Jacob Kilver

Knowledge Based AI

Project 1 Reflection

12 June 2016

# Introduction

This project set out to solve 2x2 Raven’s Progressive Matrix (RPM) problems. A production system was used as a tool to solve these matrices. While verbal descriptions were provided, this project attempted to solve the problems visually. While this method proved trivial for solving some RPMs, it proved quite difficult for others.

# Theory of Operation

A production system architecture blended with generate and test was used as the basis for project. While the production system in class contained several types of knowledge, the procedural knowledge subsystem was the focus here. Based on different “percepts” of the world (provided through the RPM data structure) various “actions” were taken (i.e., answers were generated) and then tested against the available answers.

The actions in this case were image transformations. A few operations were trivial: no transformation, rotation 90° clockwise, rotation 90° counter clockwise, reflection across the vertical axis, and reflection across the horizontal axis. Image transformations beyond this became less trivial mostly due to the complexity of some of the transforms. This portion of the solution focused on the differences between images. If none of the simple transforms above worked, the differences between pairs of images (A and B, A and C) were found. These differences were then applied to the remaining image (C + (A-B), B + (A-C)) and then compared to the possible answers.

# Implementation

Python was the language of choice for this project. Since visual solutions were generated here, the Python package PIL (<http://pillow.readthedocs.io/en/3.2.x/index.html>) was used to manipulate images. This package contained built in methods for rotation and reflection which were used for the simple transformations discussed above. For transformations beyond these, combinations of operators were used. For instance, image difference was found using ImageChops.difference and images were added using ImageChops.add.

Implementing the simple transformations was rather straightforward: manipulate image A and see if this manipulated image matches B, and similarly for A to C. Matches were determined by finding the percentage of pixels that differed between two images. If the percentage was above a certain threshold than a match was found. If multiple answers were found, then the one with the highest percentage of similarity was chosen and the transformation corresponding to that answer was saved for later use.

The one problematic aspect here was in applying the selected transforms. Since there were two transformations (one from A to B and one from A to C), they could be combined in several different ways. For this project only single transformations were used, i.e., only A to B applied to C or only A to C applied to B. From here it was expected that the answer with the greatest confidence (highest percentage of matching pixels) would result in the best answer. However this was not the case due to slight imperfections between the transformed images and the answers.

As an example, see Figure 1. In this case the transformation between A and B was a reflection along the vertical axis. A human can tell that the transformation from A to C is a reflection along the horizontal axis. Unfortunately getting the agent to distinguish between the filled and unfilled shape was not possible in this project, so the A-C transform resulted in a “no match.” However, due to the implementation details the guess that was returned was simply image A. When the guesses were compared to the answers, image A exactly matched answer 1. The second guess very closely matched answer 6 (the correct answer), but not 100%. To avoid selecting answer 1, “fuzzy logic” was used to equate these two answers as both having the same confidence. The guess of image A could then be eliminated because the transformation associated with that guess was not valid.

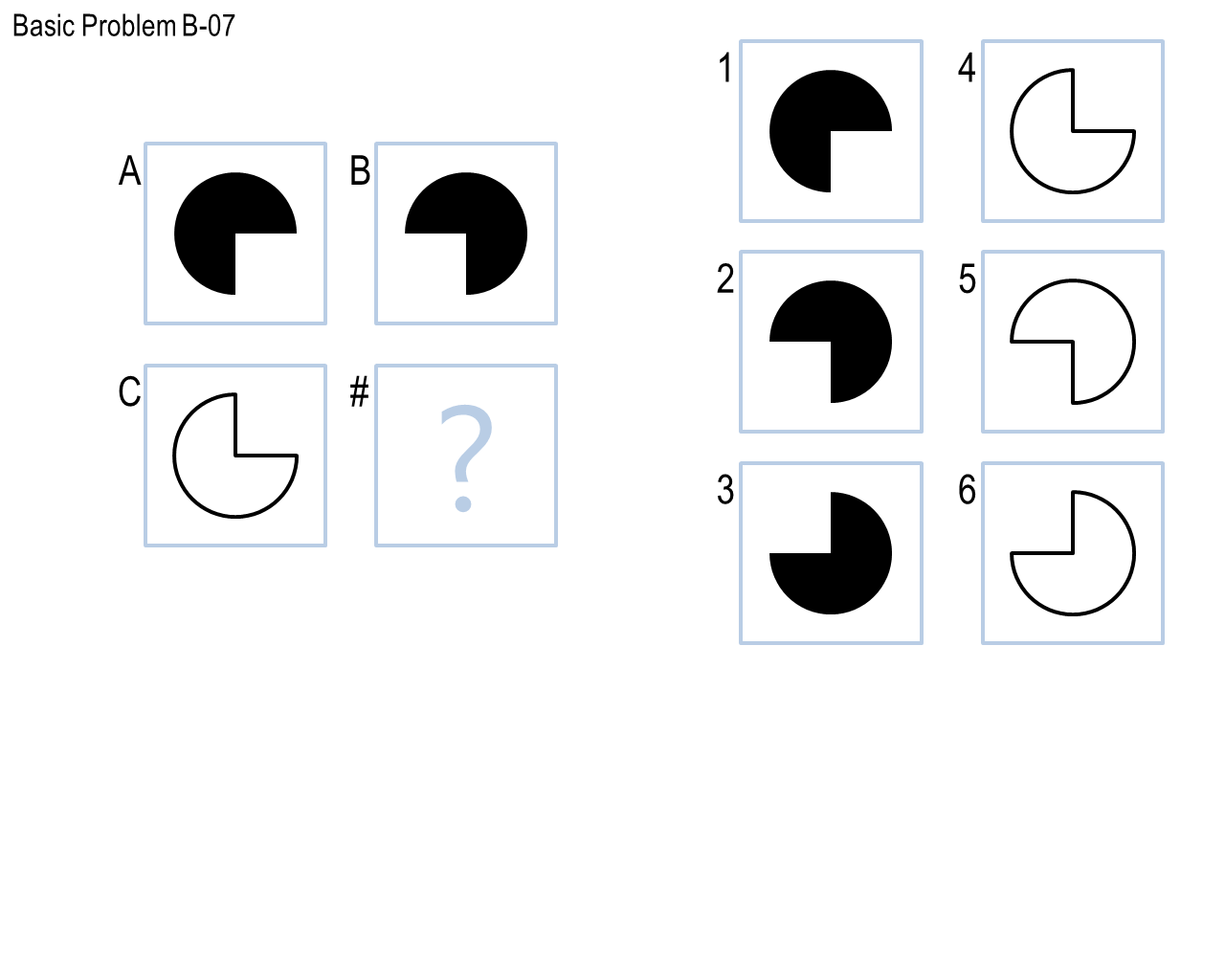


Figure : Raven's Progressive Matrix where "fuzzy logic" was used

The more difficult task was finding answers when simple transformations resulted in no matches. Take Figure 1 below. For a human this is a trivial task – the answer is clearly 1, a hexagon without the black diamond in the center.

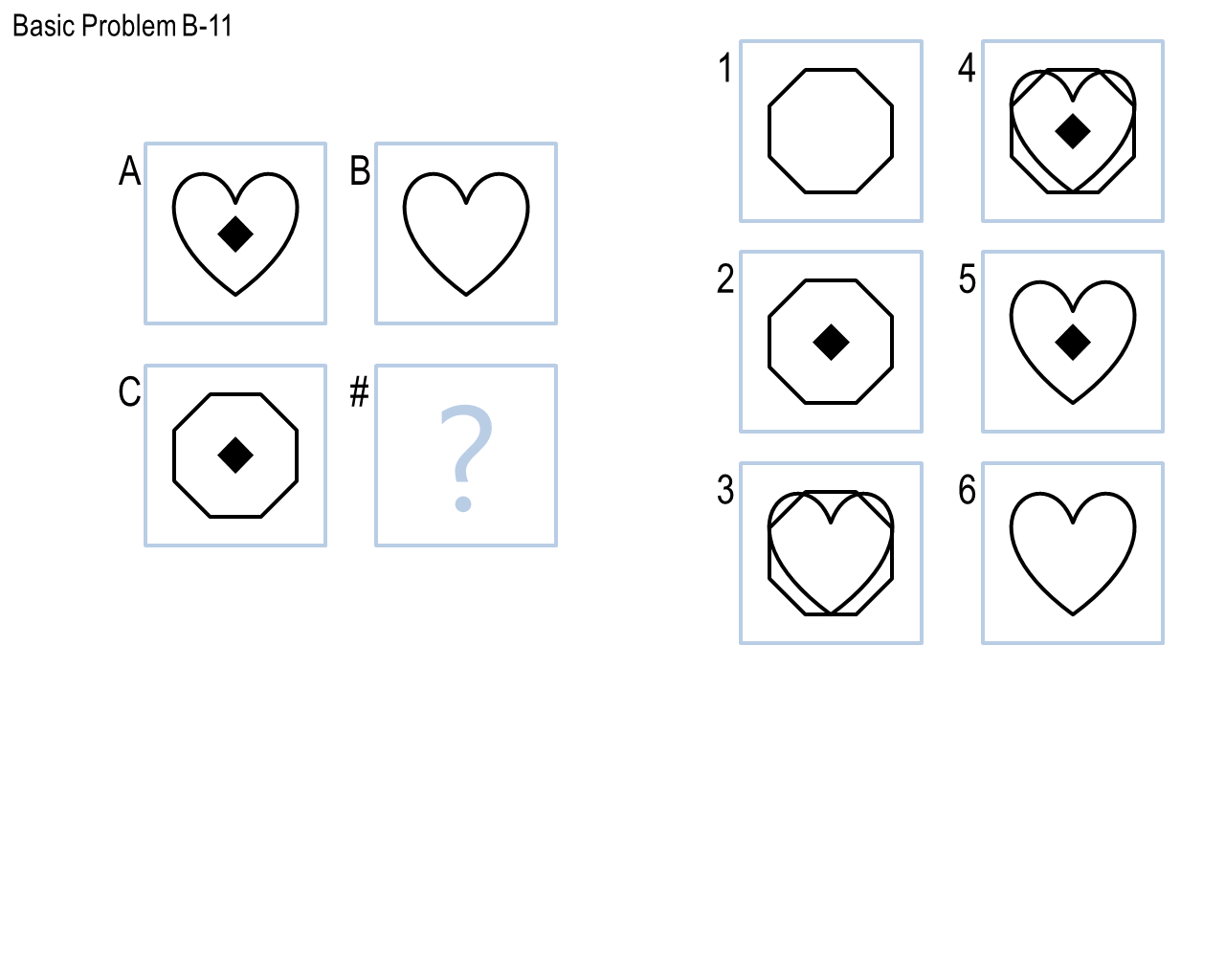


Figure : Raven's Progressive Matrix where simple transformations result in no useful information

# Results and Discussion

The implementation was tested several different ways. The first test was a set of in-sample RPMs provided by the instructor. These results can be seen in Table 1 and fig 1 below. DISCUSS RESULTS HERE.

Pie chart of correct vs. incorrect vs. not answered

The second test was using out-of-sample RPMs that were also provided by the instructor but that could only be tested a limited number of times. The latest results can be seen in \_\_\_ below. (Note: The answers were randomly ordered, so a table was not useful in this case.) DISCUSS RESULTS

# Conclusions and Future Work