**Solving the Raven’s Progressive Matrices through Frames and Case-based Reasoning**

Raven’s Progressive Matrices is known to be a very valuable test of intelligence. It involves mapping the transformations of visual representations (often shapes) of 2x2 and 3x3 matrices. For example note in Figure 1 the box in the bottom right hand corner is missing. Let’s denote this box as D. The solution to the Raven’s Matrices problem is to identify the box that fits in D such that the transformation from A to B is equivalent to transformation from C to D and the transformation from A to C is the same as that from B to D.

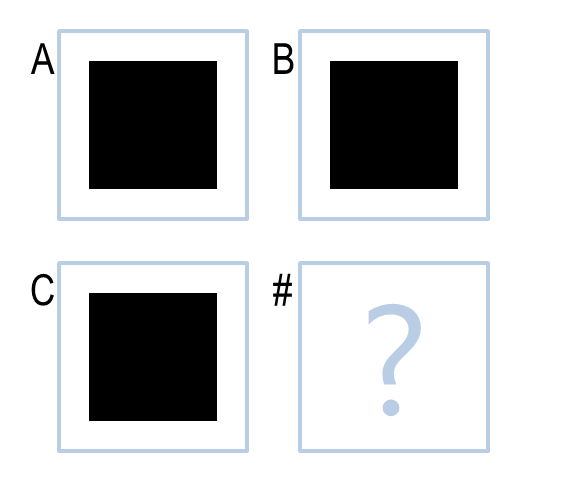


Figure 1: A 2x2 Raven’s Progressive Matrices problem.

It is fairly obvious then that there can be a number of possible transformations that can produce the image in D. For instance the transformation from A to B can be described as “unchanged” or “rotated 90 degrees clockwise”. Both these will be valid. Imagine now that C was actually a filled triangle and one of the options is a triangle “rotated 90 degrees clockwise”. Which transformation would produce the solution? This is one of the challenges. With more complex visual representations, the enumerations of transformations can become large especially when you are dealing with 3x3 matrices which are also part of the test. In the 3x3 case, in addition to the horizontal and vertical transformations, diagonal transformations also need to be satisfied. When we are presented with multiple transformations it is necessary to identify the best fit since the transformation across vertical, horizontal and/or diagonal direction could very well vary.

Another challenge with this problem is to programmatically identify the objects in each map. Instead of performing complicated and resource intensive image processing to determine the objects and their attributes, this author will read a csv file consisting of this information. This will then be translated to frames which represent the objects within the boxes in an object oriented way. An important decision to make here is how to treat objects that can be considered as one or two different objects which are either overlapping or touching as is the case shown in Figure 2. This author decided to treat these as two distinct objects.

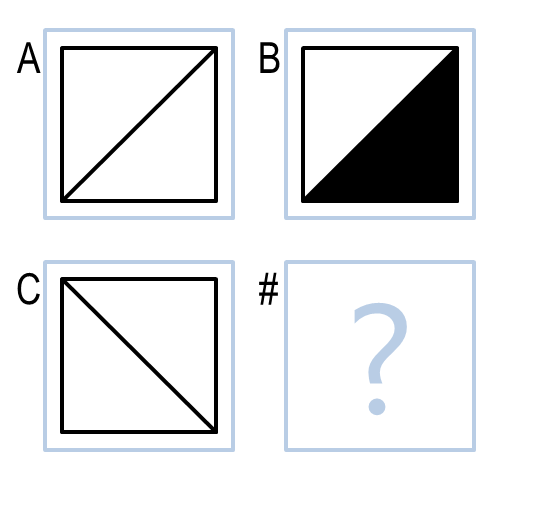
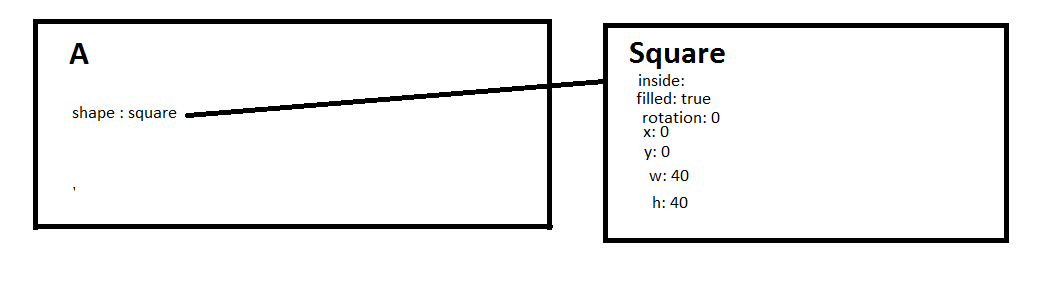


Figure 2: A 2x2 Raven’s matrices with two triangles that are touching. One may also consider this as being a single object.

Based on these, new frames will be generated to represent the transformations between boxes.

Here is an example of a Frame A and Square found in Figure 1.

Figure 3: The frame for box A and the square object in box A found in Figure 1.

The challenge involved in computing the transformations is associated with correspondence. Which shape in A corresponds to which shape in B? We can achieve some correspondence by ordering the frames associated with the objects in terms of relative position, i.e. outside, inside, above, below, left or right. We can then formulate the transformations by comparing each of the object frames for A with those of B in the sorted order. One of the challenges here, is identifying which object if any were “deleted”. It is difficult to do this deterministically however it is believed that the last object in the sorted collection would likely be the one that is flagged as being “deleted”. In the event this isn’t true the case-based reasoning approach proposed next should discover the right object.

Once we have the transformations we can then store the objects and transformations as a single case. This will be our working memory which will then be used to compare with the transformation generated between B and D (the option) and C and D for the 2x2 matrices. If the transformations aren’t a match then we will use the stored transformation to generate a new D. We will then compare this new D with the option that was tested. If there is a match we keep the stored transformation and then accept the option that was tested as the solution, otherwise we will use the transformation for the option to generate a new B and C and compare with the frames stored as a case. If we have a match we will accept the option as the answer and update the case with the new transformation. This process will be repeated for each option, until we arrive at the solution.

For the 3x3 case, we have one more horizontal, vertical and an additional diagonal transformation that will allow us to reason better. The algorithm should conceptually work for the 3x3 matrices as well with additional constraints. We will have to ensure the first and second row, first and second column transformations are consistent or match and then use that in generating the transformation for the third row. We can then validate the diagonal case.

In conclusion, this report proposes a mechanism for using frames and case-based reasoning to solve 2x2 Raven’s matrices problems and hints at how the algorithm can be used to solve 3x3 matrices. There are a number of challenges including generating frames from visual representations, identifying transformations, correspondence of objects within visual representations, and adapting transformations to be more specialized. Ways to address these challenges were presented.