Lab 3

Liam Healy

liam.healy1@marist.edu

March 12, 2019

1 Crafting a Compiler, Chapter 4, Exercise 4.7

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

E$

T plus E

T plus T plus E

T plus T times F plus E

T plus T times num plus E

F plus F times num plus T

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$

T

T times F

T times (E)

T times T plus E

T times T plus T

T times T plus T times F

F times F plus F 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.

A leftmost derivation process always chooses the leftmost possible nonterminal at each step. The nonterminal vaguely describes the terminal terms it can represent, and the result is the production sequence that a top-down parser would discover. A rightmost derivation process is the opposite, meaning the rightmost nonterminal is always expanded first, and this process is typically used by bottom-up parsers. One tricky thing about the rightmost derivation process is that the last step taken in a rightmost derivation is the first production applied by the bottom-up parser.

2 Crafting a Compiler, Chapter 5, Exercise 5.2c

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

```
parseValue() {
→ matchAndConsume(num) {
→ }
→ matchAndConsume(lparen) {
→ → parseExpr() {
→ → }
→ → matchAndConsume(rparen) {
→ → }
→ }
}
parseExpr() {
→ matchAndConsume(plus) {
→ }
→ parseValue() {
→ }
→ parseValue() {
→ }
→ matchAndConsume(prod) {
→ }
→ parseValues() {
→ }
}
parseValues() {
→ parseValue(plus) {
→ }
→ parseValues() {
→ }
}
```

3 Dragon Book exercise 4.2.1: a, b and c

a) Leftmost derivation:

S → SS → SS+S* → aS+S* → aa+S* → aa+(a)*

b) Rightmost derivation:

S → SS* → S+SS* → S+Sa* → S+aa* → a+a(a)*

c) Parse tree:

From - http://www.texample.net/tikz/examples/tree/