Pete Aguirre II

Dr. Andreas Stefik

CS 326 - Language Implementation

February 12, 2019

**Edited:**

grammar Interpreter;

// Parser Rules Section

start

: expression EOF

;

expression

:

| INT // INT: 0-9

| FLOAT // FLOAT: INT.INT

| BOOL // BOOL: yay or nay

| STRING

| expression (MUL | DIV) expression // MUL & DIV rule

| expression (ADD | SUB) expression // ADD & SUB rule

| (LPAREN) expression (RPAREN) // Parenthesis Rule

| (COMMENT) expression // COMMENT <stmt.>

| expression (SPC) expression // Space Rule

;

// JS number primitive style

FLOAT

:

| INT (DOT) INT

| (DOT) INT

;

// Booleans exactly like JS

BOOL

:

| (TRUE)

| (FALSE)

;

// Lexer Rules Section

ADD : '+'; // Given

SUB : '-'; // Given

MUL : '\*'; // C style

DIV : '/'; // C style

LPAREN : '('; // C style

RPAREN : ')'; // C style

TRUE : 'yay'; // lolcode style

FALSE : 'nay'; // lolcode style

SPC : [ \r\n] -> skip; // lolcode exact copy

COMMENT : '//' ~ [\r\n] -> skip; // C style

DOT : '.'; // JS style

INT : '0'..'9'+; // Given

STRING : 'a'..'z'+ | 'A'..'Z'+;

**Discussion:**

The three languages I picked to compare my attempt for this assignment are C, JavaScript, and LOLCODE. In the Parser rule section of the interpreter, nearly nothing was similar to the three code I’ve compared. Though I think I hit two targets with JavaScript. The FLOAT was identical to JavaScripts but it was wrongly formatted. BOOL was also nearly the same.

For the Lexer rule section, almost everything was similar besides the space and number primitives. I found that all the operators, comment, left and right parenthesis were all similar to C, because I thought was the simplest way to do it.

**Changes:**

With the various language interpreters I compared, I thought the best solution was to combine C and LOLCODE together. With that, I kept the float and boolean interpretation similar to JavaScript. To explain how this would work, I thought LOLCODE’s interpreter is very simple to read. I “copied” LOLCODE’s format in the edited version of my interpreter. Although, LOLCODE’s Lexer rules are not to be messed with. Almost all the Lexer rules were inspired by

“Simplicity is the ultimate sophistication.” -Leonardo da Vinci

C. Three of them were by LOLCODE: booleans (yay or nay) and copied the whitespace because I couldn’t figure how to properly do the space.

**Difficulties:**

At first, I had trouble deciphering the interpreter. In the exception of Lexer rules section, I was completely confused by the Parser rules section. I asked myself often, “what is the expression supposed to be?” So all my initial guesses were obtained by the given expression:

expression (PLUS | MINUS) expression

However, after looking over the other interpreters, I figured out what changes I needed to make. Last second difficulties: comment is like the space but with ‘//’, so I copied it too. Another last second difficulties, I think the booleans doesn’t do anything, they’re just there.