Alexander Marshall – 10048718?
CSCI 4020U – Assignment 2

Design Principles Behind the Grammar of L

I tackled this problem by making almost everything part of the 'expr' rule. I found this to be the easiest way to embed expressions inside of each other. Anything that gets evaluated gets sent to the 'var' rule; this includes expressions. I couldn't get the printing to print embedded messages and values in the correct order without majorly re-writing my code so I left it the way it was.

L2 Language Extension

Instead of adding one big feature to my grammar, I added several smaller ones. I added the math functions exponent, log, maximum and minimum. I also implemented a comparator that supports greater than, less than, and equal comparisons.

Accommodation of L2

Adding these features to the grammar was easy. I implemented the math functions by extending the 'expr' rule to support more options. To implement the comparator, I created a new 'comp' rule to evaluate the comparisons. Both sets of extensions should work with embedded expressions and messages.

How L Can Be A Functional Programming Language

To make L into a function programming language, we simply need to add more rules. We would need rules for flow control statements like do/while, for, if, and case switch. It would also need support for data structures like lists, arrays, maps, and others. There are infinitely many things we could add to the L programming language but I think the features I have listed are the most important.