indication of the difficulty of that puzzle.

3.1 Number of Choices

In the eight puzzles given above, the number of objects to choose the answer from ranges from 4 to 6, and increasing the number of objects is a potential way to increase (or indeed decrease) the difficulty of the puzzle. A plausible method for solving odd one out puzzles is to choose an odd one out and attempt to find a property that the others share. Another possible method is to take two objects and find a property that they share, then try to add a third with the same property and so on, until all the objects have been added, with the exception of the odd one out. In either case, having more examples will slow down the solving process, and hence can be thought of as increasing the difficulty.

Similarly, to solve analogy puzzles, a plausible method is to choose a possible answer and attempt to find a concept which provides both an analogy between the pairs of objects stated in the question and an analogy between the final object stated in the question and the answer chosen. Again, increasing the number of objects to choose an answer from may increase the time taken to find a solution.

3.2 Complexity of the concept

Another possible way to increase the difficulty of the puzzle is to increase the complexity of the embedded concept, because, to search for the answer, there must be a search for the explanatory concept. Hence, making the solver search further for that concept will increase the difficulty of the puzzle.

We noted that, in general, we've observed that the concepts embedded in the puzzle are usually not particularly complex. One explanation for this could be that it is unlikely that a simpler solution will exist if the concept chosen to embed is simple. That is, checking for uniqueness of the solution up to a particular complexity level will be easier if the concept chosen to embed is itself simple. With this observation, we can tentatively state that, if the uniqueness-checking mechanism is reliable and efficient, then there is no reason why the difficulty of puzzles couldn't be raised by choosing complicated concepts to embed in the puzzles.

3.3 Disguising Concepts

If we look at puzzle 7 above, there is clearly an attempt to disguise the puzzle by involving another concept in the examples given in the problem statement. In that case, the embedded concept is multiples of 7, and the disguising concept is multiples of 2. It could be argued that multiples of 2 is also an embedded concept, and that our characterisation should take into account the possibility of multiple concepts providing the explanation. However, we note that the answer is only dependent on the concept of multiples of 7, and the explanation could have been provided as: "every other term is a multiple of 7, hence the next is 21", with no explicit reference to the concept of multiples of 2. Therefore, we call the secondary concept the disguising concept, and note that it only serves to increase the difficulty of the puzzle by diverting the solver's attention from the real problem in the puzzle.

It is important to note that there are a number of ways to employ disguising concepts, and it is part of the solving process to (a) realise that the puzzle is disguised and (b) work out and ignore the disguising concept(s). For number sequences, interleaving the disguising concept as in puzzle 7 is a common construction, as is appending the disguising sequence onto the front (or end) of the embedded sequence, for instance puzzle 8 above, where the embedded concept is "multiples of five", and the disguising concept is placing the number 1 or 2 alternatively in front of each number.

If we assume the strategies for finding the odd one out discussed in §3.1, then we can propose a way to use a disguising concept C in odd one out puzzles: choose examples which *all* share the property prescribed by C. In this way, as the solver attempts to find a property which the majority of examples share, he or she may come across C, because as all the examples share the property. As this cannot lead to a solution to the odd one out puzzle, it simply serves to slow the solver down. For instance, in puzzle 1 above, the solver might realise that the majority of the examples provided are foodstuffs, and try to find an example which is not. In this case, they would be disappointed to realise that *all* the examples are foodstuffs and therefore another concept is embedded in the puzzle.

4 The HR Program

The HR system performs automated theory formation by inventing concepts, making conjectures, proving theorems and finding counterexamples (2). The main functionality used for the application to puzzle generation is concept formation, which is achieved by using production rules which take one (or two) old concepts as input and output a new concept. The production rules perform as follows:

- `Compose:` this composes concepts using conjugation.
- `Disjunct:` this combines concepts using disjunction.
- `Exists:` this introduces existential quantification.
- `Forall:` this introduces universal quantification.
- `Match:` this equates variables in predicate definitions.
- `Negate:` this finds compliments by negating clauses.
- `Size:` this counts set sizes.
- `Split:` this instantiates variables.

As an example of how the production rules are employed, figure 1 shows how HR can construct the concepts of squares minus one (as used in puzzle 6 above). We see that the concept of square numbers is constructed from the user-given concept of multiplication using the match and exists production rules. The concept of adding 1 (equivalently subtracting 1) is constructed using the split rule to instantiate the number being added to 1. Finally, the compose rule is used to join these two concepts into the desired concept. We say that the concept produced in figure 1 is of *complexity* 5, because five concepts (including itself) are used in the construction path. The complexity

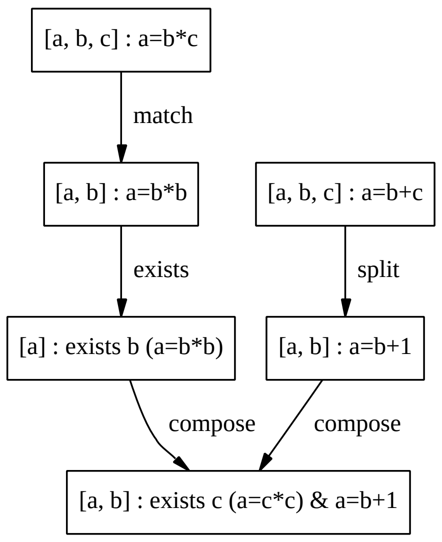

Figure 1: Example construction via production rules

of a concept gives some indication of how complicated the notion expressed in the concept is, and we use this in the puzzle generation, as discussed in §5 below.

Another fact used in puzzle generation is that each concept produces a categorisation of the objects of interest in a domain, for example the concept of squares minus one categorises the numbers 1 to 15 into two categories:

$$[3, 8, 15], [1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14]$$

The fact that HR knows when a concept is a specialisation of the objects of interest is also employed in puzzle generation. For example, once constructed, HR knows that the concept of squares minus one actually specialises the concept of integers. Furthermore, HR knows when one concept is a generalisation of another, information which is also used in the puzzle generation process. For instance, HR knows that the concept of prime numbers is a generalisation of the concept of odd prime numbers. HR also maintains knowledge of which concepts are functions. The user specifies which concepts supplied as background knowledge are functions, and HR propagates this information as the theory is built, i.e., it knows in which circumstances the application of a production rule will result in a function, and records this information. In particular, the size production rule (which counts sizes of sets) always produces a function. For more details of HR's concept formation, see (4), or chapter 6 of (2).

5 Extension for Puzzle Generation

Our original motivation for using HR for puzzle generation was the knowledge that HR can form a theory about a set of objects of interest, given some concepts about those objects. Hence it seemed that the task of producing puzzles with unique solutions could be achieved by (i) forming a theory and (ii) taking each concept in the theory and embedding it in a puzzle in such a way that no other concept in the theory could be used to provide a solution to the puzzle. Hence, we extended HR to produce puzzles of the three types mentioned above (odd one out, analogy and next in sequence). There are many ways to produce such sequences, and we have only explored one or two avenues for each puzzle type – there is still much work needed to make HR proficient at generating puzzles.

Due to the characterisation afforded above, much of the implementation was not specific to the particular puzzle types. However, the two main areas where the implementation differs for the puzzle types is in checking that no other simple solution to a puzzle was possible, and in disguising the puzzle, and we deal with these in the subsections below. Also, various observations made in the characterisation above influenced our approach to automated puzzle generation, including:

- Negated concepts are not usually provided as the solution to puzzles. Based on this observation, we decided to omit the negation production rule when producing theories from which puzzles will be generated. This helps avoid puzzles where the explanation is a negated concept (for example, the odd one out is a square number, whereas the others are non-squares).

- Concepts with disjunction are not usually provided as the solution to puzzles. Based on this observation, we decided to omit the disjunct production rule when producing theories from which puzzles will be generated. This helps to avoid puzzles where the explanation concept has disjunction (for example, the embedded concept is "primes or squares").

- The solution to a puzzle is supposed to be the simplest such solution. As mentioned above, the methods for checking uniqueness of solution is described for each puzzle type below. In essence, however, HR simply checks that there is no simpler solution to a puzzle it has generated by looking through all the concepts in the theory. Our notion of whether one solution is simpler than another was changed by some initial results from HR: we originally intended to use HR's complexity measure to determine which solution was simplest, with the least complex concept embedded being the simplest solution. However, we found that sometimes, more complex concepts were actually easier to understand than less complex ones. In particular, the match production rule will generate a concept with complexity one more than its parent, but in many cases, the definition of the concept produced will be simpler than that of the parent. Hence, we decided that, once all concepts up to a particular complexity limit (usually 4 or 5) had been generated, HR should discard puzzles which have two solutions, even if one embedded concept is more complex (according to HR's measure) than the other. While this lowers the yield of puzzles, it also reduces the possibility of a puzzle being generated with two or more plausible solutions.

5.1 Odd One Out Puzzles

After a theory containing many concepts has been formed by HR, the user decides the number of choices, n, and asks HR for odd one out puzzles. HR then looks at each specialisation concept, C, in turn and takes each subset of n objects of interest where the first $n-1$ have the property prescribed by C and the last one does not have the property. For each tuple of n objects, it checks whether C is the only concept embedded in the tuples for an odd one out puzzle. For instance, if it took the integers $3, 5, 17, 8$ to embed the concept of prime numbers (true for 3, 5 and 17, not true for 8), it would run through all the other specialisation concepts and check that no other solution was possible. In this case, when it came to the concept of integers with one digit, it would realise that 17 could be considered the odd one out instead of 8, because 17 has two digits, whereas the others have only one. Hence this quadruple would be rejected for an odd one out puzzle where the concept to be discovered is prime numbers.

Tuples of objects for embedding a concept C into a puzzle are rejected if another concept D can be embedded as the solution to the puzzle, with one exception: when C is a generalisation of D. For example, if $3, 5, 17, 22$ were chosen to embed the concept of prime numbers in a puzzle, then the concept of odd prime numbers could be used to reject this tuple, as it provides another answer (22 is not an odd prime number, the others are). However, as mentioned above, HR keeps track of which concepts are specialisations of another and it knows that the concept of prime numbers is a generalisation of odd prime numbers, so it will not reject the tuple for the puzzle. In effect, the solver is expected to be happy with the answer of prime numbers for the puzzle, if they have decided that the answer was odd prime numbers, because the former is a simpler solution. Hence, as HR knows which concepts are generalisations of the others, it can avoid rejecting a tuple when a more specialised concept can be embedded in the objects chosen.

For each concept, HR collects all the tuples which pass the uniqueness test, and determines the level of disguise of each, measured as the number of concepts in the theory which specify a property that *all* the objects satisfy. For instance, if the concept of square numbers was embedded with the examples 4, 16, 20, 36, then the concept of even numbers would be a disguising concept for this puzzle, as all the numbers are even. The choice of objects determines the number of other concepts in the theory which act as disguising concepts. We found that, even with a high level of disguise, puzzles with simple concepts were easy to solve. Hence, we used an interestingness measure which takes the average of the complexity of the embedded concept and the level of disguise. For each concept, the most interesting puzzle is taken as a representative puzzle for that concept. To finish the puzzle, HR randomly re-orders the choices, so that the odd one out is no longer the last choice.

5.2 Analogy Puzzles

Our treatment of analogy puzzles has been fairly shallow so far. After a theory has been formed, HR takes each specialisation concept and chooses objects of interest to embed the concept in a puzzle. There are many ways in which the analogy could be set up, and we have so far only experimented with one: HR produces puzzles of the form "A is to B as C is to ?", where objects A, B, C and the solution, all share the property expressed by the concept.

The user specifies the number, n, of choices which will be available in the puzzle, and then to generate puzzles from a concept C, HR takes each category, T, in the categorisation of the objects of interest afforded by C, and counts the number of objects in T. If this number is four or more, then HR randomly chooses four objects O_1, O_2, O_3 and O_4 from T. From the objects of interest which are not members of T, HR chooses objects $X_1, \ldots, X_{n-1}$ as the choices for the puzzle. It then adds O_4 to this list in a random position, so that O_4 becomes X_i for some i and there are n choices for the answer to the puzzle. The puzzle is then stated as: "O_1 is to O_2 as O_3 is to $X_1, X_2, \ldots, X_n$?".

As with odd one out puzzles, for each puzzle HR generates, it checks whether there is another equally plausible solution. In the case of analogy puzzles, to do this, it runs through every other concept D, and checks whether there is a category V in the categorisation produced by D such that V contains O_1, O_2 and O_3 and V also contains a single object from the set $X_1, \ldots, X_n$. In such cases, D also provides a solution to the puzzle, so the puzzle should be discarded. As with odd one out puzzles, the exception to this rule is when D is a specialisation of C.

For analogy puzzles, we do not yet worry about disguising the puzzle. For this reason, for each concept to embed in a puzzle, HR stops once it has found a puzzle which passes the uniqueness test. To stop biasing towards certain objects of interest, the category T is chosen randomly from all the categories, and $\{O_1, O_2, O_3, O_4\}$ and $\{X_1, \ldots, X_{n-1}\}$ are chosen randomly from all possible sets until the supply is exhausted.

5.3 Next in Sequence Puzzles

Three common formats for next in sequence puzzles are:

- The embedded concept is a number type and successive examples of the number type are given in order, with the next such number being the answer, e.g., puzzle 10 in §2 has the concept of multiples of five embedded.

- The embedded concept is a function f mapping the integers to themselves, and the function is applied to successive integers $n, n+1, \ldots, n+k$ for some n and k, with the answer being $f(n+k+1)$. For example, in puzzle 6 in §2, the function is $f(n) = n^2 - 1$ and the application of f starts with $f(2) = 3$ and continues until $f(6) = 35$, so that $f(7) = 48$ is the answer.

• The embedded concept is again a function f, but it is applied recursively to the previous terms in the sequence. An example of this is the Fibonnaci sequence: $1, 1, 2, 3, 5, 8, \ldots$, where the nth term in the sequence is gained by adding together terms $n - 1$ and $n - 2$ (with 1 and 1 chosen arbitrarily as the first two terms of the sequence).

At present, HR only has the functionality to produce puzzles of the first and second format, although we do not envisage any problems extending HR's range to include the third format.

To generate puzzles of the first type, HR forms a theory in the domain where the objects of interest are integers and then looks for concepts C which are number types, i.e., specialisation concepts. Given that the user has specified that n integers should be given as clues to the puzzle, HR chooses n successive integers from the integers which have the property prescribed by C. Instead of choosing the starting point for the clues randomly, we made HR choose them in such a way that the clue numbers were as large as possible. For example, given the numbers 1 to 30 about which to form a theory, when producing a puzzle with 5 integers as clues to a puzzle with prime numbers as the embedded concept, HR would choose the numbers $11, 13, 17, 19$ and 23, with the final prime number that HR knows about (29) being held back as the answer. We found that many sequences have similar numbers early on in the number line, and hoped that, by choosing the largest numbers possible, it would be easier to ensure the uniqueness of the solution and also perhaps increase the difficulty of the puzzle.

To generate puzzles of the second type, HR takes each concept C, which it knows to be a function, f, mapping the integers to themselves. HR then chooses a number x to start at, and provides the integers $f(x), f(x + 1), \ldots, f(x+n)$ as clues to the puzzle, with $f(x+n+1)$ held back as the answer. As with the first type, we made HR choose x to be as high as possible. With both types of sequence, HR checks that the solution is unique for all the concepts in the theory. It takes into account the fact that an alternative solution could come from a sequence generated by number types or by a function.

Rather than trying to find puzzles with the most disguise – as is the case with odd one out puzzles – HR explicitly imposes disguise on a puzzle by interleaving the clues with another sequence of similar complexity. HR does this only if the complexity of the embedded concept is very low (the user sets this, usually at complexity 2 or less). This means that only the puzzles about very simple concepts such as even numbers are disguised. For instance, HR disguises the concept of numbers with the digit 2 in them (given with examples 12, 20, 21) by interleaving them with the concept of even numbers (2, 4, 6, 8) to produce the puzzle: "what is the next in the sequence: 2, 12, 4, 20, 6, 21, 8?"

We chose the starting point for the disguising sequence to be as low as possible. Our rationale behind this was that we wanted the solver to recognise the disguised concept (thus being somehow distracted by it). After disguising, HR again checks for the uniqueness of the solution, and will backtrack over (i) the choice of start point for the disguising sequence (ii) the choice of disguising sequence and (iii) the start point for the embedded sequence, until it finds a puzzle which has a unique solution.

There are many other methods for generating and disguising next in sequence puzzles which we have not yet equipped HR with. HR will need to both generate puzzles using these methods and take them into account when checking for uniqueness of the solution. For instance, HR will eventually take into account that many sequence extrapolation puzzles are solved by taking the difference sequence, i.e., the difference between successive terms.

6 Results

Unfortunately, time has not permitted the kind of extensive development and testing of HR's puzzle generation capabilities that we wanted to report on here. In particular, we wanted to use HR to generate puzzles of a visiospatial nature, but we have not had time yet to do this.

6.1 Animals Puzzles

To generate odd one out and analogy puzzles, we chose the animals dataset which is supplied not with HR, but rather with the Progol machine learning program distribution (6). This dataset consists of 18 animals described with 12 properties, such as whether or not they produce eggs, which habitats they dwell on (air, water, land) and so on. We set HR's complexity limit to be five and ran the search to exhaustion using the compose, exists, forall, match, size and split production rules. It took 65 seconds to build the theory on a Pentium 500Mhz processor. We then asked for all odd one out and analogy puzzles with four choices which could be generated as prescribed in §5 above. This took a further 7 seconds and produced 31 puzzles, which we supply in appendix A in such a way that the choices for the answer are given, with the correct one bearing an asterix, and the embedded concept and disguising concepts stated below the choices.

With some puzzles, the complexity of the embedded concept may have made the puzzle of sufficient difficulty to be interesting, for example, the solver needed to know in puzzle 25 that of the four animals supplied, dolphin was the only one to have a single habitat (water). With other puzzles, however, the choice of the animals negated the complexity of the concept, e.g., puzzle 27 asked: dog is to cat as eagle is to (a) lizard (b) eel (c) ostrich (d) trout, with ostrich being the only plausible solution, regardless of the concept which related dog, cat, eagle and ostrich (which was being a homeothermic land-dweller).

Puzzle 7 had the most disguise: the objects to choose the odd one out from were ostrich, platypus, eagle and penguin, which share three properties; they are homeothermic, produce eggs and have two legs. However, as they are also all animals (a disguising concept for all the odd one out puzzles), and because HR invented the concept of being homeothermic and producing eggs, which they all share, this puzzle scored 5 for disguise. This suggests that an improvement in measuring disguise would be to make sure that the properties the objects share are not really being counted twice. In general, we were disappointed with the level of disguise, and it was at such a low level to be unlikely to have an effect on the solver's ability to find the solution.

An important observation is that relatively few concepts used in the puzzles were of complexity 4 and 5. This is partly due to there being slightly fewer such concepts in the theory, but also because the more complex concepts tend to be more specialised. As the complexity (and hence specialisation) of a concept increases, it becomes less likely that the concept will have sufficient positive examples to make a puzzle out of it. For instance, the concept of birds which fly has only one example in the animals dataset: eagle. Hence this concept cannot be used in an odd one out puzzle. The solution to this, of course, is to provide more objects of interest (in this case, more animals) in the background information, although this will mean that the theory formation and puzzle generation will take more time.

To summarise the animals results, as the level of both disguise and complexity of the puzzles was in general fairly low, we expect these puzzles to be easily solved. It appears, however, that a sizeable proportion of the puzzles would be satisfying to the solver, in that they would probably come up with the solution that HR prescribed.

6.2 Integer Sequences

To produce next in the sequence puzzles, we ran HR in number theory with the numbers 1 to 30 and the concepts of: multiplication, addition, divisors and digits, which are all commonly used concepts in sequence extrapolation puzzles. Due to experience in this domain, to reduce the complexity of the concepts generated, we used a complexity limit of 4 and did not use the forall production rule, using only compose, exists, match, size and split. Again, we ran the exhaustive search to completion, and then asked for all next in sequence puzzles where 6 numbers were given as clues. It took 165 seconds on a Pentium 500Mhz processor to form the theory and a further 2 seconds to generate the 24 puzzles given in appendix B.

Some of the puzzles generated are promising, for instance HR made a puzzle about prime numbers (puzzle 4), which is a mainstay of human produced puzzles. Also, puzzle 10 asked: what is next in the sequence: 21, 22, 24, 25, 26, 28, with 30 being the answer, as this is the sequence of integers where there is a digit which divides the number. This is perhaps at around the limit of difficulty for a next in sequence puzzle. Also, the disguising process for the simpler concepts seemed to work fairly well. In particular, in puzzle 3, HR used the same concept (multiples of 3) to disguise the embedded concept to produce the sequence: 21, 3, 24, 6, 27, 9, which was interesting.

However, many of the puzzles highlight problems with HR's puzzle generation which will need improvement. In particular, some of the concepts, while only at complexity 4 in HR's terms might be a little complicated to reasonably expect the solver to identify them. These include the number of even divisors of the integers (puzzle 20). The situation is exasperated by the policy of starting the puzzles as far down the number line as possible. While this is a good idea for puzzles involving number types, it seems likely that expecting the solver to realise that the sequence 6, 0, 2, 0, 4, 0 has been generated by writing down the number of even divisors of 24, 25, 26, 27, 28 and 29 is asking too much. This suggests that for sequences generated by applying a function to successive integers, the clues to the puzzle should start by applying the function as close to the number 1 as possible.

The last two puzzles in appendix B highlight the fact that puzzles with a simpler solution can sometimes slip through. It is unlikely that the solver would accept HR's convoluted answer for the next in the sequence: 0, 1, 2, 3, 4, 5 (puzzle 24). In this particular case, as HR was given the numbers 1 to 30 to work with, not 0 to 30, it never realised that the sequence 0, 1, 2, 3, 4, 5 was part of the natural number sequence. Similarly, HR should have realised that the sequence 11, 12, 12, 13, 13, 14 (puzzle 23) could be generated by repeating the natural numbers.

To summarise the next in sequence results from this session, it seems that the majority of the sequences generated from number types produced acceptable puzzles and the policy of taking the largest numbers possible worked well. However, the policy worked badly for the sequences generated from functions and this combined with the high complexity of the embedded concepts meant that the puzzles were perhaps too difficult.

7 Conclusions and Further Work

The generation of puzzles is a machine learning task which, to our knowledge, has rarely been tackled. To implement puzzle generation, we identified three common types of puzzle and found a characterisation to fit the three types. This enabled us to extend the HR program to generate odd one out, next in sequence and analogy puzzles. We presented results in two domains to highlight the strengths and weaknesses of our approach. We found that the main technical problem with puzzle generation was ensuring the uniqueness of the concept supposed to explain the puzzle solution. We used automated theory formation to build a theory so that, up to a reasonable level of complexity, HR could check that there was no other simple solution to a puzzle. While this approach is not perfect – indeed it is a very difficult problem to predict the creative approaches to problem solving that people will employ – we found that in most cases the solution found by the solver was likely to be the one they were supposed to find. We argue in (3) that the theory formation approach employed by HR is perhaps better suited for puzzle generation than other machine learning techniques which aim to find a single concept.

Puzzle generation, which, in its broadest sense encompasses many activities ranging from identifying the correct question to propagate research to the setting of exercises to educate students, is an intelligent activity which itself requires creativity, but which must take into account the perceived creativity of the intended puzzle solvers. In particular, a puzzle must be generated in such a way that the given answer is more plausible than any other the solver might think of. In terms of Boden's characterisation of creative acts (1), a puzzle setter must choose examples for the puzzle which make the solver act P-creatively (produce a solution which is a personal discovery), rather than H-creatively (produce a solution which is historically novel). This is because, by definition, a H-creative solution to a puzzle will be wrong (in the sense that it was not the answer intended by the puzzle setter). Once we have improved the quality of HR's puzzles and tested them with human solvers as mentioned below, we hope to have a greater insight into the nature of machine/human creativity with respect to puzzle generation and solving.

The results from the initial testing of HR's puzzle generation show that there is still much more to be done in order to increase the quality of the puzzles it produces. We hope to follow two main directions with this work. Firstly, we must test whether human puzzle solvers find the puzzles HR produces interesting, and we hope to use a web-based experiment which asks the solvers to rate (a) how difficult they thought the puzzle was and (b) how happy they were with the solution. We will then compare their difficulty ratings with those used by HR, and attempt to reduce the number of puzzles produced for which they gave a different (but valid) answer.

Secondly, we hope to add a layer of sophistication to the assessment of difficulty by getting HR to model the puzzle solver as well as the puzzler setter. HR has already been applied to integer sequence extrapolation (4), and we hope to combine this application with the application to puzzle generation, so that one copy of HR generates the puzzles and another copy attempts to solve them, giving an indication of the difficulty in the process.

Acknowledgments

This work is supported by EPSRC grant GR/M98012. The author is also affiliated to the Dept. of Computer Science at the University of York. This research was inspired by a very interesting meeting with Prof. Herbert Simon, whose comments and questions greatly improved the author's understanding of puzzle solving and generation.

References

[1] M Boden. *The Creative Mind*. Weidenfeld and Nicolson, 1990.

[2] S Colton. *Automated Theory Formation in Pure Mathematics*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, 2000.

[3] S Colton. An application-based comparison of automated theory formation and inductive logic programming. *Linkoping Electronic Articles in Computer and Information Science (special issue: Proceedings of Machine Intelligence 17),* forthcoming.

[4] S Colton, A Bundy, and T Walsh. Automatic identification of mathematical concepts. In *Machine Learning: Proceedings of the 17th International Conference*, 2000.

[5] Marsha J. Ekstrom Meredith. *Seek-Whence: A Model of Pattern Perception*. PhD thesis, Department of Computer Science, Indiana University, 1987.

[6] S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[7] G Ritchie. Describing verbally expressed humour. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, 2000.

[8] H Simon and A Newell. Heuristic problem solving: The next advance in operations research. *Operations Research*, 6(1), 1958.

A. Puzzles about Animals

```
1. Which is the odd one out?
i. eagle* ii. bat iii. cat iv. dolphin
answer: a produces milk
Disguising Concepts:
a is homeothermic
interestingness=1.5, disguise=2.0, complexity=1
------------------------
2. bat is to cat as dog is to:
i. platypus* ii. lizard iii. trout iv. herring
answer: a produces milk
complexity=1
------------------------
3. Which is the odd one out?
i. eagle ii. eel* iii. bat iv. dog
answer: a is homeothermic
interestingness=1.0, disguise=1.0, complexity=1
------------------------
4. bat is to cat as dog is to:
i. shark ii. dragon iii. eel iv. dolphin*
answer: a is homeothermic
complexity=1
------------------------
5. Which is the odd one out?
i. bat* ii. snake iii. lizard iv. penguin
answer: a produces eggs
interestingness=1.0, disguise=1.0, complexity=1
------------------------
6. dragon is to lizard as snake is to:
```

```
i. dog ii. t_rex* iii. dolphin iv. bat
answer: a produces eggs
complexity=1
------------------------
7. Which is the odd one out?
i. ostrich ii. platypus* iii. eagle iv. penguin
answer: a is a bird
Disguising Concepts:
a is homeothermic
a produces eggs
a has 2 legs
a is homeothermic & a produces eggs
interestingness=3.5, disguise=5.0, complexity=2
------------------------
8. Which is the odd one out?
i. snake ii. herring* iii. lizard iv. dragon
answer: a is a reptile
Disguising Concepts:
a produces eggs
a is covered by scales
interestingness=2.5, disguise=3.0, complexity=2
------------------------
9. crocodile is to dragon as lizard is to:
i. penguin ii. platypus iii. bat iv. turtle*
answer: a is a reptile
complexity=2
------------------------
10. Which is the odd one out?
i. cat ii. platypus iii. bat iv. dolphin*
answer: a is covered by hair
Disguising Concepts:
a produces milk
a is homeothermic
interestingness=2.5, disguise=3.0, complexity=2
------------------------
11. Which is the odd one out?
i. lizard ii. eagle* iii. dragon iv. herring
answer: a is covered by scales
Disguising Concepts:
a produces eggs
interestingness=2.0, disguise=2.0, complexity=2
------------------------
12. crocodile is to dragon as herring is to:
i. penguin ii. eel iii. turtle* iv. bat
answer: a is covered by scales
complexity=2
------------------------
13. Which is the odd one out?
i. snake ii. shark iii. bat* iv. dolphin
answer: a has 0 legs
interestingness=1.5, disguise=1.0, complexity=2
------------------------
14. herring is to shark as snake is to:
i. trout* ii. platypus iii. turtle iv. eagle
answer: a has 0 legs
complexity=2
------------------------
15. Which is the odd one out?
i. penguin ii. ostrich iii. cat* iv. bat
answer: a has 2 legs
Disguising Concepts:
a is homeothermic
interestingness=2.0, disguise=2.0, complexity=2
------------------------
16. bat is to eagle as ostrich is to:
i. snake ii. platypus* iii. herring iv. dragon
answer: a has 2 legs
complexity=2
------------------------
17. Which is the odd one out?
i. dog ii. dragon iii. cat iv. eel*
answer: a has 4 legs
interestingness=1.5, disguise=1.0, complexity=2
------------------------
18. cat is to crocodile as dragon is to:
i. snake ii. penguin iii. dolphin iv. t_rex*
answer: a has 4 legs
complexity=2
------------------------
19. Which is the odd one out?
i. lizard ii. snake iii. dog iv. bat*
answer: a lives in land
interestingness=1.5, disguise=1.0, complexity=2
------------------------
20. cat is to crocodile as dog is to:
i. shark ii. penguin iii. herring iv. ostrich*
answer: a lives in land
complexity=2
------------------------
21. Which is the odd one out?
i. crocodile ii. turtle iii. bat* iv. dolphin
answer: a lives in water
interestingness=1.5, disguise=1.0, complexity=2
------------------------
22. crocodile is to dolphin as eel is to:
i. dragon ii. ostrich iii. t_rex iv. trout*
answer: a lives in water
complexity=2
------------------------
23. eagle is to ostrich as penguin is to:
i. t_rex ii. platypus* iii. dog iv. trout
answer: a is homeothermic & a produces eggs
complexity=3
------------------------
24. eel is to herring as snake is to:
i. dragon ii. platypus iii. t_rex iv. trout*
answer: a produces eggs & a has 0 legs
complexity=4
------------------------
25. Which is the odd one out?
i. dragon ii. bat iii. crocodile iv. dolphin*
answer: 2=|{b: b is a habitat & a lives in b}|
interestingness=2.0, disguise=1.0, complexity=3
------------------------
26. bat is to crocodile as dragon is to:
i. cat ii. eel iii. eagle* iv. dog
answer: 2=|{b: b is a habitat & a lives in b}|
complexity=3
------------------------
27. cat is to dog as eagle is to:
i. lizard ii. eel iii. ostrich* iv. trout
answer: a is homeothermic & a lives in land
complexity=4
------------------------
28. eagle is to ostrich as snake is to:
i. cat ii. turtle iii. penguin iv. t_rex*
answer: a produces eggs & a lives in land
complexity=4
------------------------
29. crocodile is to lizard as snake is to:
i. shark ii. eel iii. bat iv. t_rex*
answer: a is a reptile & a lives in land
complexity=5
------------------------
30. cat is to dog as lizard is to:
i. t_rex* ii. eel iii. platypus iv. bat
answer: a has 4 legs & a lives in land
complexity=5
------------------------
31. eel is to platypus as shark is to:
i. snake ii. eagle iii. turtle* iv. lizard
answer: a produces eggs & a lives in water
complexity=4
------------------------
```

B. Next in Sequence Puzzles

```
1. What’s next in the sequence:
27 2 28 12 29 20 (30)?
answer: a is an integer
Disguising Concepts:
2 is a digit of a
disguise=1.0, complexity=1
------------------------
2. What’s next in the sequence:
4 3 6 6 2 9 (8)?
answer: b=|{c: c|a}|
Starting with n=27
Disguising Concepts:
3|a
disguise=1.0, complexity=2
------------------------
3. What’s next in the sequence:
21 3 24 6 27 9 (30)?
answer: 3|a
Disguising Concepts:
3|a
disguise=1.0, complexity=2
------------------------
4. What’s next in the sequence:
7 11 13 17 19 23 (29)?
answer: 2=|{b: b|a}|
disguise=0.0, complexity=3
------------------------
5. What’s next in the sequence:
2 8 4 4 6 2 (8)?
answer: b=|{c: c|a}| & 2|b
Starting with n=19
disguise=0.0, complexity=4
------------------------
6. What’s next in the sequence:
6 6 4 8 4 6 (8)?
answer: b=|{c: c|a}| & 2|a
Starting with n=9
disguise=0.0, complexity=4
------------------------
7. What’s next in the sequence:
6 4 6 4 8 4 (8)?
answer: b=|{c: c|a}| & 3|a
Starting with n=4
disguise=0.0, complexity=4
------------------------
8. What’s next in the sequence:
5 6 7 8 9 11 (22)?
answer: 1=|{b: b is a digit of a}|
disguise=0.0, complexity=3
------------------------
9. What’s next in the sequence:
19 20 21 23 24 25 (26)?
answer: 2=|{b: b is a digit of a}|
disguise=0.0, complexity=3
------------------------
10. What’s next in the sequence:
21 22 24 25 26 28 (30)?
answer: exists b (b|a & b is a digit of a)
disguise=0.0, complexity=4
------------------------
11. What’s next in the sequence:
2 1 1 0 1 0 (1)?
answer:  b=|{c: c|a & c is a digit of a}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
12. What’s next in the sequence:
3 1 2 1 2 1 (1)?
answer:  b=|{(c d): c*d=a & d|c}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
13. What’s next in the sequence:
16 17 18 19 21 24 (25)?
answer: exists b c (b*c=a & c is a digit of b)
disguise=0.0, complexity=4
------------------------
14. What’s next in the sequence:
2 1 1 1 0 1 (0)?
answer: b=|{(c d): c*d=a & d is a digit of c}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
15. What’s next in the sequence:
7 2 3 3 5 1 (7)?
answer: b=|{(c d): c+d=a & d|c}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
16. What’s next in the sequence:
20 22 23 24 25 26 (27)?
answer: exists b c (b+c=a & c is a digit of b)
disguise=0.0, complexity=4
------------------------
17. What’s next in the sequence:
2 1 3 1 3 1 (2)?
answer: b=|{(c d): c+d=a & d is a digit of c}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
18. What’s next in the sequence:
0 1 1 2 2 2 (2)?
answer: b=|{(c d): c+d=a & d is a digit of a}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
19. What’s next in the sequence:
2 5 2 2 2 3 (2)?
answer: b=|{c: c|a}| & 2=|{d: d|b}|
Starting with n=8
disguise=0.0, complexity=4
------------------------
20. What’s next in the sequence:
6 0 2 0 4 0 (4)?
answer: b=|{c: c|a & 2|c}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
21. What’s next in the sequence:
4 0 0 3 0 0 (4)?
answer: b=|{c: c|a & 3|c}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
22. What’s next in the sequence:
11 12 20 21 23 24 (25)?
answer: exists b (b is a digit of a &
b=|{c: c is a digit of a}|)
disguise=0.0, complexity=4
------------------------
23. What’s next in the sequence:
11 12 12 13 13 14 (14)?
answer: b=|{(c d): c+d=a & exists e (e+d=c)}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
24. What’s next in the sequence:
0 1 2 3 4 5 (6)?
answer: b=|{(c d): c+d=a & exists e f (e+f=d)}|
Starting with n=24
disguise=0.0, complexity=4
------------------------
```