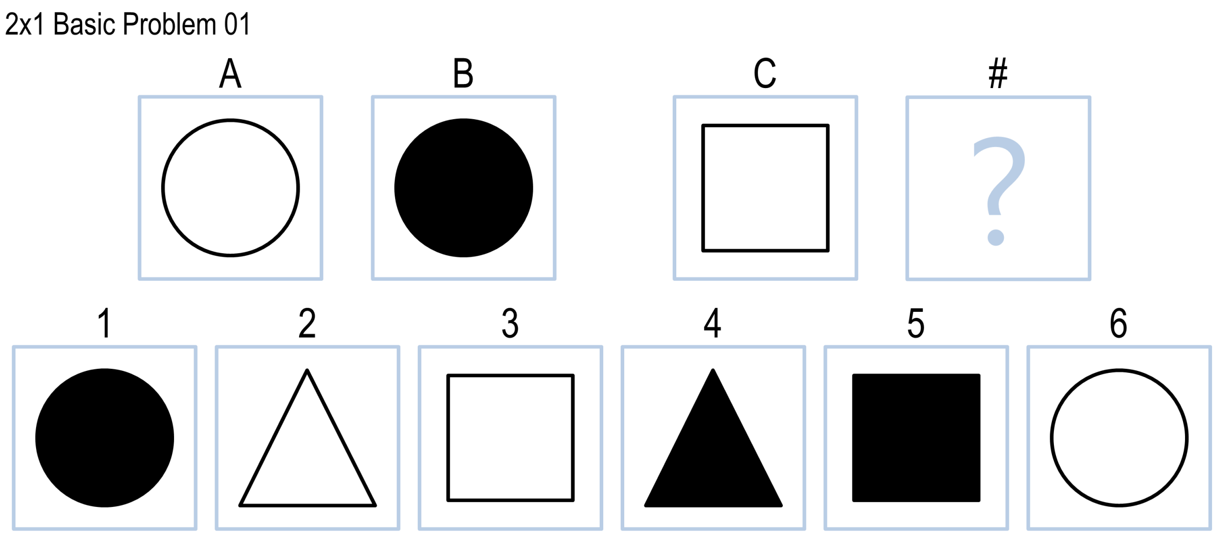
Brad Ware
bware8@gatech.edu
Assignment 3

Using Frames to solve Raven's progressive matrices

When trying to understand frames as knowledge representations, I always go back to my days in 1331 when I was learning Java, but more specifically Objet-Oriented Programming. The hardest part for me was comprehending how objects in the physical world are represented through methods, variables, and inheritance between classes. These translate to actions, descriptions, and relationships when having to describe everyday objects to someone. So when I think in terms of frames, I almost envision them as a class in Java representing an object: they have a name or classification, slots that are categorizing the frame, and fillers that answer each slot. Slots can even reference other frames as their fillers, which is directly related to inheritance in Java, or using other classes as variables in the class you are defining.

Frames that are used to represent knowledge usually are a thematic role, which describes how agents participate in an action. The agent is the objet causing the action, and he can have a co-agent that is "with" him while the action is occurring. The beneficiary is the object for who the action was performed, and the trickiest part to analyze is the thematic object: the subject or reason for why the thematic event occurred. To solve Raven's progressive matrices using frames, one could argue to categorize different figures and then try to match the right answer choice with C using that knowledge base. However, in a way we are already provided a frame of each answer choice, as they contain objects that have attributes with name value pairs that can be used. So because we are already given figures A, B, and C as well as the answer choices in a frame like object, we can gain inference about the problem in another way.

To implement a solution for solving Raven's progressive matrices using frames, I would argue to leverage them for learning, and using learning to decipher between multiple answer choices in tough problems. Because problem reduction techniques can sometimes not be very effective in more difficult problems, allowing your agent to learn from similar problems in the past can be a great way to choose a viable answer choice. Because we are given a function such as checkAnswer( ) which allows us to store the right answer to the problem, we simply just need to match the current problem with a similar one. This can be done using frame comparison. For example, if given this problem:



A viable frame to represent it may be:

{

Problem 1

Type: basic problem
A: figureA)
B: figure(B)
C: figure(C)
NumShapesEqual: true
AmountShapesAnswerChoices: 1
Answer: 5
ABDiff: 1

}

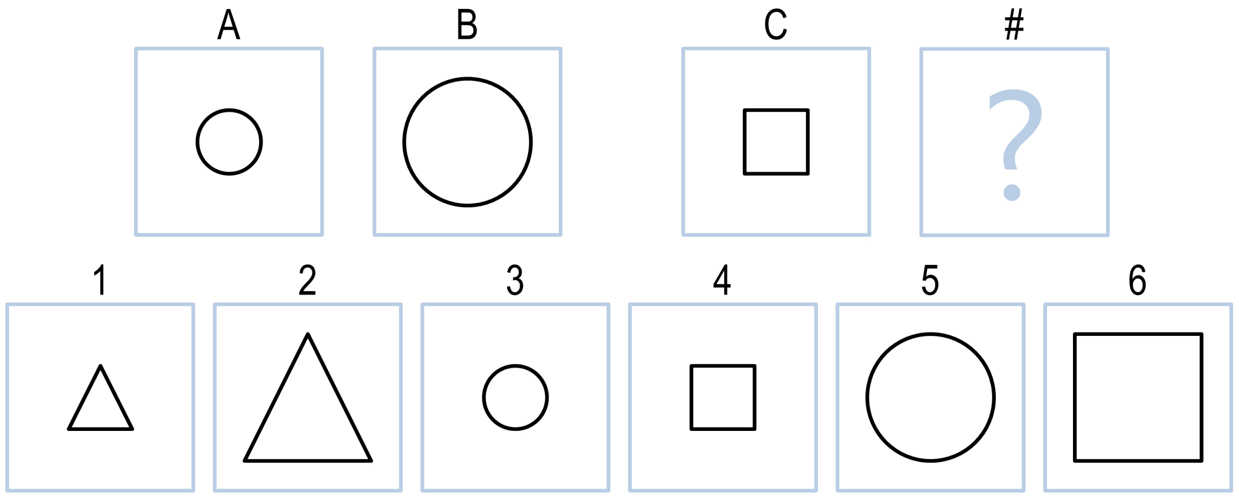
In this frame, the A, B, and C slots simply point to the RavensFigure class for each shape. This may look something like:

```
{
                    Object

          name: A
          shape: circle
          fill: no
          size: large
          angle: null
}
```

In this frame, the slots that don't apply can simply be stored as null. This frame hierarchy will allow you to break down and understand why the answer choice is correct. For a similar problem like this one below:

2x1 Basic Problem 02



We would probably map to the frame of Problem 01 and try to understand how they got the correct answer. The algorithm would label Problem 01 as similar because there is minimal differences between Figures A and B: both have figures that contain one object, and the classification of this is also a Basic Problem. All of these fillers will be the exact same as Problem 02. After it has mapped to a frame, it can then evaluate the difference between Figure C and the answer 5, and try to apply that rationale into choosing the answer for the current problem. After the algorithm has decided that the only difference between C and the answer is that figure 5 is filled, it will then look for any answer choices that have 1 difference from figure C in Problem 02. After analyzing figures A and B from Problem 02, it will know to be searching for a size difference. This makes it easy for my agent to choose figure 6 as the correct answer choice.

The only downfall about using frames for knowledge representations is that it may match to many past problems. However, this shouldn't be a major issue because they will convey similar differences to search for, which will allow answer elimination. However, it may be hard to choose all of the slots to categorize a problem type for a frame, but we are given text files that lay out the attributes. Therefore, we can use all of the attributes, and just insert null values for fillers where the slots don't apply to the problem.