Pete Aguirre II
Dr. Andreas Stefik
CS 326 - 1002
Interpreter 2
March 15, 2019

After further study of making grammars, when I looked at my first original grammar, it didn't make sense to me anymore. I emailed the professor if I would be allowed to change the grammar of my language and went ahead to recreated my grammar once I got the approval. This time, I recreated my programming language inspired by Ruby. Last time, it was lolcode though after looking at Ruby's grammar, I saw that both programming languages had a lot of similarities. I mostly considered changing the language into a less symbol and more words type of language. Instead of symbolic syntax, I made it so it's like free writing. For example, '==' is now changed to IS or is. However, I did keep some symbolic syntax like: +, -, *, /, %. I also thought that since my language contains a lot of words, it's English biased. Meaning other people who are troubled with speaking English would have a difficult time coding the language I created.

By analyzing other grammar's control structures, I've learned that a lot of languages tend to have multiplicity. What I mean by that is, programming languages implement syntax to where you can do a statement in multiple ways. My interpreter leaned towards this technique too. For example, I did 'IS GREATER THAN' or 'is greater than'. I also noticed that some of them don't have bracket closers. I wanted to implement one where there was no bracket closers but I didn't know how to.

This paper by Altadmri and Brown was the one that convinced me to take out symbolic syntax in my language. I noticed that a lot of the mistakes were symbol stuff. Like, mismatched parentheses, mistaking = with ==, etc. Though the average fix time for those errors were low, they still accumulated overtime.

My final decisions for my grammar were mostly inspired by Ruby's implementation, the paper by Altadmri and Brown, and with the help of some lab rats. Before trying to make my Interpreter, I wrote down mathematical equation and see how my non computer science peers

would read it. I also made them see which syntax looked easier for their eyes and mind. They choose the language based syntax rather than the symbolic one. However, one of my friends actually voted for the symbolic syntax rather than the language based syntax. Though he has a heavy mathematical background. If my language was published, it would be meant more for beginners and novices, which was my goal in the beginning.