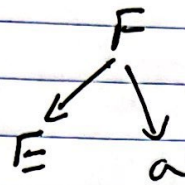
CS 181 HW 5

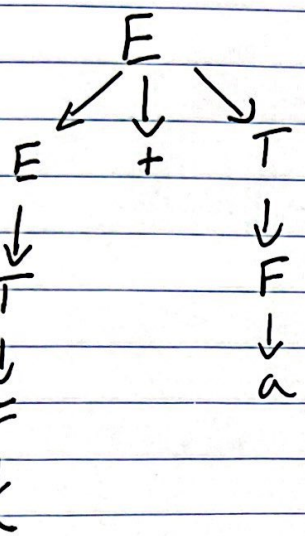
TEJAS KAMTAM, 305 749 402

2.1) a)



$= F \rightarrow a = a$

b)



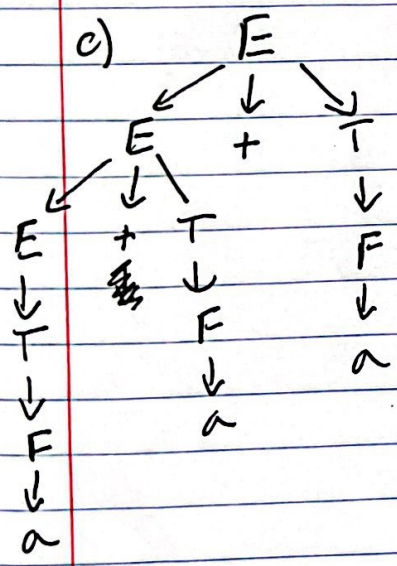
$E \rightarrow E_1 + T$

$= E_1 \rightarrow T \quad = a + a$

$T \rightarrow F$

$F \rightarrow a$

c)



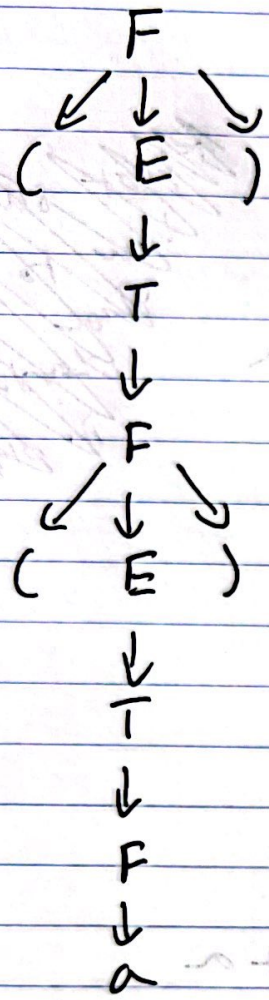
$E \rightarrow E_1 + T$

$= E_1 \rightarrow T \quad = a + a + a$

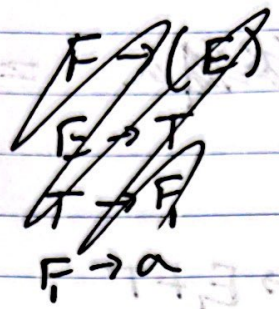
$T \rightarrow F$

$F \rightarrow a$

d)



$\equiv$



$F \rightarrow (E)$

$\rightarrow (T)$

$\rightarrow (F)$

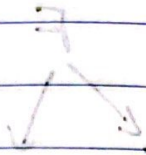
$\rightarrow ((E))$

$\rightarrow ((T))$

$\rightarrow ((F))$

$\rightarrow ((a))$

$= ((a))$



2.6) d)

$S \to aSa \mid bSb \mid \varepsilon \mid \#A\#$

$A \to S \mid \#A \mid A\#$

$S \to C \mid A\#C\#A$

$C \to aCa \mid bCb \mid \varepsilon$

$S \to C \mid C\#C\#C \mid C\#C$

$C \to aCa \mid bCb \mid \varepsilon$

2.9)

$S \to A \mid B$

$A \to Ac \mid C$ — $Accc \to Cccc \to abccc$

$B \to aB \mid D$ — ~~$aaaB \to aaaD \to aaa$~~

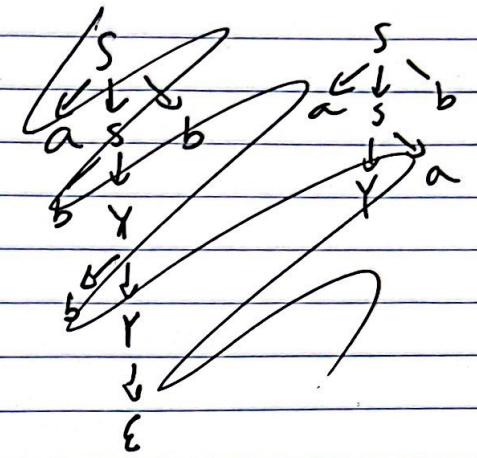
$C \to aCb \mid \varepsilon$ — $aaaB \to aaaD \to aaabc$

$D \to bDc \mid \varepsilon$

This grammar is ambiguous b/c C or D can generate ε so A or B can cause a stop which can result in the same strings but using different paths

2.19) $S \to aSb \mid bY \mid Ya$

$Y \to bY \mid aY \mid \varepsilon$



$a^n b Y b^n$ or $a^n Y a b^n$ $\quad n \geq 0$

$a^n b^m b^n$ or $a^n a^m b^n$ $\quad m \geq 0$

$a^n (a^* \cup b^*) b^n$

$a^+ \cup b^+ \cup a\Sigma^* b \cup$ ~~$b\Sigma^* a$~~ $\cup\ a b^* \cup b a^*$

~~Seeing like anything but empty string so complement CFG~~ B:

~~$S \to \varepsilon$~~

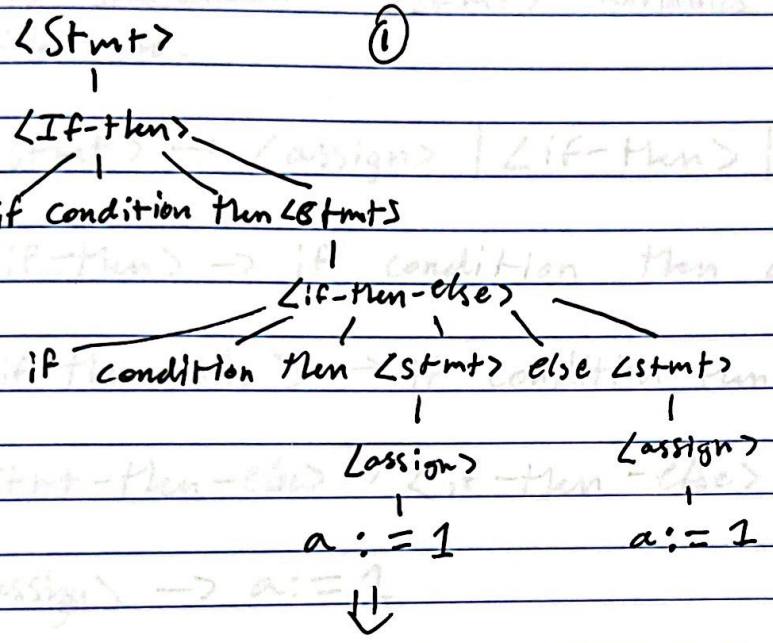
Can't generate strings ~~[illegible]~~

$\to$ like $b^+ a^*$ or ε

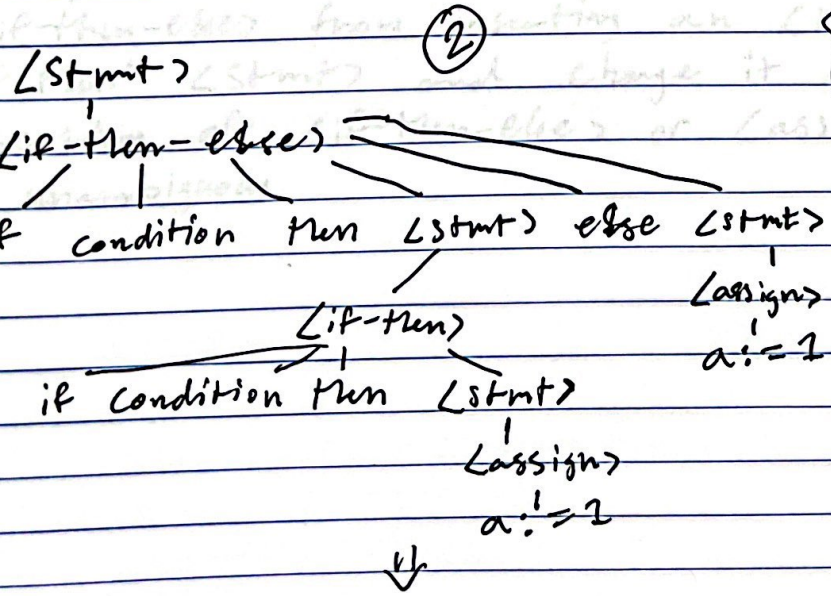
$S \to bSa \mid \varepsilon$ describes a CFG for the complement

2.27)

a) Consider the following parse trees:



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



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

Parse trees ①, ② are distinct, yet they generate the same strings, thus the CFG is ambiguous.

2.27b) To disambiguate the grammar, we need to prevent either an <if-then> from inserting an <if-then-else> or an <if-then-else> from inserting an <if-then> in the ~~student~~ <stmt> variables in their derivations, like so:

<stmt> → <assign> | <if-then> | <if-then-else>

<if-then> → if condition then <stmt>

<if-then-else> → if condition then <stmt-then-else> else <stmt>

<stmt-then-else> → <if-then-else> | <assign>

<assign> → a := 1

Here, we chose the latter option and restrict <if-then-else> from inserting an <if-then> in its "if then" <stmt> and change it to only allow continued insertions of <if-then-else> or <assign>. Thus, this CFG is unambiguous.

2.28c)

$L = \{ w : \#_a(w) \geq \#_b(w) \}$

CFG G:

$$S \to aSb \mid bSa \mid Sa \mid \varepsilon$$

This allows the variable to either generate an equal # of a's and b's or more a's or the empty string (in which $\#_a(\varepsilon) \geq \#_b(\varepsilon) = 0$).

7)

We can convert any DFA to a CFG by considering the transitions as terminal symbols and the states as nonterminals (variables). Then we can describe our grammar G for an arbitrary model M as:

Given $M = (Q, \Sigma, \delta, q_0, F)$, we describe $G = (V, \Sigma, R, q_0)$ s.t. $V = Q$

$R = \{ q \to \sigma_1 q_1 \mid \sigma_2 q_2 \mid \dots \mid \sigma_n q_n \;\; \forall q \in Q : q_i = \delta(q, \sigma_i)$ over all $\sigma_i \in \Sigma \} \cup \{ q_f \to \varepsilon \;\; \forall q_f \in F \}$

→ using G.

This union should append ε or-wise to the existing mappings on the LHS of the union for accepting states.

This can accurately describe any DFA M as a CFG G, thus all languages that are recognized by a DFA can be expressed