Kemari Legg
CMSC330
Section 121
May 7, 2013

Prolog Write-up

When I was first introduced to prolog, I thought it was a very cool language, especially it's "goal oriented" aspect of it and the short amount of code that has to be written. I quickly saw the use of the language for artificial intelligence also. However, I came to understand that the language is counter intuitive and the difficulty in the language was conceptualizing it. This project, while vastly shorter than the ones done in Ocaml and Ruby, was a difficult transition from functional programming.

With respect to Ocaml, this project for the most part was a translation from the code we wrote in Ocaml to do various list commands. The code in Ocaml was much longer with the pattern matching syntax and recursion, however it was vastly easier and intuitive to write when you come from a background in functional and object oriented programming. Prolog did the same pattern matching, however, it was more condense and helped prevent me from making numerous syntax errors. However, as mentioned before, when programming in Prolog, it was counter intuitive for me. I had to basically think of programming in Ocaml backwards while doing the project, which lead me to a lot of confusion before and still a bit now. The one thing that was vastly easier to do and more straightforward was the CFG. Prolog's built in operators made this one of the strengths of the language.

With respect to ruby, I enjoyed programming in that the most. Since I started programming with Java, and object oriented language, OOP made sense to me. And Ruby was extremely similar to Java so by far it is my second favorite language. I found that for each of the languages, Prolog and Ruby, there were strengths and weakness' for doing the maze. When making the path function and virtually all the maze function, the code was extremely short and compact. This, as mentioned before, prevents many syntax errors since we type less. You can argue that it's simpler to read from a user, but at the same time, it's more difficult to understand than the Ruby code. Also, what made the Ruby maze slightly harder to do was the fact that we had to do extra functions like printing out a visual representation of the maze. That caused us to have to create a maze structure with cells. Having the actual maze made the project easier to understand, however, I had to code up hundreds of lines of code, while the Prologue maze is less than 100 lines including the comments.

I appreciate Prolog more from completing this project; however, I still dislike it. I hope to only use Java and Ruby in the work place since understanding Prolog is hard to do.