Traci Fairchild
CS7637 KBAI
Assignment 2
September 6, 2015

Addressing Raven's Progressive Matrices Using Means Ends Analysis

With this assignment, I will be describing how I would implement Means Ends Analysis (MEA) within an *AI agent* design to solve one or more of the Raven's Progressive Matrices problems (RPM's). In this way I can further examine a process often used in computational problem resolution.

The Problem

One of the main problems with trying to computationally solve Raven's Progressive Matrix is the amount of information the agent has to work with. As we saw in assignment 1, and again in this assignment, designing an agent to programmatically process data to get to a goal can be relatively straightforward when you have lots of data to work with. In the case of the 'Basic' problems, we have the textual representation (as provided by in problemData.txt) to manipulate in various ways to get to our end goal.

What Makes Using Means Ends Analysis Difficult?

Using MEA to get from the initial state to the goal state is difficult because we have to find a good way to measure whether the transition to a state will bring us nearer or farther away from the goal state. This is particularly difficult when dealing with images. How does the agent measure the difference between 2 pictures? We can apply mathematical operations on images, and that would suggest there are ways to measure differences, but a knowledge of those operations, and what their results mean must be understood. We can also apply basic image manipulation operations such as rotate and flip. Measuring the results of image manipulation operations sound easier but then again, with this, how do you measure if you are closer to the goal state or further away? Do you perform another operation on top of the ones already done? Do you fall back to a previous state? Determining the answer to this is another problem I anticipate. I must design a way to measure differences between 2 images. In my early attempts I will use the textual representation provided and come up with an algorithm to measure the differences of two images. As I eventually move away from reliance of the textual representation, I will need to develop ways to measure differences as I attempt to solve the challenge problems.

The Design

Leveraging means ends analysis in the design of an AI agent does have it's advantages. It is a brute-force method that can be used repeatedly, and systematically in hopes that the goal state is achieved. I will continue along a series of steps, each one bringing me closer or further away from the goal. I must do this by looping through each image, and again for each object within the image. I will attempt to manipulate an object in one way to reach my next state. With this next state, I will use my 'measurement' algorithm to determine if the images are now closer or further apart. Upon review, I will either move forward and apply another manipulation to the current state or I will back out to the previous state and try something else.

The Details

My potential MEA solution being explored in this assignment is to loop through each figure of objA within a problem and apply one transformation at a time to one attribute at a time. The transform is tested and then saved or discarded. At the end, apply the list of saved transforms to objC to come up with the answer for objX:

```
# For ease of understanding the pseudo-code, we are comparing objA to objB
# Define valid transforms according to the attribute:
# shape, size and fill transforms:
        shapeShift = ['circle','square','pentagon','triangle','right triangle',...]
        sizeShift = ['huge','very large','large','medium','small',...]
        fillShift = ['yes','no']
# inside transform methods:
        getInsideList()       # get figures from the inside attr
        removeInsideList()    # remove one figure from the inside attr
        addInsideList()       # add one figure to the inside attr

# for the objectA, loop through figures
    figkeylist = problem.objects.keys()
    for figkey in figkeylist:
        # loop through attributes of each figure
        attrlist = problem.objects[figkey].attributes.keys()
        for akey in attrlist:
            # select an appropriate transform for the type of attr
            if akey = 'shape':
                transformlist = shapeShift
            else if akey = 'size':
                transformlist = sizeShift
            …

            # for each transform, apply it and test
```

```
for tkey in transformlist:
    # apply the transform, returning a new figure
    newTransform = [attr, transform[tkey]] # ex: ['size','large']
    newFig = manip(newTransform,problem.objects[skey])
    # build new object replacing the previous figure with the new
    tmpObj = replaceFig(objA,oldFig,newFig)
    # measure if closer or further from goal
    score = measure(tmpObj, objB)
    # if score is higher, save the transform
    saveTransform.append(newTransform)
    # if score is less, discard and continue with transforms

# we have exhausted the possible transforms, so now it is time to
# apply what we have saved to objC.  The resulting object should
# closely match objX.
objX = applyTransform(objC, saveTransform)
answer = findMatchingObj(objX)
```

Note: The above design is generalized for a 2x2 problem set.

Conclusion

Means Ends Analysis is a repetitive process that can, with processing power and maybe a little luck, produce an answer that is close to the desired goal. Since changing from one state to the next can exponentially alter the possibilities of the next states thereafter, the programmer must be careful to limit the processing to a reasonable and finite amount. With this in mind, MEA can be leveraged to achieve many different state spaces before reaching the goal. In this project, the agent could first use logic for Semantic Networks and in the absence of a discovered solution, the agent could move on to MEA and attempt to solve the problem there.

As I am discovering, there are many ways for achieving and displaying artificial intelligence in an agent. Lots of consideration, deliberation and coding must go into building an intelligent agent that can produce expected results.