Brad Ware
bware8@gatech.edu
Assignment 1

Generate & Test, Means-End Analysis, Problem Reduction

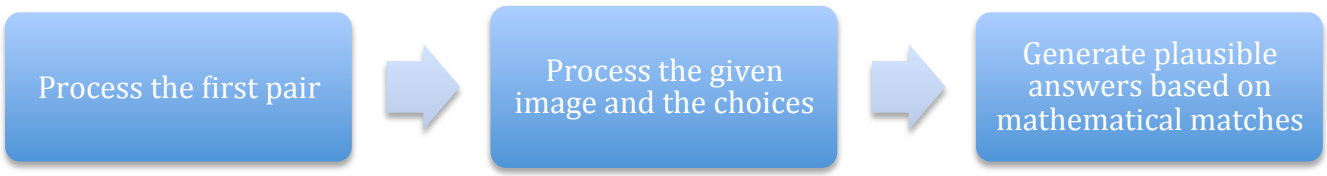
So to build a knowledge base that can solve Raven's Progressive Matrices I have decided to use a multitude of strategies for different steps in the process. Ultimately in this problem, the example images that are paired together are the most important information we are given. So to make this challenge easier, I want to simplify it through Problem Reduction. We know that we need to understand what changes occurred in the example, so by first analyzing that we are gaining knowledge to make a more educated decision. By doing this, I am assuming that my machine can interpret images and identify different object shapes and sizes. From there, I want to pre-process the example pair by giving numerical ratings for the **change** from the first image to the second. These ratings will be for the following categories: shape type, shape size, and shape location. By analyzing the changes from the first image to the second in these following categories, we are able to estimate a numerical difference based on percentage for (initial – final) / initial.

After this, we have a clear picture on what we should be looking for in the answer, but still have no idea what the actual image contains that we're basing our decision from. Means-End analysis will be a useful strategy from here, as we're trying to understand how to reach our goal. The same pre-processing step on analyzing the change in the first two example images will be applied here as well, except now we're just recording the number of shapes, the type, and the size on the screen. From there, we now have a more educated picture on how we should strategize in choosing a pair. Now, we can go through every option of image we have to choose and provide a similar processing calculating the size, type, and location of the objects that the image contains.

Finally, my last strategy will be to Generate and Test. However, we now can generate images that we find to be a more mathematically sound match for it's pair. Because we know the mathematical difference between the example images through the object type, size, and location, we can apply a simulated version to the image we're trying to find a match. Then from there, mathematically pick the highest rated choice we have to find the differences we're expected to calculate. We can keep generating new choices based on the mathematical similarities it has to our predicted answer, because we have already processed our given image. These three strategies require image processing but after that, the shape we're looking for should be found quite easily. I call the last step generate and test, but it should clearly be a simple choice after performing calculations on the choices. Basically, we

are providing intelligence for our generations by performing these pre-processing steps.

Here's a flow chart to detail this process:



Here's some pseudo code to simulate solving this problem:

Reference:

process(image) – takes in an image and then mathematically gives it a rating based on it's objects size, type, and location and returns the stats in a wrapper object.

compare(imageAStats, imageBStats) – takes in two image wrapper objects with their ratings and compares the difference.

```
ravensProgressiveMatrices(imageMatch1, imageMatch2, imageA, totalChoices) {

//Sections resemble breaking down the problem into smaller steps

        image1Object = process(imageMatch1)
        image2Object = process(imageMatch2)
        imageAObject = process(imageA)


        stats = compare(image1Object, image2Object)

        for image i in totalChoices:
                imageCurrObject= process(i)
                currStats = compare(imageAObject, imageCurrObject)
                abs(diff) = stats – currStats
                store diff and object in max heap or list

        imageAnswer = pop off max heap/structure
```

```
    return imageAnswer
}
```

This algorithm runs in *O(n)* as you are processing at most n total images. This shouldn't be very computationally expensive as in our examples in class we were given around 5 choices, which means less than 10 images will need to be processed overall. However, the only downfall is that processing these images could be very computationally expensive. But in this scenario, I am assuming that we have reasonable technology that can detect pixels formed in a particular object, the location on the images, and the total area that these object encompass. The knowledge base for identifying an object type will be from square, rectangles, triangle, pyramid, trapezoid, and pentagon.

This is a high-level overview for how I plan to solve Raven's Progressive Matrices.