Lab 3

Andrew Hatch

Andrew.Hatch1@Marist.edu

March 12, 2024

1 Crafting a Compiler Exercises

1.1 Chapter 4.7

Part a) Show the leftmost derivation for the following string: num plus num times num plus num $

Derivation is as follows:

Start -> E $

T plus E $

T plus T $

F plus T $

num plus T $

num plus T times F $

num plus F times F $

num plus num times F $

num plus num times E $

num plus num times T plus E $ (you can include parentheses for (E) if you want)

num plus num times F plus E $

num plus num times num plus E $

num plus num times num plus T $

num plus num times num plus F $

num plus num times num plus num $

Part b) Show the rightmost derivation for the following string: num times num plus num times num $

Derivation is as follows:

Start -> E $

T plus E $

T plus T $

T plus T times F $

T plus T times num $

T plus F times num $

T plus num times num $

T times F plus num times num $

T times num plus num times num $

F times num plus num times num $

num times num plus num times num $

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

This grammar structures expressions based on the type of derivation we want to use - in left hand, expressions are structured based on left-associativity since they are capable of expansion when evaluated from the left (i.e. E is capable of expanding into (T times F). Likewise, when using right hand derivation we want to structure using right-associativity.

1.2 Chapter 5.2 Part c

Construct a recursive-descent parser based on the given grammar.

The order of tokens is as follows (variables will be marked):

parseStart() {

parseValue()

match($)

}

parseValue() {

if match(num) is true {

match(num)

}

else {

match(lparen)

parseExpr()

match(rparen)

}

}

parseExpr() {

```
if match(plus) is true {
match(plus)
parseValue()
parseValue()
}
else {
match(prod)
parseValues()
}
}
parseValues() {
if match(Value) is true {
parseValue()
parseValues()
}
else {
//Nothing here...
}
}
```

2 Dragon Book

2.1 Chapter 4.2.1

Part a) Give a leftmost derivation for the given context-free grammar.

Start -> S

S S *

S S + S *

a S + S *

a a + S *

a a + a *

Part b) Give a rightmost derivation for the given context-free grammar.

Start -> S

S S *

S a *

S S + a *

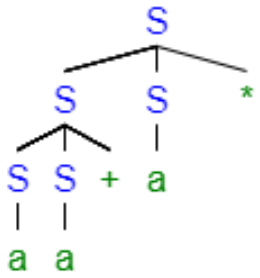

Figure 2.1: Parse Tree of given grammar for 4.2.1 Part c

S a + a *

a a + a *

Part c) Give a parse tree for the string.

See given figure.