Scott Dickerson
rdickerson3@gatech.edu
September 14, 2014

Project 1 - RPM AI Agent

Design Document

Introduction

The goal of Project 1 was to create an AI agent to answer Ravens Progressive Matrices 2x1 problems. My agent attempts to determine the correct answer to these problems using the Generate and Test method of problem solving. The problem representations provided are converted into RDF (Resource Description Framework) documents where the object attributes become RDF facts. As RDF documents, the problems as solved without any inherent knowledge or Ontology of the types of attributes provided in the problem representation.

Conversion to RDF Documents

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. 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.

The Brain

The agent has a brain. The brain can retrieve and utilize methods to attempt to solve the problems. For each problem the brain receives it sends to the GenerateAndTest method to solve.

The brain contains memory that can be used to store information as needed (will be utilized in future projects).

Generate And Test Method

The GenerateAndTestMethod is passed collection of RDF documents representing a Ravens Progressive Matrices 2x1 problem. It first extracts the RDF document for figures A and B. These documents are passed to the GenerateBasicTransformation which returns a document describing as a collection of facts how figure A transformed into figure B.

A...	B...	ABX...
:Z :shape :circle	:Z :shape :circle	:Z :shape :circle:same
:Z :size :large	:Z :size :large	:Z :size :large:same
:Z :fill :no	:Z :fill :no	:Z :fill :no:same
:Y :shape :diamond	:Y :shape :diamond	:Y :shape :diamond:same
:Y :size :small	:Y :size :large	:Y :size :small:different
:Y :inside :Z	:Y :fill :no	:Y :inside :Z:missing
:Y :fill :no	:Y :above :Z	:Y :fill :no:same
:X :shape :circle		:X :shape :circle:missing
:X :size :small		:X :size :small:missing
:X :fill :yes		:X :fill :yes:missing
:X :above :Z,Y		:X :above :Z,Y:missing
		:Y :above :Z:added

Once the A to B transformation is created, the GenerateAndTestMethod extracts the RDF document for figure C and each of the possible solutions. It performs a test using figure C, the A to B transformation and each possible solution returning a score for each in the range of 0 to 100 where 100 would indicate a perfect match. If this test results in one high score (greater than

90%), the corresponding figure name from the possible solutions is returned as the answer to the problem.

If no clear winner is found with the previous test, the GenerateAndTestMethod generates possible solutions using figure C and the A to B transformation. These generated solutions are tested against each of the possible solutions collecting scores for each test. The scores are evaluated to determine the best answer to the problem.

Generating a transformation

A transformation between two RDF documents (containing sets of facts) is generated as follows:

1. Compare each fact of document A to document B.
2. If the fact is found to be the same, the state is set to SAME.
3. If the fact is found, but the value is different, the state is set to DIFFERENT.
4. If the state is DIFFERENT and the value is numeric, the state is set to INCREASE or DECREASE depending on if document A value is greater or less than document B value.
5. If the fact is not found, the state is set to MISSING.
6. For each fact is document B that does not appear in document A, the state is set to ADDED.

See column ABX in figure above.

Learning from possible solutions

Before generating solutions as described in the next section, information about the values present in the possible solutions is collected in memory. The information is collected according to the RDF fact predicate field (i.e. shape or fill) and the RDF object (i.e. circle or yes/no). Once this process is complete memory contains a list of each value that appears in the possible solutions. These values will be used when generating solutions when the transformation state is DIFFERENT, INCREASE or DECREASE.

Generating solutions from source figure and transformation

Solutions are generated using the facts from the source figure and the facts from a transformation (i.e. A to B transformation). Multiple solutions are generated to provide for all possible options to values when a fact has a state of DIFFERENT, INCREASE or DECREASE in the

transformation facts. A list of possible options for values (i.e. circle, square) that appear in the possible solutions is used to fill in values.

Conclusion

The agent does a decent job of solving the problems correctly with a score of 12 out of 20 on the Basic problems and 2 out of 5 on the Challenge problems. It misses some problems where the value is a comma separated list of values (i.e X,Y,Z). This indicates a weakness in how the test is performed to determine if a value changed from figure A to figure B. One solution to this issue is to treat each value as a separate fact (e.g. change X:above:Y,Z to X:above:Y and X:above:Z). Another problem the agent has is when the object names don't match up exactly. It needs to be smarter in comparing facts about two figures without being so dependent of the object names. This has been an interesting project. I tried to not code very Ravens Progressive Matrices problem solving techniques into the agent in this project. That is, I wanted to see how well I could solve the problems without defining very specific rules and actions, but do plan to do more of that in the next project.