CMPT 308 - Spring 2024 - Dr. Labouseur

Lab 3 – Lexer

Patrick Tyler
Patrick.Tyler1@marist.edu

March 11, 2024

1 Crafting A Compiler

1.1 4.7 derivations

```
A)
E $
T plus E $
F plus E $
num plus E $
num plus T plus E $
num plus T times F plus E $
num plus F times F plus E $
num plus num times num 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 $
B)
E $
T plus E $
T plus T plus E $
T plus T plus T $
T plus T plus F $
T plus T plus num $
T plus T times F plus num $
T plus T times num plus num $
T plus F times num plus num $
```

```
T plus num times num plus num $
F plus num times num plus num $
num plus num times num plus num $
```

c)
The left to right associatvity of operators can derive the same set of terminal symbols with different trees. If I actually followed the rules it probably would have happend this way. This is because the langauge is not LL1.

1.2 5.2c recursive parser

```
let tokens = new Tokens;
let current_node = new Node;
let tree = new Tree(current_node);

fn match(token) {
    if tokens.next() == token {
        tree.add(token);
    } else {
        err;
    }
}

fn move_up() {
    current_node = current_node.parent;
}

fn do_start() {
    do_value();
    match($);
    move_up();
}

fn do_value() {
    next_token = tokens.peek();
    if (next_token == num) {
        match(num);
    } else if (next_token == lparen) {
        match(lparen);
        do_expr();
        match(rparen);
    } else {
        err;
    }
    move_up():
}
```

```
fn do_expr() {
    next_token = tokens.peek();
    if (next_token == plus) {
        match(plus);
        do_value()
        do_value()
    } else if (next_token == prod) {
        match(prod);
        do_values()
    } else {
        err;
    }
    move_up():
}

fn do_values() {
    next_token = tokens.peek();
    if (next_token in [num, lparen]) {
        do_value();
        do_values();
    }
    move_up():
}
```

2 Dragon

2.1 Exercise 4.2.1

```
A)
S
SS*
SS+S*
aS+S*
aa+S*
aa+a*
B)
S
Sa*
SS+a*
Sa+a*
aa+a*
```

```
C)
S
- S
-- S
--- a
-- S
--- a
-- +
- S
-- a
- *
```