Amol Parikh
amolparikh21@gatech.edu
October 5, 2014

Project 2

Reasoning the agent uses

The agent that i have designed uses a kind of a generate and test method with problem reduction. It first of all identifies a transition from A to B. With each transition is generates a similar transition on C to get an intermediate formation of D. It iterates through each of the transitions between A and B thus generating D step by step. However, there are some cases in which D is not correctly generated. For such cases, the transitions from A to C are iterated over and similar transitions are generated over B to get D. This also generates an intermediate steps of D. Comparing this D to the one generated by comparing A and B we get a final formation of D. Now testing all the given options with the D that we have generated, we can get the correct answer. This is an example of a smart generator and dumb tester.

How the agent comes to some of its correct answers

The first compares the first attribute of the first figure in A. It then takes the value and searches for a similar value in B. If found, it takes the first attribute of the first figure in C and assigns it to D. Thus the transition from A to B is mimicked in C to D. In this way it iterates through each and every figure in A and B and mimics the operation from C to D. This works only as long as the values of A are equal to that of B. If not equal then we check the name of the attributes. If the fill attributes changes from 'yes' to 'no' in A to B we do the same to generate D. Similarly we check for shape. If the shape in A and B is different, we check whether A and C have the same shape, we set the shape in D as the open in D. So the transition from A to B is the same as one from C to D. Similarly we check the size attribute and set appropriate values of D.

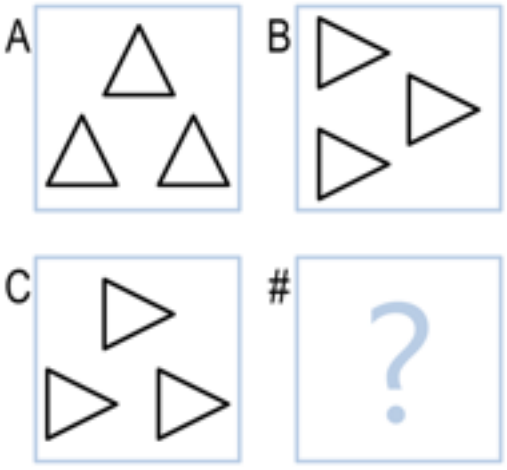
In case of the angle attribute, we calculate the difference between the angles if figure A and that of figure B. Then the same difference is subtracted from figure in C to generate D. However if the either angle A or angle B is greater than 180 , we subtract 180 and then calculate the difference. Also of the final answer is greater than 360 we subtract 360. Another condition that we have checked is if the shape of the figure is a square or circle If the shape is a circle then the angle doesn't matter and we set it to 0. In case it is a square then we consider that angle%90 = 0.In case of attributes like overlaps, left-of, above etc, we check the figure number which is mentioned in the attribute. (Eg if it is above B it is read as above 2 as B is the second figure in A). We then iterate over all the objects in all figures and generate D in such a way that 2 is substituted as the 2 object in that figure.

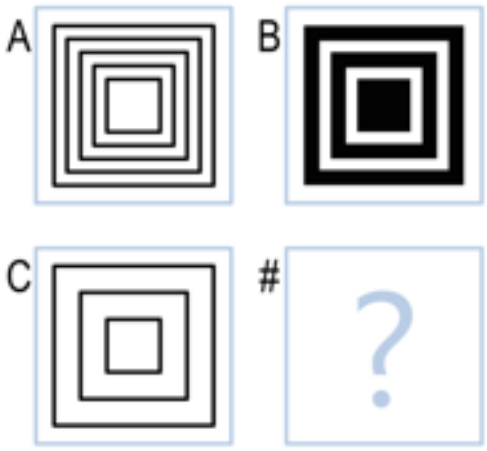
The D then generated, is then checked against each option and the correct option is chosen.

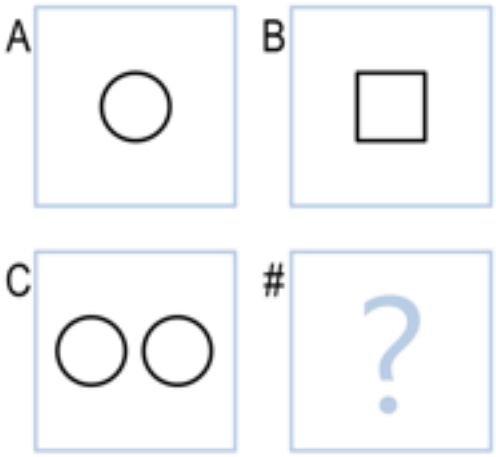
Why your agent makes some of the mistakes it does and how it can be improved

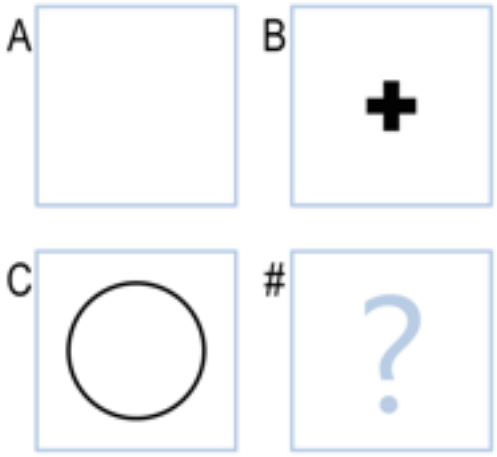
In the implementation i am unable to check the transitions from A to C and generates D based on these transitions being applied on B . Because of this if A and C have single figure transitioning to multiple figures, the output generated has only a single figure. Also in case A is empty and B isn't, then no transitions from C to D are generated as C is not empty. So if we compare A to C and generate D based on them, we could arrive at the correct answer. I have traced my algorithm along these lines and I eliminate most of these errors. Also if certain images get deleted from A to B, it cannot deal with that while generating D. Also it cannot deal with transitions that start with the left-of or above and end up with completely different ones like inside or below. In these cases, it is unable to track the transitions happening from the figure in A to the one in B as neither the name of the object nor its shape are the same or the shape is the same but its orientation is completely different.

Examples of problems in which the agent gives incorrect answers:









The agent is quite efficient in terms of time complexity and give the output almost immediately. This is because there is only 1 file reads. The rest is stored in memory in an array. So the read /write time is much faster resulting in a quick output.

Relationship between your agent and human cognition

The agent is quite similar to human cognition. This is because when we try ti solve such questions what we do it pick a transition between A and B and then apply it to C to generate D. However the difference is that we simultaneously compared A,C to B,D. This isn't part of the agent that i have created. Another similarity is the problem reduction part. In this case, it reduces the problem to individual transitions which is what we also do to solve ravens matrices.