Amol Parikh
amolparikh21@gatech.edu
September 13, 2014

Project 1

For project 1 we have to create an agent that is able to solve Raven's Matrices. To help the agent to so I have made the following assumptions:

1) In the figure A, if an object is labeled X which is a particular shape, there has to be an object with the same label X in C.

2) In figure A, if an object has a particular shape, the same object type in B will also be present with the same shape but with different attributes.

3) We assume that there is a object X in figure C which has the same attributes as object X in figures A.

How the agent works?

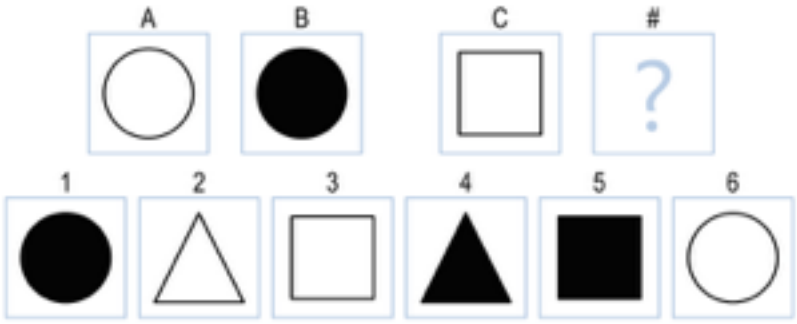
The agent initially read the text file and identifies a figure in A (namely X, Y, Z etc.). It then stores the shape of the object in a variable.(E.g circle, square etc.). Then it searches through all the objects in B till it finds an object with the shape as the one on A. So helps in case there is a square labeled X in A its corresponding mapping is to a square labeled Z in B. It then stores all the attributes and attribute names of that particular object in two arrays namely battr and bname.

After doing so, in figure C, the agent searches for the object name that is selected in A (namely X or Y or Z). According to the assumption listed above it will have the same attributes as the object in figure A expect the shape which may be different. In a loop which iterates 6 times (because we have 6 option to choose from), we search through each of the figures in each option. In each figure we search through all objects of the figure to find an object that has the same

shape as the object selected in C. For that particular object then we get the number of attributes and compare the name and the value of each attribute to the arrays bname and battr. So what happens here is that for each attribute of the object selected from the figures in each options, it first compares the name of the attribute in the array with that of the selected figure. It then compares the value of that particular attribute. If it is the same, a flag is incremented. It does so for all the attributes of the selected figures and if all the names as well as the values of the attributes match, the value of the flag would be equal to the number of attributes of that particular figure. Also a new string is created which stores the values of the correct option. Thus it eliminates those options that do not have the correct mapping for that particular figure. So in case when the next figure in A is selected, those option are never evaluated.

This entire process is repeated for all the figures in A. At the end of the iteration, if the flag is equal to the total number of attributes in all selected figures, the option which it was currently iterating will be displayed as the answer.

How the agent comes to some correct answers?



Taking the above problem as an example we first have the label X for the object in A. The shape of X is a circle. We then search for an object with the shape circle in B. Here we find that the object with a shape circle has a label X as well. We then store the name of the attributes of X i.e fill in an array and the value of the attribute of X i.e yes in another array. The in C we iterate through all the attributes of the object X. During each iteration we compare the name the attribute with that in the array with all name stored. If it finds a match it then checks if the value

of that attribute is the same as that stored in the array. If it also satisfies that condition then, flag is incremented. If the value of the flag is same as that of the total number of attributes of the object that means that all attributes in the option and B are same. Hence it is most closet match.

Where the agent fails and why?

The agent fails in problems where a particular object is deleted in the corresponding figure from A to B. This because the agent just checks whether all the attributes of B are present in the option so at time seven though the second object is deleted in B it checks them and just matches it to the first object in the option. This creates two answers in the output. (E.g. The output for 2X1 Basic Problem 7). Also when the number of attributes in B is not the same as the number of attributes as the object in the selected option an array out of bounds exception. Here instead of just assuming that the number of attributes of both B and the option selected are the same we can check the attribute name individually thus eliminating this error.

Future Improvements

The major improvement i feel is needed in my agent is the ability to identify that a particular object in a figure has been deleted in the figure corresponding to the semantic network. Also another major aspect that the agent is not able to do is learn from its mistakes. So a method by which, if it gets an incorrect answer the agent can retrospect as to how to get to the correct answer and use that knowledge while solving future problems is what i would envision a good agent that solves Ravens Problems to do.