CMPT 432 - Spring 2018 - Dr. Labouseur

Lab Three

Marcus A. Zimmermann

Marcus.Zimmermann1@Marist.edu

March 16, 2018

Crafting a Compiler

Exercise 4.7

Grammar for infix expressions:

$$
\begin{array}{lcl}
Start & \rightarrow & E \ \$ \\
E & \rightarrow & T \ plus \ E \\
 & | & T \\
T & \rightarrow & T \ times \ F \\
 & | & F \\
F & \rightarrow & (\ E \ ) \\
 & | & num
\end{array}
$$

A Show the leftmost derivation of the following string.

num plus num times num plus num $

$$
\begin{array}{lcl}
Start & \Rightarrow & E \ \$ \\
 & \Rightarrow & T \ plus \ E \ \$ \\
 & \Rightarrow & F \ plus \ E \ \$ \\
 & \Rightarrow & num \ plus \ E \ \$ \\
 & \Rightarrow & num \ plus \ T \ plus \ E \ \$ \\
 & \Rightarrow & num \ plus \ T \ times \ F \ plus \ E \ \$ \\
 & \Rightarrow & num \ plus \ F \ times \ F \ plus \ E \ \$ \\
 & \Rightarrow & num \ plus \ num \ times \ F \ plus \ E \ \$ \\
 & \Rightarrow & num \ plus \ num \ times \ num \ plus \ E \ \$ \\
 & \Rightarrow & num \ plus \ num \ times \ num \ plus \ T \ \$ \\
 & \Rightarrow & num \ plus \ num \ times \ num \ plus \ F \ \$ \\
 & \Rightarrow & num \ plus \ num \ times \ num \ plus \ num \ \$
\end{array}
$$

B Show the rightmost derivation of the following string.

num times num plus num times num $

$$
\begin{array}{lll}
Start & \Rightarrow & E\ \$ \\
 & \Rightarrow & T\ plus\ E\ \$ \\
 & \Rightarrow & T\ plus\ T\ \$ \\
 & \Rightarrow & T\ plus\ T\ times\ F\ \$ \\
 & \Rightarrow & T\ plus\ T\ times\ num\ \$ \\
 & \Rightarrow & T\ plus\ F\ times\ num\ \$ \\
 & \Rightarrow & T\ plus\ num\ times\ num\ \$ \\
 & \Rightarrow & T\ times\ F\ plus\ num\ times\ num\ \$ \\
 & \Rightarrow & T\ times\ num\ plus\ num\ times\ num\ \$ \\
 & \Rightarrow & F\ times\ num\ plus\ num\ times\ num\ \$ \\
 & \Rightarrow & num\ times\ num\ plus\ num\ times\ num\ \$
\end{array}
$$

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

This grammar ensures that "times" has a higher precedence than "plus." Different nonterminals are used for each precedence level. By placing the rewrite rule for "times" lower in the grammar, it ends up lower in the tree.

Exercise 5.2

Grammar suitable for LL(1) parsing:

$$
\begin{array}{lll}
Start & \rightarrow & Value\ \$ \\
Value & \rightarrow & num \\
 & | & lparen\ Expr\ rparen \\
Expr & \rightarrow & plus\ Value\ Value \\
 & | & prod\ Values \\
Values & \rightarrow & Value\ Values \\
 & | & \lambda
\end{array}
$$

C Construct a recursive-descent parser based on the grammar (pseudo code)

Match

```
match(currentToken, expectedToken) {
    if (currentToken == expectedToken) {
        consume currentToken;
    } else {
        //error
    }
}
```

Parse Start

```
parseStart() {
    parseValue();
    match(currentToken, {EOP});
}
```

Parse Value

```
parseValue() {
    if (currentToken == num) {
        match(currentToken, {num});
    } else if (currentToken == lparen) {
        match(currentToken, {lparen});
        parseExpr();
        match(currentToken, {rparen});
    } else {
        // error
    }
}
```

Parse Expression

```
parseExpr() {
    if (currentToken == plusop) {
        match(currentToken, {plusop});
        parseValue();
        parseValue();
    } else if (currentToken == prodop) {
        match(currentToken, {prodop});
        parseValues();
    } else {
        // error
    }
}
```

Parse Values

```
parseValues() {
    if (currentToken == num || currentToken == lparen) {
        parseValue();
        parseValues();
    } else {
        // no error
        // lambda
    }
}
```

Compilers: Principles, Techniques, and Tools

Exercise 4.2.1