Unit-3

* Context Free Language *

Ambiguity in CFG:-

A terminal string $w \in L(G)$ is ambiguous if there exist two (or) more derivations for w (or there exist two (or) more left most derivations of w)

A Context-Free Grammar, G is ambiguous if there exists some $w \in L(G)$, which is ambiguous

1) Example:-

Let $G = (\{S\}, \{a, b, +, *\}, P, S)$ where P consists of $S \rightarrow S+S/S*S/a/b$. The derivation tree for a+a*b will be as follows

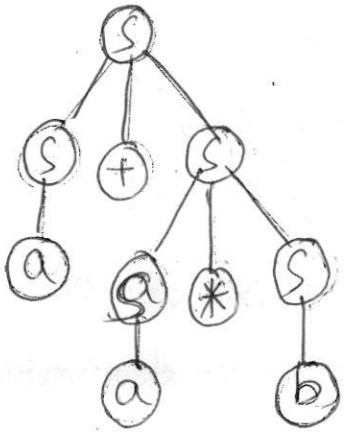


Fig:- one kind of derivation tree

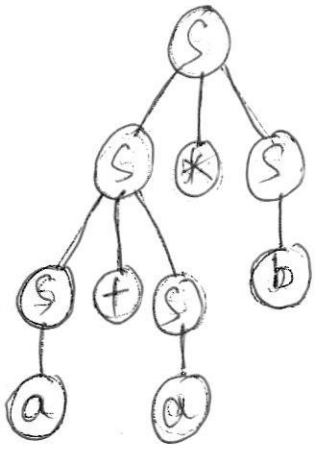


Fig:- Second kind of derivation tree

The left most derivations induced by the two derivation trees are:-

① $S \rightarrow \underline{S}+S$

$\rightarrow a+\underline{S}$ [$\because S \rightarrow a$]

$\rightarrow a+\underline{S}*S$ [$\because S \rightarrow S*S$]

$\rightarrow a+a*\underline{S}$ [$\because S \rightarrow a$]

$\rightarrow a+a*b$

② $S \rightarrow \underline{S}*S$ [$\because S \rightarrow S+S$]

$\rightarrow S+S*S$ [$\because S \rightarrow a$]

$\rightarrow a+S*S$

$\rightarrow a+a*S$ [$\because S \rightarrow b$]

$\rightarrow a+a*b$

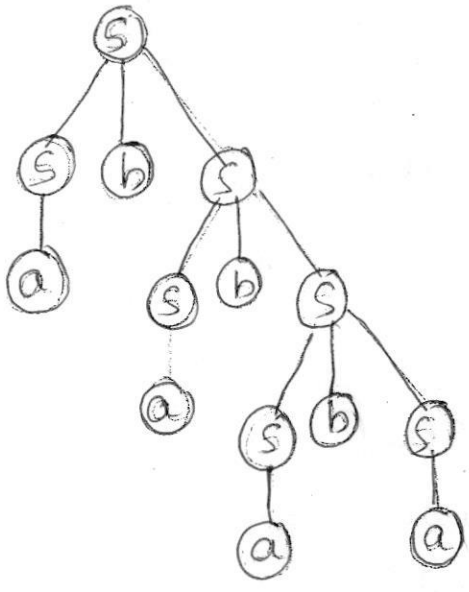
By definition of ambiguous grammar, there exists two derivation trees for the given grammar. So, there is ambiguity in the grammar

(2) (2) If G is the grammar $S \rightarrow SbS/a$, show that G is ambiguous

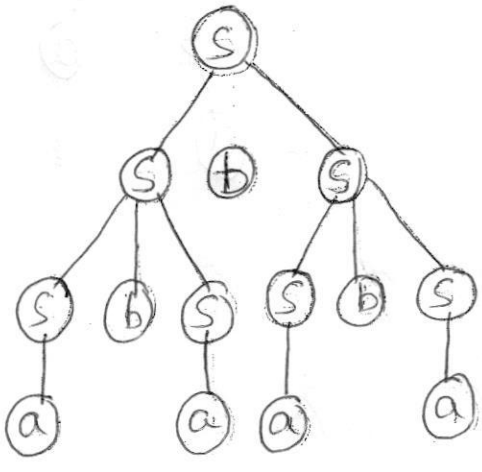
Sol:- To prove that G is ambiguous, we have to find a $w \in L(G)$, which is ambiguous.

Consider, $w = abababa \in L(G)$. The derivation trees that are possible for this is.

(1)



(2)



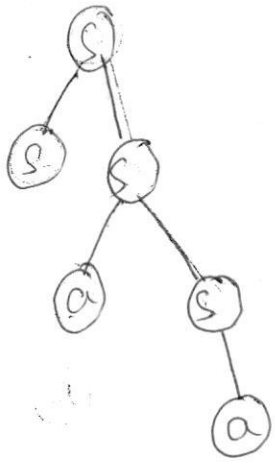
Two derivation trees exists for the string abababa. So, the grammar is ambiguous

(3) Show That the language of all non-null strings of a's can be defined by CFG as follows

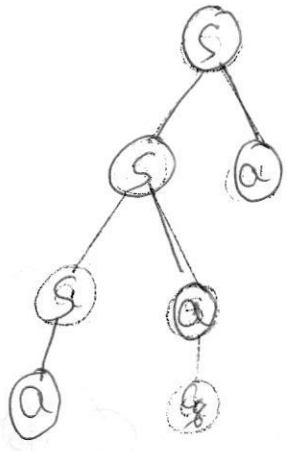
$S \rightarrow aS/Sa/a$ is ambiguous

Sol:- Let us consider, w as a^3. The possible derivations are:-

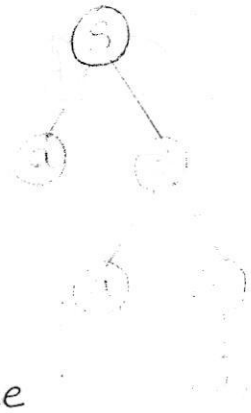
① $S \rightarrow aS$ [$\because S \rightarrow Sa$]
$\rightarrow aSa$ [$\because S \rightarrow a$]
$\rightarrow aaa$



② $S \rightarrow Sa$ [$\because S \rightarrow aS$]
$\rightarrow aSa$ [$\because S \rightarrow a$]
$\rightarrow aaa$



③ $S \rightarrow aS$ [$\because S \rightarrow Sa$]
$\rightarrow aaS$ [$\because S \rightarrow a$]
$\rightarrow aaa$



There are many possible solutions (more than 1) for the string aaa. So, the grammar has ambiguity

(4) Consider the CFG $S \rightarrow aSb/SS/\epsilon$ and show that the word aabb is ambiguous.

Sol:- Given,

$$S \rightarrow aSb/SS/\epsilon$$

① $S \rightarrow aSb$

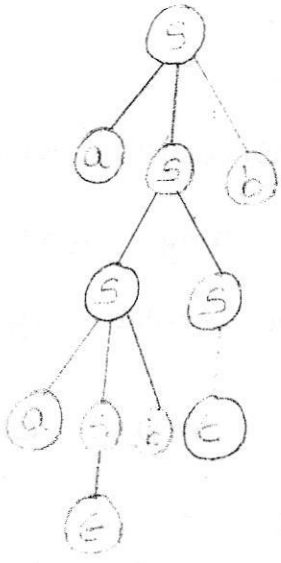
$\rightarrow aSSb$

$\rightarrow aaSbSb$

$\rightarrow aa\epsilon bSb$

$\rightarrow aab\epsilon b$

$\rightarrow aabb$



② $S \rightarrow SS$

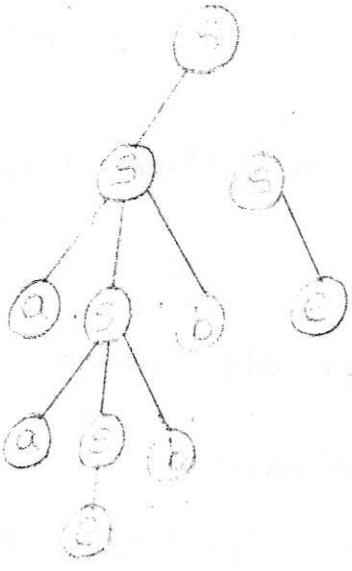
$\rightarrow aSbS$

$\rightarrow aaSbbS$

$\rightarrow aa\epsilon bbS$

$\rightarrow aabb\epsilon$

$\rightarrow aabb$



The word/string aabb has more than one derivation tree as well as more than one leftmost derivation. So, the word is ambiguous & grammar has ambiguity.

Reduction of CFG:-

Let L be the non-empty context free language, then it can be generated by a context free grammar, G with the following properties:-

(a) we must eliminate useless symbols, those variables (or) terminals that does not appear in any derivation of terminal string from the start symbol.

(b) we must eliminate ϵ-productions, those of the form $X \to \epsilon$, for some variable X

(c) we must eliminate unit productions, those of the form $X \to Y$, for some variables X & Y.

Eliminating useless Symbols:-

In this, first we need to identify the symbols, which donot play any role in the derivation of any string, w in $L(G)$. Next, eliminate the identified production, which contains useless symbols from the CFG.

A symbol α in a CFG is useful if and only if

(a) $\alpha \overset{*}{\Rightarrow} w$, where $w \in L(G)$ and w in V_t^*, i.e,

Push Down Automata :- (PDA)

→ In order to accept CFL, we need a m/c called PDA.

PDA = FA to which some mem. element is added & that mem. element is stack

PDA = FA + stack

→ PDA is defined by 7-tuples $(Q, \Sigma, \delta, q_0, z_0, F, \Gamma)$

Q : Set of states

Σ : I/P symbol

δ : Transition function

q_0 : Initial state

z_0 : Bottom of the stack

F : Set of final states

Γ : Stack alphabet → symbols allowed to be present on top of stack.

F: Set of final states, which mean, we might use final states to repr. whether a string is accepted by PDA or not.

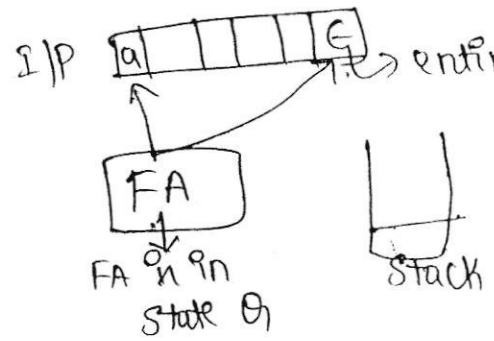
* Sometimes, this can be empty also, which mean that without a final state also we could accept some strings.

→ PDA in 2 types :

1) Deterministic PDA
2) Non-Deterministic PDA

→ In DPDA, $\delta : Q \times \{\Sigma \cup \in\} \times \Gamma \rightarrow Q \times \Gamma^*$

If we are in a state (Q), & if we see some I/P alphabet or you see $\in \{\Sigma \cup \in\}$ & if we see (Γ) one symbol from top of the stack ⇒ it will go to some state (or) same state (Q) & print/push something on to the top of the stack (Γ^*)



FA present state i/p q_1 &
it look into I/P a (δ) $\in$ (entire I/P i/p over) $(\Sigma \cup \epsilon)$ &
it look at the top of stack Γ
Depending on above three, it will decide what to do.
it could either go to some other state / in same state &
print / push something on to the top of the stack.

→ In NPDA,

$$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \longrightarrow 2^{(Q \times \Gamma^*)}$$

it decides to go to more than one state & print / push more than one symbol on to top of the stack.

Ex:- $L = a^n b^n \mid n \geq 1$

we have to see all a's 1st & count a's & whenever we see a 'b', we have match b's against a's

Let $L = aabb$
whenever we see 'a', push it
" " " 'b' pop 'a'. & once we see '$', ie, entire string i/p over & if top of stack = z_0, we can stop the procedure.

Implement this, using PDA,
whenever we say z_0 i/p top of the stack, remember that it is no longer on top of stack, I have removed out & i'm holding it in my hand, next if you have to push anything on to top of stack, we have to push z_0 1st

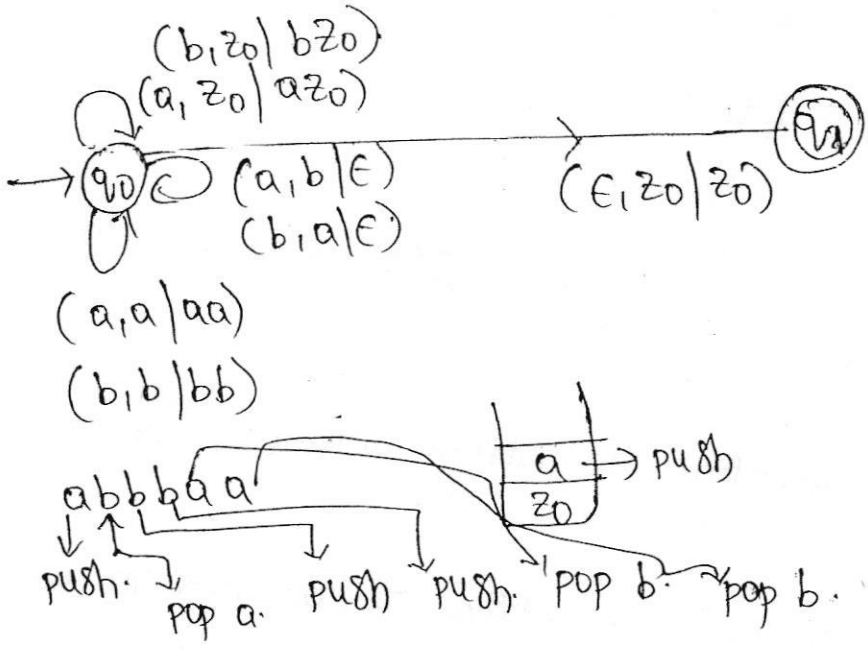
(6) Ex:- $\omega \mid n_a(\omega) = n_b(\omega)$

we have to count all a's & match them against b's.

a, b can come in any order.

So, previous procedure of pushing a's & poping b's for every a is not a valid approach.

Instead of that, if we see 'a' we push the symbols, when you don't have chance of poping & pop " " " " have chance of poping.



1) BR 8) CN
2) AM 9) 69.
3) DA 10) DN
4) AE 11) BG
5) CD
6) CX
7) CA

(7) Ex:- $a^n b^n c^m \mid n, m \geq 1$

a's & b's have to be compared & they should be equal

∴ For every a, we will push a,
" " b, " " pop a.

we need not to count 'c' at all, $c^m = c^*$ BD, BL, BR, AM, BH, C9, CM, CT, CC, DE, DA, AG, AN, B8, AP, DQ, CG, DV, AC, CR, AV, BI, AR, BG, FR 'c', we need not require PDA, FA is enough. CN, C9, DN, CD, CX, BG, BK, B3, 73, AP, B7

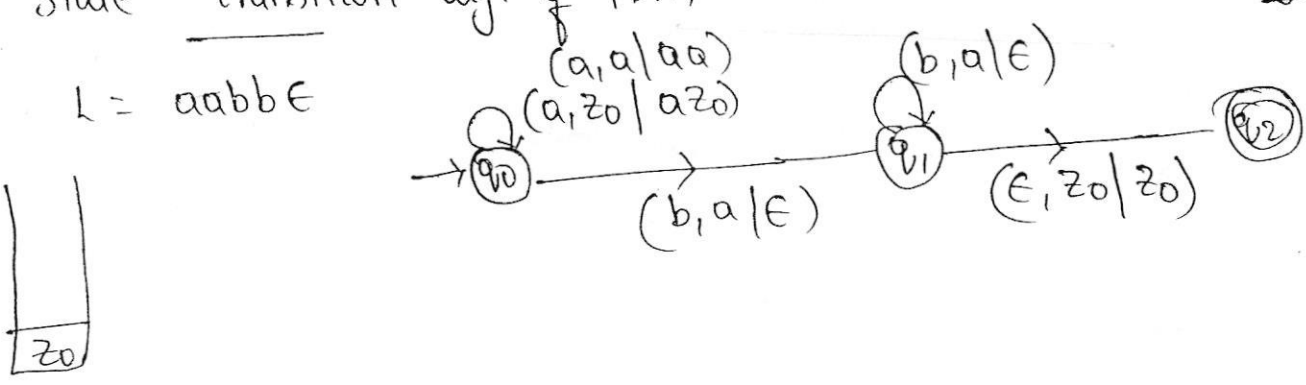
compare a's & b's & simply reach 'c' & these should be atleast one 'c'.

So, if we see one 'c', go to final state & after that any no. of c's may come

State transition dig: of PDA,

Fig 1

$L = aabb\epsilon$



$ Transition function for PDA :-

Another way of repr. PDA

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

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

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

$\delta(q_1, b, a) = (q_1, \epsilon)$ (&) $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

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

→ Here, we are accepting the strings, whenever we reach the final state.

→ Another way of accepting a string is, no need to use the final state, we can just empty the stack.

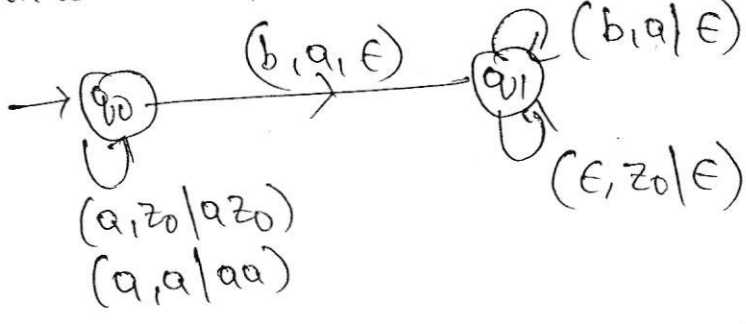


Fig (2)

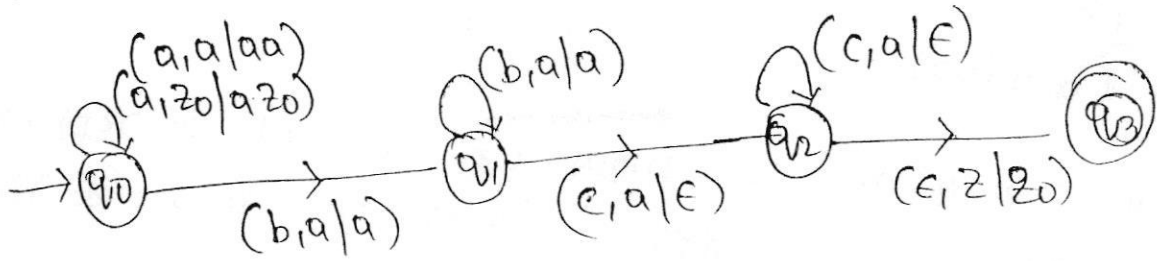
∴ String acceptance by PDA is

1) acceptance by final state (Fig 1)

2) " empty stack. (Fig 2)

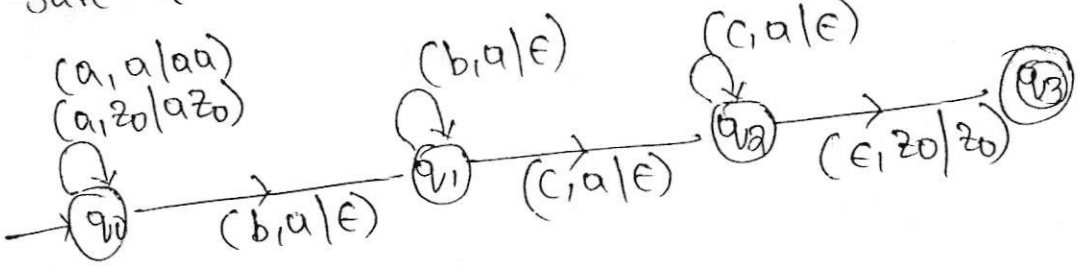
(8a) Ex:- $a^n b^m c^n \mid n, m \geq 1$

a's have to be matched against 'c' & b's can be left alone.



(8b) Ex:- $a^{m+n} b^m c^n \mid m, n \geq 1$

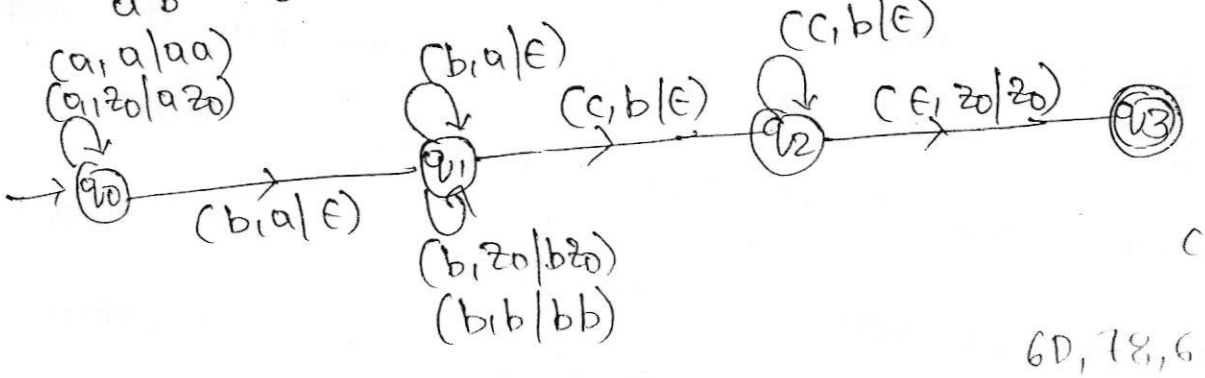
Save & count all a's & match them against b's & c's.

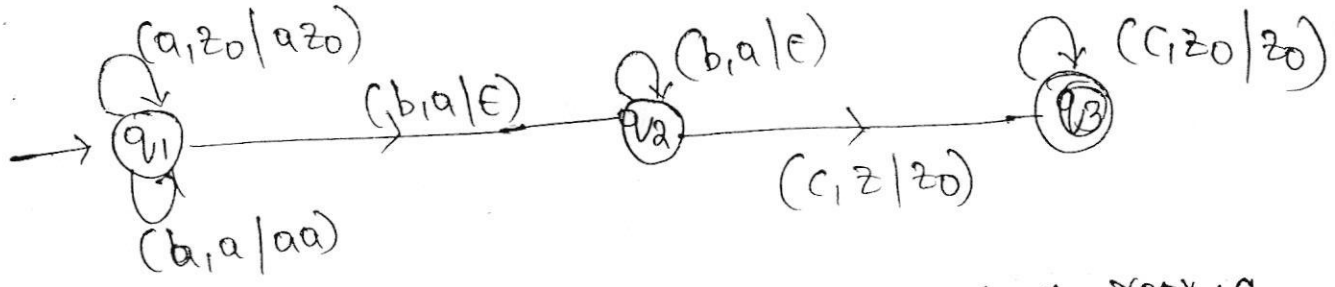


(8c) Ex:- $a^n b^{m+n} c^m \mid m, n \geq 1$

whenever we see 'a', push them, whenever you see b's, we have to match them against a's & remaining b's you push them on to the stack & match them against c's.

$a^n b^{m+n} c^m = a^n b^n b^m c^m$





As in case of DFA, in DPDA we no need to repr. a dead state, if a string not in the lang in there, PDA will halt at the state

if string = bc

at q_1, we haven't shown transition when for (b, Z_0) $\Rightarrow$ ∴ PDA will halt at q_1 itself.

if string = aabcc.

a → pushed

a → pushed

b → a is poped.

next I/P = c, top = a.

we haven't shown transition for $\delta(c, a)$

∴ PDA will halt at q_2

ie, if I/P is present in the lang. we will reach final state,

if " " not there in the lang, then PDA will come across dead configuration & it is going to halt.

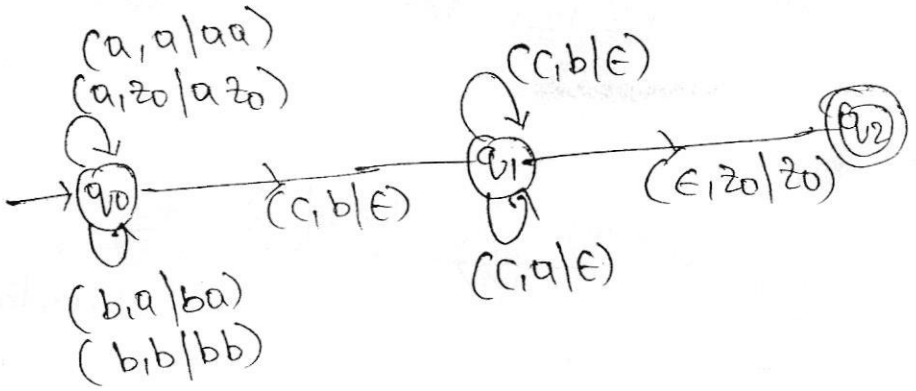
we will halt either at final state → accepting string.

" " " ∴ non-final " → not accepting

PDA will never go to infinite loop.

(8d) Ex:- $a^n b^m c^{n+m} \mid n, m \geq 1$

push a's, push b's & then pop c's for every a & b



Examples:-

1) $a^n b^n c^m d^m$ 2) $a^n b^m c^m d^n$ 3) $a^n b^m c^n d^m$ (not CFL)

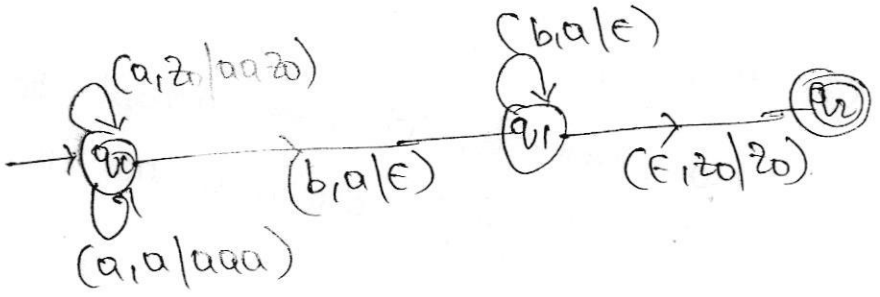
4) $a^n b^{2n} \mid n \geq 1$ 5) $a^n b^{2n+1}$.

a) one sol. is, whenever you see 'a', instead of pushing one a, push 2 a's on to the stack.

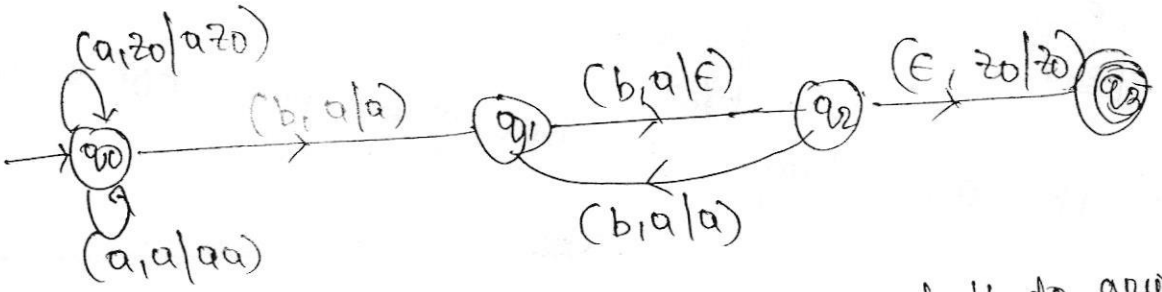
b) another sol. is, for every 2 b's, pop one 'a'.

abb, aabb, aaabbbbbb, ---

a)



b)

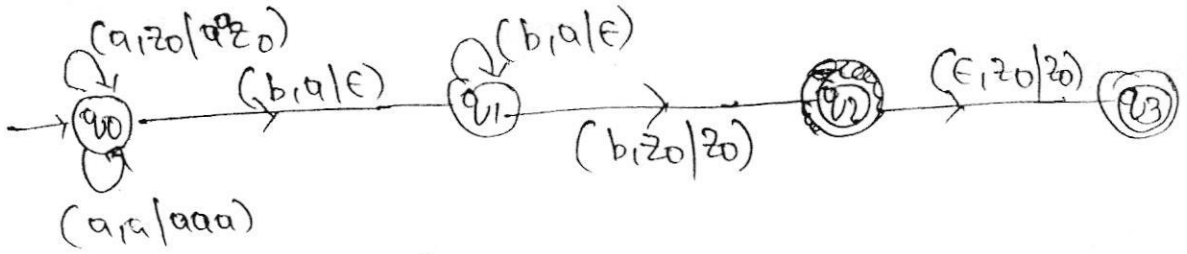


For every 2 b's, pop one a, mean, don't do anything

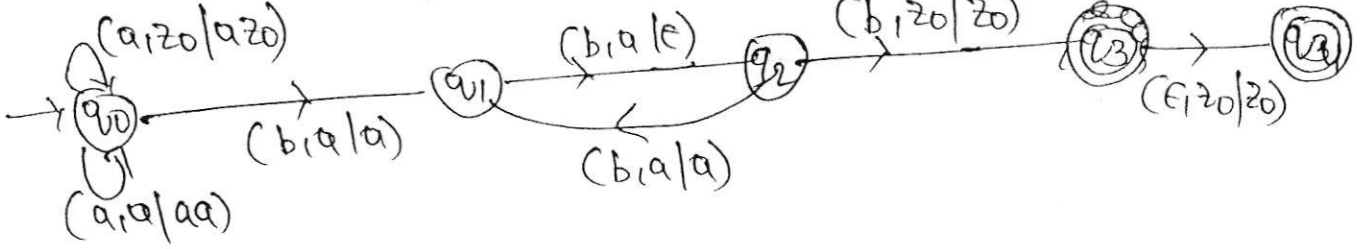
for 1st b, for every 2nd b pop 'a'

" 3rd " , " 4th " " "

Ex:- $a^n b^{2n+1} = a^n b^{2n} b$



(or)



(or)

$a^n b^{2n+1} = a^n b \cdot b^{2n}$

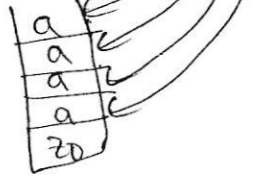
Ex:- $a^n b^n c^n / n \geq 1$

if we push a's, pop b's & match them against b's, by the time we reach 'c', they will be nothing to compare against c's

So, instead of pushing one a, push 2 a's & match them against b's & c's.

By we do like this, if

$w = aabbcc$



but prob. is, it even accept the $w = aabccc$ which is not in lang.

∴ for given lang. we can't give PDA. not a CFL.

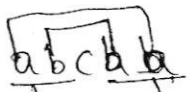
(9) Ex:- $wcw^R \mid w \in (a,b)^+$

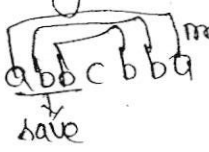
w^R = Reverse of string

wcw^R = set of all odd palindromes

aabcbaa , abacaba, baacaab, ...

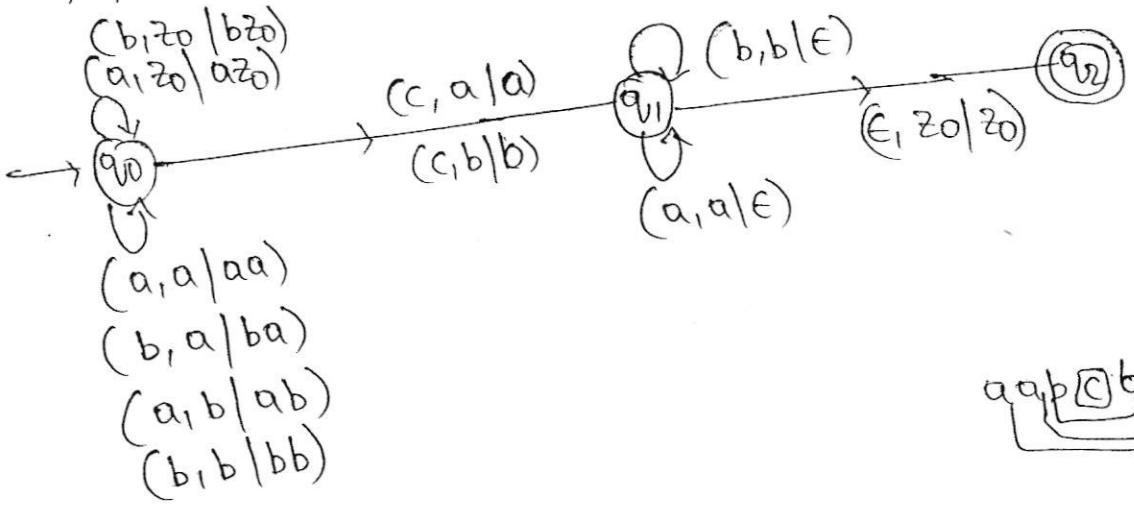
push w, c, & pop w^R matching correspondingly.





how to know,
when to push w & when to pop w^R ?

in center we have 'c'. so when you keep on pushing 'w' until you see center (c), when you see a center, pop w correspondingly. No ambiguity.



q_0 → pushing entire 'w'.

whenever we see a 'c', don't do anything.

Equivalence of CFG & PDA

Theorem:

→ A lang. is context free iff some PDA recognizes it.

Proof: 1) Given a CFG, show how to construct a PDA th recognizes it

2) Given a PDA, show how to construct a CFG that recogni the same lang.

Converting CFG to PDA:-

→ In this, the PDA we r going to construct is a NDPDA & this NPDA accepts the I/P by empty stack condition & (assumptions) we always have z_0 at the bottom of stack.

→ with these assumptions, we are going to construct a PDA for a given CFG.

→ CFG can be expressed as CFG = (V, T, P, S) &

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

→ So we need to extract 7 tuples from 4 tuples.

→ Here, in PDA we are going to construct have "one" state

∴ $Q = \{q\}$

Σ = I/P symbols - terminals in CFG

Γ = stack symbols: $z_0 \cup \{T\} \cup \{V\}$

$q_0 = q$

$F = \phi$ [∵ PDA here accepts I/P by empty stack]

$z_n = z_n$

§ There are 4 steps for constructing a 'δ'

Transition (δ)

Step 1: for starting symbol.

$\delta(q_0, \epsilon, z_0) = \delta(q, Sz_0)$ [Non determinism]

S → starting symbol.

without consuming any I/P, if the stack contain z_0, we will push starting symbol into the stack

Step 2: for all variables, like $X \rightarrow Y/z$

$\delta(q, \epsilon, X) = \{(q, Y), (q, z)\}$ [Non determinism].

without consuming any I/P, if top of the stack contain a variable, instead of the variable, insert rth productions

Step 3: for all terminals 'a'

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

If the I/P & top of stack contain same element, then we will pop that element from the stack.

Step 4: final transition

$\delta(q, \epsilon, z_0) = (q, \epsilon)$ [Non determinism]

after processing all the I/P symbols, if the stack contain z_0 alone, then empty the stack condition

Ex:- $E \rightarrow E+E \mid E*E \mid a$

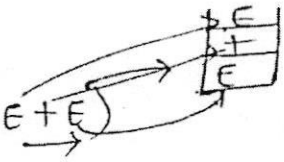
CFG: $\{\{E\}, \{a, +, *\}, P, E\}$

PDA: $\{\{q\}, \{a, +, *\}, \{z_0, E, a, +, *\}, \delta, q, z_0, \phi\}$.

δ 1) starting symbol

$\delta(q, \epsilon, z_0) = (q, E z_0)$ [Non-deterministic with step 1]

2) for variable 'E'



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

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

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

} Non-deterministic

3) for all terminals $a, +, *$

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

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

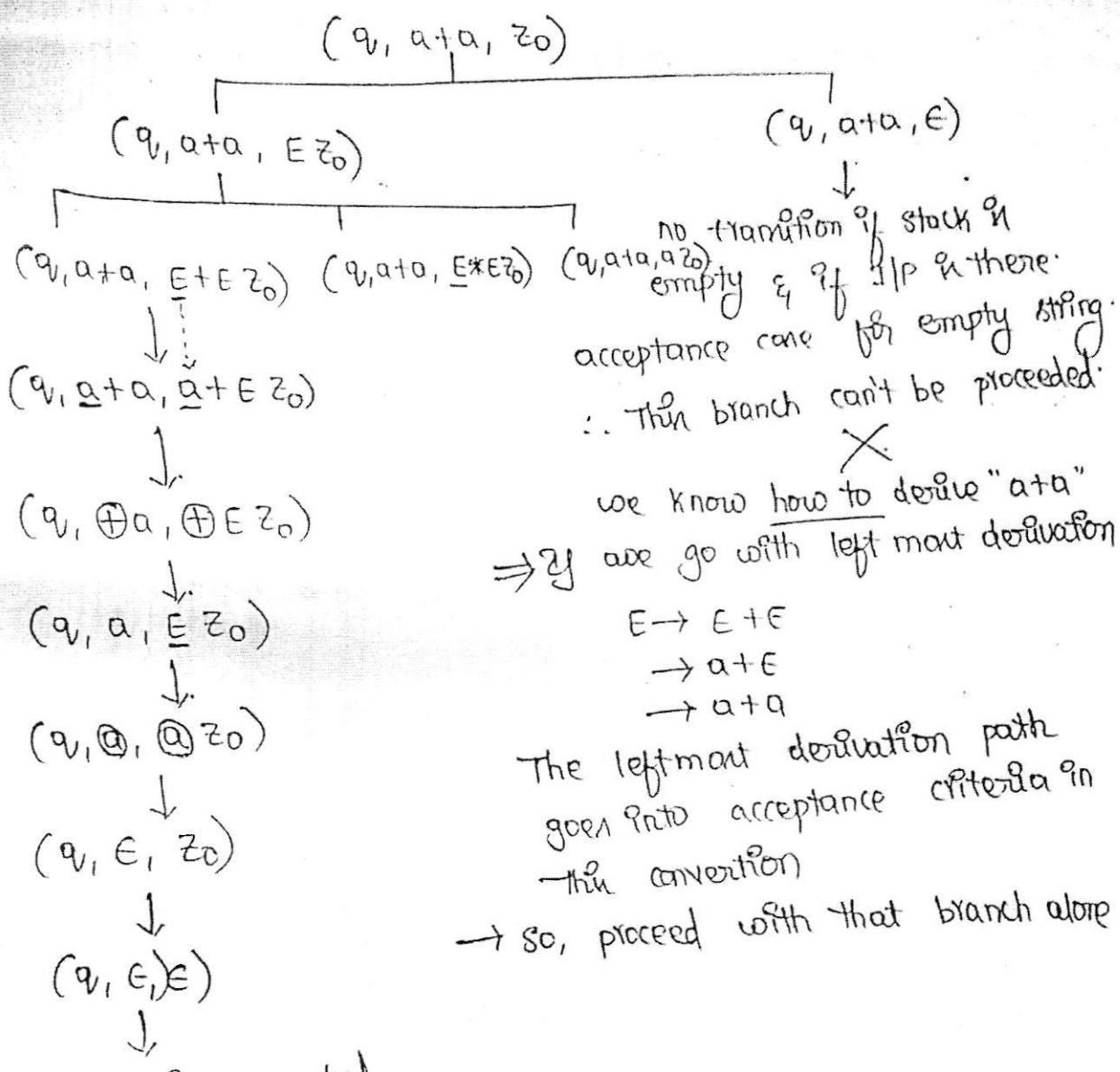
$\delta(q, *, *) = (q, \epsilon)$

4) final transition

$\delta(q, \epsilon, z_0) = (q, \epsilon)$.

→ Take a string & check whether the string is accepted by above PDA or not. using instantaneous description

let the string $w = a+a$



∴ I/P is accepted.

no transition if stack is empty & if I/P is there.

acceptance case for empty string.

∴ This branch can't be proceeded.

we know how to derive "a+a"

⇒ We go with left most derivation

$E \rightarrow E+E$
$\rightarrow a+E$
$\rightarrow a+a$

The leftmost derivation path goes into acceptance criteria in this convention

→ So, proceed with that branch alone

Ex:- $S \rightarrow aSb \mid ab$

$\delta(q_1, \epsilon, z_0) = (q_1, Sz_0)$

$\delta(q_1, \epsilon, S) = \{(q_1, aSb), (q_1, ab)\}$

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

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

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

PDA = $\{ \{q\}, \{a, b\}, \{z_0 \cup S \cup \{a, b\}\}, \delta, q_1, \phi \}$

Instantaneous description for $\omega = aabb$.

$(q, aabb, z_0)$

- $(q, aabb, \epsilon)$
- $(q, aabb, Sz_0)$
  - $(q, \underline{a}abb, \underline{a}Sbz_0)$
    - (q, abb, Sbz_0)
      - $(q, \underline{a}bb, \underline{a}bBz_0)$
        - $(q, \underline{b}b, \underline{b}bz_0)$
          - $(q, \underline{b}, \underline{b}z_0)$
            - (q, ϵ, z_0)
              - (q, ϵ, ϵ)
                - I/P accepted by empty stack
  - $(q, aabb, abz_0)$
    - (q, abb, bz_0)
      - No transition defined.

Ex:- $S \rightarrow OBB$

$B \rightarrow OS \mid 1S \mid 0$

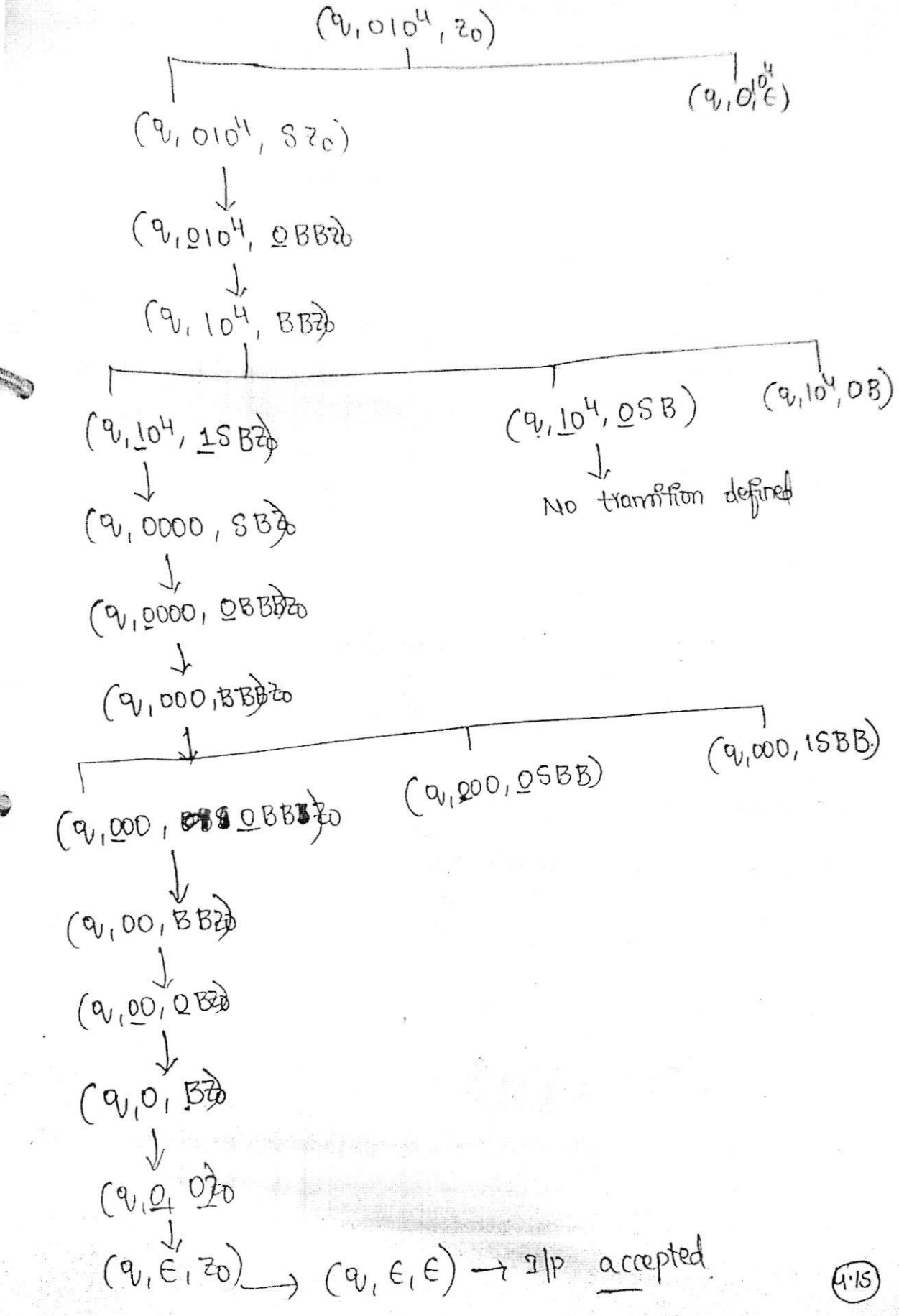
$\delta(q, \epsilon, z_0) = (q, Sz_0)$

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

$\delta(q, \epsilon, B) = \{(q, OS), (q, 1S), (q, 0)\}$

$\delta(q, 0, 0) = (q, \epsilon)$ & $\delta(q, 1, 1) = (q, \epsilon)$

Instantaneous description for, $w = 010^4$



4.15

Ex:- $S \to XY$

$X \to aX \mid bX \mid a$

$Y \to Ya \mid Yb \mid a$

CFG = $\{\{S, X, Y\}, \{a, b\}, P, S\}$.

PDA = $\{Q, \Sigma, \sim, \delta, q_0, z_0, F\}$

$= \{\{q\}, \{a, b\}, \{z_0 \cup \{a, b\} \cup \{S, X, Y\}, \delta, q, z_0, \emptyset\}$

& δ का

1) For start symbol

$\delta(q_0, \epsilon, z_0) = (q, Sz_0)$

2) For all variables

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

$\delta(q, \epsilon, X) = \{(q, aX), (q, bX), (q, a)\}$

$\delta(q, \epsilon, Y) = \{(q, Ya), (q, Yb), (q, a)\}$

3) For all terminals

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

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

4) For acceptance, final transition

$\delta(q, \epsilon, z_0) = (q, \epsilon)$

ID for aaab

$(q, aaab, z_0)$

$\downarrow$

$(q, aaab, Sz_0)$

$\downarrow$

$(q, aaab, XYz_0)$

$\downarrow$

$(q, aaab, aXYz_0)$

$\downarrow$

(q, aab, XYz_0)

$\downarrow$

(q, aab, aYz_0)

$\downarrow$

(q, ab, Yz_0)

$(q, \epsilon) \leftarrow (q, \epsilon, z_0) \leftarrow (q, b, bz_0) \leftarrow (q, ab, abz_0) \leftarrow (q, ab, Ybz_0) \leftarrow (q, ab, Yz_0)$

Converting PDA to CFG:-

PDA = $\{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

CFG = $\{V, T, P, S\}$

Convert 7-tuples of PDA to 4-tuples of CFG

Σ (in PDA) = T (in CFG)

S = starting symbol

V = variable is made up of 3 combinations

[state stack symbol state] = $[Q_i \; \Gamma \; Q_i]$

If $Q = \{q_0, q_1\}$ then variables are,

$[Q_i \; \Gamma \xrightarrow{a} Q_i]$ Consider all possibilities

$q_0 \; q_1$ $\quad$ $q_0 \; q_1$

$V = \{S, [q_0 \; a \; q_0], [q_0, a, q_1], [q_1, a, q_0], [q_1, a, q_1]\}$

P = derived from δ

δ is a rule of PDA, P is a rule of CFG

→ In PDA, there are 2 types of transition available

1) one transition is for pushing

2) " " popping

i.e $\delta[q_0, a, z_0) = (q_0, \epsilon)$ → pop

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

Production rule (P) for "popping" can be written as

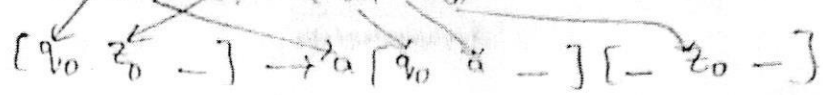
$\delta(q_0, a, z_0) = (q_0, \epsilon)$

$[q_0 \; z_0 \; q_0] \rightarrow a$

Variable combination of 3 $[Q \; \Gamma \; Q_i]$ (4.16)

Prod. rule (P) for "pushing" can be written as,

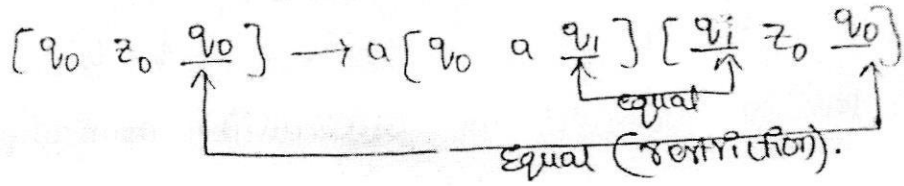
$\delta(q_0, a, z_0) = (q_0, a z_0)$



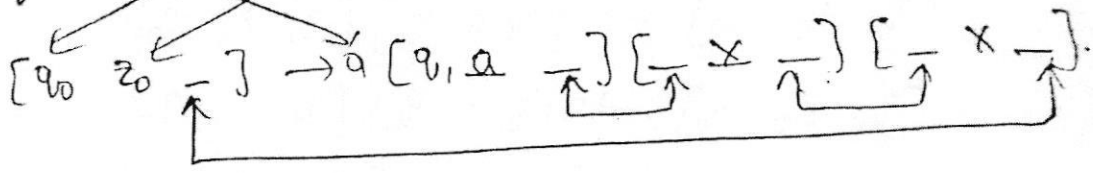
all the empty spaces should be filled with states.

Here $Q = \{q_0, q_1\}$ [assume].

Then fill the spaces with all combination of $\{q_0, q_1\}$



if $\delta(q_0, a, z_0) = (q_1, aX)$ [pushing 3 ele's]



→ In this conversion also, we are going to accept the I/P string by "empty stack" condition

Ex:- $PDA = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

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

δ is given as

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

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

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

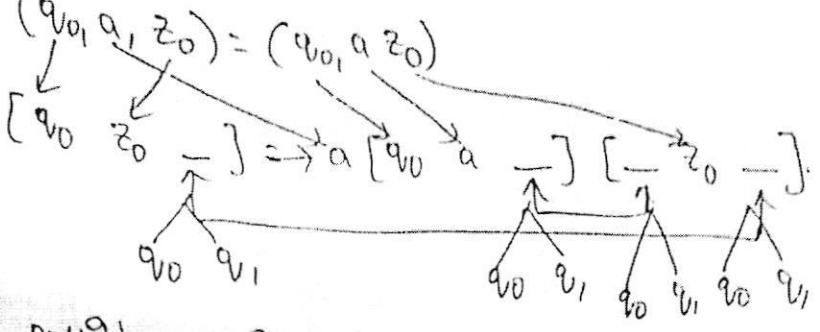
$\delta(q_1, b, a) = (q_1, \epsilon)$

Sol:

$G_1 = (V, T, P, S)$

$T = \Sigma = \{a, b\}$. $S = S$

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



∴ possible variables are (all possible combination)

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

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

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

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

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

possible var's for above transition are,

$[q_0\ a\ \underline{q_0}] \rightarrow a\,[q_0\ a\ \underline{q_0}]\,[\underline{q_0}\ a\ \underline{q_0}]$

$[q_0\ a\ \underline{q_0}] \rightarrow a\,[q_0\ a\ \underline{q_1}]\,[\underline{q_1}\ a\ \underline{q_0}]$

$[q_0\ a\ \underline{q_1}] \rightarrow a\,[q_0\ a\ \underline{q_0}]\,[\underline{q_0}\ a\ \underline{q_1}]$

$[q_0\ a\ \underline{q_1}] \rightarrow a\,[q_0\ a\ \underline{q_0}]\,[\underline{q_0}\ a\ \underline{q_1}]$.

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

possible variable q_1,

$[q_0\ a\ q_1] \rightarrow b$.

d) $\delta(q_1, b, a) = (q_1, \epsilon)$ & the possible variable is

$[q_1\ a\ q_1] \rightarrow b$

e) $\delta(q_1, \epsilon, z_0) \rightarrow (q_1, \epsilon)$

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

All the transition are converted to production

f) Now, Starting symbol production is,

$S \rightarrow [q_0\ z_0\ -]$

↓ starting state, → bottom of stack

$S \rightarrow [q_0\ z_0\ q_0] \mid [q_0\ z_0\ q_1]$

→ Total no: of productions generated are 13; Give some names to each variable, as

$[q_0\ z_0\ q_0] \rightarrow A$

$[q_0\ a\ q_0] \rightarrow B.$

$[q_0\ a\ q_1] \rightarrow C$

$[q_1\ z_0\ q_0] \rightarrow D$

$[q_0\ z_0\ q_1] \rightarrow E$

$[q_0\ a\ q_1] \rightarrow F$

$[q_1\ z_0\ q_1] \rightarrow G$

$[q_1\ a\ q_1] \rightarrow H$

$[q_1\ a\ q_0] \rightarrow I.$

We write the productions as,

$S \rightarrow A$ → as 'A' is eliminated, due to D. ∴ This can be removed ✗

$S \rightarrow E$

$A \rightarrow aBD$ ✗

$A \rightarrow aCD$ → no prod. for D. ∴ eliminate this prod ✗

$B \rightarrow aBB$ → recursion ∴ non generating ✗

$B \rightarrow aFI$ → no prod. for I, F is non-generating ∴ ✗

$C \rightarrow b$

$E \rightarrow aBE$ → 'B' is non-generating ✗

$E \rightarrow aCG$

$F \rightarrow aFH$

$F \rightarrow aBF$ → as 'B' is non-generating, ∴ F ✗

$G \rightarrow \epsilon$ ✓

$H \rightarrow b$ → generating.

Eliminate useless symbols (non-generating, not reachable)

∴ simplified CFG is,

$S \rightarrow E$

$C \rightarrow b$

$E \rightarrow aCG$

$F \rightarrow aFH$

$G \rightarrow \epsilon$

$H \rightarrow b$

4.18

Ex:- PDA = $\{Q, \Sigma, \Gamma, \delta, q_0, z_0, \phi\}$

= $\{\{q_0, q_1\}, \{a, b\}, \{z_0, z\}, \delta, q_0, z_0, \phi\}$

and δ is given by

$\delta(q_0, b, z_0) = (q_0, z z_0)$ $\quad$ $\delta(q_0, a, z) = (q_1, z)$

$\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$ $\quad$ $\delta(q_1, b, z) = (q_1, \epsilon)$

$\delta(q_0, b, z) = (q_0, z z)$ $\quad$ $\delta(q_1, a, z_0) = (q_0, z_0)$

Sol: $G = \{V, \{a, b\}, P, S\}$

$S \rightarrow [q_0\ z_0\ \underline{q_0}]$

$S \rightarrow [q_0\ z_0\ \underline{q_1}]$

Next transition is, $\delta(q_0, b, z_0) = (q_0, z z_0)$ & the corresponding variables are,

$[q_0\ z_0\ \underline{q_0}] \rightarrow b\,[q_0\ z\ \underline{q_0}]\,[\underline{q_0}\ z_0\ \underline{q_0}]$

$[q_0\ z_0\ \underline{q_0}] \rightarrow b\,[q_0\ z\ \underline{q_1}]\,[\underline{q_1}\ z_0\ \underline{q_0}]$

$[q_0\ z_0\ \underline{q_1}] \rightarrow b\,[q_0\ z\ \underline{q_0}]\,[\underline{q_0}\ z_0\ \underline{q_1}]$

$[q_0\ z_0\ \underline{q_1}] \rightarrow b\,[q_0\ z\ \underline{q_1}]\,[\underline{q_1}\ z_0\ \underline{q_1}]$

Next transition is, $\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$ which is a pop operation & the corresponding variable is,

$[q_0\ z_0\ q_0] \rightarrow \epsilon$

Next transition is, $\delta(q_0, b, z) = (q_0, z z)$ & the corresponding variables are,

$[q_0\ z\ \underline{q_0}] \rightarrow b\,[q_0\ z\ \underline{q_0}][\underline{q_0}\ z\ \underline{q_0}]$

$[q_0\ z\ \underline{q_0}] \rightarrow b\,[q_0\ z\ \underline{q_1}][\underline{q_1}\ z\ \underline{q_0}]$

$[q_0\ z\ \underline{q_1}] \rightarrow b\,[q_0\ z\ \underline{q_0}][\underline{q_0}\ z\ \underline{q_1}]$

$[q_0\ z\ \underline{q_1}] \rightarrow b\,[q_0\ z\ \underline{q_1}][\underline{q_1}\ z\ \underline{q_1}]$

Next transition q_1, $\delta(q_0, a, z) = (q_1, z)$ & the corresponding variables are,

$[q_0\ z\ \underline{q_0}] \rightarrow a\,[q_1\ z\ \underline{q_0}]$

$[q_0\ z\ \underline{q_1}] \rightarrow a\,[q_1\ z\ \underline{q_1}]$

Next transition q_1, $\delta(q_1, b, z) = (q_1, \epsilon)$ which is a pop operation & the corresponding variable is,

$[q_1\ z\ q_1] \rightarrow b$

Next transition q_1, $\delta(q_1, a, z_0) = (q_0, z_0)$ & the corresponding variables are,

$[q_1\ z_0\ \underline{q_0}] \rightarrow a\,[q_0\ z_0\ \underline{q_0}]$

$[q_1\ z_0\ \underline{q_1}] \rightarrow a\,[q_0\ z_0\ \underline{q_1}]$

Now give names to all the variables & re-write the productions:

$[q_0\ z_0\ q_0] = A$

$[q_0\ z_0\ q_1] = B$

$[q_0\ z\ q_0] = C$

$[q_0\ z\ q_1] = D$

$[q_1\ z_0\ q_0] = E$

$[q_1\ z_0\ q_1] = F$

$[q_1\ z\ q_0] = G$

$[q_1\ z\ q_1] = H$

Now re-write the prod's with new variables & remove useless symbols (not reachable & non-generating symbols).

$S \to A$ ✓ as 'A' is generating

$S \to B$ ✗ 'B' is non-generating.

$A \to bCA$ ✗ as 'C' is non-generating

$A \to bDE$ ✓

$B \to bCB$ ✗ as 'C' is non-generating

$B \to bDF$ [$B \to b(aH)(aB)$] ✗ 'B' non-generating, calling itself.

$A \to \epsilon$ ✓ Generating

$C \to bCC$ ✗ non-generating

$C \to bDG$ ✗ no production for 'G'

$D \to bAB$ ✗ 'B' is non-generating

$D \to bBH$ ✗ 'B' is non-generating

$C \to aG$ ✗ as 'G' does not doesn't have any prod.

$D \to aH$ ✓ generating

$E \to aA$ ✓ generating

$F \to aB$ ✗ 'B' is non-generating.

$H \to b$ ✓ Generating

∴ CFG for given PDA is, $\{V, T, P, S\}$

where, $T = \Sigma = \{a, b\}$, $S \to$ Start symbol

$P =$ $S \to A\epsilon$ | $V = \{S, A, D, E, H\}$.

$A \to bDE \mid \epsilon$

$D \to aH$

$E \to aA$

$H \to b$.