Scott Dickerson
rdickerson3@gatech.edu
September 27, 2014

# KBAI Project 2 Design Report

## Ravens Progressive Matrices 2x2 Problems

# Introduction

The goal of Project 2 was to create an AI agent to answer Ravens Progressive Matrices 2x2 problems. My agent attempts to determine the correct answer to these problems by applying multiple problem solving methods including Generate and Test, Means Ends Analysis, Production System, Frames and Learning by Recording Cases. The problem representations provided are converted into RDF (Resource Description Framework) documents where the object attributes become RDF facts. So, a Raven's Progressive Matrices problem gets converted to a list of facts with a subject (object name), predicate (attribute type) and object (value).

# Summary

My agent solves these 2x2 problems by determining what the transformation between figure A and the missing figure D should be and using this transformation to generate a solution that can be matched to the choices in deciding on the correct answer. The agent performs the the following tasks in picking an answer to the problem:

| | | |
|---|---|---|
| 1 | RavensProblem is converted to a collection of RDFDocuments | Each figure in the problem is loaded into a RDFDocument by flattening the objects and attributes into a collection of facts. See Appendix A - RavensProblem to RDFDocuments for details. |
| 2 | Normalize object names | The object names across all figures are normalized by converting labels to sequential numbers. The end result is that object names in each figure are named 1, 2, 3, etc. Any object references in attribute values (i.e. above:B) are updated to reference the renamed object. |
| 3 | Generate A>B and A>C transformations | Creates a RDFXDocument containing RDFXFacts that define the change between two figures. See Appendix B - Basic transformation for details. |
| 5 | Generate A>D transformation | Creates a RDFXDocument representing the A>D transformation by merging the A>B and A>C transformations. See Appendix C - Merging transformations for details. |
| 6 | Generate possible solution from the A>D transformation deltas | Creates a new RDFDocument that contains facts about the potential solution figure by applying the rules defined in the A>D transformation to figure A. See Appendix D - Generate delta solution for details. |

| | | |
|---|---|---|
| 7 | Calculates a score for the comparison of the delta solution and each answer option. | For each fact that is an exact match between figures add 100 to a running total. This is done twice, once from figure A to B and again from figure B to A which increasing the weight of facts that do appear in both figures. Once all the facts have been tested, calculate the average score for all facts. |
| 8 | Pick the best answer. | |
| 9 | Return answer if only one answer got a score of 100, otherwise continue. | |
| 10 | Calculates a score for the comparison of the delta solution and each answer option again, but this time score using all combinations of object comparisons. | Potentially creates more solutions that are tested by shifting object names in figure A so that all combinations of object names are tested (i.e. figure A has objects 1 and 2 two solutions would be tested, first as is and again after exchanging object 1 facts for object 2). |
| 11 | Return the best answer. | This time just return the best answer. If more than one answer gets the same best score, return the answer with lower number (i.e. answer 2, before answer 6). |

## Getting the correct answer

My agent currently gets 16 of the 20 basic 2x2 problems and all of the challenge and classmate problems correct. Here I will examine some of the problems my agent got correct and explain why it chose the correct answer.

Problem: 2x2 Basic Problem 01

It determined that there was a 180 degree difference between the angle of object 0 in figure B and C (i.e. |135-315|=180). Since the shape of object 0 didn't change it looked for an answer choice that had a shape of Pac-Man and an angle of 225 (i.e. 45+180=225) which matched exactly to answer choice 5.

Problem: 2x2 Basic Problem 11

It saw that an object was added between figure A and B and a different object was added between figure A and C. It combined these two differences and looked for an answer choice that contained both added objects. At first the generated solution didn't match perfectly to any answer

choice mostly due to the object mapping being off (i.e. plus was object 0 and circle was object 1). Once the object names were shifted the generated solution almost matched exactly to answer choice 4. The only difference was the 4 also include the fact that plus was inside the circle.

## Making mistakes

Yes, my agent does make some mistakes. Here I will examine some of the problems my agent missed and explain why it chose the wrong answer.

Problem: 2x2 Basic Problem 12

My agent determined that only difference between figure A, B and C was that the shape changed to a square, so it looked for an answer choice that contained a square, unfilled and angle 0. It didn't find an exact match so I picked the first answer with the best score. Answers 2 and 3 to the same score, but 2 was selected.
Giving the fill attribute a higher weight would have gotten answer 3 instead, but that would probably cause it to miss a different problem.

## How my agent could be improved

If I had more time I would develop a submodule for my agent that could analyze spacial relationships between objects in a figure and objects from one figure to another. My agent currently does not consider relationships between objects.

## Agent Efficiency

The elapsed time from start to finish to process all 2x2 problems currently ranges from 80ms to 116ms depending on how many print statements are executed (i.e. minimal output 80ms). If I include the 2x1 problems the time increase by about 20ms.
Encountering problems with more objects per figure would increase elapsed time some.

## My agent vs. human cognition

When I analyze some of the 2x2 problem it's so easy to see that the object in figure B and C are mirror images of each other. For my agent to understand that requires examining the object's angle attributes and there relationship with figure A. It's really interesting that the addition of the ability to visually compare the figures is my first choice. Since my agent doesn't have that ability it is reliant on the angle values. I wonder if my agent could see would it solve the problem more like I do.

## Appendix A - RavensProblem to RDFDocuments

Each problem consists of figures A, B, C, 1, 2, 3, 4, 5 and 6. Each figure consists of objects and object attributes. For each attribute, a RDF fact is created where the mapping is as follows:

object name ==> RDF subject

attribute name ==> RDF predicate

attribute value ==> RDF object

A RDF document is created to contain all facts relating to a figure. Also, if the attribute value is a comma separated list of values the list is flattened so that each value in the list is a separate RDF fact.

Although the structure of a RDF document does not differ substantially from the RavensProblem class used to contain the problem representation, working with RDF documents and facts allowed for development of components that could be easily repurposed in other agents in the future.

## Appendix B - Basic transformation

Given two figures (1 and 2) this process produces a collection of facts describing the difference between the figure 1 and 2 as indicated by assigning a state (e.g. same, different, missing) to each attribute comparison. A typical transformation would be as follows:

| 1... | 2... | 1X2... |
|---|---|---|
| :0 :shape :circle | :0 :shape :circle | :0 :shape :circle:same |
| :0 :size :large | :0 :size :large | :0 :size :large:same |
| :0 :fill :no | :0 :fill :no | :0 :fill :no:same |
| :1 :shape :diamond | :1 :shape :diamond | :1 :shape :diamond:same |
| :1 :size :small | :1 :size :large | :1 :size :small:different |
| :1 :inside :0 | :1 :fill :no | :1 :inside :0:missing |
| :1 :fill :no | :1 :above :0 | :1 :fill :no:same |
| :2 :shape :circle | | :2 :shape :circle:missing |
| :2 :size :small | | :2 :size :small:missing |
| :2 :fill :yes | | :2 :fill :yes:missing |
| :2 :above :0,Y | | :2 :above :0,Y:missing |
| | | :1 :above :0:added |

Since the figures can contain multiple objects, the object naming from figure 1 to 2 may not indicate the best match for objects. In the above example the object naming does indicate the best possible match, but sometimes the object names need to be remapped. This is done using Means Ends Analysis to find the best match for an object in the other figure. If the figures do not have the same number of objects, the figure with fewer objects is remapped. Here's an example of a problem where the object names needed to be remapped:

Problem: 2x2 Basic Problem 09

| 1… | 2… | 2…remapped |
|---|---|---|
| :0 :shape :circl | C... | C... |
| :0 :size :large | :0 :shape :circle | :0 :shape :circle |
| :1 :shape :circle | :0 :size :large | :0 :size :large |
| :1 :size :medium | :1 :shape :circle | :2 :shape :circle |
| :1 :inside :0 | :1 :size :small | :2 :size :small |
| :2 :shape :circle | :1 :inside :0 | :2 :inside :0 |
| :2 :size :small | | |
| :2 :inside :0 | | |
| :2 :inside :1 | | |

# Appendix C - Merging transformations

To generate the transformation from A to missing figure D, I merged the transformations from A to B and A to C. The transformations consist of a collection of facts with a object, attribute type (e.g. angle), value from figure 1, value from figure 2 and the change state (same, different, missing, added, delta). When merging a fact from each transformation the process adheres to the following rules in producing the A to D transformation:

1. State **different** takes precedence over states **same** or **missing**.
2. State **missing** takes precedence over state **same**.
3. State **different** in both converts to state **same**.
4. State **different** in both and attribute type is **shape** could result in a shape change (ie. square to hexagon) determined by net change in number of sides of the shapes. For instance, if figure A object is square, figure B object is hexagon and figure C object is pentagon the shape would be changed to a heptagon (7 sides). Formula is (sides of figure A object) + (sides of figure B object - sides of figure A object) + (sides of figure C object - sides of figure A object).
5. State **delta** and attribute type is **angle** results in a new calculated delta based on the delta for figure A and B objects and figure A and C objects or the angles of figure B object and figure C object. Additional rules apply in this case to determine new angle.

# Appendix D - Generate delta solution

The idea is to generate a potential solution that is based on the differences defined in the provided transformation. The rules are fairly straight-forward:

1. If state **same** include the fact from figure A. For instance, the object is a **circle** in figure A, B and C.
2. If **state** different create a new fact using the transformation value. For instance, object is **circle** in figure A and B, but a **square** in figure C, the new value will be a **square**.
3. If **state** added create a new fact using the transformation value.
4. If state **delta** and attribute type is angle create a new fact where the value is angle of object in figure A + the combined delta from comparing figure A, B and C. If result is greater than 360, subtract 360 from the result.