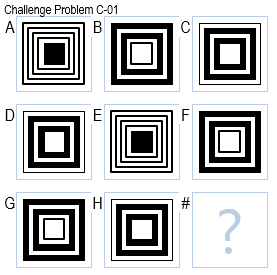
CS 7637: Project 3 Journal

Zachary Todd  
ztodd3@gatech.edu

# 1 Introduction

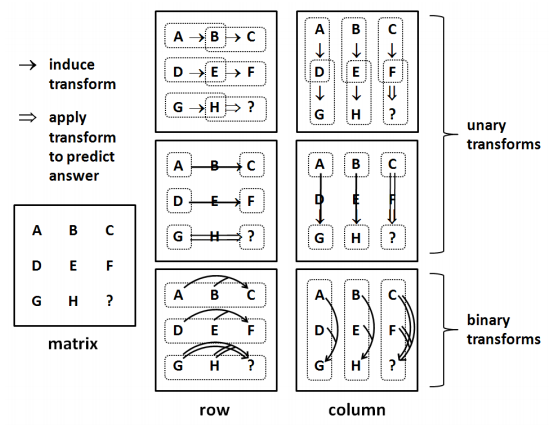
This paper will outline the third iteration of developing a KBAI agent capable of solving Raven’s Progressive Matrices (RPMs). In this iteration, I will build on the agent I have already developed. Iteration 2 was able to solve visual 2x2 and 3x3 RPMs by identifying transformations and patterns between images. The agent used techniques of unary transformation detection, pixel addition/subtraction, and shape detection. At submission, the agent was able to solve 17/24 on the basic and test problems and 14/24 for the Raven’s and challenge problems for problem set C. It also got 14 of these problems incorrect and skipped 3. The ultimate goal of this iteration is to implement a fractal approach by identifying individual shapes in an image and looking for transformations of these shapes over images.



***Figure 1—***For clarity, I will refer to different Images in a 3x3 Raven’s Problem as A, B, C, D, E, F, G, H, and # as shown above for the rest of this paper. I will also refer to possible answers as 1-8.

## 1.1 Initial approach

Before getting into fractal transformation detection, I plan on adding binary transformation detection to the agent’s toolbelt. Binary transformations are those which combine information from two images to produce a third (Kunda). Figure 2 shows how images such as A and B can be combined to produce image C. These combinations typically occur as logical AND, OR, XOR, and Subtraction and will be capable of solving problems like Basic Problem E-05.



***Figure 2—***By detecting patterns in the given problem, the agent can make a guess at the correct answer.

I will then move on to a fractal approach using connected component labelling (CCL) to identify shapes in an image. A fractal approach is just one which breaks an image down into sub-images and looks at the transformations associated with these smaller parts rather than the whole image at once (Kunda). Each of these shapes will have a frame associated with it. I will then use analogical reasoning to pair shapes between images and identify what transformations have taken place.

# 2 Submission #1 (2020-07-09 00:09:58 UTC) 608.467s

In this submission, I mainly cleaned up code from the last iteration and changed the final decision metric the agent uses. The agent will now only choose an answer if there are no other answers with a close confidence level. For instance, if the agent is 98% sure answer 2 is correct and 97% sure answer 3 is correct, it would choose to skip the problem. Additionally, it will not choose an answer unless it is at least 85% sure the answer is right.

The intent of this change is to get an agent that does not guess when it doesn’t have a high confidence that the answer is correct. Using this method will highlight problems the agent struggles with and will ensure the agent is not just getting lucky with its guesses.

The agent’s performance on problem set C very minorly decreased from 31/48 to 27/48 but the number of incorrect answers drastically decreased from 14 to 5. This shows that we don’t have to sacrifice much performance in order to get an agent that will only guess when it is confident in its answer. I have also displayed the agent’s performance on problem sets D and E in Figure 3.

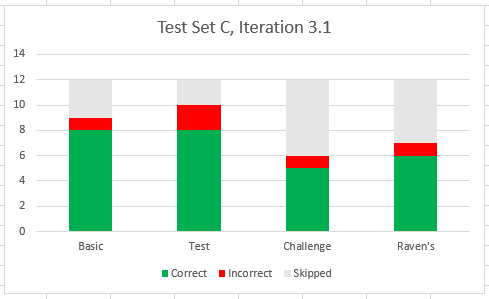


Figure 2—Results of the iteration 3.1 agent against sets C.

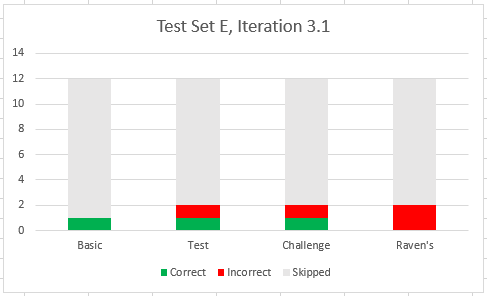
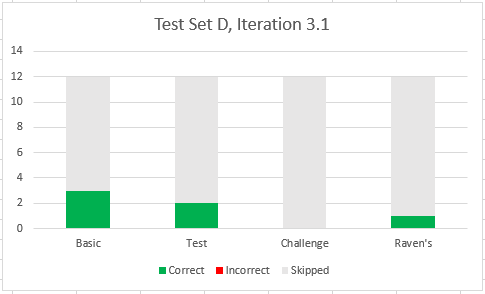


Figure 3—Results of the iteration 3.1 agent against set D and E.

It is interesting to note that the agent did not guess incorrectly on problem set D and had a 100% confidence rating for Basic Problem D-01. I will continue to have the agent strive for a low number of incorrect guesses as this iteration continues until I feel confident with its performance. Once it can determine what questions it is not comfortable answering, I will have it guess the answer for which it has highest confidence. This is more along the lines of human performance as I will explain later in the paper.

# 3 Submission #2 (2020-07-14 16:11:43 UTC) 640.449s

The agent now has an image comparison threshold of 2%. This is reduced from its previous 7% to further increase the accuracy of the agent. This percentage is used to check whether a transformation has occurred as well as to compare generated guesses to the possible answers.

Additionally, I added a binary transform check. The agent can now check for union, subtraction, intersection, and XOR between rows and columns.

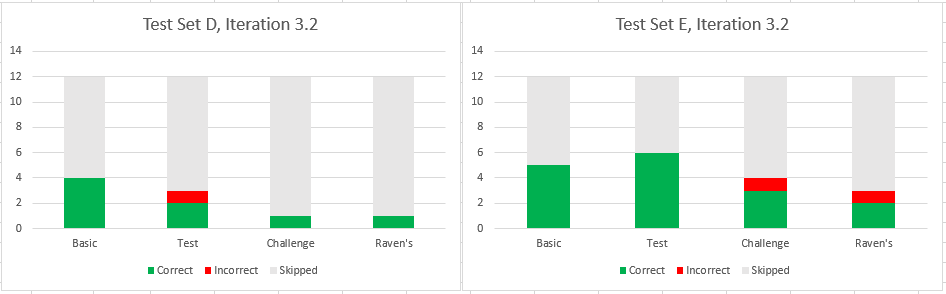


Figure 4—Results of the iteration 3.2 agent against set D and E.

# 3 Submission #2 (2020-07-14 16:11:43 UTC) 640.449s

added the first version of fractal image comparison. The agent now does all of its previous checks and if it cannot determine a good answer, will break the image apart into sub-images. These sub-images consist only of a single connected blob of black pixels, differentiated through Connected Component Labeling. Figure 5 shows how one such RPM image may be broken up into sub-images.

The agent currently only looks at the sub-image which contains a black pixel in the center of the image. By analogical reasoning, all of these center shapes *should* be relatable in the RPM. There are a few cases where this is not true, but the agent can recognize these cases most of the time through its other comparisons. In problems where there is not a colored pixel in the center, this check does not occur. The agent will then compare these center images to find unary patterns.

# XXXX References

1. “Connected-Component Labeling.” Wikipedia, Wikimedia Foundation, 15 Jan. 2020, en.wikipedia.org/wiki/Connected-component\_labeling.
2. Kunda, Maithilee, et al. “A Computational Model for Solving Problems from the Raven’s Progressive Matrices Intelligence Test Using Iconic Visual Representations.” *Cognitive Systems Research*, vol. 22-23, 2013, pp. 47–66., doi:10.1016/j.cogsys.2012.08.001.