Project 1 ~~Design Report~~ Post Mortem

**For the past 24 hours I have been looking to this moment in dread. My code fails miserably. More functionality has broken in the past 48 hours than was working the 48 hours before that. I am submitting the agent and associated classes that worked reasonably well on Wednesday – although it did not do much. I figure I’ll submit the code, as well as this report which, except for last minute changes, reflected the state of things on Wednesday. Better than not to submit anything.**

**I’m going to have a beer, crawl into bed with the e-book chapter on inheritance in Java open and ready to read. Then wake up in the morning to begin working on project #2.**

# How Does Agent Work?

The Agent class is the root class of what is basically a structured, procedural approach to arriving at an answer. An instance of a RavensProblem class object is passed to the Agent, and it expands the knowledge of the problem set into several standard Java Collections, such as ArrayLists and HashMaps.

My proposition was that, by rule-based logic, the AI agent should be able to prune wrong answer frames before a resource-intensive problem-solving method even begins its slow iterative computations. I expected this to result in a quicker, and perhaps even more accurate, AI agent. And, honestly, it was the only way forward because I was mentally stuck not knowing how to embody ***is-a*** and ***has-a*** relationships required of a Semantic Network. By essentially pruning logically wrong answers, I would have a higher probability of success.

The problem solving method was initially going to be based on Means-Ends Analysis techniques. As the days left to finish the agent lessened, the problem solving method technique is a greatly reduced Generate and Test strategy. The “weightings” process whereby certain relationships have greater significance over others was not developed. To that end, as a last resort the agent makes its decision on random selection of non-pruned answer figures.

# How Agent Arrives at Correct Answers?

The agent has a poor success rate. The expectation that good results would be achieved by first reducing the number of answer figures does not appear to be the case. As much as it pains me to say it, the agent’s success rate is probably due to random selection as much as anything else.

There are some good aspects to report on, though. Even though I was stuck on how to design the Semantic Network component, I was not idle. One of the first pieces of code that I finished was an instrumentation class. I can monitor a single RavensProblem class instance or all instances. Tracing through the interim results at each major processing step are surprisingly easy to analyze, greatly aiding those times when I had a problem to troubleshoot. This instrumentation is already a Java class that I can carry over to subsequent projects.

# What Causes Agent to Make Mistakes?

Once one gets past the parts of the agent that are incomplete, I am faced with the fact that my whole approach relied on providing the agent with good – no Great! – rules choices. The agent could not learn! It was I who was learning, and in the process make agent solve the problems – sometimes making changes that broke earlier fixes.

There is also the brute-force aspect of the problem-solving method design. Humans don’t think that way! While we may approach problems by using problem reduction thought processes, humans do not think through all possibilities and iterative permutations to get at an answer. We are prone to instant reaction, especially when faced with immediate fight-or-flight situations.

With only a few days remaining before the submittal deadline, something profound crossed my mind that has been hard to shake off. Say for argument sake that my agent could indeed solve most if not all the problems put to it, and also say that it was due to computing every permutation of states until arriving at a correct answer: what does that say about the RPM as a test for intelligence? If the rules are known, and a person is told the rules, that hat person can be scored as having a higher intelligence than one that doesn’t know the rules? Like I said, I have not quite shook that thought from my mind.

# What Could Be Done to Improve Agent?

I have learned more lessons over these past days than I prudently ought to burden the reader with. Let’s stick to those issues that I can reasonably address in upcoming class projects:

* Have better clarity in my questions to Instructors as to what I need to know in order to get past problems. Early into week 2, in as much as I understood the theory of Semantic Networks, I had no idea how to apply the learning in a real world Java-based program. I got around to asking for help in week 3. However I never got across that my impediment was not knowing how to represent relationships (i.e. ***is-a***, ***transforms-to***, …) between objects. Instead, I wrote several Piazza postings asking “What do I do?” Our class TA was wanting to help, and I was eager to be helped. I should have asked “Can you provide an example in Java or pseudo-code how relationships are stored and evaluated?” I still don’t know the answer to that question, and I hope to learn it when I analyze program submittals by my classmates.
* Having nothing to show for the first three weeks, I began coding – thinking that if I incrementally add functionality), by trial and error, through luck and excessive caffeine, I could have a working AI agent. My error was regressing the design into traditional (i.e. procedural) rather than OOPs-based design. I was finally making progress toward a working AI agent, not realizing I would likely have to begin all over again in Project 2. It was easier for me to “see” the brute force steps needed to find a correct answer to a Rubin’s Progressive Matrix problem. Big, big mistake!
* Whereas I was clueless to implementing a Semantic Network knowledge representation, I did have ideas how to implement two problem-solving methods: Generate & Test, and Means-End Analysis, or even some hybrid design between the two. (I had not given Problem Reduction much thought, perhaps due to the emphasis placed on the other two problems solving methods.) I still believe that to be the case. On my next project, I think I should begin by considering how to design the problem-solving method, and use that design when designing the Semantic Network. (Not the other way around, as I did with this project.)
* Before coding had begun, it occurred to me that agent would be slow and inefficient in the problem-solving method task due to its inherent iterative nature, and not having the skillset to write efficient Java code. I thought that, if I had time, I would try to use some degree of multiple-thread concurrency to spread the compute load across more than a single (thread) CPU. I think the idea still has appeal, although what little I have read about java concurrency suggests it is difficult to get right, and even harder to debug. I would venture to try it only if I had a reasonably intelligent AI agent to start with.

In summary: I am highly critical of what little I have to show for this project. Still, it is not a complete loss because I have a good idea on what I have to begin working on for project #2. I see it involves Frames, which at first glance doesn’t look that different from what we have already covered in class. I am already reviewing Java inheritance and polymorphism. Starting tomorrow my questions to the instructors will have clarity. And I can try to work more collaboratively with one or more classmates in those things that I have trouble bridging theory and working programs.