Jacob Kilver
Knowledge Based Artificial Intelligence
Assignment 1
5/29/2016

## Introduction

This assignment describes the means by which the transformations between images can be determined both verbally and visually. These methods can then be used to solve tests of intelligence such as the Raven's Progressive Matrices (RPMs). While such a task can be simple for humans it can be quite a challenge for computers. Humans have an amazing ability to process both verbal and visual data and infer relationships. Computers need to have all these relationships explained; they are not able to infer these relationships as easily as humans. Production systems are one method that computer scientists use to endow computer with some of the intelligence of humans to infer these relationships. The remainder of this paper describes how Production Systems (and its related techniques) can be used to solve RPMs.

## Problem Description

For an example of a RPM see Figure 1. As a human we can easily tell that the correct answer to this problem is answer #2, the solid square. But how do we explain this pattern to a computer, especially when the problems become more difficult, like in Figure 2 which many humans have difficulty answering?

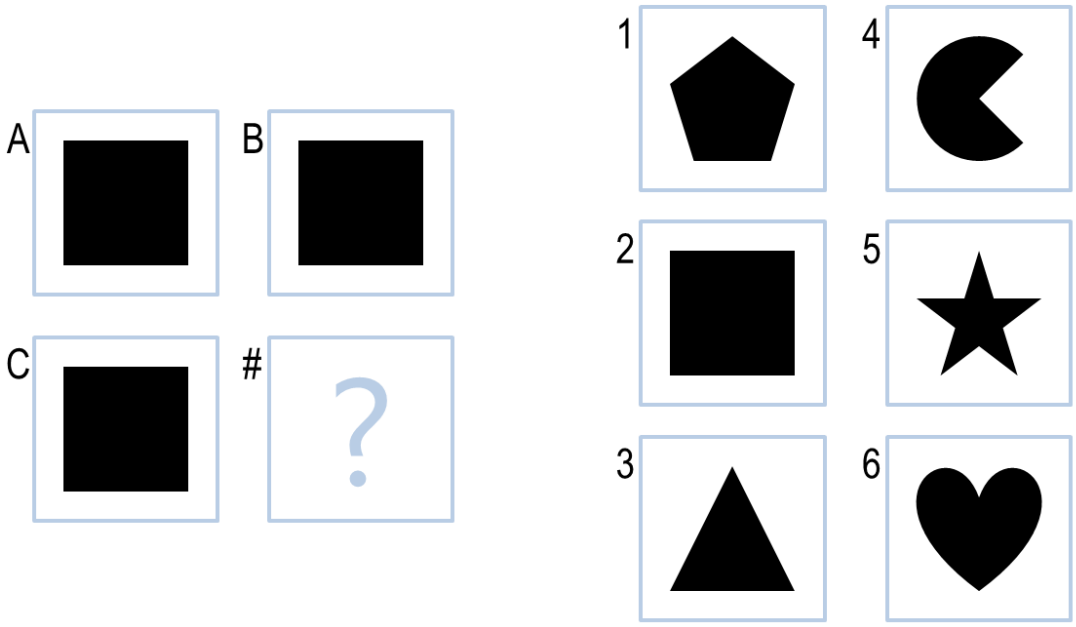


*Figure 1: Example of a basic Raven's Progressive Matrix*

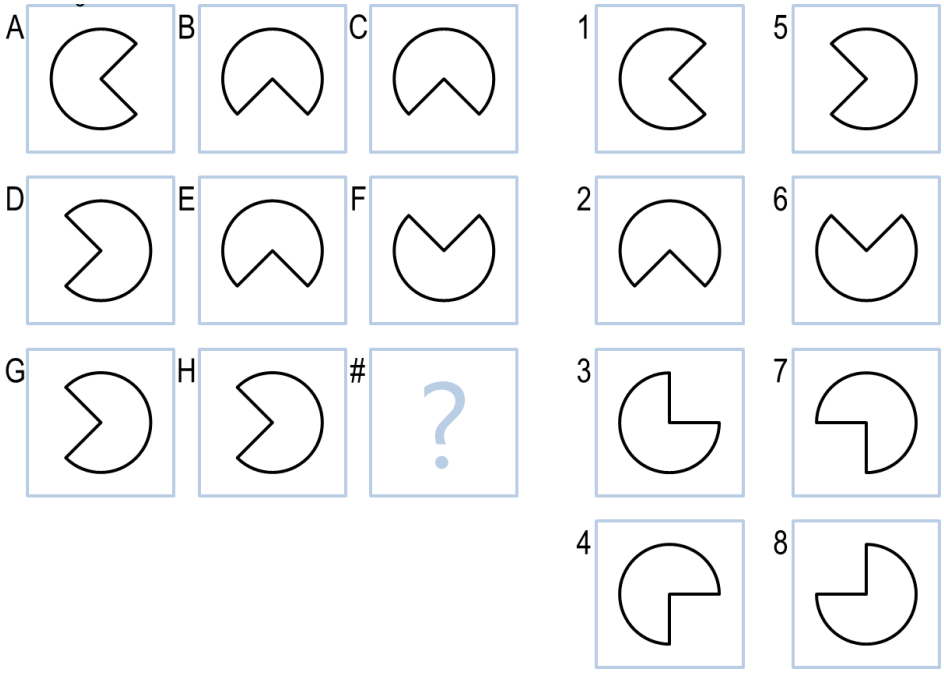


*Figure 2: Example of a challenging Raven's Progressive Matrix*

Here are some of the nuances that the humans can identify that we use to solve these types of problems:

- Shapes
  - Sizes
  - Whether shape is filled
- Relationships between shapes in a single cell
  - Relative locations
  - Relative sizes
- Relationship between shapes in other cells
  - What shapes are different?
  - What shapes are similar?
  - What shapes moved?
  - What shapes are new?
  - What shapes were removed?

Solving these problems with just the visual description can be difficult. It is difficult to take the vast input space of an image and quantify the similarities and differences. Fortunately some RPMs have verbal descriptions that describe the RPMs in a more limited but also more tractable input space. Solving visually requires either many different image transformations to be attempted or some more advanced form of visual reasoning.

## Approach

As mentioned previously, a production system is one tool that can be used to solve the RPMs. One such widely used production system is known as SOAR. A production system is, as the name implies, a system, and like most systems it has an architecture. See Figure 3 for the architecture of SOAR which consists of a long term memory divided into three parts, each with its own set of knowledge, and a working memory for "scratch work." The procedural knowledge is essentially a set of *if-then* rules that

describe how to do certain tasks. Certain percepts in the world lead to certain actions to be taken. The semantic knowledge contains models and views of the world. The episodic knowledge contains particular instances of events, a means of keeping track of what has happened in the past. All these pieces working together can be a very powerful problem solving tool.

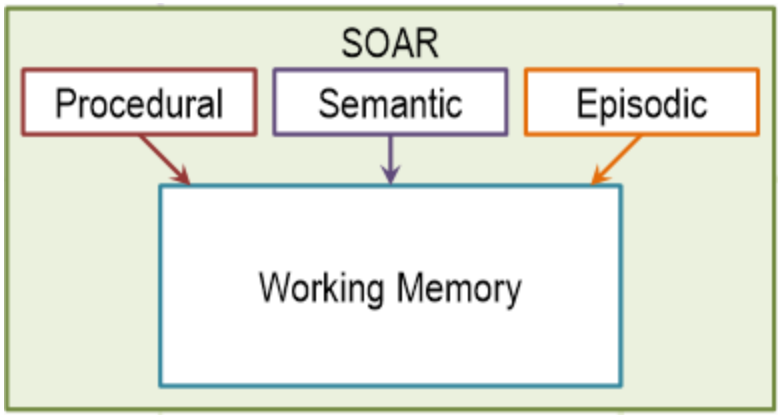


*Figure 3: Production System Architecture*

Applying the production system to RPMs, we will assume that we have semantic knowledge about the RPMs – how the elements of an RPM relate to each other for the various size (2x2 and 3x3). The episodic knowledge would store RPMs that have been encountered in the past along with the solution (if it is available). The procedural knowledge would take information about the RPM and determine what the answer should be. For example, one of the easiest and most important distinctions to make would be the size of the RPM. Solving a 2x2 matrix involves different techniques than solving a 3x3 matrix.

The procedural knowledge is the most interesting portion of the production system. Here is one way it could be used to solve an RPM. (See pseudocode in Listing 1.) The first several rules would involve determining what class of RPM is currently in use, and the class of problem would be stored in the working memory.

Next the RPM could be queried for whether there is a verbal description of the problem. If so, the production system could construct a semantic network and store that in working memory. This semantic network could then be applied to the missing image to solve the RPM.

If there is no verbal description the production system should attempt to solve the problem visually. This would involve a series of rules that would run transformations on the images (rotation, reflection, scaling, image differences, etc.) and compare them. For example, for a 2x2 matrix we know that A is to B as C is to D. Thus, we would need to identify what transformation connects A and B. Once this is known the same transformation can be applied to C. The resultant image can then be compared to the given answer choices and the closest match can be selected as the answer. This method includes some concepts of generate and test. The potential answers are generated and then these solutions are tested and unacceptable answers are pruned.

*Listing 1: Example of procedural knowledge rules for solving RPMs*

| | |
|---|---|
| Rule 1 | If matrix size is 2x2<br>Then write size = 2x2 to working memory |
| Rule 2 | If matrix size is 3x3<br>Then write size = 3x3 to working memory |

| | |
|---|---|
| Rule 3 | If verbal description is available<br>Then write verbal = True to working memory |
| Rule 4 | If verbal == True<br>Then construct semantic network and write to working memory |
| Rule 5 | If verbal == False<br>Then run transformation #1 and write to working memory and<br>write transform_available = true to working memory |
| Rule 6 | If transform_available == True<br>Then compare transformed image and write confidence_rating to working memory |
| Rule 7 | If confidence_rating > minimum_confidence_rating (in working memory)<br>Then write best_answer to working memory |

The example in Listing 1 is to illustrate some of the rules that could be used when solving an RPM. However, in practice it is much easier to use a standard higher level programming language. The example in Listing 1 is more like assembly language than a language like Python. When the production system is actually coded up, the `if` statements would be nested rather than writing to a working memory. The style is different but the outcome is the same.

When solving these problems it is reasonable to include some form of a confidence rating – a measure of how closely the generated answer matches the answer that is selected (1, 2, 3,...). If this confidence value is below a certain threshold or if there are multiple answers that have the same best confidence rating, the production system needs some method of selecting an answer. The process whereby a production system does this is known as chunking. The production system essentially learns a new rule and adds it to its procedural knowledge by searching for previous similar events in its episodic knowledge where the selected answer was correct. Note: this assumes that this system has some knowledge of whether previous answers were correct or not. If the system has no knowledge of whether past answers were correct than chunking cannot be as effective. Because there is no knowledge of whether a previously answered problem was correct or not, a new rule should not be added. In such cases the production system has no recourse but to simply choose the match with the highest confidence rating even if it is below the threshold or to simply pick one of solutions if there are multiple best answers.

## Conclusion

In this paper we have seen what makes RPMs hard to solve for computers – primarily their large input spaces and need to determine relationships. A method for solving RPMs, production systems, was introduced. When combined with semantic networks (if a verbal description is available), the production system can limit the large input space down to a fixed vocabulary which can then be used to solve the RPM. When no verbal description is available a visual approach must be used. Here generate and test imbedded in the production system rules can be used to transform the input images and test how closely they fit the pattern in order to find the best match. In this way the set of relationships is limited to the transformations in the generation phase.

Admittedly the semantic network details were left out here since this paper focused on production systems. Additionally the specifics for generate and test, namely the type and number of transformations that should be attempted, were not included, again because this paper focused on the production system portion. These details will be available in the project source code. The point to emphasize now is that production systems are a powerful tool especially when coupled with other artificial intelligence methods.