Module 3

Mrs. Kavya S
Asst. professor
C. design dept
PESITM

Context free Grammer

Definition of context free grammer (CFG):

It is a formal grammar, which consists of a set of production rules. These production rules are used to generate the string of a language.

CFG 'G' can be defined by a 4 tuples as

$$G = (V, T, P, S)$$

T → set of terminals symbols (lowercase)

V → set of non terminal symbols (uppercase)

S → start symbol (from V)

P → set of production rules. which are used to replace non-terminals symbols to a string with other terminal or non terminal symbols.

Problems:

1) construct CFG for the language having any number of a's

$L = \{\epsilon, a, aa, aaa, aaaa \cdots\}$

$G = (V, T, P, S)$ $\quad \therefore V = \{S\}$

$T = \{a\}$

$\delta(S, a) = S$

production rule $P = \begin{cases} S \rightarrow aS & \text{— Rule 1} \\ S \rightarrow \epsilon & \text{— Rule 2} \end{cases}$

S is Start symbol.

Let us drive one example string "aaaa" CL using production Rule.

Begin with start symbol S

$\Rightarrow aS$ Rule 1

$\Rightarrow aaS$ Rule 1

$\Rightarrow aaaS$ Rule 1

$= aaaa\epsilon$ Rule 2 Replace S by ϵ

$=$ 'aaaa' string derived finally

So grammer is correct.

(2) construct CFG for the language having any number a's & b's

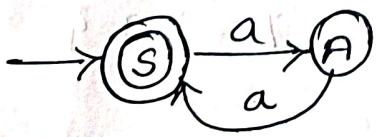
$L = \{\epsilon, a, b, aa, ab, bb, ba, bbb---\}$

$P = \{S \to aS \mid bS \mid \epsilon S\}$ or $S \to aS$, $S \to bS$, $S \to \epsilon$

CFG $G = (V, T, P, S)$

$G = (\{S\}, \{a, b\}, [S \to aS|bS|\epsilon\}, S)$

(3) Design CFG for the language having even number of a's



$G = (V, T, P, S)$

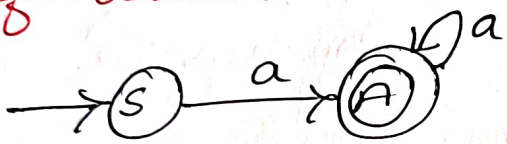
$V = \{S, A\}$

$T = \{a\}$

$P = \{ S \to aA \mid \epsilon$
$A \to as \}$

S is start symbol

(4) obtain a grammer to generate string consisting of atleast 1a



$G = (V, T, P, S)$

$V = \{S, A\}$

$T = \{a\}$

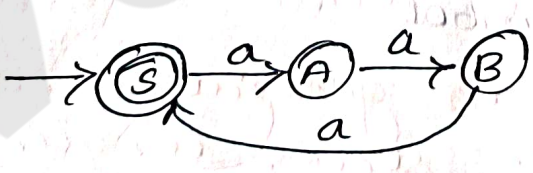
$P = \{ S \to aA$
$A \to aA \mid \epsilon \}$

S is start symbol.

(5) obtain a grammer to generate string consting of multiple of 3 a's

$S \to aaaS \mid \epsilon$

or



$G = (V, T, P, S)$

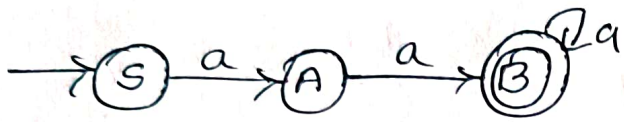
$V = \{S, A, B\}$

$T = \{a, b\}$

$P = \{ S \to aA$
$A \to aB$
$B \to aS$
$S \to \epsilon \}$

S is start symbol

⑥ obtain grammer to generate string consisting of atleast two a's.



$G = (V, T, P, S)$

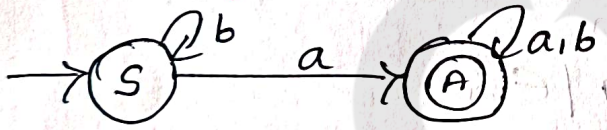
$V = \{S, A, B\}$

$T = \{a\}$

$P = \{ S \to aA$
$A \to aB$
$B \to aB \mid \epsilon \}$

S is start symbol.

⑦ obtain grammer to generate string consisting of a's & b's atleast one a.



$G = (V, T, P, S)$

$V = \{S, A\}$

$T = \{a, b\}$

$P = \{ S \to bS$
$S \to aA$
$A \to \epsilon \mid aA \mid bA \}$

S is start symbol

⑧ obtain grammer to generate string of a's & b's such that string length is multiple of 3

$S \to \epsilon \mid AAAS$

$A \to a \mid b$

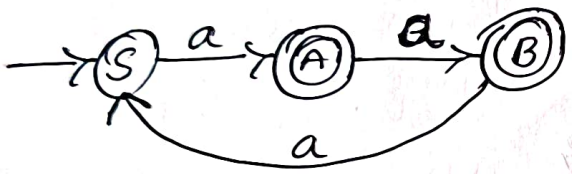
$G = (V, T, P, S)$

$V = \{S, A\}$

$P = \{ S \to \in | AAAS$
$A \to a|b \}$

S is start symbol.

) obtain grammer to accept the following language $L = \{ w : |w| \bmod 3 > 0$ where $w \in \{a, b\}$



or $S \to a|aa|aaaS$

$G = (V, T, P, S)$
$V = \{S, A, B\}$
$T = \{a\}$
$P = \{ S \to aA$
$A \to aB | \in$
$B \to aS | \in \}$
S is start symbol

) obtain a grammer to generate the following language

$L = \{ a^n b^n | n \geq 0 \}$

$L = \{ ab, aabb, aaabbb -- \}$

$G = (V, T, P, S)$
$V = \{ S \}$
$T = \{a, b\}$
$P = \{ S \to \in | asb \}$
S is start symbol.

(11) Obtain a grammer to generate the following language.

$L = \{a^n, b^n : n \geq 1\}$

$L = \{ab, aabb, aaabbb \cdots\}$

$G = (V, T, P, S)$

$V = \{S\}$

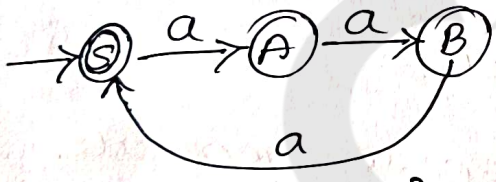
$T = \{a, b\}$

$P = \{S \to ab \mid asb\}$

S is start symbol.

(12) obtain a grammer to generate the following language

$L = \{w : |w| \bmod 3 = 0, \text{ where } w \in \{a\}^*\}$



$G = (V, T, P, S)$

$V = \{S, A, B\}$

$T = \{a\}$

$P = \{S \to aA \mid \epsilon$
$A \to aB$
$B \to aS\}$

S is start symbol

(13) obtain a grammer to generate the following language $L = \{a^{n+1}, b^n : n \geq 0\}$

$L = \{a\epsilon, aab, aaabb, aaaabbb \cdots\}$

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \to a \mid asb\}$ S is start symbol

4) obtain a grammer to generate the following language

$L = \{a^n b^{n+1} : n \geq 0\}$

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \to b \mid asb\}$

S is start symbol.

5) obtain a grammer to generate the following

$L = \{a^n b^{n+2} : n \geq 0\}$

Two extra b's should be generated so the final grammar to generate given lang is correct

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \to bb \mid asb\}$

S is start symbol

6) obtain a grammer to generate the following language $L = \{a^n b^{2n} : n \geq 0\}$

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \to \epsilon \mid asbb\}$

S is start symbol.

(17) * Obtain a grammer G & generating set of all palindrome over $\Sigma = \{a, b\}^*$

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{a, b\}$

$P = \{ S \to \epsilon$
$S \to a \mid b$
$S \to asa \mid bsb \}$

S is start symbol.

(18) Obtain a grammer to generate the following

$L = \{ww^R$ where $w \in \{a, b\}^*\}$

$L = \{aa, bb, abba, baab - - -\}$

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{a, b\}$

$P = \{S \to \epsilon \mid asa \mid bsb\}$

S is start symbol

(19) Obtain a grammer to generate a language consisting of all non-palindrome over $\{a, b\}^*$

$G = (V, T, P, S)$

$V = \{S, A, B\}$

$T = \{a, b\}$

$P = \{ S \to asa \mid bsb$
$S \to A$
$A \to aBb \mid bBa$
$B \to aB \mid bB \mid \epsilon \}$

S is start symbol

20) obtain grammer to generate the language

$L = \{ 0^m 1^m 2^n \mid m \geq 1 \ \& \ n \geq 0 \}$

$L = \{ \underbrace{0^m 1^m}_{A} \underbrace{2^n}_{B} \mid m \geq 1 \ \& \ n \geq 0 \}$

$S \rightarrow A \quad B$

The variable A should produce ~~more~~ no of 0's and followed by m no of 1's

B should produce any no of 2's

$B \rightarrow \in | 2B$

$S \rightarrow AB$

$A \rightarrow 01 | 0A1$

$B \rightarrow \in | 2B$

$G = (V, T, P, S)$

$V = \{ S, A, B \}$

$T = \{ 0, 1, 2 \}$

$P = \{ S \rightarrow AB,$
$A \rightarrow 01 | 0A$
$B \rightarrow \in | 2B \}$

S is start symbol

21) obtain the grammer to generate the language $L = \{ w \mid n_a(w) = n_b(w) \}$

eg: abba, baab

$G = (V, T, P, S)$

$V = \{ S \}$

$T = \{ a, b \}$

$P = \{ S \rightarrow \in | aSb | bSa | SS \}$

S is start symbol.

(22) obtain a grammer to generate a string balanced parenthesis.

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{(,)\}$

$P = \{ S \to (S) \mid SS \mid \epsilon \}$

S is start symbol

(23) obtain a grammer to generate the langu

$L = \{a^i b^j \mid i \neq j, i \geq 0 \text{ & } j \geq 0\}$

$L = \{ a^i b^j \mid i \neq j, i \geq 0, j \geq 0$

$G = (V, T, P, S)$

$V = (S, A, B)$

$T = \{a, b\}$

$P = \{ S \to aSb \mid A \mid B$

$A \to a \mid aA$

$B \to b \mid bB$

S is start symbol

$L = \{ 0^i 1^j \mid i \neq j, i \geq 0, j \geq 0$

$S \to 0S1 \mid A \mid B$

$A \to 0 \mid 0A$

$B \to 1 \mid 1B$

$G = (V, T, P, S)$

$V = \{S, A, B\}$

$T = \{0, 1\}$

$P = \{ S \to 0S1 \mid A \mid B$

$A \to 0 \mid 0A$

$B \to 1 \mid 1B \}$

S is start symbol.

(24) obtain a grammer to generate the langu

$L = \{ a^n b^m \mid n \geq 0, m > n \}$

$n=0$ $m \geq 1$ $\quad$ $n=1$ $m \geq 2$ $\quad$ $n=2$ $m \geq 3$

$L = \{ \epsilon bb^*, abbb^*, aabbbb^* - - - \}$

$G = (V, T, P, S)$

$V = \{S, B\}$

$T = \{a, b\}$

$P = \{ S \to aSb \mid B,$ [generating $a^n b^n \mid n \geq 0$ followed by atleast 1 b]

$B \to bB \mid b \}$ [generate one or more b's]

S is start symbol

(25) obtain a grammer to generate the language

$L = \{ a^n b^{n-3} \mid n \geq 3 \}$

$L = \{ \underset{n=3}{aaa}, \underset{n=4}{aaaab}, \underset{n=5}{aaaaabb} --- \}$

$G = (V, T, P, S)$

$V = \{S, A\}$

$T = \{a, b\}$

$P = \{ S \to aaaA,$

$A \to aAb \mid \in \}$

S is start symbol.

Derivation:

"The process of obtaining string of terminal and/or non terminals from the start symbol by applying some or all production is called derivation."

⇒ If one or more only one production are applied to get string it is called one step derivation. one or more production are applied to get string $\alpha\beta\gamma$ from A then

$$A \overset{+}{\Rightarrow} \alpha\beta\gamma$$

If zero or more production are applied to get the string $\alpha\beta\gamma$ from A then we write

$$A \Rightarrow^{*} \alpha\beta\gamma$$

(i) **Leftmost derivation:**

In the derivation process if a left most variable is replaced at every step then the derivation is said to be leftmost

(ii) **Rightmost derivation:**

In the derivation process if a right most variable is replaced at every step then the derivation is said to be rightmost

1) consider a grammer shown below from which any arithmetic expression can be obtained. $E \to E+E \mid E-E \mid E*E \mid E/E \mid id$
E is start symbol. obtain the string id+id*id by applying leftmost & rightmost derivation

Leftmost

$$\begin{aligned} E &\overset{+}{\underset{lm}{\Rightarrow}} E+E \\ &\Rightarrow id+E \\ &\Rightarrow id+E*E \\ &\Rightarrow id+id*E \\ &\Rightarrow id+id*id \end{aligned}$$

The string id+id*id is obtained from the start symbol E by applying leftmost derivation & can be written as

$$E \overset{+}{\underset{lm}{\Rightarrow}} id+id*id$$

Rightmost

$$\begin{aligned} E &\underset{rm}{\Rightarrow} E+E \\ &\Rightarrow E+E*id \\ &\Rightarrow E+E*id \end{aligned}$$

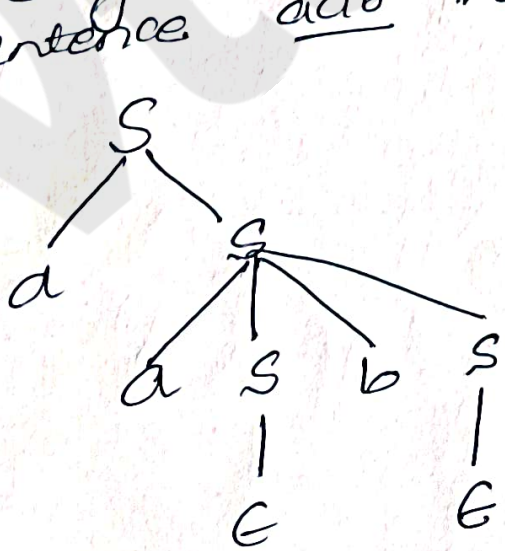
$\therefore E \underset{rm}{\Rightarrow} id+id*id$

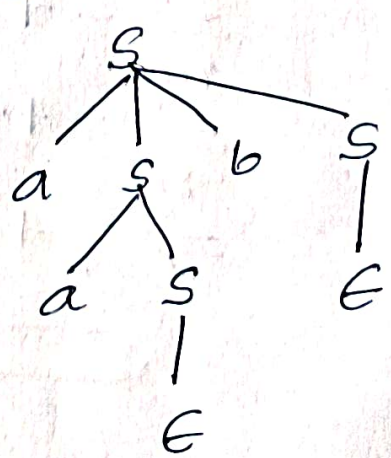
(1) Obtain the leftmost derivation for the string aaabbabbba using following grammer

$S \to aB/bA$
$A \to aS/bAA/a$
$B \to bS/aBB/b$

$S \underset{lm}{\Rightarrow} aB$ (Applying $S \to aB$)
$\Rightarrow aaBB$ ($B \to aBB$)
$\Rightarrow aaaBBB$ ($B \to aBB$)
$= aaabBB$ ($B \to b$)
$= aaabbB$ ($B \to b$)
$= aaabbaBB$ ($B \to aBB$)
$= aaabbabB$ ($B \to b$)
$= aaabbabbS$ ($B \to bS$)
$= aaabbabbbA$ ($S \to bA$)
$= aaabbabbba$ ($A \to a$)

(2) Show that grammer below is ambiguous
$S \to aSbS/aS/\epsilon$
the grammar is ambiguous because the sentence aab has two parse trees.





(3) consider the grammar $E \to +EE \mid *EE \mid -EE \mid x \mid y$

(a) Find LMD & RMD for string $-+*xyxy$

(b) Is the grammer ambiguous? Justify

(a) Leftmost Derivation.

$$E \underset{lm}{\Rightarrow} -EE$$
$$\underset{lm}{\Rightarrow} -+EEE$$
$$\underset{lm}{\Rightarrow} -+*EEEE$$
$$\underset{lm}{\Rightarrow} -+*xEEE$$
$$\underset{lm}{\Rightarrow} -+*xyEE$$
$$\underset{lm}{\Rightarrow} -+*xyxE$$
$$\underset{lm}{\Rightarrow} -+*xyxy$$

(b) Rightmost derivation

$$E \underset{rm}{\Rightarrow} -EE$$
$$\underset{rm}{\Rightarrow} -Ey$$
$$\underset{rm}{\Rightarrow} -+EEy$$
$$\underset{rm}{\Rightarrow} -+*EExy$$
$$\underset{rm}{\Rightarrow} -+*EyXy$$
$$\underset{rm}{\Rightarrow} -+*xyxy$$

eg ① $S \Rightarrow_{rm}$ aB

aaBB

aaBaBB

aaBaBbS

aaBaBbbA

aaBaBbbba

aaBabbba

aaaBBabbba

aaabbabbba

aaabbabbba

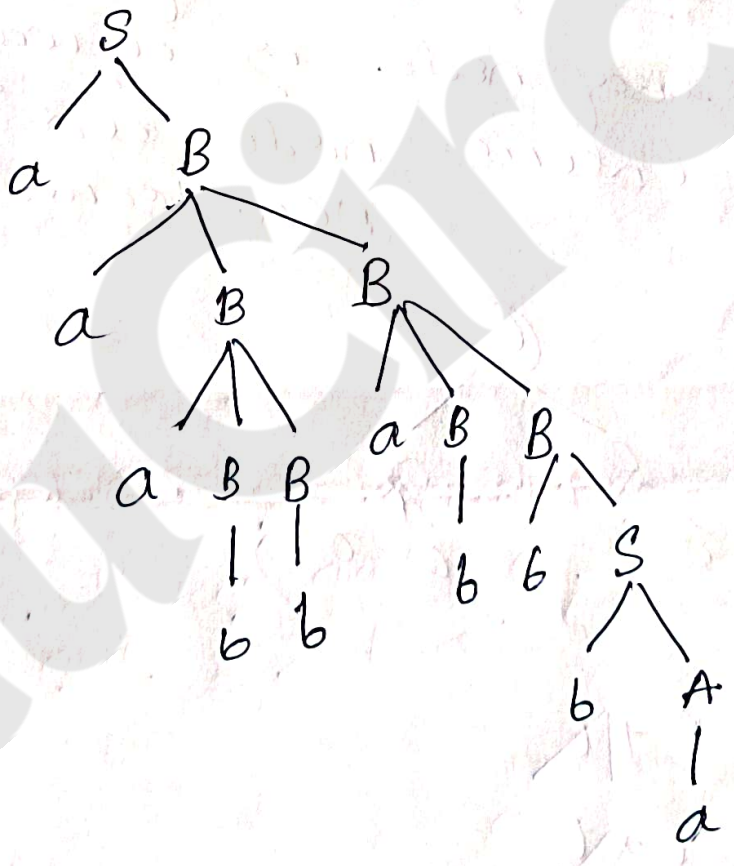
Grammer

$S \rightarrow aB \mid bA$

$A \rightarrow aS \mid bAA \mid a$

$B \rightarrow bS \mid aBB \mid b$

rightmost derivation

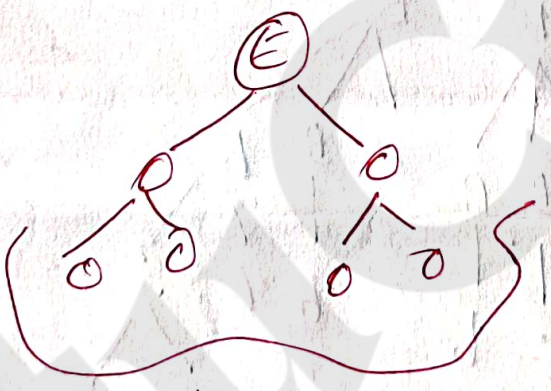


parse tree

It is a graphical representation for the derivation of the given production rule for a given CFG.

Pro..

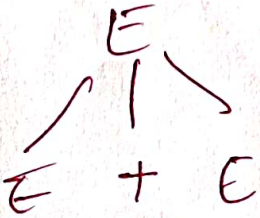
* The root node is always a node indicating start symbol
* The der is read from left to rig
* The leaf node is always terminal nodes
* The Interior nodes are always non-termina

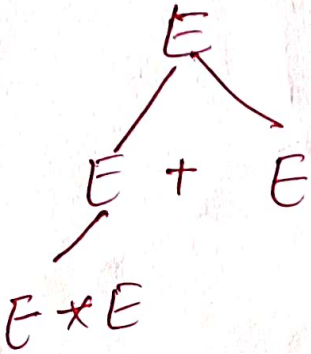


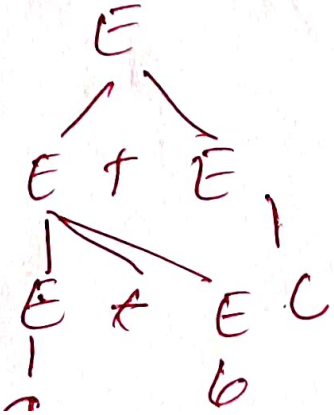
E = E + E

E = E * E

E = a | b | c







Ambiguity [ambiguous grammer]:

A grammer is said to be ambiguous if there exists, more than one leftmost derivation (or) more than one rightmost derivation (or) more than one parse tree for given input string.

⇒ If the grammer is not ambiguous then we call unambiguous, then we call

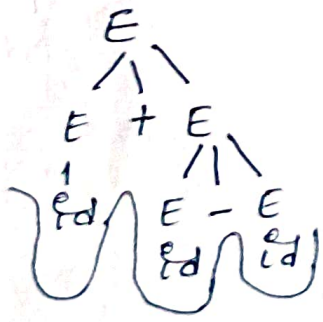
⇒ If the grammer has ambiguity, then it is not good for compiler construction

⇒ no method can automatically detect & remove the ambiguity, but we can remove ambiguity by rewriting the whole grammer without ambiguity.

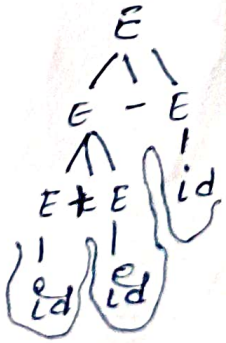
(1) check whether the grammer is ambiguous or not. the string is id+id-id

$E \to E+E$
$E \to E-E$
$E \to id$

$E \Rightarrow E+E$
$id+E$
$id+E-E$
$id+id-E$
$id+id-id$



$E \Rightarrow E-E$
$E+E-E$
$id+E-E$
$id+id-E$
$id+id-id$



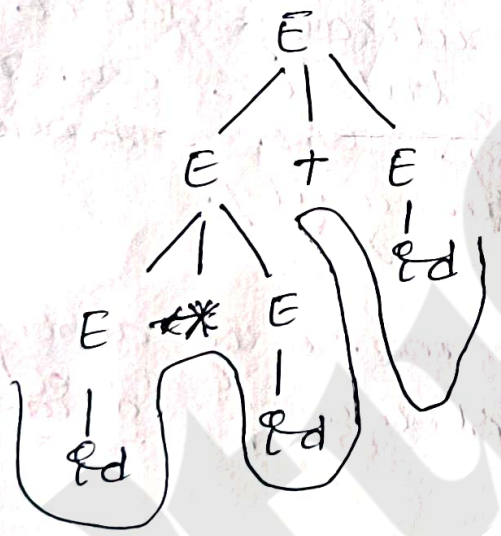
The given grammer is ambiguous.

(8) Given grammer, $E \to E+E$
$E \to E*E$
$E \to (E)$
$E \to id$

obtain derivation tree of expression
(i) id * id + id (ii) (id + id) * id

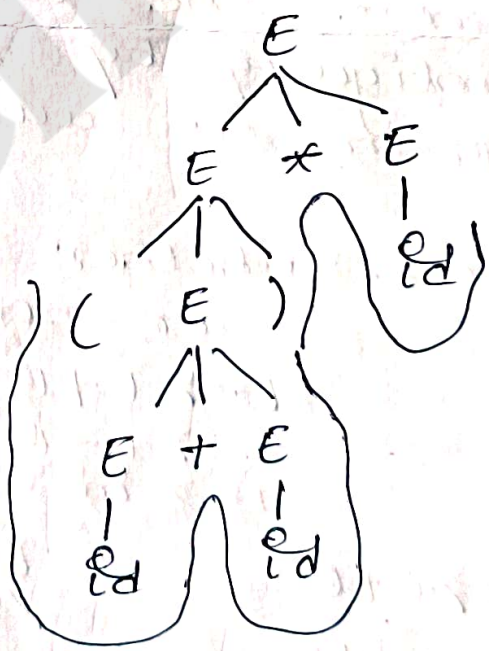
(i) id * id + id

$E \underset{lm}{\Rightarrow} E + E$
$\underset{lm}{\Rightarrow} E*E + E$
$\Rightarrow id*id + id$



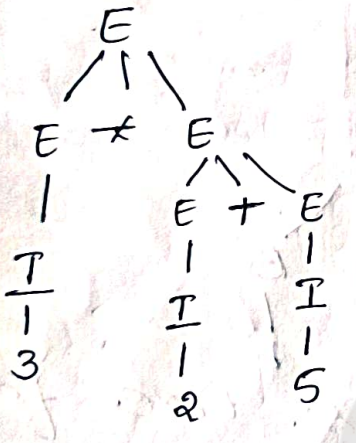
(ii) (id + id) * id

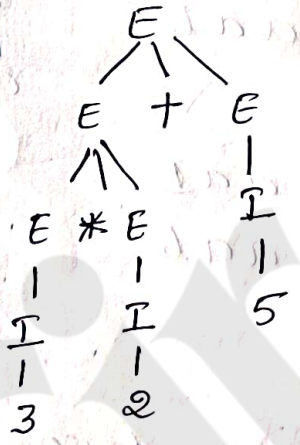
$E \underset{lm}{\Rightarrow} E*E$
$\underset{lm}{\Rightarrow} (E)*E$
$\underset{lm}{\Rightarrow} (E+E)*E$
$\underset{lm}{\Rightarrow} (id+id)*id$



② consider a grammer with production rule check whether grammer is ambiguous or not? string 3*2+5.

$E \to I$
$E \to E+E$
$E \to E*E$
$E \to (E)$
$I \to \epsilon \mid 0 \mid 1 \mid 2 \mid \ldots 9$

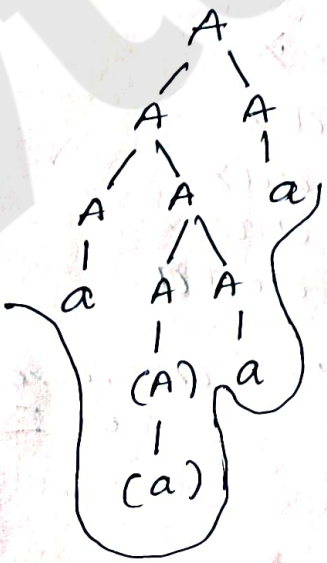




Since there are two parse for single string the grammer is ambiguous.

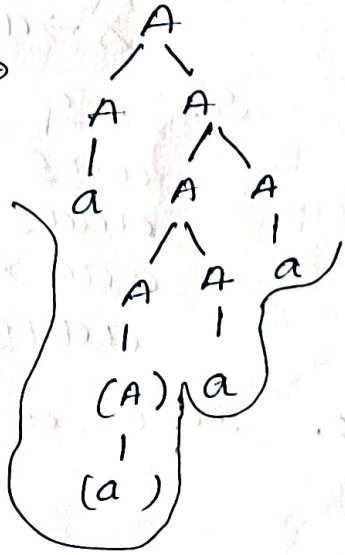
③ check whether the grammer is ambiguous or not. $A \to AA$, $A \to (A)$, $A \to a$

For the string "a(a)aa". It can generate 2 parse tree



a(a)aa

→



the given grammer is ambiguous.

④ I) the grammer ambiguous

$S \to AB \mid aaB$

$A \to a \mid Aa$

$B \to b$

String aab

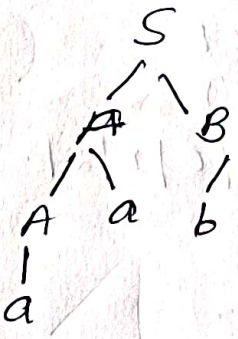
Leftmost

$S \Rightarrow AB$

$\Rightarrow AaB$

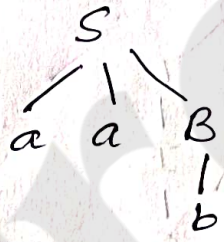
$\Rightarrow aaB$

$\Rightarrow aab$



$S \Rightarrow aaB$

$\Rightarrow aab$



Since there are two parse tree for the string aab, so given grammer is ambiguous.

⑤ I) the following grammer ambiguous

$S \to aS \mid X$

$X \to aX \mid a$

$S \Rightarrow aS$

$\Rightarrow aaS$

$\Rightarrow aaaS$

$\Rightarrow aaaX$

$\Rightarrow aaaa$

$S \Rightarrow X$

$\Rightarrow aX$

$\Rightarrow aaX$

$\Rightarrow aaaX$

$\Rightarrow aaaa$

Since two derivation ∴ ambiguous

⑥ If the following grammer ambiguous?

$S \to iC+S \mid iC+SeS \mid a$

$C \to b$

The string ibtibtaea can be obtained by applying the leftmost derivation as shown below

$S \underset{lm}{\Rightarrow} iC+S$

$\underset{lm}{\Rightarrow} ib+S$

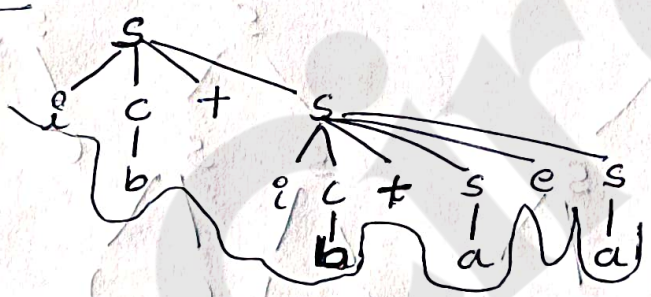
$\underset{lm}{\Rightarrow} ib+iC+SeS$

$\underset{lm}{\Rightarrow} ib+ib+SeS$

$\underset{lm}{\Rightarrow} ib+ib+aeS$

$\underset{lm}{\Rightarrow} ib+ib+aea$

parsetree



$S \underset{lm}{\Rightarrow} iC+SeS$

$\Rightarrow ib+SeS$

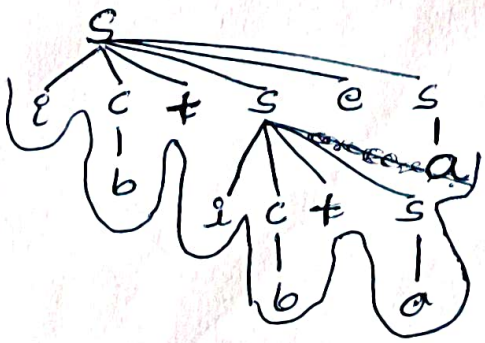
$\Rightarrow ib+iC+SeS$

$\Rightarrow ib+ib+SeS$

$\Rightarrow ib+ib+aeS$

$\Rightarrow ib+ib+aea$

parsetree

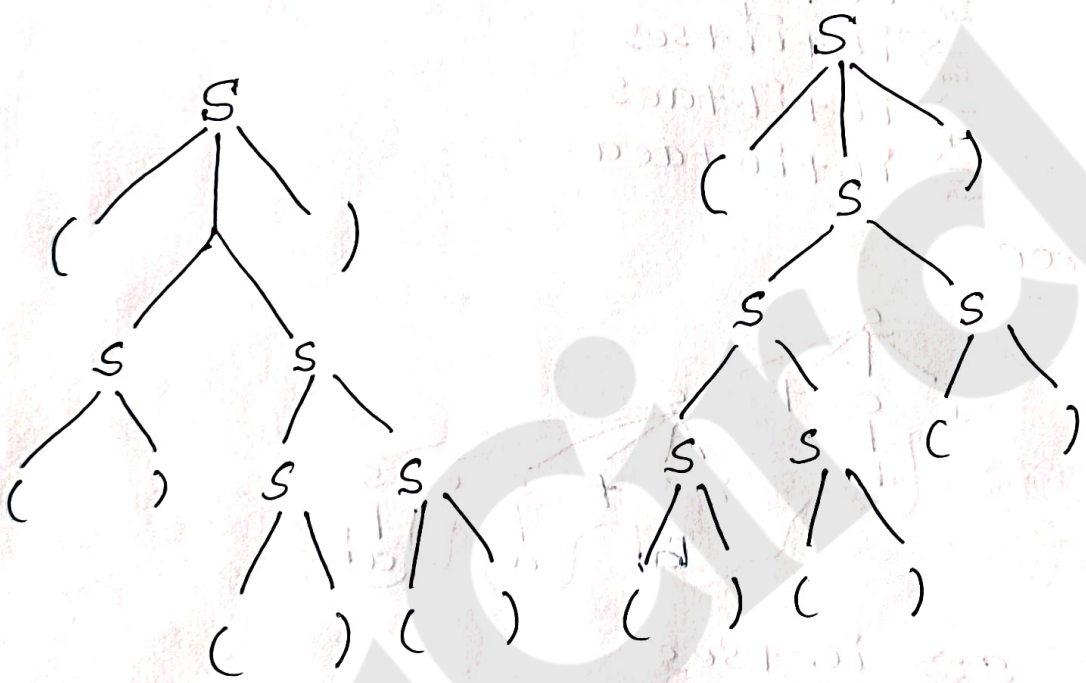


∴ since it has two parse tree for same string, so grammer is ambiguous.

(7) Show that the grammar given below to generate non empty balanced parenthesis is ambiguous.

$$S \rightarrow SS \mid (S) \mid ()$$

The sentence ( ) ( ) ( ) has two parse tree as given below.



It has more than one derivation tree is

( ) ( ) ( )

Pushdown Automata

A pushdown automata is way to implement a context free grammar in a similar way we design DFA a Regular Grammer.

⇒ A "DFA" can remember a "finite" amount of information but,

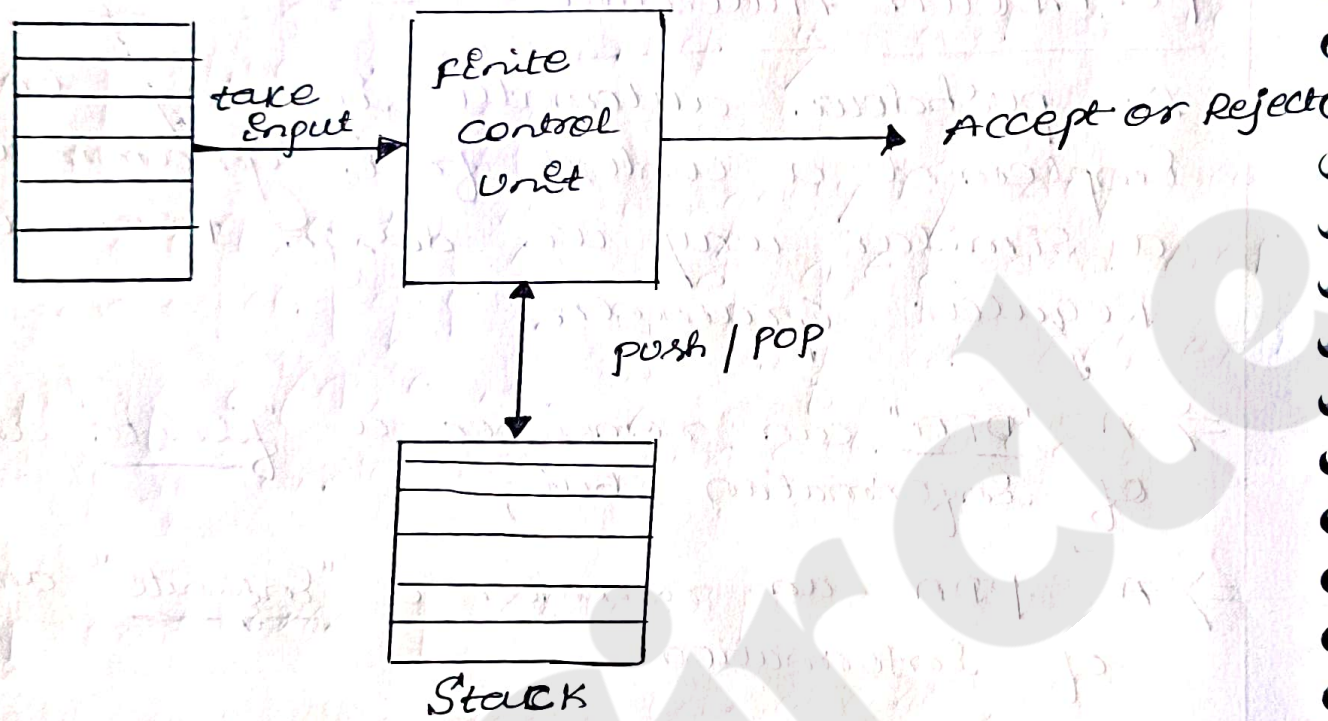
⇒ A PDA can remember a "infinite" amount of information.

Basically a PDA is

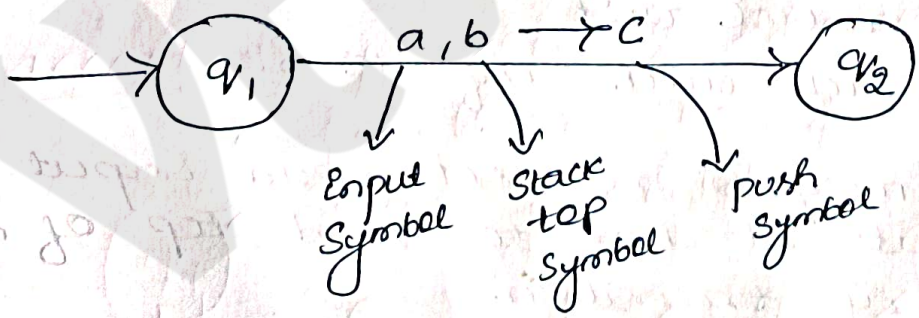
"finite state machine" + "A stack"

⇒ A PDA has 3 component
  * An input tape
  * A control unit
  * A stack with infinite size.

⇒ A PDA may or may not read input symbol but it has to read top of stack in every transition.



The following figure shows a transition in a PDA from a state q_1 to q_2 labeled as $a, b \rightarrow c$



Define PDA

PDA can be mathematically, a PDA defined

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

(i) Q is a set of states
(ii) Σ is the input symbol alphabet
(iii) Γ stack alphabet.
(iv) δ is the transition function
(v) $q_0 \in Q$ is the start state
(vi) $z_0 \in \Gamma$ is initial stack
(vii) $F \subseteq Q$ is a set of accepting states.

Terminologies:

Instantaneous Description ID (ID):

The ID of PDA is represented by triple (q, w, s) where

$q \rightarrow$ state
$w \rightarrow$ unconsume Input
$s \rightarrow$ stack content.

Turnstile notation ($\vdash$)

It is used for connecting pairs of ID's that represent one or many moves of PDA.

The process of transition denoted by turnstile symbol ($\vdash$)

$\vdash$ one move
$\vdash^*$ sequence of moves.

Deterministic PDA (pushdown Automata)

A pushdown automata is said to be deterministic iff

(i) $\delta(q, a, A)$ is uniquely by defined for each $q \in Q$ and $a \in \Sigma$ and $A \in \Gamma$

(ii) If $\delta(q, \epsilon, A)$ is defined, $\delta(q, a, A)$ is undefined for all $a \in \Sigma$.

problems:

(1) Design PDA for language,

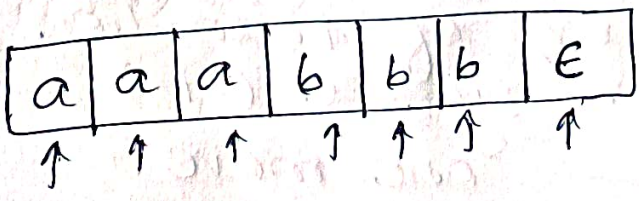
$$A^n B^n = \{ a^n b^n : n \geq 1 \}$$

sol whenever a comes push into stack
whenever b comes, pop 'a' from stack.

accept: → Input string should be consumed
→ should be in final state
→ Stack should be empty

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

$$L = \{ ab, aabb, aaabbb \ldots \}$$



$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, a, a) = (q_0, aaz_0)$

$\delta(q_0, a, a) = (q_0, aaaz_0)$ ×

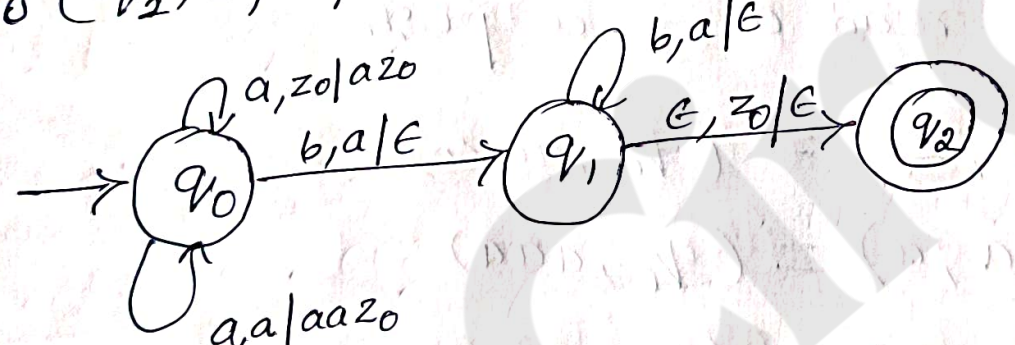
$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_2, \epsilon)$



String aabb

$(q_0, aabb, z_0) \vdash (q_0, abb, az_0)$

$\vdash (q_0, bb, aaz_0)$

$\vdash (q_0, b, az_0)$

$\vdash (q_1, \epsilon, z_0)$

$\vdash (q_2, \epsilon)$

$M = \{\{q_0, q_1, q_2\}, \{a, b\}, \{z_0 a\}, \delta, q_0, z_0, q_2\}$

(2) construct PDA $L = a^n b^{2n} \mid n \geq 1\}$

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

$L = \{abb, aabbbb \text{ - - - - - - }\}$

a	a	b	b	b	b	ϵ
↑	↑	↑	↑	↑	↑	

logic1 : read a → 2as
read b → POP

logic2 : read a → a push

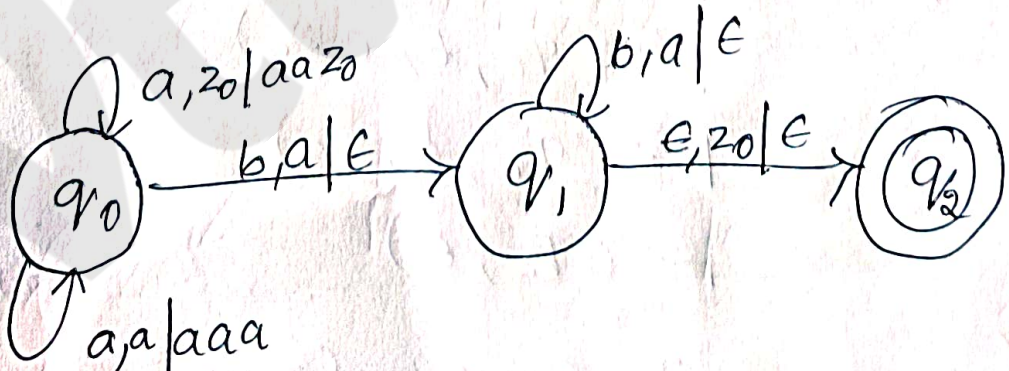
$\delta(q_0, a, z_0) = (q_0, aaz_0)$

$\delta(q_0, a, a) = (q_0, aaa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_2, \epsilon)$



$M = (\{q_0, q_1, q_2\}\ \{a, b\}\ \{z_0, a\}, \delta$
$q_0, z_0\ \{q_2\})$

(3) PDA M= construction PDA lang $L=\{wcw^R | w\in(a+b)^*\}$

$M=(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

$L=\{abacaba, ababacbaba \cdots\}$

$\delta(q_0, a, z_0) = (q_0, az_0)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, b, b) = (q_0, bb)$

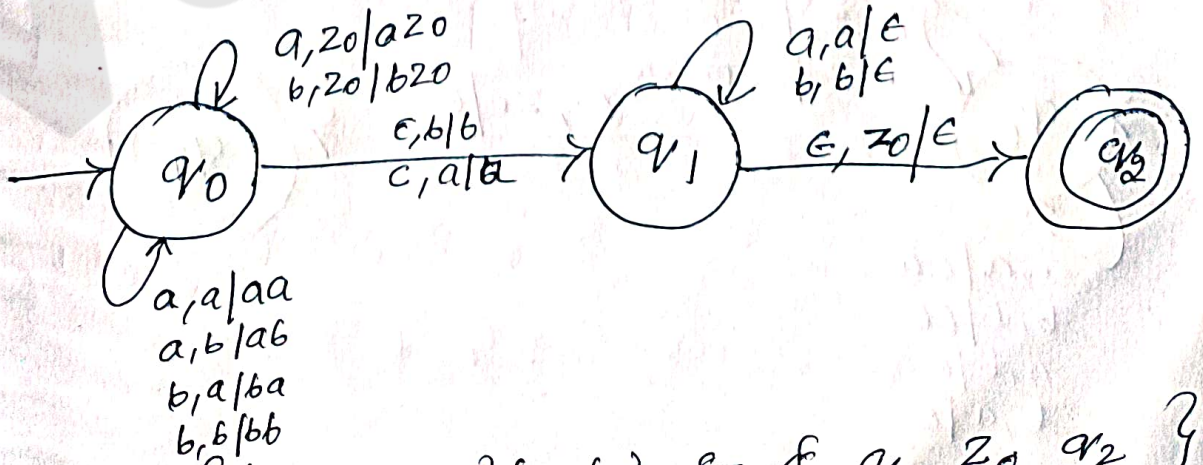
$\delta(q_0, c, a) = (q_1, a)$

$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_1, a, a) = (q_1, \epsilon)$

$\delta(q_1, b, b) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_2, \epsilon)$



$M=\{\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, z_0, q_2\}$ //

④ Construct PDA for $L = \{ w / n_a(w) = n_b(w) \}$

$L = \{ \epsilon, ab, aabb, abab, abba, bbaa \cdots \}$

1. stack empty a, b push
2. top → a input a — push
3. top → a input b — pop
3. top → b input b — push
   input a — pop

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

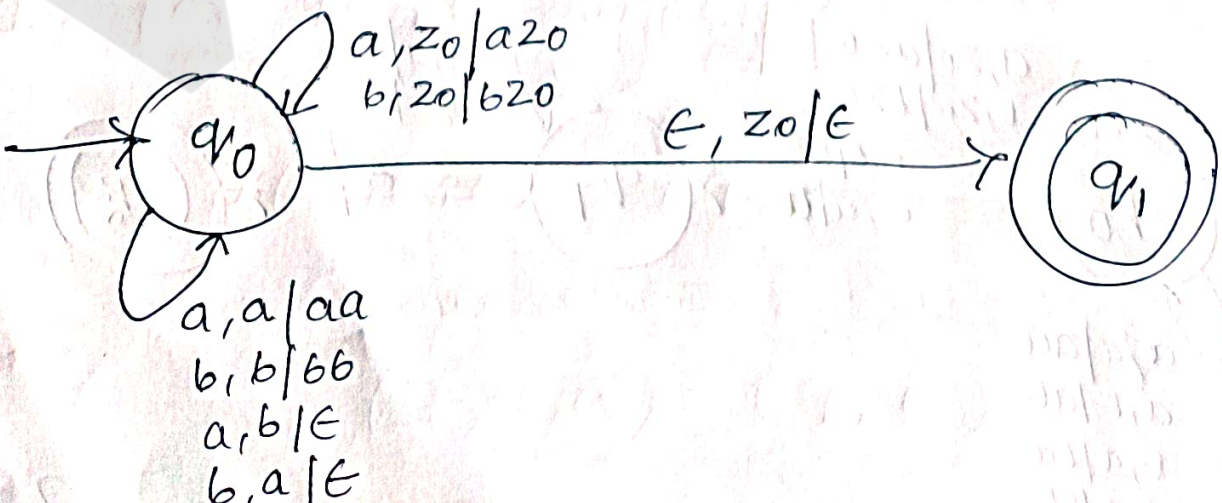
$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, b) = (q_0, bb)$

$\delta(q_0, a, b) = (q_0, \epsilon)$

$\delta(q_0, b, a) = (q_0, \epsilon)$

$\delta(q_0, \epsilon, Z_0) = (q_1, \epsilon)$



PDA $-M = (\{q_0, q_1\}, \{a, b\}, \{a, b, z_0\}, \delta, \{q_0\}, z_0, \{q_2\})$

(5) construct PDA $A^n B^n = \{a^n b^n \mid n \geq 0\}$

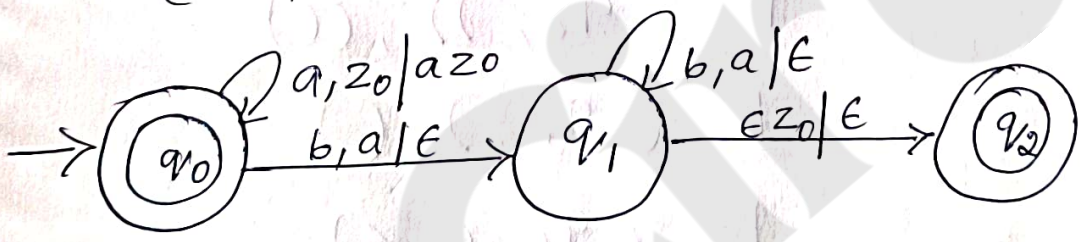
$L = \{\epsilon, aabb, aaabbb \cdots\}$

$\delta(q_0, a, z_0) = (q_0, a z_0)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) = (q_2, \epsilon)$



$(q_0, aabb, z_0) \vdash (q_0, abb, a z_0)$

$\vdash (q_0, bb, aa)$

$\vdash (q_1, b, a)$

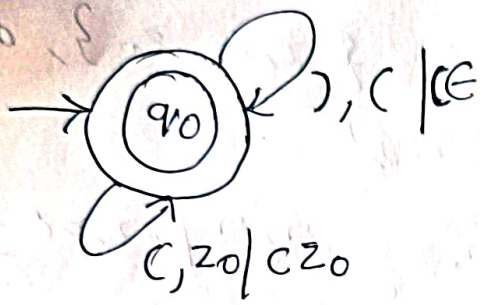
$\vdash (q_1, \epsilon, \epsilon)$

(6) Design PDA for balanced parenthesis $\{w \in \{), (\}^*\}$.

$\{(), (())^\checkmark, ()()^\checkmark, (()^\times\}$

logic: (push into stack

) pop from stack.



$\delta(q_0, (, z_0) = (q_0, (z_0)$

$\delta(q_0,), () = (q_0, \epsilon)$

$M = \{\{q_0\}, \{(,)\}, \delta, z_0, \Gamma, F\}$

$\delta(q_0, (()), z_0) \vdash (q_0, \underline{(})), ()$

$\vdash (q_0,)), (()$

$\vdash (q_0,), ())$

$\vdash (q_0, \epsilon, \epsilon)$

Equivalence of PDA :

construct CFG to PDA

1. For Non terminal | variable

$A \to \alpha$

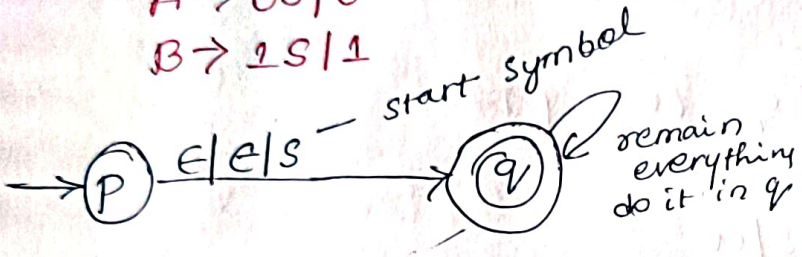
$\delta(q, \epsilon, A) = (q, \alpha)$

2) For terminal

$\delta(q, a, a) = (q, \epsilon)$

(1) construct PDA for grammer:

$S \to AB$
$A \to 0S \mid 0$
$B \to 1S \mid 1$



$\delta(q, \epsilon, S) = (q, AB)$
$\delta(q, \epsilon, A) = (q, 0S)$
$\delta(q, \epsilon, A) = (q, 0)$
$\delta(q, \epsilon, B) = (q, 1S)$
$\delta(q, \epsilon, B) = (q, 1)$
$\delta(q, 0, 0) = (q, \epsilon)$
$\delta(q, 1, 1) = (q, \epsilon)$

(2) Construct PDA for grammer

$E \to E+T \quad E \to T \quad T \to T*F \quad T \to F \quad F \to (E), F \to id$

$\delta(q, \epsilon, E) = (q, E+T)$
$\delta(q, \epsilon, E) = (q, T)$
$\delta(q, \epsilon, T) = (q, T*F)$
$\delta(q, \epsilon, T) = (q, F)$
$\delta(q, \epsilon, F) = (q, (E))$
$\delta(q, \epsilon, F) = (q, id)$
$\delta(q, id, id) = (q, \epsilon)$
$\delta(q, (, () = (q, \epsilon)$
$\delta(q,),)) = (q, \epsilon)$

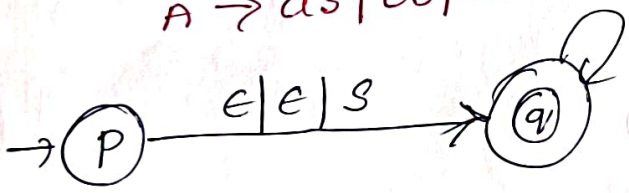
$\delta(q, +, +) = (q, \epsilon)$

$\delta(q, *, *) = (q, *)$

(3) Construct CFG to PDA.

$S \rightarrow aAA$

$A \rightarrow aS \mid bS \mid a$



$\delta(q, \epsilon, S) = (q, aAA)$

$\delta(q, \epsilon, A) = (q, aS)$

$\delta(q, \epsilon, A) = (q, bS)$

$\delta(q, \epsilon, A) = (q, a)$

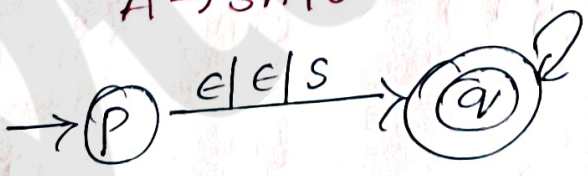
$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

(4) Construct CFG to PDA

$S \rightarrow AA \mid a$

$A \rightarrow SA \mid b$



$\delta(q, \epsilon, S) = (q, AA)$

$\delta(q, \epsilon, S) = (q, a)$

$\delta(q, \epsilon, A) = (q, SA)$

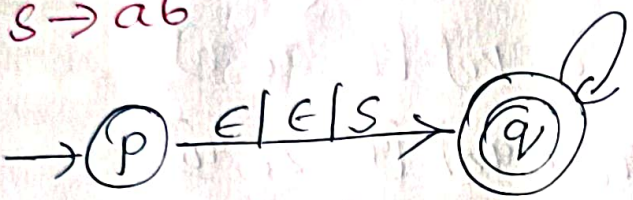
$\delta(q, \epsilon, A) = (q, b)$

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

(5) $S \to aSb$ convert into PDA.

$S \to ab$



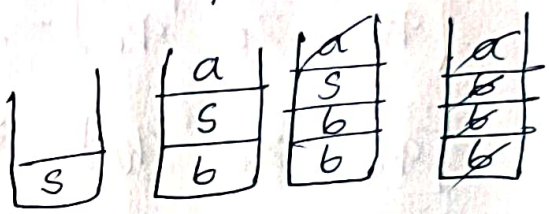
$\delta(q_1, \in, S) = (q_1, aSb)$

$\delta(q_1, \in, S) = (q_1, ab)$

$\delta(q_1, a, a) = (q_1, \in)$

$\delta(q_1, b, b) = (q_1, \in)$

ex: aaabbb



Construction from PDA to CFG

(1) Design PDA to CFG,

$M = (\{q_0, q_1\}, \{a, b\}, \{a z_0\}, \delta, q_0, z_0, F)$

$\delta(q_1, a, z_0) = (q_0, a z_0)$

$\delta(q_0, b, a) = (q_1, a)$

$\delta(q_1, a, a) = (q_1, \in)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_1, b, a) = (q_1, a)$

$\delta(q_1, \in, z_0) = (q_1, \in)$

Solution

$CFG(G) = (V, T, P, S)$

$V = \{S \cup T\}$

$\therefore V = (S, [q_0, a, q_0] [q_0, a, q_1] [q_1, a, q_0] [q_1, a, q_1]$
$[q_0, z_0 q_0] [q_0 z_0 q_1] [q_1, z_0 q_0] [q_1, z_0 q_1]$

$T = \{a, b\}$

$S \to [q_0, z_0, q_0]$

$S \to [q_0, z_0, q_1]$

① $\delta(q_0, a, z_0) = (q_0, a\ z_0)$

$[q_0\ z_0\ q_0] \to a\ [q_0\ a\ q_0]\ [q_0, z_0\ q_0]$

$[q_0\ z_0\ q_0] \to a\ [q_0, a, q_1]\ [q_1, z_0, q_0]$

$[q_0\ z_0\ q_1] \to a\ [q_0, a, q_0]\ [q_0, z_0, q_1]$

$[q_0\ z_0\ q_1] \to a\ [q_0, a, q_1]\ [q_1, z_0, q_1]$

same

same

② $\delta(q_0, a, a) = (q_0, aa)$

$[q_0, a, q_0] \to a\ [q_0, a, q_0]\ [q_0, a, q_0]$

$[q_0, a, q_0] \to a\ [q_0, a, q_1]\ [q_1, a, q_0]$

$[q_0, a, q_0] \to a\ [q_0, a, q_0]\ [q_0, a, q_1]$

$[q_0, a, q_0] \to a\ [q_0, a, q_1]\ [q_1, a, q_1]$

③ $\delta(q_0, b, a) = (q_1, a)$

$[q_0, a, q_0] \to b\ [q_1, a, q_0]$

$[q_0, a, q_1] \to b\ [q_1, a, q_1]$

④ $\delta(q_1, b, a) = (q_1, a)$

$[q_1, a, q_0] \to b\ [q_1, a, q_0]$

$[q_1, a, q_1] \to b\ [q_1, a, q_1]$

(5) $\delta(q_1, a, a) = (q_1, \epsilon)$

$[q_1, a, q_1] \rightarrow a$

(6) $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

$[q_1, z_0, q_1] \rightarrow \epsilon$ //

(2) construct CFG from given PDA

$M = (\{q, p\}, \{0, 1\}, \{x, z\}, \delta, q_0, z, F)$

$\delta(q, 1, z) = (q, xz)$ $\quad \delta(q, 0, x) = (p, x)$

$\delta(q, 1, x) = (q, xx)$ $\quad \delta(p, 1, x) = (p, \epsilon)$

$\delta(q, \epsilon, x) = (q, \epsilon)$ $\quad \delta(p, 0, z) = (q, z)$

Sol CFG $(G) = (V, T, P, S)$

$V = \{S, [q_0, x, p], [q, x, q], [p, x, p], [p, x, q], [q, z, p], [q, z, q], [p, z, p], [p, z, q]\}$

$T = \{0, 1\}$

$S \rightarrow [q_0, z, q]$

$S \rightarrow [q_0, z, p]$

① $\delta(q, 1, z) = (q, xz)$

$[q, z, q] \to 1\ [q, x, q]\ [q, z, q]$

$[q, z, q] \to 1\ [q, x, p]\ [p, z, q]$

$[q, z, p] \to 1\ [q, x, q]\ [q, z, p]$

$[q, z, p] \to 1\ [q, x, p]\ [p, z, p]$

② $\delta(q, 1, x) = (q, xx)$

$[q, x, q] \to x\ [q, x, q]\ [q, x, q]$

$[q, x, q] \to x\ [q, x, p]\ [p, x, q]$

$[q, x, p] \to x\ [q, x, q]\ [q, x, p]$

$[q, x, p] \to x\ [q, x, p]\ [p, x, p]$

③ $\delta(q, \epsilon, x) = (q, \epsilon)$

$[q, x, q_1] \to \epsilon$

④ $\delta(q, 0, x) = (p, x)$

$[q, x, q] \to 0\ [p, x, q]$

$[q, x, p] \to 0\ [p, x, p]$

⑤ $\delta(p, 0, z) = (q, zz)$

$[p, z, q] \to 0\ [q, z, q]$

$[p, z, p] \to 0\ [q, z, p]$

⑥ $\delta(P, \frac{1}{\cdot}, X) = (P, \epsilon)$

$[P, X, P] \rightarrow 1$ //

$P = \{ S \rightarrow Aa \mid aB \mid b \mid Ca$

$B \rightarrow aB \mid b$

$C \rightarrow Db \mid ab \mid d$

$D \rightarrow d \mid ab$

$E \rightarrow ab \}$

S is Start Symbol.