CS 181 - Homework 6

Kuruvilla Cherian, Jahan
UID: 104436427
Section 1B

November 8, 2017

Problem 2.1

We first begin by creating the derivation and then build out a top down parse tree from this derivation.

a

Derivation:

$$E \Rightarrow T \Rightarrow F \Rightarrow a$$

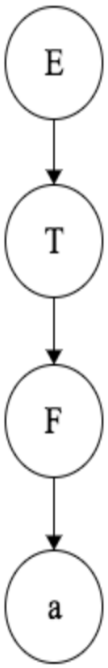

Figure 1: Parse tree for a

b

Derivation:

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow F + F \Rightarrow a + F \Rightarrow a + a$$

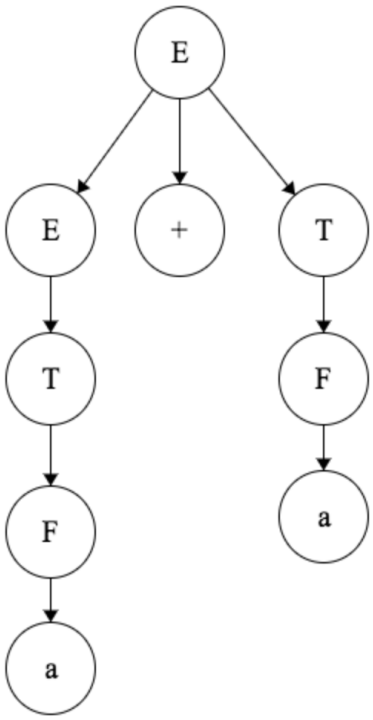

Figure 2: Parse tree for $a + a$

c

Derivation:

$E \Rightarrow E{+}T \Rightarrow E{+}T{+}T \Rightarrow T{+}T{+}T \Rightarrow F{+}T{+}T \Rightarrow F{+}F{+}T \Rightarrow F{+}F{+}F \Rightarrow a{+}F{+}F \Rightarrow a{+}a{+}F \Rightarrow a{+}a{+}a$

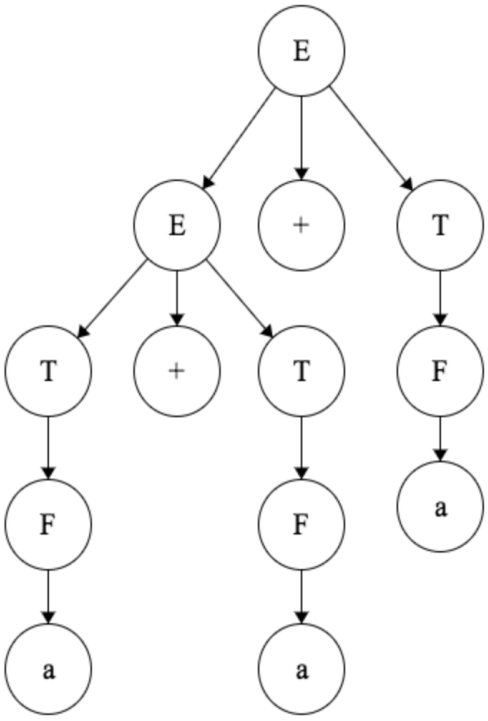

Figure 3: Parse tree for $a + a + a$

d

Derivation:

$$E \Rightarrow T \Rightarrow F \Rightarrow (E) \Rightarrow (T) \Rightarrow (F) \Rightarrow ((E)) \Rightarrow ((T)) \Rightarrow ((F)) \Rightarrow ((a))$$

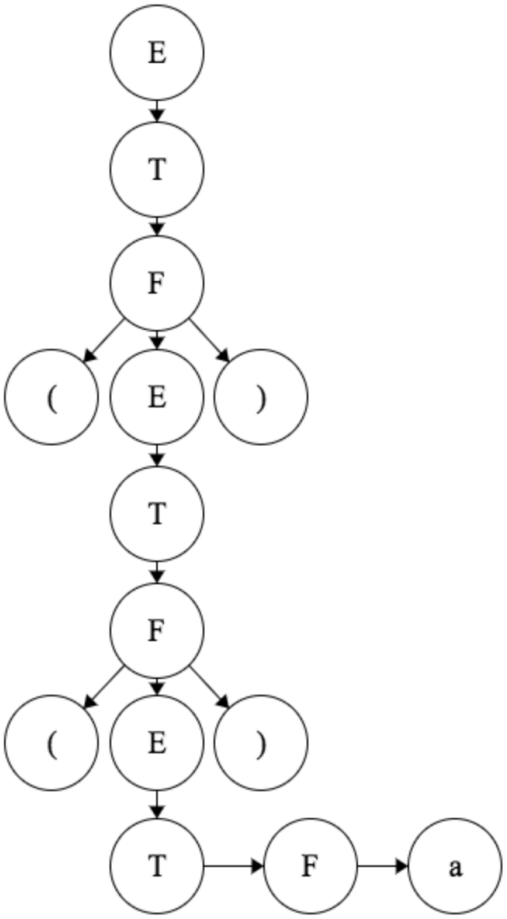

Figure 4: Parse tree for $((a))$

Problem 2.6(d)

$$S \to M\#P\#M \mid P\#M \mid M\#P \mid P$$
$$P \to aPa \mid bPb \mid a \mid b \mid \varepsilon \mid \# \mid \#M\#$$
$$M \to aM \mid bM \mid \#M \mid \varepsilon$$

Problem 2.19

Looking at our start terminal we see that the productions indicate that this language should have an unequal amount of a's and b's. This means that in all instances there should be an imbalance between the two symbols, with the ability for one symbol to never appear. This means this language represents the complement of $a^k b^k : k \geq 0$. Therefore, the complement of $L(G)$ is

$$\neg L(G) = \{a^k b^k : k \geq 0\}$$

This can be represented by the following CFG:

$$S \to aSb \mid \varepsilon$$

Given $\Sigma = \{a, b\}$ and starting symbol as S.

Problem 2.27

a

The problem comes from the following program that can be generated by the CFG:

if condition then if condition then a:=1 else a:=1

We see this can be produced by two left-most derivations as follows:

<STMT>
⇒ <IF-THEN-ELSE>
⇒ if condition then <STMT> else <STMT>
⇒ if condition then <IF-THEN> else <STMT>
⇒ if condition then if condition then <STMT> else <STMT>
⇒ if condition then if condition then a:=1 else <STMT>
⇒ if condition then if condition then a:=1 else a:=1

or we could have

<STMT>
⇒ <IF-THEN>
⇒ if condition then <STMT>
⇒ if condition then <IF-THEN-ELSE>
⇒ if condition then if condition then if condition then <STMT> else <STMT>
⇒ if condition then if condition then a:=1 else <STMT>
⇒ if condition then if condition then a:=1 else a:=1

Thus because we have two different derivations, it means we can generate two different parse trees, which means the language is *ambiguous*.

b

The problem here is that we have no rule to match the *else* with the closest *if*. What this means is that the *else* can match to both the nearest and farthest *if* statements, which leads to ambiguity. We can mitigate this by explicitly introducing two new terminals, and replacing the old grammar's <IF-THEN-ELSE> rule. These two new terminals are defined as follows:

<NEW-STMT> → <ASSIGN> | <NEW-IF-THEN-ELSE>

<NEW-IF-THEN-ELSE> → if condition then <NEW-STMT> else <NEW-STMT>

IF-THEN-ELSE → if condition then <NEW-STMT> else <STMT>

Problem 2.28(c)

$$
\begin{aligned}
E &\to aAbE \mid bBaE \mid F \\
A &\to aAbA \mid \varepsilon \\
B &\to bBaB \mid \varepsilon \\
F &\to aAF \mid \varepsilon
\end{aligned}
$$