Type 2 Grammar

CONTEXT FREE GRAMMAR (CFG) / CFL

- P : LHS $\longrightarrow$ RHS

$$\boxed{V \longrightarrow (VUT)^*}$$

- LHS : single capital letter
  RHS : can be any combination of variables and terminals.

eg: $S \longrightarrow \varepsilon$ $\qquad$ $S \longrightarrow aaBa$

$S \longrightarrow ab$

$S \longrightarrow A$

- All regular grammar is CFG but not viceversa.

Q. $L = \{ a^n b^n , n \geq 0 \}$ ...aaabbb...

$S \rightarrow aSb / \varepsilon$

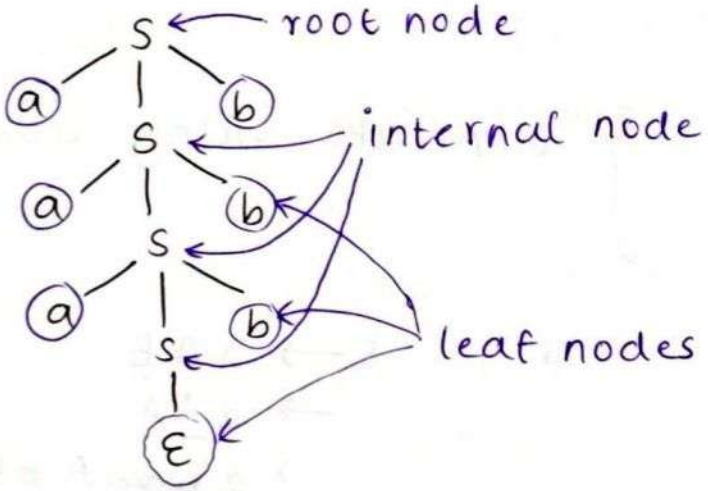
Derivation of :

- $S \rightarrow aSb \xrightarrow{S \rightarrow \varepsilon} ab$
- $S \rightarrow aSb \xrightarrow{S \rightarrow aSb} aaSbb \xrightarrow{S \rightarrow aSb} aaaSbbb \xrightarrow{S \rightarrow \varepsilon} aaabbb$

Derive the string a^3b^3:

$S \rightarrow aSb \xrightarrow{S \rightarrow aSb} aaSbb \xrightarrow{S \rightarrow aSb} aaaSbbb \xrightarrow{S \rightarrow \varepsilon} aaabbb$

Parse tree of a^3b^3:

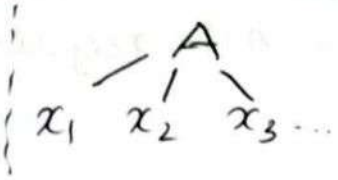


Parse Trees

- Parse trees of a grammar G are trees with following properties:

1. Root will be the start symbol
2. Each ~~node~~ internal node will be variable V
3. Each leaf node is labelled by a variable or terminal or ε. If ε is a node, that will be the only child of that parent.
4. If A is an internal node and its children $x_1, x_2, \ldots x_n$, then $A \rightarrow x_1 x_2 \ldots x_n$ will

be a production in grammar G



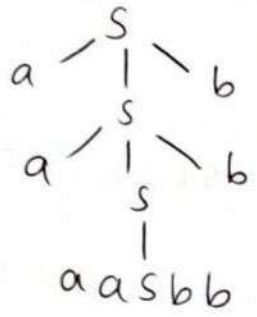
Yield of a Parse ~~Field~~ Tree

- word generated from a grammar
- Read from left to right

Sentential form

- can include variables and terminals
- intermediate ~~of~~ stage of derivation

eg:-



9/9/25

Q. $S \rightarrow aAB$
$A \rightarrow bBb$
$B \rightarrow A/\epsilon$

} generate string abbbb

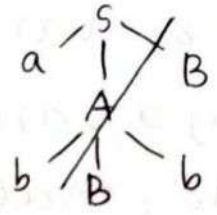
→ $S \rightarrow aAB$
$\rightarrow abBbB$
$\rightarrow abAbB$
$\rightarrow abbBbbB$
$\rightarrow abbbbB$
$\rightarrow abbbb$

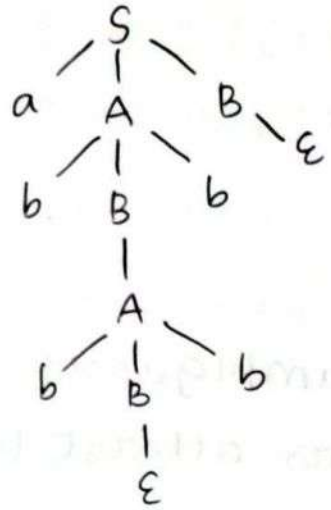
(leftmost derivation)

or

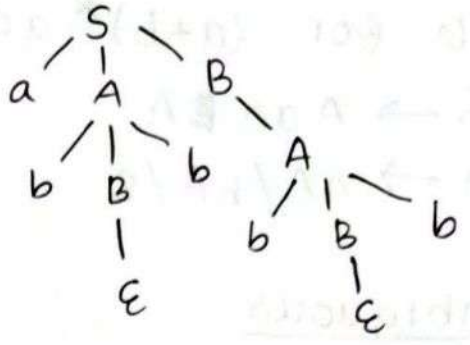
$S \rightarrow aAB$
$\rightarrow aAA$
$\rightarrow a$ ~~bBb~~ $A\ bBb$
$\rightarrow$ ~~aAbAb~~ $aAbb$
$\rightarrow abBbbb$
$\rightarrow abbbb$

(rightmost derivation)





(Leftmost)



(Rightmost)

Q.1 Write a context free grammar for all palindromes over 0 and 1

→ $S \rightarrow 0S0 / 1S1 / 0 / 1 / \varepsilon$

~~$A \rightarrow \varepsilon$~~

Even palindrome : 010010

Odd palindrome : 0110110

2. Write CFG for generating balanced paranthesis.

→ $S \rightarrow (S)S / \varepsilon$

or $S \rightarrow (S) / SS / \varepsilon$ or $S \rightarrow (S) / ()S / \varepsilon$

3. Write CFG for generating $L = \{ w \in (a,b)^*, n_a = n_b \}$

→ $S \rightarrow aSbS / bSaS / \varepsilon$

or $S \rightarrow abS / SbA / \varepsilon$

or $S \rightarrow aSb / bSa / SS / \varepsilon$

4. Write CFG for $0^*1(0+1)^*$

→ $\underset{A}{0^*}\ \underset{1}{1}\ \underset{B}{(0+1)^*}$

$S \rightarrow A | B$

$A \rightarrow 0A / \varepsilon$

$B \rightarrow 0B / 1B / \varepsilon$

5. CFG for $(a+b)^* aa (a+b)^*$

$S \rightarrow AaaA$

$A \rightarrow aA / bA / \varepsilon$

11/9

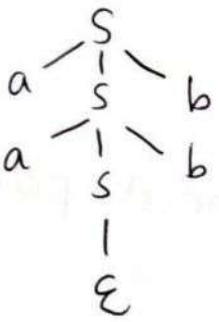
Ambiguous

A given CFG G is said to be ambiguous if there exists some $w \in L(G)$ that has atleast two different parse trees/derivation trees.

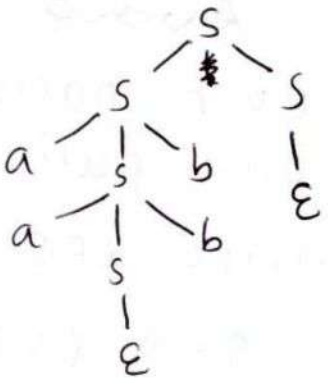
eg: $S \rightarrow aSb / SS / \varepsilon$

$w = aabb$

- $S \rightarrow aSb$
  $\rightarrow aaSbb$
  $\rightarrow aabb$



- $S \rightarrow SS$
  $\rightarrow aSbS$
  $\rightarrow aaSbbS$
  $\rightarrow aabbS$
  $\rightarrow aabb$



Unambiguous

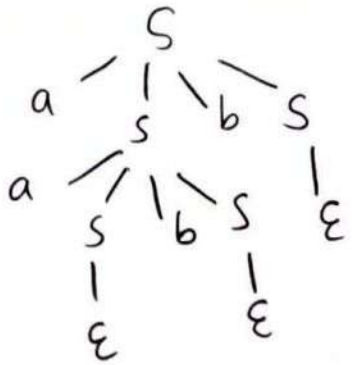
eg: $S \rightarrow aSbS / \varepsilon$

- $S \rightarrow aSbS$
  $\rightarrow aaSbbS$
  $\rightarrow aa\varepsilon bbS$
  $\rightarrow aabb$

  (leftmost)

- $S \rightarrow aSbS$
  $\rightarrow aSb$
  $\rightarrow aaSbSb$
  $\rightarrow aaSbb$
  $\rightarrow aabb$

  (rightmost)



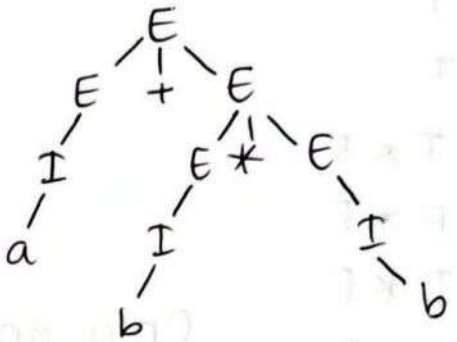
(same parse tree)

Q.1) $E \rightarrow I / E+E / E*E / (E)$

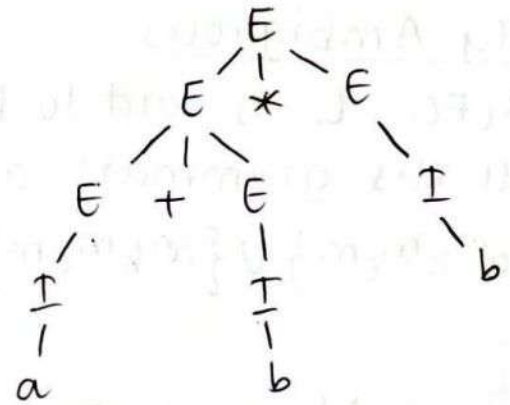
$I \rightarrow a/b/Ia/Ib/Io/I1$

$\omega = a+b*b$

- $E \rightarrow E+E$
  $\rightarrow I+E$
  $\rightarrow a+E$
  $\rightarrow a+E*E$
  $\rightarrow a+I*E$
  $\rightarrow a+b*E$
  $\rightarrow a+b*I$
  $\rightarrow a+b*b$



- $E \rightarrow E*E$
  $\rightarrow E+E*E$
  $\rightarrow I+E*E$
  $\rightarrow a+E*E$
  $\rightarrow a+I*E$
  $\rightarrow a+b*E$
  $\rightarrow a+b*I$
  $\rightarrow a+b*b$



For the string ω, since we got two different parse trees, the given grammar is ambiguous.

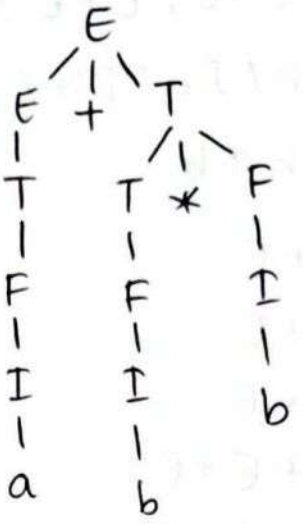
Q.2) $E \rightarrow T / E+T$

$T \rightarrow E / T*F$

$F \rightarrow I / (E)$

$I \rightarrow a/b/Ia/Ib/Io/I1$

$E \to E + T$
$\to T + T$
$\to F + T$
$\to I + T$
$\to a + T$
$\to a + T * F$
$\to a + F * F$
$\to a + I * F$
$\to a + b * F$
$\to a + b * I$
$\to a + b * b$



Only one parse tree, hence unambiguous.
[unambiguous version of Q.1]

Inherently Ambiguous

A ~~cont~~CFG L is said to be inherently ambiguous if all its grammars are ambiguous

eg: $L = \{a^n b^m c^m\} \cup \{a^n b^m c^m\}$

Normal Forms

Simplification of CFG

1. Elimination of useless symbols
2. " ε productions
3. " unit "

1. Elimination of useless symbols

- A ~~sym~~ variable ~~s~~ in a grammar is useful if $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$
  ① reachable ② generating

eg: $S \to AB / a$
$A \to b$

Generating		Reachable	
S ✓ A ✓ B ×	$S \to a$ $A \to b$	$S \to a$ $A \to b$ ×	$S \to a$

Simplification of CFG

Q.1) $S \to aB / bX$

$A \to BAd / bSX / a$

$B \to aSB / bBX$

$X \to SBD / aBX / ad$

ans:- ⇒ A ✓

X ✓

S ✓ ($S \to bX \to bad$)

B and D not generating

∴ $S \to bX$

$A \to bSX / a$

$X \to ad$

⇒ Remove A as A is not reachable

Hence, $S \to bX$

$X \to ad$

2. Elimination of null production or ε productions

Two conditions for a variable to be nullable:

i) if $A \stackrel{*}{\Rightarrow} \varepsilon$

ii) $B \to C_1 C_2 \dots C_k$, C_i is nullable, then B nullable.

Q.2) $S \to aA$

$A \to b / \varepsilon$

ans:- A is nullable.

Hence: $S \to aA / a$

$A \to b$

Q.3) $S \to ABC$

$A \to BC / a$

$B \to bAC / \varepsilon$

$C \to CAB / \varepsilon$

ans:- S, A, B and C are nullable.

~~Hence: S → ABC/AC/BC/AB/A~~

Hence:

$S \to ABC / AC / AB / BC / A / B / C$

$A \to B / C / a / BC$

$B \to bC / bA / b / bAC$

$C \to CAB / C / CA / CB / AB / A / B$

3. Elimination of Unit Productions

15/9

$A \to B$
$B \to \alpha$
$A \to \alpha$ (α: non-unit productions)

Steps of simplification of CFG

1. Null ~~pe~~ productions
2. Unit productions
3. Useless symbols

Q 4) $S \to A/bb$
$A \to B/b$
$B \to S/a$

ans: $S \to bb/b/a$
$A \to b/a/bb$ } these are useless
$B \to a/bb/b$ }

Q.5) $S \to Aa/B$
$B \to A/bb$
$A \to a/bc/B$

ans: $S \to Aa/a/bc/bb$
$B \to a/bc/bb$ } useless
$A \to a/bc/bb$

16/9

Chomsky Normal Form (CNF)

A context free grammar G is said to be in CNF if all its production rules are of the form $A \to a$ or $A \to BC$

Eg Q. $S \to bA/aB$
$A \to bAA/aS/a$
$B \to aBB/bS/a$
} convert into CNF

→ $S \to CA/DB$
$A \to CAA/DS/a$
$B \to DBB/CS/a$
$C \to b$
$D \to a$

$\Rightarrow$ $S \to CA/DB$
$A \to CE/DS/a$
$B \to DF/CS/a$
$C \to b$
$D \to a$
$E \to AA$
$F \to BB$

Q.2) Convert the given CFG to CNF:

$S \to ABC$
$A \to aAb/\varepsilon$
$B \to b$
$C \to aC/a/\varepsilon$

→ 1. Removing null productions:

$S \rightarrow ABC / BC / AB / B$

$A \rightarrow aAb / ab$

$B \rightarrow b$

$C \rightarrow aC / a$

2. Removing unit productions

$S \rightarrow ABC / BC / AB / b$

$A \rightarrow aAb / ab$

$B \rightarrow b$

$C \rightarrow aC / a$

3. Useless symbols : no useless symbols.

Convert to CNF

$S \rightarrow ABC / BC / AB / b$

$A \rightarrow DAB / DB$

$B \rightarrow b$

$C \rightarrow DC / a$

$D \rightarrow a$

$\Rightarrow$

$S \rightarrow EC / BC / AB / b$

$A \rightarrow DE / DB$

$B \rightarrow b$

$C \rightarrow DC / a$

$D \rightarrow a$

$E \rightarrow AB$

Greibach

~~Gray Back~~ Normal Form (GNF)

A context free grammar G is said to be GNF if all its production rules are of the form $A \rightarrow a$ or $A \rightarrow aX_1X_2 \ldots X_n$

Steps

1. Convert the grammar to CNF
2. Rename the variables to $A_1, A_2 \ldots A_n$ in the order of occurence
3. Check all production rules where first symbol on RHS is a non-terminal or variable.
   a) $i < j$: proceed

b) $i > j$, replace A_j with A_j's production rules.

c) $i = j$: Left recursion : must remove them.

eg : $A_1 \rightarrow A_2A_3 / A_4A_2$ (i = 1, j = 2)

$A_2 \rightarrow a$

$A_3 \rightarrow b$

$A_4 \rightarrow A_2A_4 / A_4A_3$ (i = 4, j = 2) (Left recursion)

$\Rightarrow$

$A_1 \rightarrow A_2A_3 / A_4A_2$

$A_2 \rightarrow a$

$A_3 \rightarrow b$

$A_4 \rightarrow aA_4$

4. Replace every first non-terminal in any production rule until it is in GNF.

Q. $S \rightarrow XA / BB$

$B \rightarrow b / SB$

$X \rightarrow b$

$A \rightarrow a$

→ 1. Already CNF

2. Renaming : $A_1 \rightarrow A_2A_3 / A_4A_4$

$A_4 \rightarrow b / A_1A_4$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

3. Replacing A_1 with all its production rule (in A_4)

ie, $A_1 \rightarrow A_2A_3 / A_4A_4$

$A_4 \rightarrow b / A_2A_3A_4 / A_4A_4A_4$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

because

$A_4 \rightarrow A_1A_4$

$i = 4$, $j = 1$

$i > j$

↳ $A_1 \rightarrow A_2A_3 / A_4A_4$

$A_4 \rightarrow b / bA_3A_4 / A_4A_4A_4$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

here

$A_4 \rightarrow A_2A_3A_4$

$i = 4$, $j = 2$

$i > j$

Steps to remove left recursion

1. Create a new variable Z with production rules containing trailing path of left recursive production once without Z and once with Z.
2. Remove the left recursive production and add all other production rules once followed by Z

ie, • $Z \rightarrow A_4A_4 / A_4A_4Z$

• $A_1 \rightarrow A_2A_3 / A_4A_4$

$A_4 \rightarrow b / bA_3A_4 / bZ / bA_3A_4Z$

$A_2 \rightarrow b \qquad A_3 \rightarrow a$

$Z \rightarrow A_4A_4 / A_4A_4Z$

$A_1 \rightarrow bA_3 / bA_4 / bA_3A_4A_4 / bZA_4 / bA_3A_4ZA_4$

$A_4 \rightarrow b / bA_3A_4 / bZ / bA_3A_4Z$

$A_2 \rightarrow b \qquad A_3 \rightarrow a$

$Z \rightarrow bA_4 / bA_3A_4A_4 / bZA_4 / bA_3A_4ZA_4 / bA_4Z /$
$bA_3A_4A_4Z / bZA_4Z / bA_3A_4ZA_4Z.$

17/9

Q. Convert ~~th~~ to GNF

$S \rightarrow ABA$

$A \rightarrow aA / \varepsilon$

$B \rightarrow bB / \varepsilon$

→ Convert to CNF

1. Remove null productions

$S \rightarrow ABA / BA / AA / AB / A / B$

$A \rightarrow aA / a$

$B \rightarrow bB / b$

2. Remove unit productions

$S \rightarrow ABA / BA / AA / AB / a / bA / b / bB$

$A \rightarrow aA / a$

$B \rightarrow bB / b$

3. No useless symbols

4. Convert to CNF

$S \rightarrow AE / AB / BA / AA / CA / a / DB / b$

$A \rightarrow CA / a$

$B \rightarrow DB / b$

$C \rightarrow a$

$D \rightarrow b$

$E \rightarrow BA$

Convert CNF to GNF

1. Renaming

$A_1 \rightarrow A_2A_3 / A_2A_4 / A_4A_2 / A_2A_2 / A_5A_2 / a / A_6A_4 / b$

$A_2 \rightarrow A_5A_2 / a$

$A_4 \rightarrow A_6A_4 / b$

$A_5 \rightarrow a$

$A_6 \rightarrow b$

$A_3 \rightarrow A_3A_2A_4A_2$

2. No situation when $i<j$ or $i=j$

3. Convert to GNF

$A_2 \rightarrow aA_2 / a$

$A_4 \rightarrow bA_4 / b$

$A_5 \rightarrow a$

$A_6 \rightarrow b$

$A_3 \rightarrow bA_4A_2 / bA_2$

$A_1 \rightarrow aA_2A_3 / aA_3 / aA_2A_4 / aA_4 / bA_4A_2 / bA_2 / aA_2A_2 / aA_2 / aA_2 / a / bA_4 / b$

Push Down Automata (PDA)

Formal Definition

PDA can be defined by a 7 tuple format,

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

where, Q : non empty finite set of states

Σ : input alphabet

τ : stack alphabet

q_0 : initial state ($q_0 \in Q$)

z_0 : initial stack symbol/top

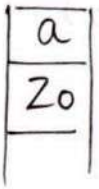
F : set of final states

$\delta : Q \times \Sigma \times \tau \rightarrow Q \times \tau^*$

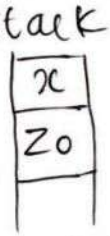
(τ^* : stack operations)

Eg: 1. $(q, a, z_0) = (q', z_0)$: no operation "nop"

2. $(q, a, z_0) = (q', a z_0)$ (push)

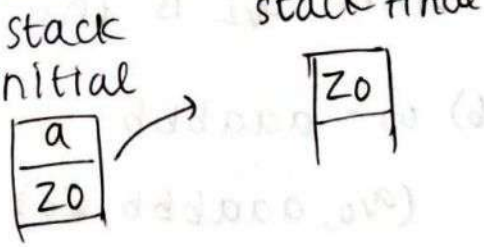


3. $(q, a, z_0) = (q', x z_0)$



4. $(q, a, z_0) = (q', a a z_0)$

5. $(q, a, z_0) = (q', \varepsilon)$ (pop)



→ (q) —curr. i/p, stack top/stack op.→ (q')

eg: . q —a, Z0/Z0→ (q') for $(q, a, z_0) = (q', z_0)$

. q —a, Z0/aZ0→ (q') for $(q, a, z_0) = (q', a z_0)$

. q —a, Z0/ε→ (q') for $(q, a, z_0) = (q', \varepsilon)$

Instantaneous Description (ID) of PDA

ID of PDA is (q, w, γ)

where, q : current state

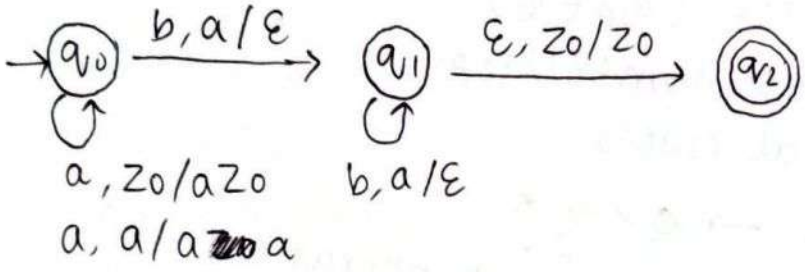
w : remaining input

γ : stack operation

eg: $(q_0, aabb, Z_0) \vdash (q_0, abb, aZ_0)$

18/9

Q.1) $L = \{a^n b^n , n \geq 1\}$. [deterministic]



a) Trace $w = aabb$

$(q_0, aabb, Z_0) \vdash (q_0, abb, aZ_0)$

$\vdash (q_0, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_2, \varepsilon, Z_0)$

Since q_2 is the final state, the string is accepted.

b) $w = aaabbb$

$(q_0, aaabbb, Z_0) \vdash (q_0, aabbb, aZ_0)$

$\vdash (q_0, abbb, aaZ_0)$

$\vdash (q_0, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

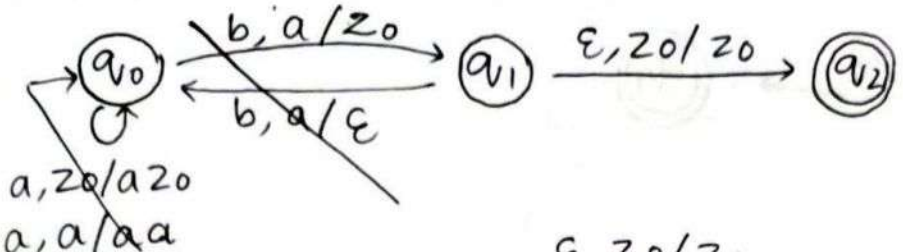
$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

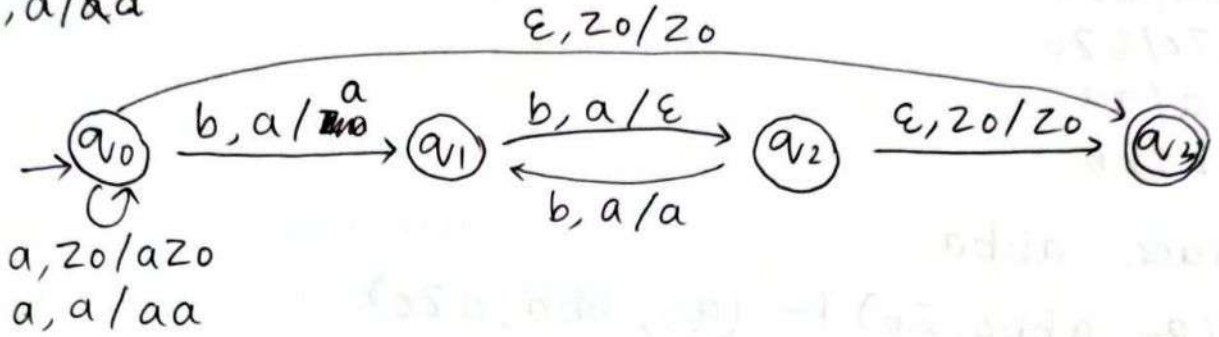
$\vdash (q_2, \varepsilon, Z_0)$

String accepted.

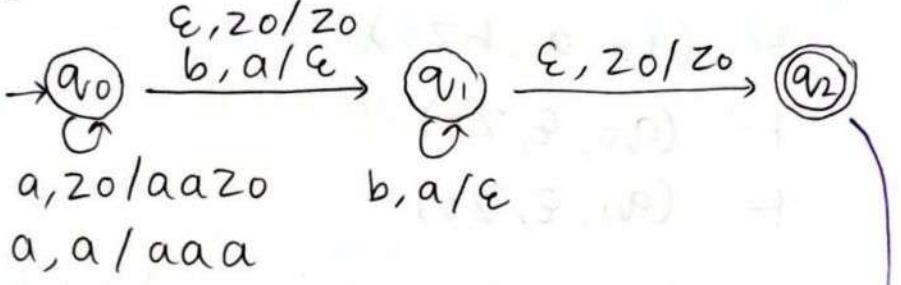
2) Design PDA for the language $L=\{a^n b^{2n}, n \geq 0\}$



[non-deterministic]



or



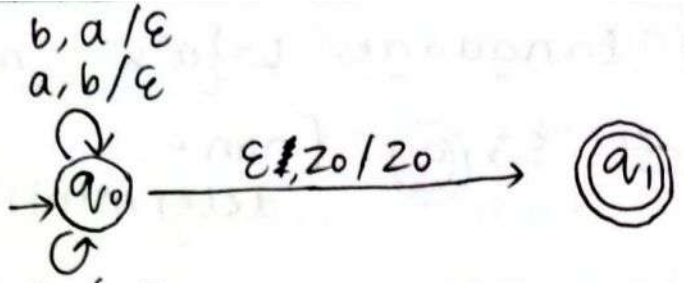
Trace $w = aabbbb$

$(q_0, aabbbb, z_0) \vdash (q_0, abbbb, aaz_0)$
$\vdash (q_0, bbbb, aaaaz_0)$
$\vdash (q_1, bbb, aaaz_0)$
$\vdash (q_1, bb, aaz_0)$
$\vdash (q_1, b, az_0)$
$\vdash (q_1, \varepsilon, z_0)$
$\vdash (q_2, \varepsilon, z_0)$

Hence string accepted.

3. Design a ~~push~~PDA to accept set of all strings over $\{a,b\}$ with equal no. of 'a's and 'b's.

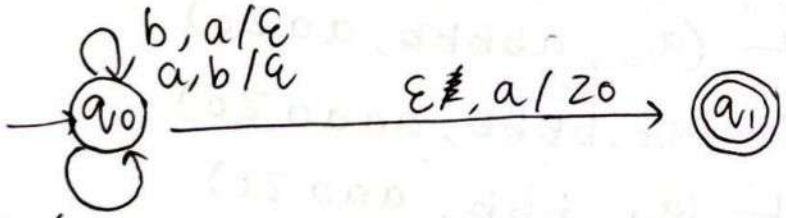
[Eg for non-deterministic PDA (condition 2)]



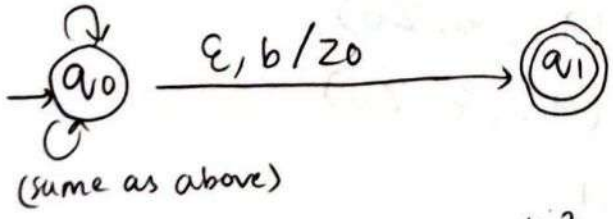
a) Trace abba

$(q_0, abba, Z_0) \vdash (q_0, bba, aZ_0)$
$\vdash (q_0, ba, Z_0)$
$\vdash (q_0, a, bZ_0)$
$\vdash (q_0, \varepsilon, Z_0)$
$\vdash (q_1, \varepsilon, Z_0)$

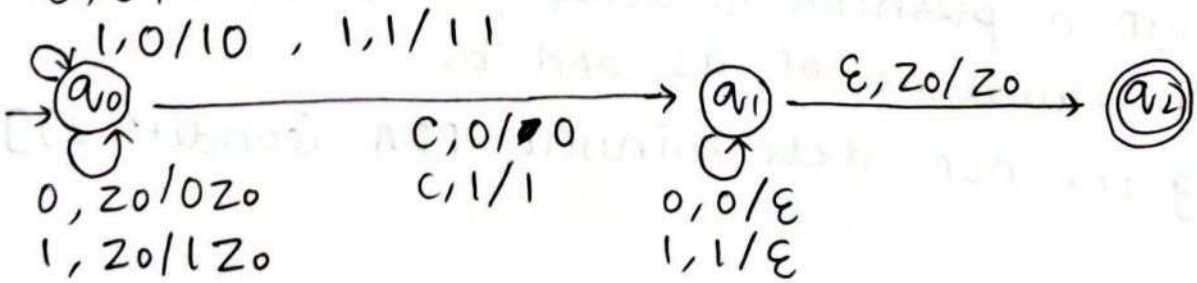
4) PDA to accept set of all strings over {a, b} such that $n(a) > n(b)$



5) $n(a) < n(b)$

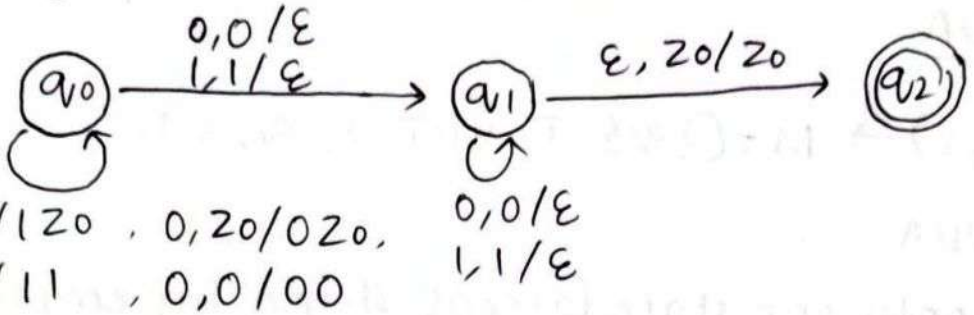


6) $L = \{w c w^R, w \in (0,1)^*\}$ eg: 01 c 10 (w c w^R)



19/9

7) Design PDA for $L = \{wwR, w \in (0,1)^*\}$



1, z0/1z0 . 0, z0/0z0.
1, 1/11 , 0,0/00
0,1/01 , 1,0/10

→ like trial and error
→ one type of non deterministic approach.

8) - A PDA is said to be non deterministic if it follows any of the two conditions:
1. $\delta(q, a, \alpha)$ has multiple choices
2. $\delta(q, \varepsilon, \alpha)$ and $\delta(q, a, \alpha)$ are present
(a: any i/p symbol)

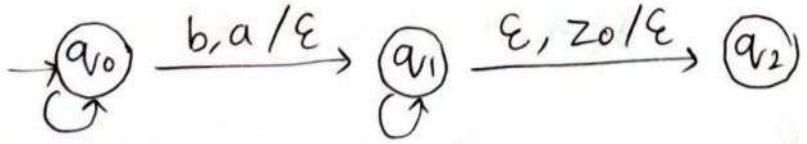
23/9

Language of PDA

1. Acceptance by Final state
$L(M) = \{w / (q_0, w, z_0) \vdash^* (q_f, \varepsilon, \alpha)\}$

2. Acceptance by empty stack
$L(M) = \{w / (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$

Eg: $L = \{a^n b^n, n \geq 1\}$ in acceptance by empty stack:



a, z0/az0
a, a/aa
b, a/ε

- no final state
- popping of z0 at last step proves the acceptance of the string.

Equivalence of CFG & PDA

I CFG to PDA

$G = (V, T, P, S) \rightarrow M = (\{q\}, T, VUT, \delta, q, S)$

here, for PDA

- $\{q\}$: only one state (accept string by empty stack)
- $\Sigma = T$: terminals
- $\Gamma = VUT$: can be terminals & non terminals
- $q_0 = q$: same starting state
- $Z_0 = S$
- No importance for F (final state)
- δ in PDA

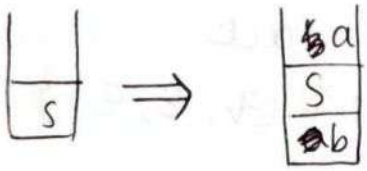
1) For every variable A, add $\delta(q, \varepsilon, A) = \{(q, B) / A \rightarrow B$ is a production rule$\}$.

Eg: $S \rightarrow aSb/a/b/\varepsilon$

if we consider $\delta(q, \varepsilon, S) = (q, aSb)$

when taking $S \rightarrow asb$

then

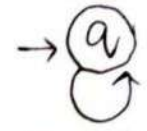


2) For every terminal add a pop transition

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

Eg: $S \rightarrow 0BB$

$B \rightarrow 0S/1S/0$



$\varepsilon, S/0BB$

$\varepsilon, B/0S$, $\varepsilon/B/1S$, $\varepsilon/B/0$

$0, 0/\varepsilon$, $1, 1/\varepsilon$

Trace 010000

$S \rightarrow 0\underline{B}B$

$\rightarrow 01\underline{S}B$

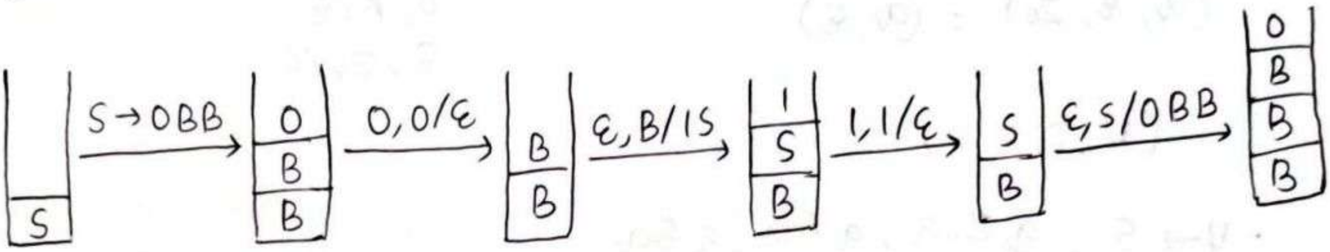
$\rightarrow 010\underline{B}BB$

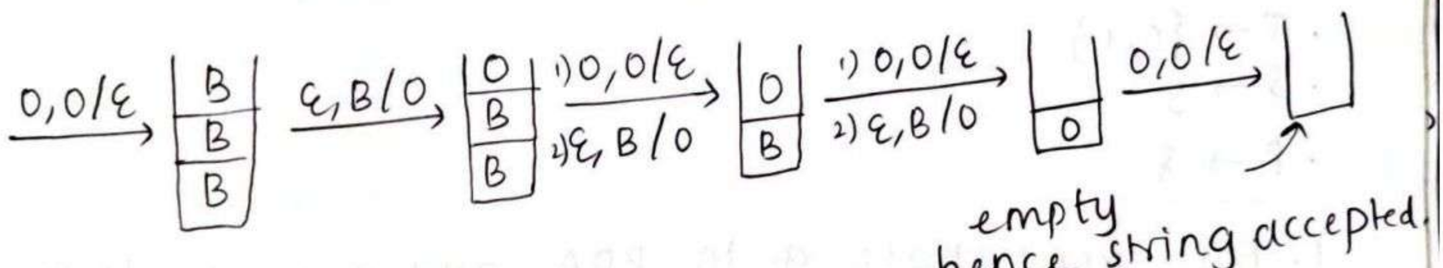
$\rightarrow 0100\underline{B}B$

$\rightarrow 01000\underline{B}$

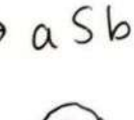
$\rightarrow 010000$

(By leftmost)





Q. $S \rightarrow aSb/a/b$



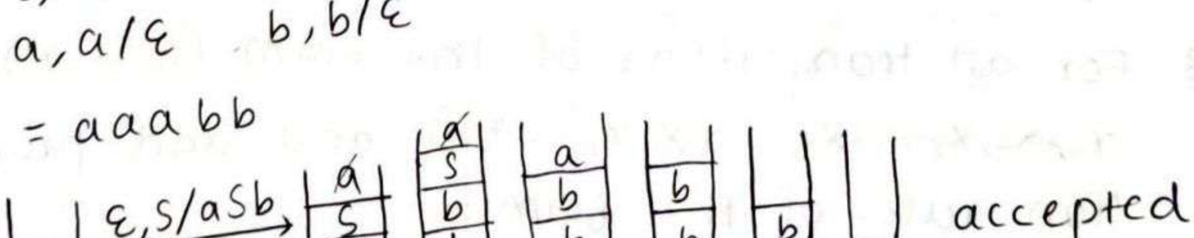
ε,S/aSb

ε,S/a

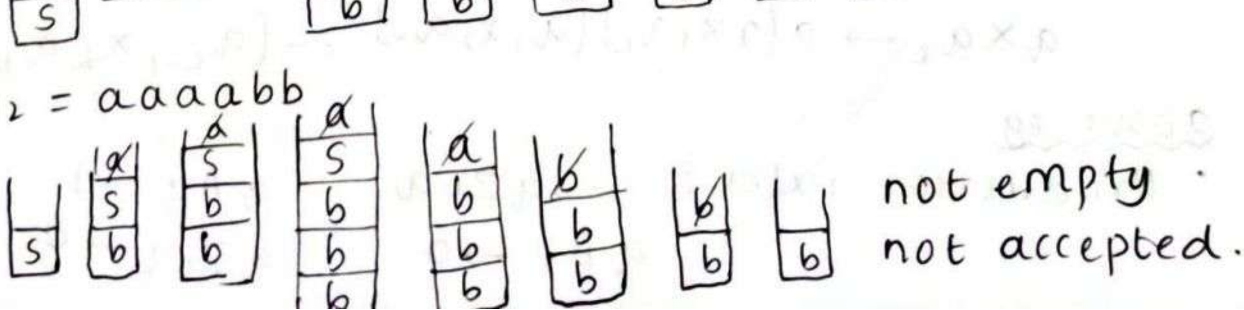
ε,S/b

a,a/ε b,b/ε

a) w_1 = aaabb



b) w_2 = aaaabb



25/9

Conversion of PDA to CFG

Eg: ① PDA = ({q}, {0,1}, {Z0, A, B}, δ, q, Z0)

$(q, 0, Z_0) = (q, AZ_0)$
$(q, 1, Z_0) = (q, BZ_0)$
$(q, 0, A) = (q, AA)$
$(q, 1, B) = (q, BB)$
$(q, 0, B) = (q, \varepsilon)$
$(q, 1, A) = (q, \varepsilon)$
$(q, \varepsilon, Z_0) = (q, \varepsilon)$

ie

0, Z0/AZ0
1, Z0/BZ0
0, A/AA
1, B/BB
0, B/ε
0, A/ε
ε, Z0/ε.

(V, T, P, S)

- V → S, qZ0q, qAq, qBq
- T → {0,1}
- S → S
- P → δ

1. For every state q in PDA, add a production rule in the format $S \rightarrow q_0 Z_0 q$

   Eg: Q = {q0, q1}

   then $S \rightarrow q_0 Z_0 q_0$, $S \rightarrow q_0 Z_0 q_1$

2. For every pop transition of the form $(q, a, X) = (r, \varepsilon)$, add a production $\boxed{qXr \rightarrow a}$

   Eg: $(q, 1, A) = (q, \varepsilon) \Rightarrow qAq = 1$

3. For all transitions of the form $(q, a, X) = (r X_1 X_2 \ldots X_k)$ and add production rules of the form:

   $qXq_k \rightarrow a[rX_1q_1][q_1X_2q_2] \ldots [q_{k-1}X_kq_k]$

above eg

- Production rules: $S \rightarrow qZ_0q$, $qBq \rightarrow 0$, $qAq \rightarrow 1$, $qZ_0q \rightarrow \varepsilon$

- For $(q_1, 0, Z_0) = (q, AZ_0)$:

  $q_1 Z_0 q \rightarrow 0\ [qAq][qZ_0q]$

- $q Z_0 q \rightarrow 1\ [qBq][qZ_0q]$
- $qAq \rightarrow 0\ [qAq][qAq]$
- $qBq \rightarrow 1\ [qBq][qBq]$

Let: $V \rightarrow S$, $\underbrace{qZ_0q}_{A}$, $\underbrace{qAq}_{B}$, $\underbrace{qBq}_{C}$

then,

$S \rightarrow A$
$C \rightarrow 0$
$B \rightarrow 1$
$A \rightarrow \varepsilon$
$A \rightarrow 0BA$
$A \rightarrow 1CA$
$B \rightarrow 0BB$
$CB \rightarrow 1CC$

$S \rightarrow A$
$A \rightarrow 0BA / 1CA / \varepsilon$
$B \rightarrow 0BB / 1$
$C \rightarrow 1CC / 0$

(2) PDA $= (\{q_0, q_1\}, \{0, 1\}, \{Z_0, X\}, \delta, q_0, Z_0)$

1. $(q_0, 0, Z_0) = (q_0, XZ_0)$
2. $(q_0, 0, X) = (q_0, XX)$
3. $(q_0, 1, X) = (q_1, \varepsilon)$
4. $(q_1, 1, X) = (q_1, \varepsilon)$
5. $(q_1, \varepsilon, X) = (q_1, \varepsilon)$
6. $(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$

ans: $V \rightarrow S$, $\underset{A}{q_0 Z_0 q_0}$, $\underset{B}{q_0 Z_0 q_1}$, $\underset{C}{q_1 Z_0 q_0}$, $\underset{D}{q_1 Z_0 q_1}$,
$\underset{E}{q_0 X q_0}$, $\underset{F}{q_0 X q_1}$, $\underset{G}{q_1 X q_0}$, $\underset{H}{q_1 X q_1}$

$S \rightarrow q_0 Z_0 q_0 / q_0 Z_0 q_1$

Pop transitions

$q_0 X q_1 \rightarrow 1$
$q_1 X q_1 \rightarrow 1 / \varepsilon$
$q_1 Z_0 q_1 \rightarrow \varepsilon$

$q_0 Z_0 q_0 \rightarrow 0\,[q_0 X q_0][q_0 Z_0 q_0]$

$q_0 Z_0 q_1 \rightarrow 0\,[q_0 X q_1][q_1 Z_0 q_0]$

$q_0 Z_0 q_1 \rightarrow 0\,[q_0 X q_0][q_0 Z_0 q_1]$

$\rightarrow 0\,[q_0 X q_1][q_1 Z_0 q_1]$

$q_0 X q_0 \rightarrow 0\,[q_0 X q_0][q_0 X q_0]$

$\rightarrow 0\,[q_0 X q_1][q_1 X q_0]$

$q_0 X q_1 \rightarrow 0\,[q_0 X q_0][q_0 X q_1]$

$\rightarrow 0\,[q_0 X q_1][q_1 X q_1]$

Then $S \rightarrow A/B$

$F \rightarrow 1$

$H \rightarrow 1/\varepsilon$

$D \rightarrow \varepsilon$

$A \rightarrow 0EA/0FC$

$B \rightarrow 0EB/0FD$

$E \rightarrow 0EE/0FG$

$F \rightarrow 0EF/0FH$

Removing useless symbols

- C, E, G, A not generating

Hence, $S \rightarrow B$

$F \rightarrow 1/0FH$

$H \rightarrow 1/\varepsilon$

$B \rightarrow 0FD$

$D \rightarrow \varepsilon$

Closure Properties of CFL

1. Union

Let For a CFL, L_1 the grammar $G_1 = (V_1, T, P_1, S_1)$ and

For a CFL, L_2 " " $G_2 = (V_2, T_2, P_2, S_2)$

then $L_1 \cup L_2 = (V_1 \cup V_2 \cup S, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \to S_1/S_2\}, S)$

Hence ~~el~~ union is closed.

2. Concatenation

Let L_1 be CFL, it means there exist $G_1 = (V_1, T_1, P_1, S_1)$

" L_2 " " " " " " " $G_2 = (V_2, T_2, P_2, S_2)$

$L_1 \cdot L_2 \to$ also of the form (V, T, P, S)

∴ It is closed under concatenation

3. Reversal

$L \to G \quad A \to \alpha$ reverse RHS of all production

$L^R \to G^R \quad A \to \alpha^R$ rules.

4. Homomorphism

Let L be CFL with grammar G

$L = a^n b^n$, $G(L) \Rightarrow S \to aSb/\varepsilon$

$h(a) = 01$
$b(b) = 1$ $\Rightarrow G(h(L)) \Rightarrow S \to 01S1/\varepsilon$

Replace all ~~images~~ terminals using homomorphic images.

5. CFG is not closed under intersection

IMP 6. Intersection of a Regular language and a context free language is context free.

- $L_1 \to CFL \to PDA \quad (Q_1, \Sigma_1, \tau, \delta_1, q_{01}, Z_0, F_1)$
- $L_2 \to RL \to DFA \quad (Q_2, \Sigma_2, \delta_2, q_{02}, F_2)$
- $L_1 \cap L_2 \to PDA \quad (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \tau, \delta(q_{01}, q_{02}), Z_0, F_1 \times F_2)$

RL ∩ CFL = CFL

$\delta \to ((q_1, q_2), \alpha, \beta), (P_1, P_2), r)$

iff $(q_1, \alpha, \beta) = (P, r) \in \delta_1$ [if a tran^n of PDA]

$(q_2, \alpha) = P_2 \in \delta_2$ [transitn of DFA]

if and both transitions intersection comes. it is context free.

Decision Properties of CFL

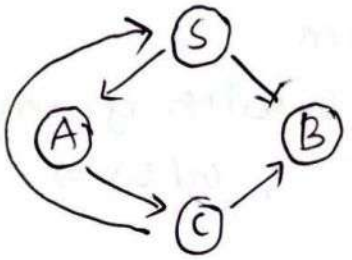
1. Emptiness

Remove all the useless symbols, then if the start symbol S is removed, it is said to be empty.

2. Finiteness

Remove all useless symbols or simplify grammar. Construct dependency graph if it has loop, it is infinite.

$S \rightarrow AB$
$A \rightarrow aCb/a$
$B \rightarrow bB/bb$
$C \rightarrow cBS$



infinite

3. Membership

Let G be a grammar and w is a string, checks if $w \in L(G)$?

CYK Algorithm used to check membership

- Given grammar should be in CNF.

1) Each row corresponds to one length of substrings.

$S \rightarrow AB/BC$
$A \rightarrow BA/a$
$B \rightarrow CC/b$
$C \rightarrow AB/a$

$w = baaba$

x_{15}				
x_{14}	x_{25}			
x_{13}	x_{24}	x_{35}		
x_{12}	x_{23}	x_{34}	x_{45}	
x_{11}	x_{22}	x_{33}	x_{44}	x_{55}

S, A, C				
ϕ	S, A, C			
ϕ	B	B		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

- $X_{14} = baab$
  $= baa + b = \phi + \{B\}$
  $= b + aab = \{B, B\}$
- $X_{2,5} = aaba = aab + a$
  $= \{B\} \cdot \{A, C\}$
  $= \{BA, BC\}$
  $= a + aba = \{A, C\} \cdot \{B\}$
  $= \{AB, CB\} = \{S, A, C\}$
- $X_{1,5} = (X_{1,1}, X_{2,5}) \cup (X_{1,2}, X_{3,5}) \cup (X_{1,3}, X_{4,5}) \cup (X_{1,4}, X_{5,5})$

We can see $X_{15} = S, A, C$ then, if $S \in X_{15}$ then $baaba \in L(G)$.

$\Rightarrow$ ababa ?

[15] S, A				
[14] B	[24] B			
[13] ϕ	[23] A, S	[33] ϕ		
S	A, S	S	A, S	
A, C	B	A, C	B	AC

$X_{1,2} = (X_{i,i}, X_{i+1,j})$
$= (X_{1,1}, X_{2,2})$
$X_{2,2} = X_{2,2}, X_{3,3}$
$X_{3,3} = X_{3,3}, X_{4,4}$
$X_{4,4} = X_{44}, X_{55}$
$X_{13} = (X_{11}, X_{23}) \cup (X_{1,2}, X_{3,3})$
$= \{B\}\{B\} \cup \{S, A\}\{A, C\}$
$= \{BB, SA, SC, AA, AC\}$
$X_{2,4} = (X_{2,2}, X_{3,4}) \cup (X_{2,3}, X_{44})$
$= \{A, C\}\{S, C\} \cup \{B\}\{B\}$
$= \{AS, AC, CS, CC, BB\}$
$X_{35} = a + ba$
$= \{A, C\} \cdot \{S, A\}$
$= ab + a$
$= \{S, C\} \cdot \{A, C\}$
$= \{(AS, AA, CS, CA) \cup (SA, SC, CA, CC)\}$
$= B$

X_{13}

a ba : ab a

A,C A,S : S A,C

ie, AA, AS, CA, CS : ie, SA SC

X_{23}

b ab ba b

B SA A, S B

ie, BS BA ie, AB, SB

S S

X_{15}

a baba abab a

A,C B B A,C

ie, AB CB BA, BC S, A

S

ab aba aba ab

S S

X_{33}

a ba ab a

A,C AS S

X_{14}

ab ab a bab aba b

A, A,S

AA, AS, CA, CS

9/10

Pumping Lemma For CFL

Let L be a ~~regular~~ CF language, then there exists a constant ~~m~~ n such that for any string w from the language with length $\geq n$. we can divide w into ~~u~~, five parts u, v, x, y, z satisfying the following conditions

1. $|vy| > 0$
2. $|vxy| \leq n$
3. $uv^ixy^iz \in L \quad \forall i \geq 0$

Q. PT $L = \{a^n b^n c^n, n \geq 1\}$ is not CFL.

→ Let L be a ~~n~~ CFL. Then w be a string in L and m be the constant such that

$w = a^m b^m c^m$, $|w| = 3m > M$

$u = \varepsilon$
$v = \varepsilon$
$x = a^{m-k}$
$y = a^k$, $k \geq 1$
$z = b^m c^m$

$uv^ixy^iz = a^{m-k} a^{ki} b^m c^m$

when $i = 0$

$\Rightarrow a^{m-k} b^m c^m$

$k \geq 1$

It doesnot exist in the L. Hence assumpth is wrong and L is not CFL.

OR

$u = \varepsilon$
$v = a$
$x = a^{m-2}$
$y = a$
$z = b^m c^m$

$a^i a^{m-2} a^i b^m c^m$

$i = 0 \Rightarrow a^{m-2} b^m c^m$

Type 0 : Unrestricted Grammar

- Recursively enumerable L

Turing Machine : Formaly defined as,

$(Q, \Sigma, \tau, \delta, q_0, B, F)$

Q : set of ~~sy~~ states

Σ : alphabets

τ : Tape symbol (can be Σ or any symbol)

q_0 : initial state

B : special blank symbol

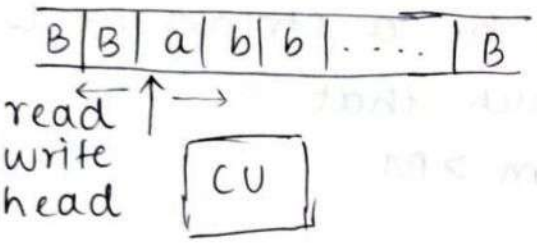
F : set of final states

$\delta : Q \times \tau \longrightarrow Q \times \tau \times \{L, R\}$

R — move right

L — move left

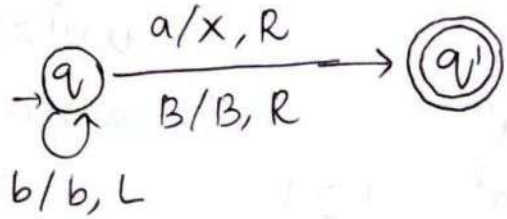
eg : $\delta(q, a) = (q', X, R)$



Eg : $\delta(q, a) = (q', X, R)$

$\delta(q, b) = (q', b, L)$

$\delta(q', B) = (q', B, R)$



Transition table

	a	b	X	B
q	(q', X, R)	(q, b, L)	ϕ	(q', B, R)
q'	ϕ	ϕ	ϕ	ϕ

Tracing : Instantaneous description (ID)

$(\alpha\, q\, \beta)$

where, α : contents to left of pointer

q : current state

β : contents to right of pointer.

Eg : B a a b b B $\qquad ID_1 \rightarrow B q_0 a a b b B$

$(q_0, a) = (q_1, X, R)$

B	X	a	b	b	B

$ID_2 \rightarrow BXa_1abbB$

$(a_1, a) = (a_1, X, R)$

B	X	X	b	b	B

$ID_3 \rightarrow BXXa_1bbB$

$(a_1, b) = (a_0, Y, L)$

B	X	X	Y	b	B

$ID_4 \rightarrow BXa_0XYbB$

110

Q.1) Design a turing machine for the language

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

→ B a a b b c c B

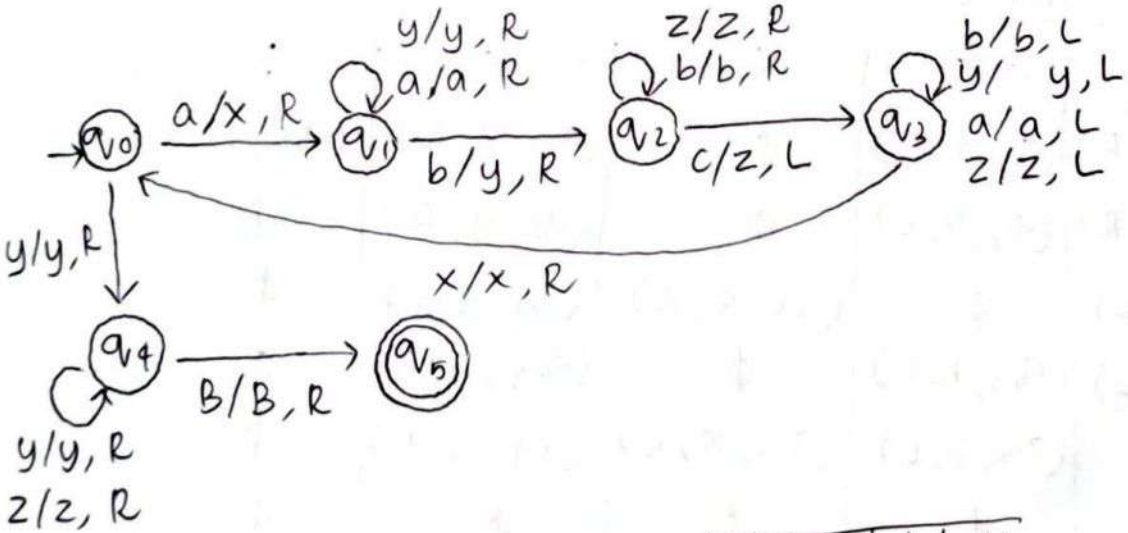
1st cycle : B X a b b c c B → B X a Y b c c B → B X a Y b Z c B

(moves R to look for b)

(comes back to x to see if a is there).

IInd cycle : B X X Y Y Z Z B

- comes back to see if a is there
- no a's found, hence moves repeatedly to R to see if only y and z exists (hence accepted).



⇒ Trace w = aabbcc

B	a	a	b	b	c	c	B

$Ba_0aabbccB \vdash BXa_1abbccB \vdash BXaa_1bbccB \vdash$

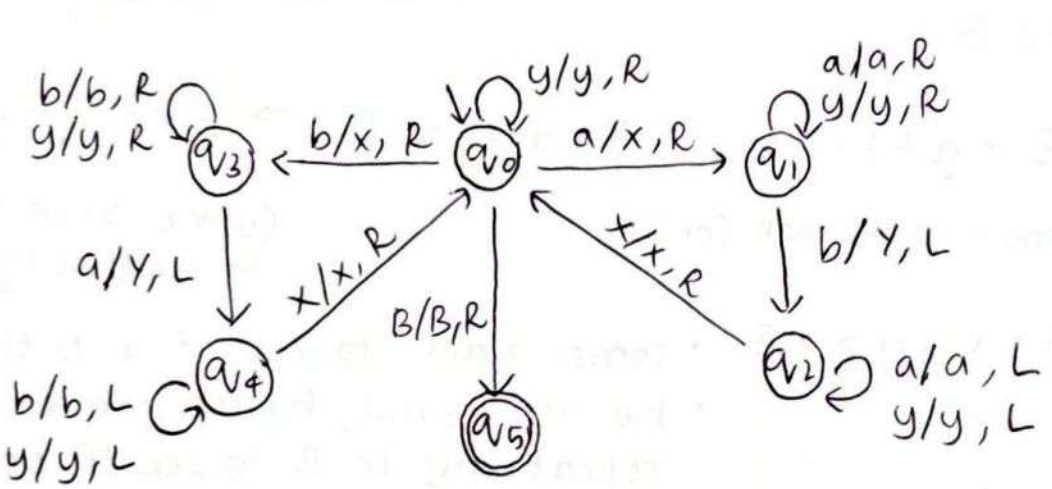
$BXaYa_2bccB \vdash BXaYba_2ccB \vdash$ ~~$BXaYbZa_3cB \vdash$~~

$BXaYa_{3}bZcB \vdash BXaa_3YbZcB \vdash BXa_3aYbZcB \vdash$

B	X	a	Y	b	Z	c	B

$Bq_3xayb\,zcB \vdash Bxq_0ayb\,zcB \vdash Bxxq_1yb\,zcB \vdash$
$Bxxyq_1b\,zcB \vdash Bxxyyq_2\,zcB \vdash Bxxyyzq_2cB \vdash$
$Bxxyyq_3zzB \vdash Bxxyq_3yzzB \vdash Bxxq_3yyzzB \vdash$
$Bxq_3xyyzzB \vdash Bq_3xxyyzzB \vdash Bxq_0xyyzzB \vdash$
$Bxxq_0yyzzB \vdash Bxxyq_4yzzB \vdash BxxyyyzzB \vdash$
$BxxyyzzB \vdash BxxyyzzBq_5$

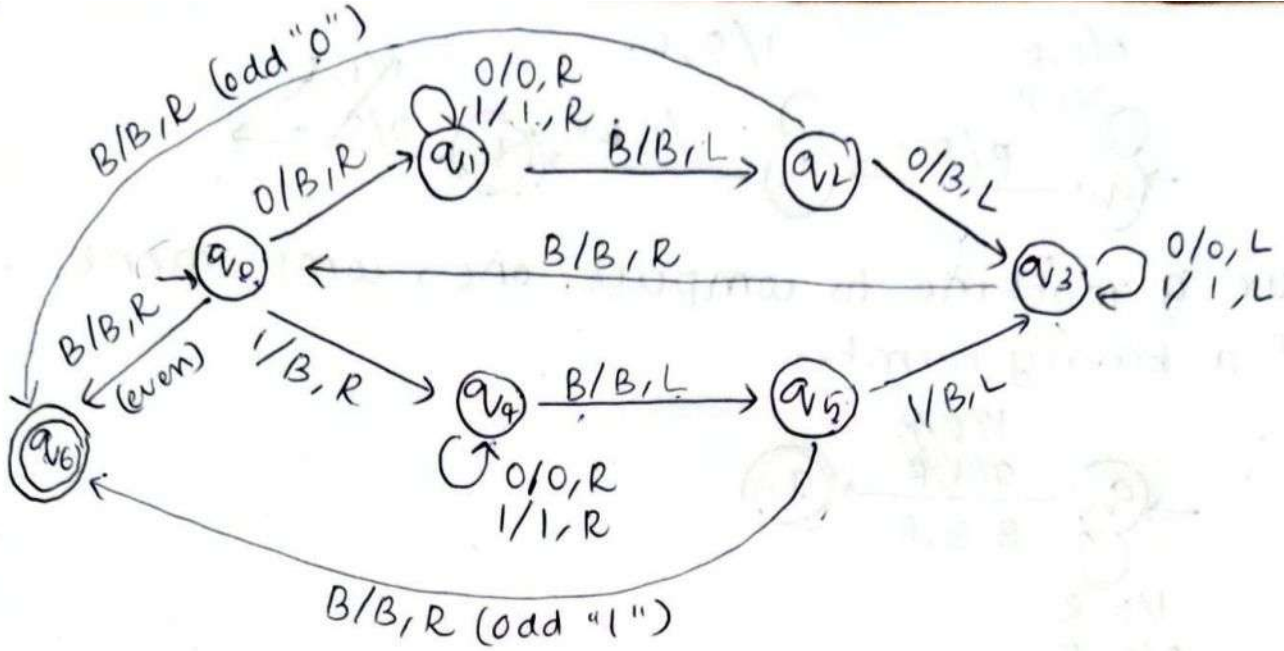
2. Design a turing machine to accept set of all strings with equal no. of a's and b's



Transition table

	a	b	X	Y	B
q_0	(q_1, X, R)	(q_3, X, R)	ϕ	(q_0, Y, R)	(q_5, B, R)
q_1	(q_1, a, R)	(q_2, Y, L)	ϕ	(q_1, y, R)	ϕ
q_2	(q_2, a, L)	ϕ	(q_0, X, R)	(q_2, y, L)	ϕ
q_3	(q_4, Y, L)	(q_3, b, R)	ϕ	(q_3, y, R)	ϕ
q_4	ϕ	(q_4, b, L)	(q_0, X, R)	(q_4, y, L)	ϕ
q_5	ϕ	ϕ	ϕ	ϕ	ϕ

3) Design a turing machine to accept all paliandromes over 0 and 1



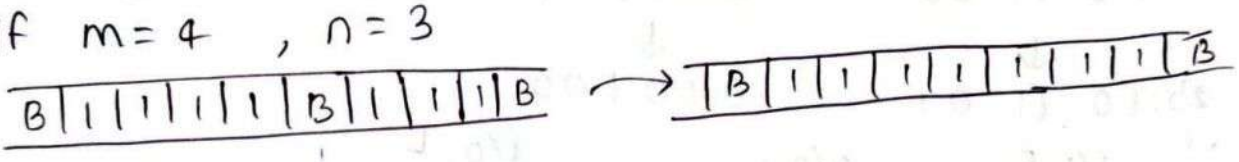
10/10

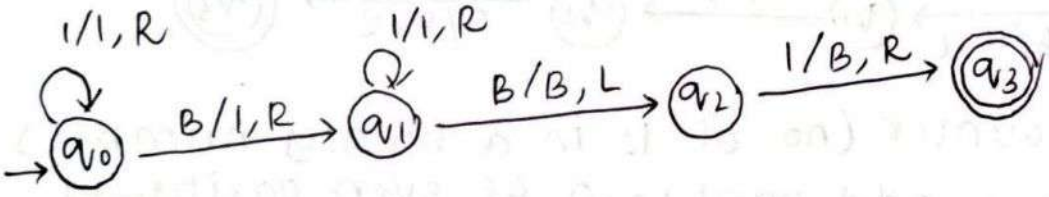
Transducers

Q1) Sum of two numbers : $f(m,n) = m+n$

ans: Using 1^n tape format

if $m=4$, $n=3$

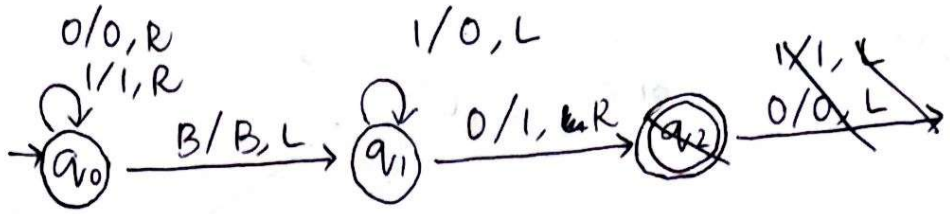




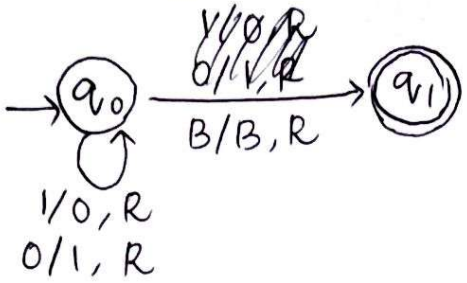
Q.2) Design a turing machine to increment a binary number.

0 0 0 0	1 0 0 0
0 0 0 1	1 0 0 1
0 0 1 0	1 0 1 0
0 0 1 1	1 0 1 1
0 1 0 0	1 1 0 0
0 1 0 1	1 1 0 1
0 1 1 0	1 1 1 0
0 1 1 1	1 1 1 1

- If last bit is 0 convert to 1
- If " " " 1 : all consecutive 1s to 0 and the first 0 to 1 (from right → left).



3) Turing machine to compute one's complement of a binary number.

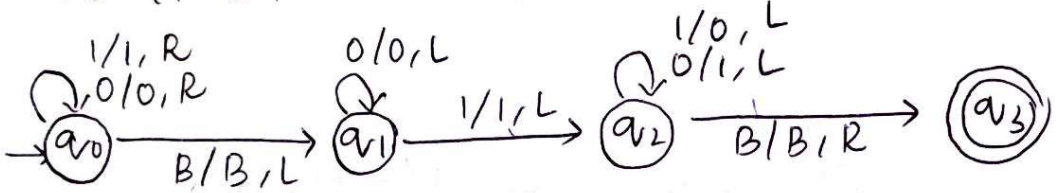


4) Compute 2's complement of a binary number.

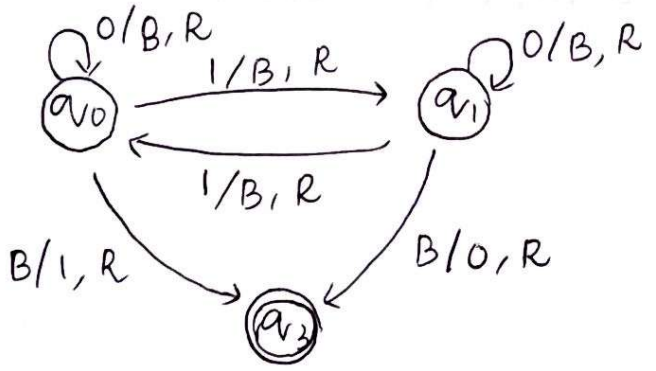
In: 0 1 0 0 1 1 | 1 0 1 1 0 0

1's: 1 0 1 1 0 0 | 0 1 0 0 1 1

2's: 1 0 1 1 0 1 | 0 1 0 1 0 0



5) Parity counter (no. of 1s in a binary number)
Consider : odd parity → 0 & even parity → 1



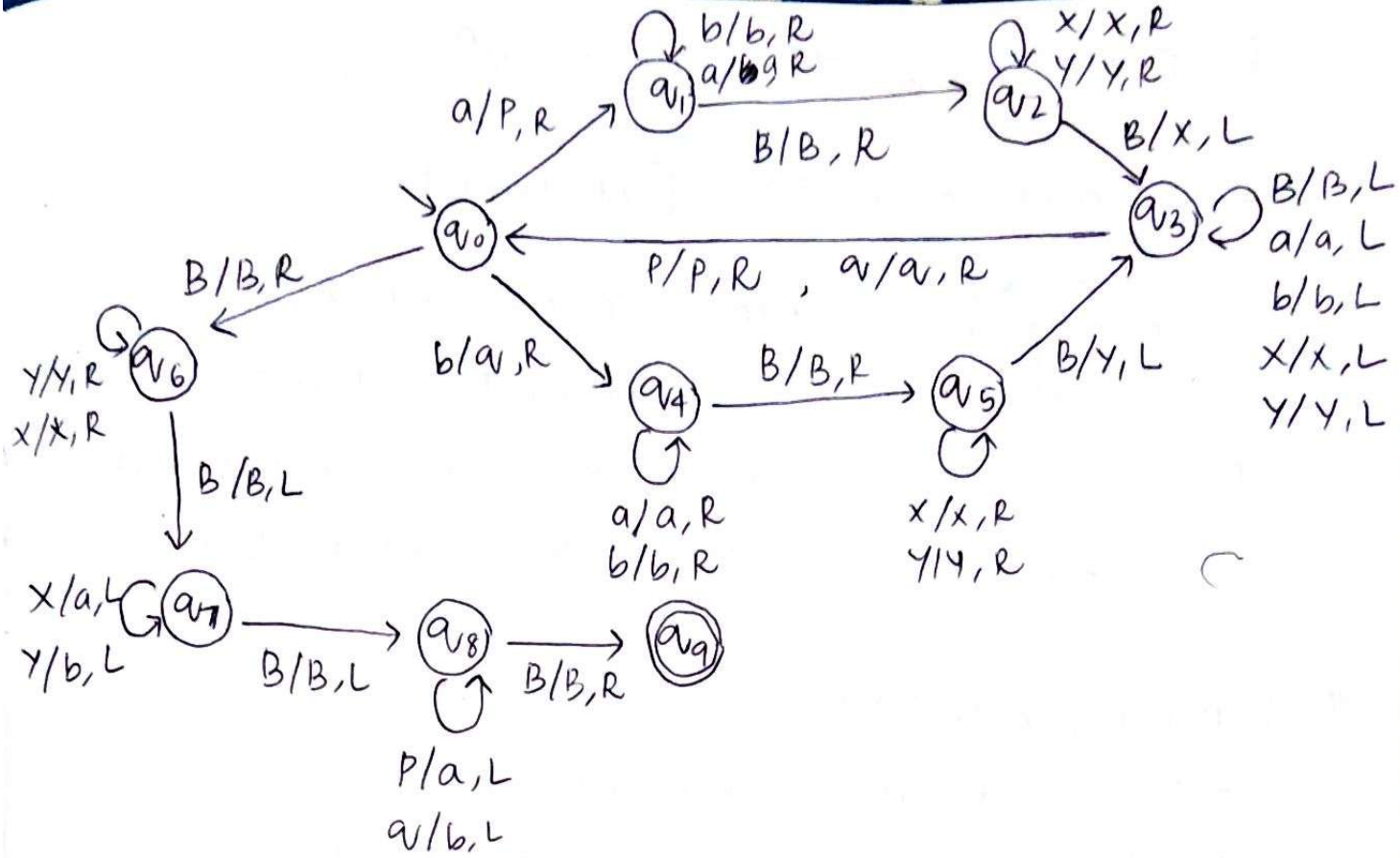
19/10

6) Design a turing machine which works as a copy machine. Let $L = \{a, b\}$:

eg: B a b a b B

⇓

B a b a b B a b a b B

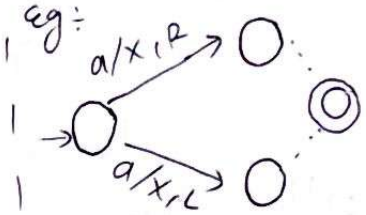


Variance of Turing Machine

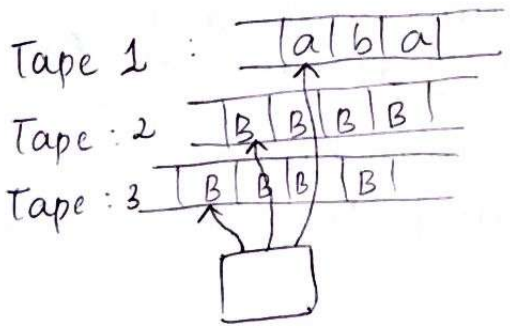
1. Non deterministic TM (NDTM)
   - Can be formally defined as,

     $(Q, \Sigma, \tau, \delta, q_0, B, F)$

   where, $\delta : Q \times \tau \longrightarrow 2^{Q \times \tau \times \{L, R\}}$



2. Multitape TM
   - Turing machine with multiple i/p tapes.



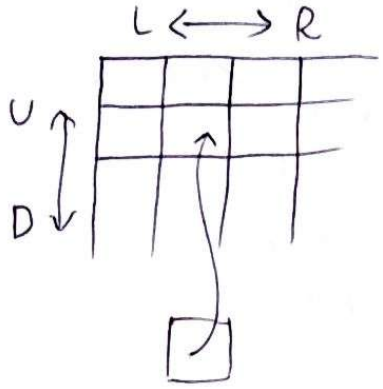
- multiple tapes
- each have seperate read and write head.

→ Read write head at start of input for Tape1 but head is at random locations for other tapes.

→ ~~Transi~~ Movement can be $\{L, R, S\}$, S: stay.

$$\delta : Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, R, S\}^n$$

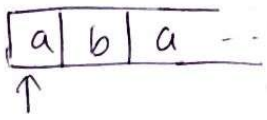
3. Multidimensional TM [2D tape]



- movement {L, R, U, D}
  U: up and D: down
- it is infinite to right and down but not to L & U.

4. Semi inifinite TM
   - It is a restricted variant of TM
   - can't move left when reading 1st symbol of the input, it is bounded.

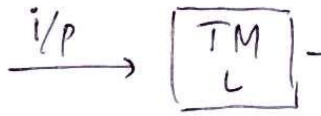
Eg:



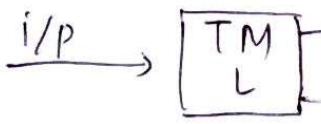
"here" can't move left

21/10 Theory

- Turing Machine : recursively enumerable lang.
- Diff b/w recursively enumerable & recursive lang.

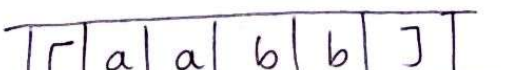


} recursively enumerable lang. (Turing recognisable)



} recursive lang. (Turing decidable)

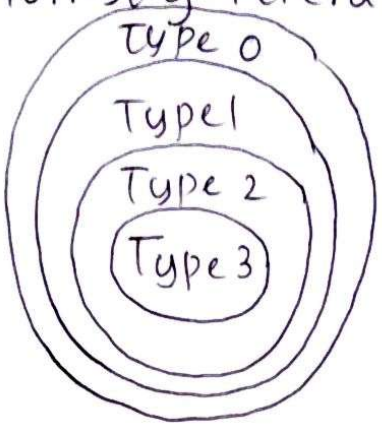
- Type 1 Grammar : CSG ⫋ LBA



$(Q, \Sigma, \Gamma, \delta, q_0, [,], F)$

CSG : context sensitive grammar.

- Chomsky Hierarchy



⇒ Explain each type including
- grammar
- rules
- Language
- Automata

	Grammar	Rules	Language	Automata
3 :	RG	$A \to a/aA/\varepsilon$ or $A \to a/Aa/\varepsilon$	RL	Finite A.
2 :	CFG	$A \to (VUT)^*$	CFL	PDA
1 :	CSG	$\alpha \to \beta$ $\alpha, \beta \in (VUT)^+$ and $\lvert\alpha\rvert \leq \lvert\beta\rvert$	CSL	LBA (Linear bounded automata)
0 :	unrestricted grammar	$\alpha \to \beta$	RE	TM

- Decidable & undecidable
- Halting problem and post correspondonce prob. (proove undecidable).