Lab Three

Emily Doran

Emily.Doran1@Marist.edu

March 12, 2021

Crafting A Compiler

4.7 (Derivations)

A grammar for infix expressions follows:

```
1 Start -> E $
2 E     -> T plus E
3         | T
4 T     -> T times F
5         | F
6 F     -> ( E )
7         | num
```

(a) Show the leftmost derivation of the following string.

```
num plus num times num plus num $
```

$Start \Rightarrow E\$$

$E\$ \Rightarrow T$ plus E

T plus $E \Rightarrow T$ plus T

T plus $T \Rightarrow T$ plus T times F

T plus T times $F \Rightarrow T$ plus T times (E)

T plus T times $(E) \Rightarrow T$ plus T times $(T$ plus $E)$

T plus T times $(T$ plus $E) \Rightarrow T$ plus T times $(T$ plus $T)$

T plus T times $(T$ plus $T) \Rightarrow F$ plus T times $(T$ plus $T)$

F plus T times T plus $T \Rightarrow F$ plus F times $(T$ plus $T)$

F plus F times $(T$ plus $T) \Rightarrow F$ plus F times $(F$ plus $T)$

F plus F times $(F$ plus $T) \Rightarrow F$ plus F times $(F$ plus $F)$

F plus F times (F plus F) ⇒num plus F times (F plus F)

num plus F times (F plus F) ⇒ num plus num times (F plus F)

num plus num times (F plus F) ⇒ num plus num times (num plus F)

num plus num times (num plus F) ⇒ num plus num times (num plus num) $

(b) Show the rightmost derivation of the following string.

```
num times num plus num times num $
```

Start ⇒ $E$$

$E$$ ⇒ T

T ⇒ T times F

T ⇒ T times num

T times num ⇒ F times num

F times num ⇒ (E) times num

(E) times num ⇒ (T plus E) times num

(T plus E) times num ⇒ (T plus T) times num

(T plus T) times num ⇒ (T plus F) times num

(T plus F) times num ⇒ (T plus num) times num

(T plus num) times num ⇒ (T times F plus num) times num

(T times F plus num) times num ⇒ (T times num plus num) times num

(T times num plus num) times num ⇒ (F times num plus num) times num

(F times num plus num) times num ⇒ (num times num plus num) times num $

(c) Describe how this grammar structures expressions, in terms of the precedence and left- or right- associativity of operators.

During leftmost derivations, this grammar structures this expression differently from the order of operations. When we need to change F to be (E) everything that the (E) becomes is inside parenthesis and will be evaluated first according to the order of operations. The final 'num plus num' will be evaluated before the rest of the equation, whereas from initial glance, we would evaluate that part last. During the rightmost derivation, our string meaning also changes, but opposite. Instead of the last thing being evaluated first due to parenthesis like in the leftmost derivation, the first part will be evaluated first. The 'num times num plus num' will be evaluated, and then the result of that will be multiplied by the final num.

5.2c (Recursive Descent Parser)

Consider the following grammar, which is already suitable for LL(1) parsing:

```
1 Start  -> Value $
2 Value  -> num
3         | lparen Expr rparen
4 Expr   -> plus Value Value
5         | prod Values
6 Values -> Value Values
7         | lambda
```

(c) Construct a recursive-descent parser based on the grammar.

```
procedure parseStart(ts)
    parseValue()
    match($)

procedure parseValue(ts)
    switch(ts.peek())
        case num
            match(num)
        case lparen
            match(lparen)
            parseExpr()
            match(rparen)
    end

procedure parseExpr(ts)
    switch(ts.peek())
        case plus
            match(plus)
            parseValue()
            parseValue()
        case prod
            match(prod)
            parseValues()
    end

procedure parseValues(ts)
    switch(ts.peek())
        case num
            match(num)
            parseValues()
        case lparen
            match(lparen)
            parseExpr()
            match(lparen)
            parseValues()
        case rparen
            match(rparen)
    end
```

Dragon

4.2.1 (Derivations and a Parse Tree)

Consider the context-free grammar:

$S \Rightarrow SS+|SS^*|a$

and the string *aa + a

a) Give a leftmost derivation for the string.

$S \Rightarrow SS^*$

$SS^* \Rightarrow SS+S^*$

$SS+S^* \Rightarrow aS+S^*$

$aS+S^* \Rightarrow aa+S^*$

$aa+S^* \Rightarrow aa+a^*$

b) Give a rightmost derivation for the string.

$S \Rightarrow SS^*$

$SS^* \Rightarrow Sa^*$

$Sa^* \Rightarrow SS + a^*$

$SS + a^* \Rightarrow Sa + a^*$

$Sa + a^* \Rightarrow aa + a^*$

c) Give a parse tree for the string.