Push down automata (PDA)

- It is used as a parser of compiler
- Finite automata can recognize finite comparison in a lang. but it fails to recog. ∞ infinite lang. which are having comparison b/w symbols of lang. because to remember infinite comparison, infinite no. of states should be the there.
- Hence to accept such type of lang., we need to use push down automata.
- PDA can recognize lang. by using empty stack mechanism or final state mechanism.

⇒ Empty stack mechanism

- By reading complete string from L to R if end of the string gives empty stack then PDA is accepted irrelevant about no. of final states, then string is said to be accepted.

⇒ Final state mechanism

- By reading complete string from L to R if end of the string gives final state of PDA then given string is accepted by PDA, irrelevant about stack elements.

c) $\{a^i b^j c^k \mid j=i \text{ or } j=k\}$

S aabbc / abbcc

S → A/B
A → aMN
M → aMb/ε
N → cN/ε
B → PQ
P → aP/ε
Q → bQc/ε

d) $\{a^i b^j c^k \mid i=j \text{ or } i=k\}$

S → A/B
A → ~~aMb~~ MN
M → aMb/ε
N → cN/ε
B → ~~PQ~~ aBc/P
P → bP/ε

e) $\{a^i b^j c^k \mid i<j \text{ or } i>k\}$

S → A/B
A → PQ
Q → cQ/ε
~~P → aPM/N~~ ~~aMN/N~~ P → aPb/M
~~bN/b~~ M → bM/b
M → ~~bMb~~ bM/b

NOTE :- The no. of lang. accepted by empty stack mechanism = final state mechanism.

Tutorial 6 CFG

1) a) $\{a^i b^j c^k \mid i = j+k\}$

$a^{j+k} b^j c^k$

$\Rightarrow a^j a^k b^j c^k$

S → ~~ABCD~~ ABAB
A → ~~ab~~ aAb / ~~a~~ ϵ
B → aBC / ϵ

abac

S → AB
A → aAb / ϵ
B → bBC / ϵ

S → aSC / A
A → aAb / ϵ

b) $\{a^i b^j c^k \mid j = i+k\}$

S → aSbC / ~~aAbb~~ A
A → aAb / ϵ

$a^i b^j b^k c^k$
S → AB
A → aAb/ϵ
B → bBC/ϵ

c) ~~$\{a^i b^j c^k / j = i \text{ or } j = k\}$~~

S → ~~aSc / a~~ ~~aAc / abc~~
A →

S → aA / bB
A → aAb

B → ~~aXc / Y~~ aXYc / Z
~~Y → b bY / ε~~

eg- $\{a^i b^j c^k \mid j > i+k\}$

S → ~~aS~~ ~~AB~~ ACB
A → aAb / ~~b~~ ε
B → bBC / ε
C → bC / b

eg- $\{a^i b^j c^k \mid j > k\}$

S → A
A → aAC / B
B → ~~aB~~ XY
X → aX / a
Y → bY / ε

eg- $\{0^i 7^j 0^k \mid j > i+k\}$

⇒ S → ACB
A → 0A1 / ε
B → 1B0 / ε
C → 1C / 1

→

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

$S \rightarrow aSb \mid ab$

eg - For, Grammar, derive Left most & right most deriv^n of "1001"

$S \rightarrow A1B$

$A \rightarrow 0A \mid \epsilon$

$B \rightarrow 0B \mid 1B \mid \epsilon$

1) Leftmost (i.e. substitute leftmost symbol first)

$\Rightarrow A1B$ (using start S)
$\Rightarrow 1B$ ($A \rightarrow \epsilon$)
$\Rightarrow 10B$ ($B \rightarrow 0B$)
$\Rightarrow 100B$ ($B \rightarrow 0B$)
$\Rightarrow 1001B$ ($B \rightarrow 1B$)
$\Rightarrow 1001$ ($B \rightarrow \epsilon$)

2) Rightmost (i.e. subs. rightmost symbol first)

$\Rightarrow A1B$ (S)
$\Rightarrow A10B$ ($B \rightarrow 0B$)
$\Rightarrow A100B$ ($B \rightarrow 0B$)
$\Rightarrow A\epsilon 1001B$ ($B \rightarrow 1B$)
$\Rightarrow A1001$ ($B \rightarrow \epsilon$)
$\Rightarrow 1001$ ($A \rightarrow \epsilon$)

→ In parseTree,

1) Root :- Start symbol
2) Intermediate nodes :- variables (capital)
3) Leaf nodes :- terminals (small / ϵ)

eg- $S \to S+S \mid S-S \mid S*S \mid S \div S \mid a$;

f) $\{a^i b^j \mid i \leq 2j\}$

$S \to AA$
$A \to aaAb \mid B$
$B \to bB \mid \epsilon$

g) $\{a^i b^j \mid i < 2j\}$

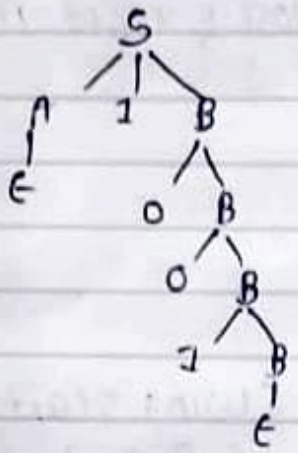
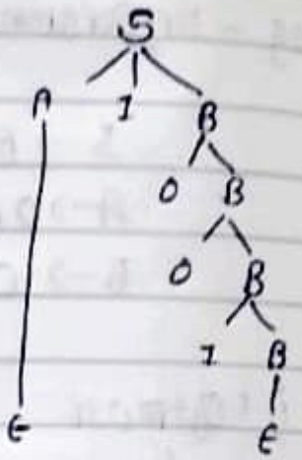
$S \to A$
$A \to aaAb \mid B$
$B \to bB$

h) $\{a^i b^j \mid i < j \leq 2i\}$

a	b	condit^n	string
0	0	$0 \leq 0 \leq 0$	ϵ ✓
0	1	$0 \leq 1 \in 0$	b X

try $2i \leq j \leq 3i$

- using parse tree for last example:-

Leftmost deriv^n (1001)	Rightmost deriv^n (1001)
	

- In sentential form, just write product in sentence

eg → $S \to AB$
↓ random example. $\to BAB$
$\to BOB$
$\to AOA$
$\to OOO$

write sentence. $(A \to BA)$
$(A \to 1)$
$(B \to A)$
$(A \to 0)$

→ write grammar

eg- $L = \{a^n, n \geq 0\}$

$\therefore S \to aS / \epsilon$

eg- $L = \{a^n, n \geq 1\}$
$\therefore S \to aS / a$

eg - $L = \{ b, ab, aab, aaab, \ldots \}$
$= \{ a^n b, n \geq 0 \}$

$S \to aA \; Ab$
$A \to aA / \epsilon$

$\Rightarrow S \to aS / b$

eg - $L = \{ w \in \{a,b\}^* \}$
aS bS
$\therefore S \to aA / bA / \epsilon$

eg - $L = \{ a^n b^n ; n \geq 0 \}$

$\therefore S \to aSb / \epsilon$

eg - $L = \{ w \in \{a,b\}^*$, w is a palindrome of odd length & $|w| > 0 \}$

$\therefore S \Rightarrow$ strings $\Rightarrow$ aba, abba, abaaba, bbabb
(b o b; a S a; b S b)

$S \to aSa / bSb / a / b$

eg - $L = \{ w \in \{a,b\}^*$, w is palindrome of even Length$\}$.

$S \to aSa / bSb / \epsilon$

eg - $L = \{ w \in \{a,b\}^*$, w is a palindrome$\}$

$S \to aSa / bSb / a / b / \epsilon$

eg - $L = \{a^m b^n \;,\; m \geq n \;,\; n \geq 0\}$

$S \rightarrow AB$
$A \rightarrow aA / \epsilon$
$B \rightarrow aBb / \epsilon$

eg - $L = \{a^m b^n \;,\; n > m\}$.

$S \rightarrow AB$
$A \rightarrow aAb / \epsilon$
$B \rightarrow bB / b$

eg - $L = \{$ no. of a = no. of b $\}$

~~$a^n b^n$, $b^n a^n$~~ .

$S \rightarrow aSbS / bSaS / \epsilon$ ~~Sb / bSa~~ ϵ

strings → abab abba

a S b
a ε b

Q: $S \rightarrow aSb / bSa / SS / \epsilon$

all abab.. ~~or abba~~ aabb
all baba.. ~~or baab~~ bbaa
all abba.. or baab..

eg - $L = \{ w \in \{a,b\}^* , w \text{ is palindrome}, |w| > 0 \}$

$S \to A=B \; aSa / bSb / a / b / aa / bb$

eg - $L = \{ a^n b^{n+2}, n \geq 0 \}$

$S \to aSb / bb$

eg - $L = \{ a^{2n} b^n, n \geq 0 \}$

$S \to aaSb / \epsilon$

eg - $L = \{ a^{2n} b^n / n \geq 1 \}$

$S \to aaSb / aab$

eg - $L = \{ a^{2n+3} b^n, n \geq 0 \}$

$S \to aaSb / aaa$

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

$S \to AAB / aa$
$A \to aA / a$
$B \to bB / \epsilon$

OR

$S \to AB$
$A \to aA / a$
$B \to aBb / \epsilon$

eg. $L = \{a^i b^j c^k, \ i = j+k, \ j,k \geq 1\}$

$S \to a^i a^j b^j c^k$

$S \to aSc \mid aAbc$
$A \to aAb \mid ab$

⇒ if $j,k \geq 0$ ↳ so ϵ present in grammar

$S \to aSc \mid A$
$A \to aAb \mid \epsilon$

eg : $L = \{a^i b^j c^k, \ i+j = k, \ i,j \geq 1\}$

$S \to aSc \mid aAc$
$A \to bAc \mid bc$

eg- $L = \{a^i b^j c^k, \ j = i+k, \ i,k \geq 1\}$

$S \to AB$
$A \to aAb \mid ab$
$B \to bBc \mid bc$

⇒ If $i,k \geq 0$,

$L = \{a^i b^j c^k$

$S \to AB$
$A \to aAb \mid \epsilon$
$B \to bBc \mid \epsilon$

eg - $L = \{a^i b^j c^k \mid i=j,\ i,j,k \geq 1\}$

$S \to AB$
$A \to aAb / ab$
$B \to cB / c$

eg - $\{a^i b^j c^k,\ j=k,\ i,j,k \geq 1\}$

$S \to AB$
$A \to aA / a$
$B \to bBc / bc$

eg - $\{a^i b^j c^k,\ i=k,\ i,j,k \geq 1\}$

$S \to aSc / aAc$
$A \to bA / b$

eg - $L = \{0^i 1^j 0^k,\ j > i+k,\ i,k \geq 1\}$

let $a^i b^j c^k$

~~$S \to ABC$~~
~~$A \to a$~~

~~$S \to aS \mid aAc$~~
$S \to AB$
$A \to aAb / ab$
$C \to bCc / bc$
$B \to bB / b$

now replace $a=0$, $b=1$, $c=0$.

eg - $L = \{a^n b^m, n \neq m\}$

$n > m$, $n < m$

$S \rightarrow A / B$

$A \rightarrow aAb / aPb$

$P \rightarrow aP / a$

$B \rightarrow aBb / aQb$

$Q \rightarrow bQ / b$

OR

$S \rightarrow aSb / A / B$

$A \rightarrow aAb / a$

$B \rightarrow bB / b$

eg - $\{a^n b^m c^k, \quad m \leq k\}$

$S \rightarrow AB$

$A \rightarrow aA / \epsilon$

$B \rightarrow bBc / C / \epsilon$

$C \rightarrow cC / \epsilon$

OR

$S \rightarrow XYZ$

$X \rightarrow aX / \epsilon$

$Y \rightarrow bYC / \epsilon$

$Z \rightarrow cZ / \epsilon$

eg - $L = \{ 0^i 1^j 0^k \mid j > i+k \;,\; i,k \geq 0 \}$

let $a^i b^j c^k$

$S \rightarrow A\,B\,C$
$A \rightarrow aAb \mid \epsilon$
$B \rightarrow bB \mid b$
$C \rightarrow bBc \mid \epsilon$

now, replace $a \Rightarrow 0,\ b = 1,\ c = 0$

eg - $\{ w \in \langle a,b \rangle^* \mid |w| \text{ is even} \}$

$S \rightarrow aSb \mid bSa \mid aSa \mid bSb \mid \epsilon$

eg - $\{ w \in \langle a,b \rangle^* \mid |w| \text{ is odd} \}$

$S \rightarrow aSb \mid bSa \mid aSa \mid bSb \mid a \mid b$

trick :) for $\langle a,b \rangle$

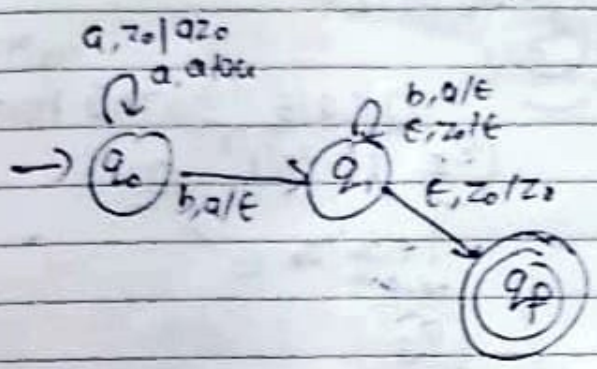
$S \rightarrow aX \mid bX$
$X \rightarrow aS \mid bS \mid \epsilon$

for $\langle a,b,c \rangle$
$S \rightarrow aX \mid bX \mid cX$
$X \rightarrow aS \mid bS \mid cS \mid \epsilon$

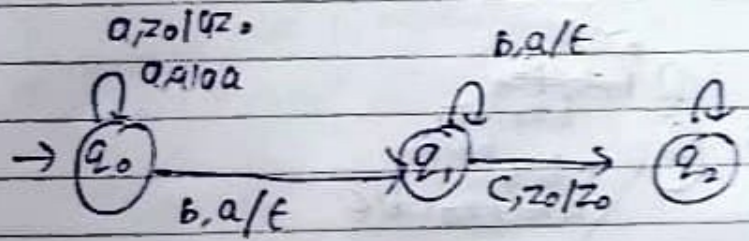
$\delta(q_1, \epsilon, Z_0) = \{q_1, \epsilon\}$ ← empty stack mechanism

$\equiv$

$\{q_f, Z_0\}$ ← final state mechanism



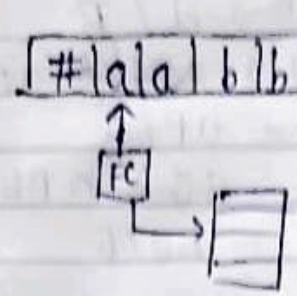
$L = \{a^n b^n c^m, n, m \geq 1\}$.



Here we are not using c as it is not pushed in stack.

→

FA → PDA → more expressive power than DFA by 1 comparison



PDA = $\{Q, \Sigma, \delta, F, q_0, Z_0, \Gamma\}$ → stack symbol

- Q → states
- Σ → symbols
- δ → transition funct^n
- F → final state
- q_0 → initial state

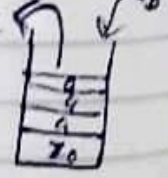
- Here Z_0 is a initial stack element
- Here Γ is a stack alphabet which contains finite no. of elements with push & pop oper^n.

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

↳ transition funct^n defin^n. (push / pop).

Q- Const. a PDA for :-

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



transition. → TOP of stack → new state.

init. state

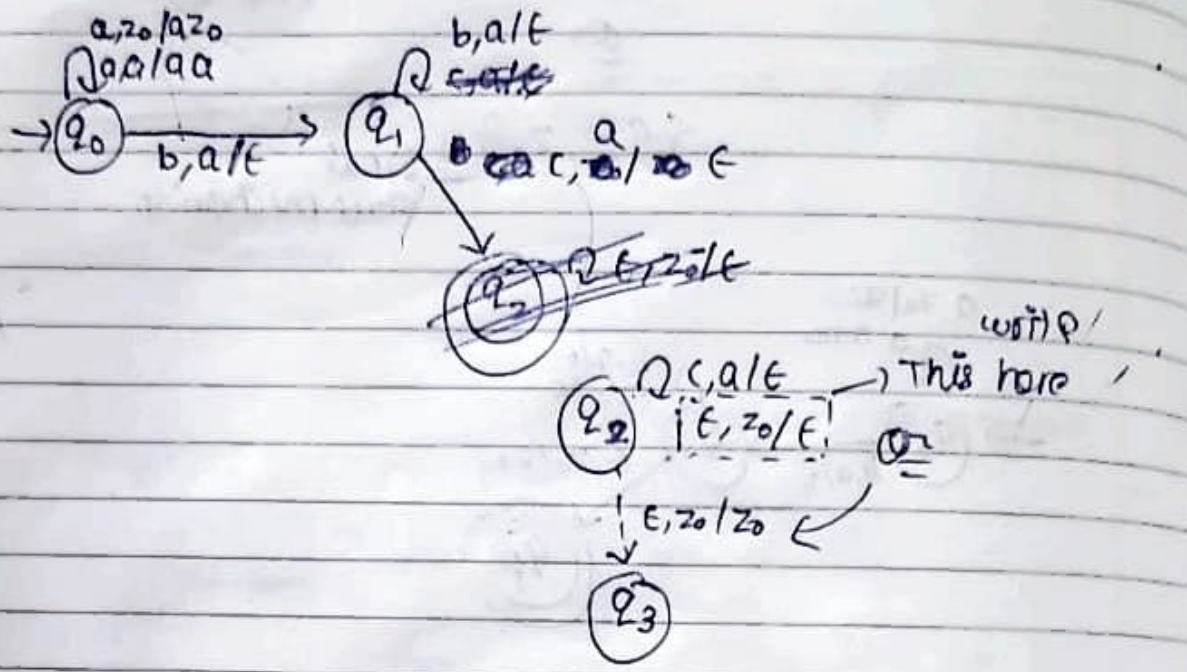
$\delta(q_0, a, Z_0) = \{q_0, aZ_0\}$ } push

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

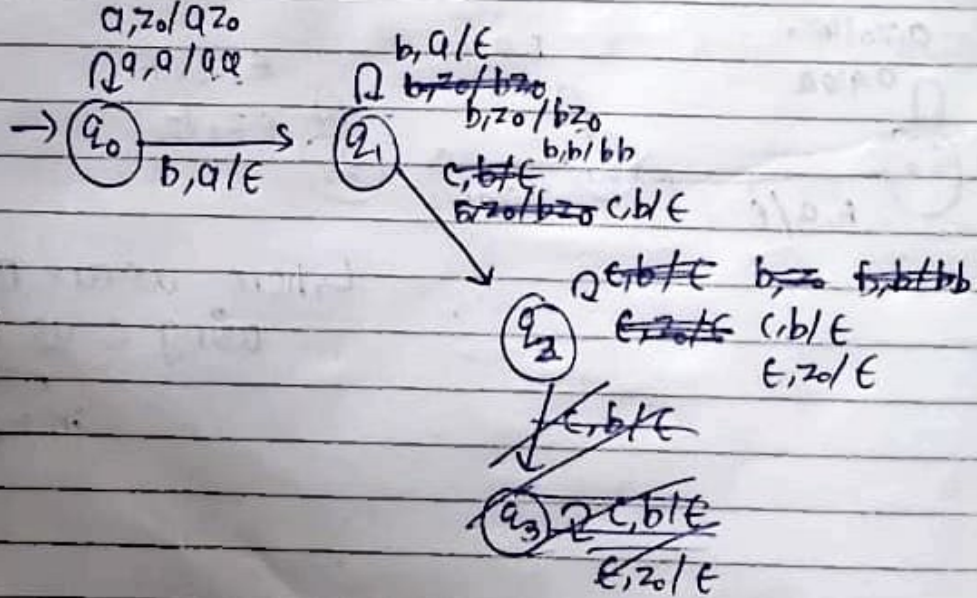
$\delta(q_0, b, a) = \{q_1, \epsilon\}$ } pop - change state

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

Q- $L = \{ a^{m+n} b^m c^n \mid n, m \geq 1 \}$.

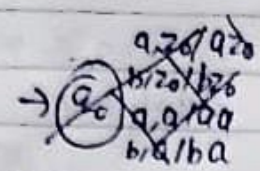


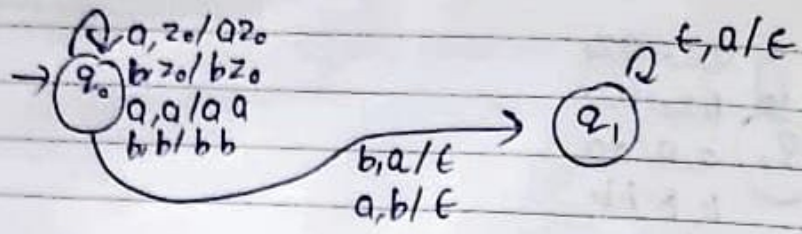
Q- $L = \{ a^m b^{m+n} c^n \}$



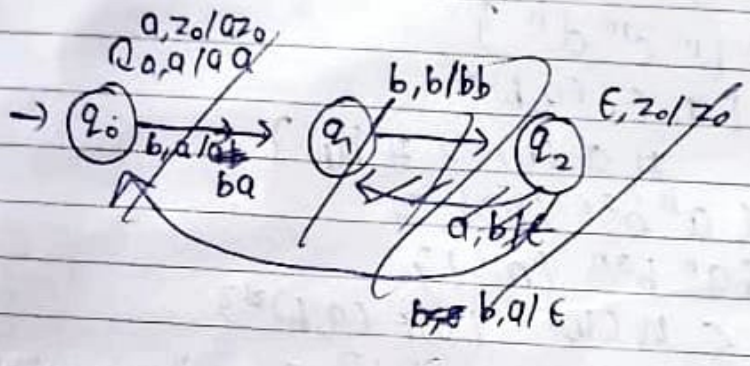
☆. we can do ⇒ push 2a's
a, z₀ / a a z₀ tag / ha.

⇒ 2)

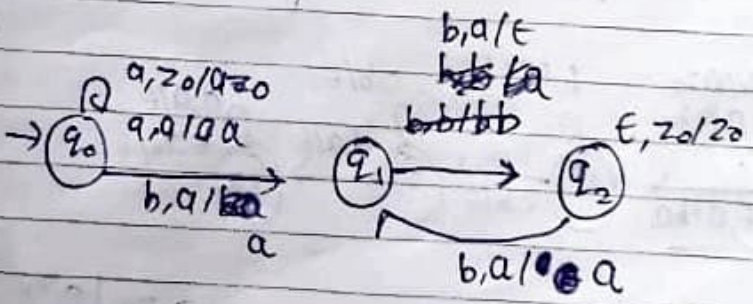




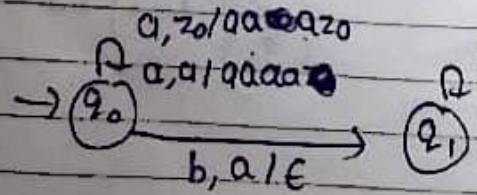
3) $\{a^n b^{2n} \quad , n \geq 1\}$



4)

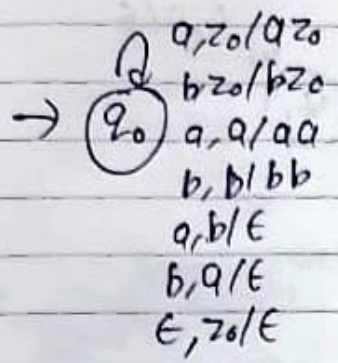


5) $\{a^n b^{3n} \quad , n \geq 1\}$



Q- Const. PDA of all strings of a's & b's Accept ~~Except~~ equal a's & b's. (only loc^n of a & b eg - abab abba etc.)

A-



Q 1) $L = \{a^n b^m c^m d^n\}$

2) $L = \{n \mid n \in (a,b)^* \; \# a(n) > \# b(n)\}$

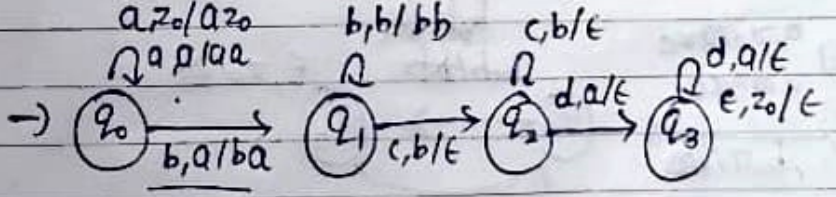
3) $L = \{a^n b^{2n} \mid n \geq 1\}$

4) $L = \{a^n b^{3n} \mid n \geq 1\}$

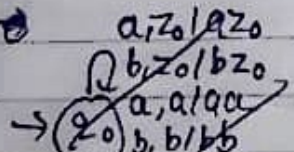
5) $L = \{w w^R \mid w \in (a,b)^*\}$

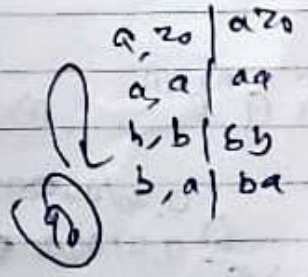
6) $L = \{a^n b^n c^n ; n \geq 1\}$ → not possible

⇒1)



⇒2)

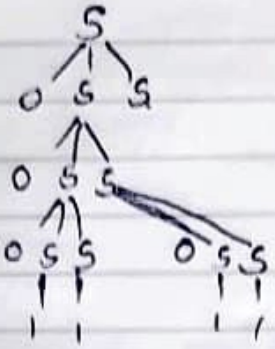




⇒ minimize.

Q- $S \rightarrow 1 / 1S / 0SS / SS0 / SOS$

check :) 000110111O



Q- ⇒ 000110111O

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

$A \rightarrow 0S / 1AA / 0$

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

0 S

⇒ R1: $(q_0, 0, 0) \rightarrow (q_0, B)$

R2: $(q_0, 1, S) \rightarrow (q_0, A)$

R3: $(q_0, 0, A) \rightarrow (q_0, S)$

R4: $(q_0, 1, A) \rightarrow (q_0, AA)$

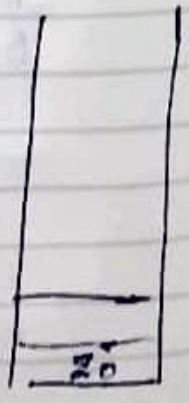
R5: $(q_0, 1, B) \rightarrow (q_0, S)$

R6: $(q_0, 0, B) \rightarrow (q_0, BB)$

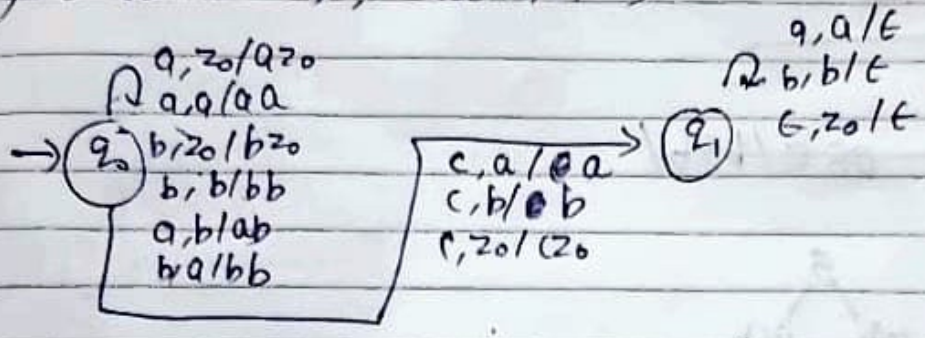
⇒ 000110111O

1) 0

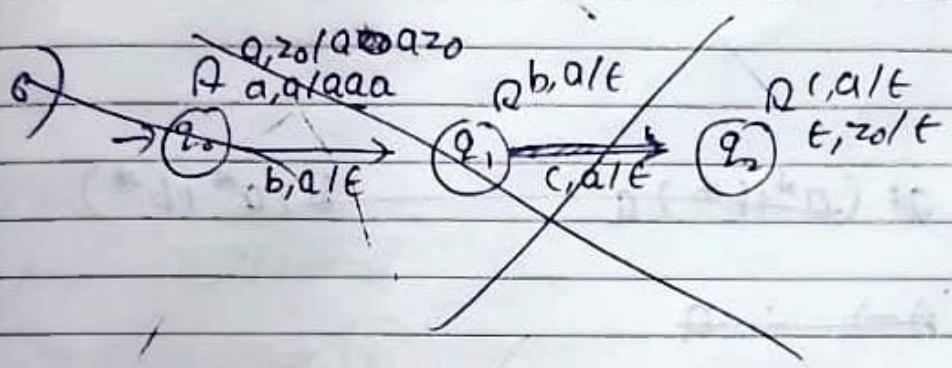
$(q_0, 0, S) \rightarrow (q_0, B)$



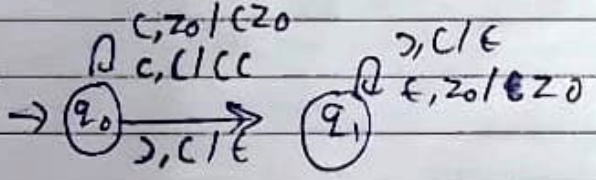
5) $L=\{wcw^R \mid w \in \{a,b\}^*\}$



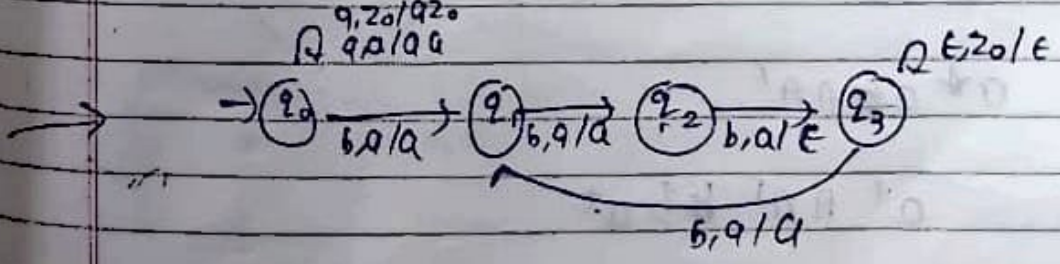
6)



7) ((((()))))



8) 4) II)



NOTE
- every D-PDA will have exactly 1 transit^n, & N-PDA will have more than 1 transition
- All D-PDA are N-PDA ; but reverse is not true.
- The language accepted by N-PDA are called CFL or N-CFL → context-free
- Hence, all DCFL (deter. CFL) are are CFL.

eg- where PDA is not possible

1) $\{wcw,\ w\in\{a,b\}^*\}$
2) $\{w,w^R,\ w\in\{a,b\}^*\}$
3) $\{a^nb^nc^n,\ n\geq 1\}$

Remember →

Conversion of CFG to PDA

[and Conversion of PDA to CFG (self study).
- mealy & moore machine (self study)
↳ Long. accep generator.]

- CFG → PDA

Type:- $A \to \alpha/\alpha/\epsilon$ $(V+T)^*$

① $\delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$
② $\delta(q_1, \epsilon, S) = (q_1, \lambda)$

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

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

ex-1) $S \to aSb \mid ab$

$\delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$

$\delta(q_0, \epsilon, S) = (q_1, aSb)$ or (q_1, ab)

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

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

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

eg - 2) $S \to aAb$

$A \to bB/a$

$B \to bB/b$

1) $\delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$

2) $\delta(q_1, \epsilon, A) = (q_1, bB)$ or 4) (q_1, a)

3) $\delta(q_1, \epsilon, B) = (q_1, bB)$ or 6) (q_1, b)

$\delta(q_1, \epsilon, S) = (q_1, aAb)$

5) $\delta(q_1, A, S) = (q_1, SA)$

6) $\delta(q_1, B, A) = (q_1, BA)$

~~5)~~ ϵ

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

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

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

classmate

Date
Page

Using GNF,

1) $\delta(q_0, \epsilon, z_0) \rightarrow (q_1, Sz_0)$
2) $\delta(q_1, a, S) \rightarrow (q_1, aA)$
3) $\delta(q_1, a, A) \rightarrow (q_1, aABD)$
4) $\delta(q_1, b, A) \rightarrow (q_1, B)$
5) $\delta(q_1, a, A) \rightarrow (q_1, \epsilon)$
6) $\delta(q_1, b, B) \rightarrow (q_1, \epsilon)$
7) $\delta(q_1, d, D) \rightarrow (q_1, \epsilon)$
8) ~~$\delta(q_1, a, a) \rightarrow (q_1, \epsilon)$~~
9) ~~$\delta(q_1, b, b) \rightarrow (q_1, \epsilon)$~~
10) ~~$\delta(q_1, d, d) \rightarrow (q_1, \epsilon)$~~
11) ~~$\delta(q_1, \epsilon, z_0) \rightarrow$~~
8) $\delta(q_1, \epsilon, z_0) \rightarrow (q_f, \epsilon)$

Watchup - class questions repractice

→ non GNF → PDA $(Q, \Sigma, \delta, \Gamma, z_0, q_0, F)$

↳ (VUT) → we push both variables & terminals

eg-1) $S \rightarrow aSb \mid ab$

we have ϵ moves for variables

① ⇒ Pushing S

1) $\delta(q_0, \overset{\epsilon}{\emptyset}, z_0) = \delta(q_0, Sz_0)$

② transitions for variables

2) $\delta(q_0, \overset{\epsilon}{a}, S) = \delta(q_1, aSb)$
3) $\delta(q_0, \overset{\epsilon}{a}, S) = \delta(q_1, ab)$

③ transit^ for terminals → only pop.

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

④ Final state

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

eg - 3) $S \to aA$

$A \to aABD \mid bB \mid a$

$B \to b$

$D \to d$

→ 1) $\delta(q_0, \epsilon, z_0) \to (q_1, Sz_0)$

2) $\delta(q_1, \epsilon, S) \to (q_1, aA)$

3) $\delta(q_1, \epsilon, A) \to (q_1, aABD)$ or (q_1, bB) or (q_1, a)

4) $\delta(q_1, \epsilon, B) \to (q_1, b)$

5) $\delta(q_1, \epsilon, D) \to (q_1, d)$

6) $\delta(q_1, a, a) \to (q_1, \epsilon)$

7) $\delta(q_1, b, b) \to (q_1, \epsilon)$

8) $\delta(q_1, d, d) \to (q_1, \epsilon)$

9) $\delta(q_1, \epsilon, z_0) \to (q_f, \epsilon)$

→ If

→ If grammar in GNF

eg - $S \to aA/bB$ ⇝ eg - $S \to aSB/bB$

$B \to b$

1) $\delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$

2) $\delta(q_1, a, S) \to (q_1, SB)$

$\delta(q_1, b, S) \to (q_1, B)$

$\delta(q_1, bB, B) \to (q_1, \epsilon)$

3) $(q_1, \epsilon, z_0) \to (q_f, \epsilon)$

Eg-2

S → aAb
A → bB/a
B → bB/b

① ⇒ pushing S

1) $\delta(q_0, \epsilon, Z_0) = \delta(q_0, SZ_0)$

② ⇒ transitn for variables

A ⇒ 2) $\delta(q_1, \epsilon, A) = \delta(q_1, bB)$
3) $\delta(q_1, \epsilon, A) = \delta(q_1, a)$
B ⇒ 4) $\delta(q_1, \epsilon, B) = \delta(q_1, bB)$
5) $\delta(q_1, \epsilon, B) = \delta(q_1, b)$
S ⇒ 6) $\delta(q_1, \epsilon, S) = \delta(q_1, aAb)$

} ε moves

③ transitn for terminals

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

④ Final state

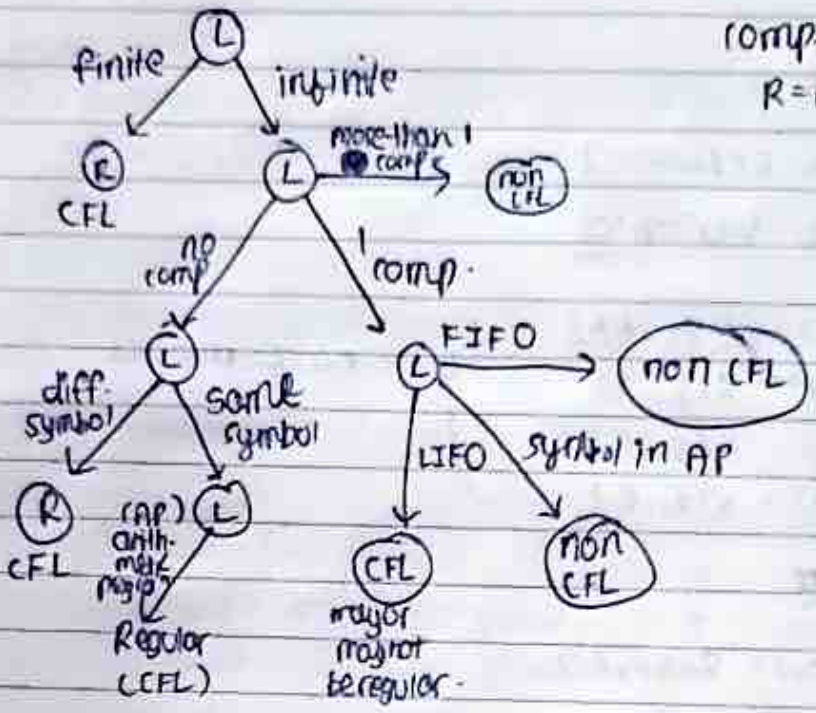
9) $\delta(q_1, \epsilon, Z_0) = \delta(q_f, Z_0)$

→ GNF

· PDA $(q_0, Q, \delta, F, \Gamma, \Sigma, Z_0)$
↳ only V ⇒ we only push variables

· Here, we don't have ε moves

→ Identific^n Tree



comp = comparison.
R = regular.

Q- which of the following lang. is ~~Regular & CFL~~. follow following opt's:

(a) Reg. & CFL
(b) CFL but not reg.
(c) non CFL.

1) $L = \{a^n b^n c^n \;,\; n \geq 1\}$
(a) Fin. & ~~Reg~~ (Reg.) CFL

2) $L = \{a^n b^n c^m d^m$, where $n, m > 1\}$
(B)

eg- $S \to OBB$
$B \to OS / IS / O$

① pushing s

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

② transits for variables

2) $\delta(q_0, 0, S) = \delta(q_1, BB)$
3) $\delta(q_1, 0, B) = \delta(q_1, S)$
4) $\delta(q_1, 1, B) = \delta(q_1, S)$
5) $\delta(q_1, 0, B) = \delta(q_1, \epsilon)$

} no ϵ moves

③ Final state

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

7)

eg- $S \to aA$
$A \to aABD / bBa$
$B \to b$
$D \to d$

1) $\delta(q_0, \epsilon, z_0) = \delta(q_0, Sz_0)$ } pushing S
2) $\delta(q_0, a, S) = \delta(q_1, A)$
3) $\delta(q_1, a, A) = \delta(q_1, ABD)$
4) $\delta(q_1, b, A) = \delta(q_1, B)$
5) $\delta(q_1, a, A) = \delta(q_1, \epsilon)$
6) $\delta(q_1, b, B) = \delta(q_1, \epsilon)$
7) $\delta(q_1, d, D) = \delta(q_1, \epsilon)$

} transit^ for variables

8) $\delta(q_1, \epsilon, z_0) = \delta(q_f, \epsilon)$ } final state

3) $\{a^i b^j c^k , j = i+k\}$
(B)

4) $\{a^i b^j c^k d^l , i=k \& j=l\}$
(C)

*5) $\{a^i b^j , i = j^2\}$
(C)

6) $\{ww^R , w \in \{a,b\}^* \& len(w) = 1\}$
(A)

7) $L = \{www , w \in \{a,b\}\}$
(A)

8) $L = \{a^{2^n} , n \geq 1\}$
(C)

9) $L = \{a^p , p = prime\}$
(C)

*10) $L = \{\Sigma^* - L1 , L1 = ww , w \in \{a,b\}^+\}$
(B)

*11) $L = \{1, 2, 4, 6, \cdots, 2^n$ contain in unary$\}$
(C)

12) $L = \{$ set of expressn statements using variables without declarn in programing$\}$
(C)

3) Klleen closure, +ve closure, subset

- CFL is closed under Kleen
- DCFL is may or may not be closed under

4) Intersect

- Both CFL & DCFL may or may not be closed.

5) ~~Not~~

5) Intersect with regular

- CFL ∩ Regular = CFL. ↳ closed. , DCFL ∩ reg ⇒ DCFL closed

6) complement

- CFL & DCFL may or may not be closed
  ↳ L_1, L_2 are closed.

7) Difference

- CFL & DCFL may or may not be closed.

8) Reversal

- ~~closed~~ CFL closed under reversal.

→

13) I-L total no. of class conducted for 2CSE013

(A)

NOTE → • If lang. is Reg., it should be CFL, or lang. is not Reg., it may or may not be CFL

→ Closure properties of CFL & DCFL.

1) Union

~~True for~~

· Let L1 & L2 be any 2 CFLs & S1 & S2 are 2 CFGs respective to L1 & L2., then L1∪L2 is always CFL.

★ · Hence CFLs are closed under union oper^.

· While union of 2 DCFL may or may not be DCFL. hence

★ DCFL is not closed under union oper^.

2) concater^

· CFL closed under concat^

· DCFL may or may not be closed under concat^.

9). Diff. with regular

CFL ∩ R
• CFL & DCFL closed under ∩ R

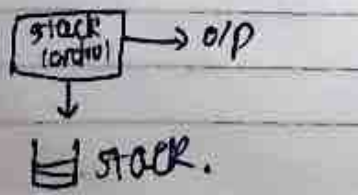
Q- (L1 - L3) ∪ (L2 - L4)
L1, L2 → CFL
L3, L4 → reg.
⇒ CFL

Q- $(L_1^* \cap L_3^R) \cap ((\Sigma^* - L_3) \cap L_4)$
L1 → CFL, L3 → CFL → reg, (Σ* - L3) → reg, L4 → CFL
CFL

→ Turing Machine

• Type 0 Language → Recursively enumerable Language.

• PDA has stack control.

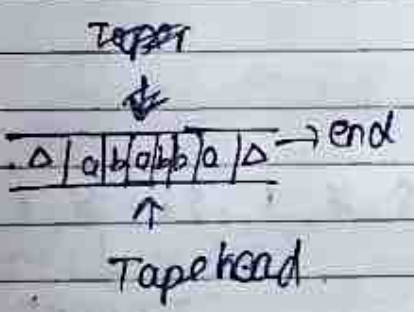


classmate
Date
Page

· Turing machine has:-

· i/p string → to read i/p
· Tape → starting & ending & inbetween string
· special symbol → blank symbol (δ) to define dead states
· control portion
  ↳ 2 control - 1) (read & write) to read a string & overwrite.
  2) also left or right movement

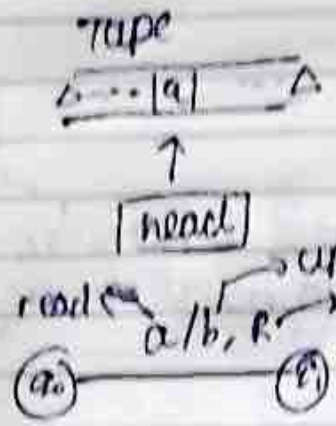
b c a
left move ✓ allowed
right move ✓ allowed



· Tape is capable of doing 3 oper^n -
  - Read a symbol ve above tape head
  - modify / update a symbol above tape head.
  - Shift either L or R

→ L or R movement or read / write is control movement.

* Tape read + control = Turing machine



now, we perform this till we get blank state.

⇒ b/a , L
⇒ a/Δ , R → replace a by Δ.
⇒ a/a , R
↳ skipping to R

7 tuples :- for Turing machine

$T = (Q, \Sigma, \Gamma, q_0, \delta, F, B)$

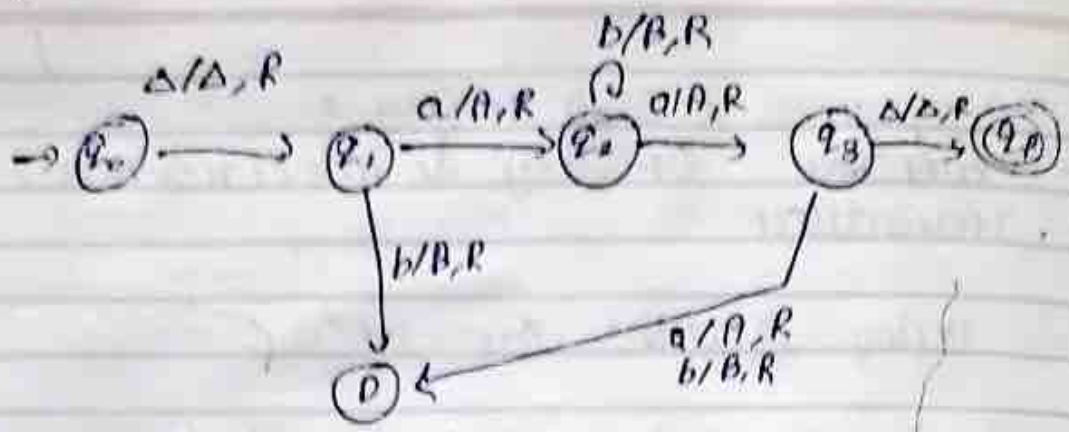
- Γ → final
- δ → Transition
- B → Blank symbol (end of string or dead state)

$\delta \Rightarrow (\delta : Q \times \Sigma \to Q \times \Gamma \times (L/R))$

B ⇒ Δ or B

eg- 1's complement generator :-

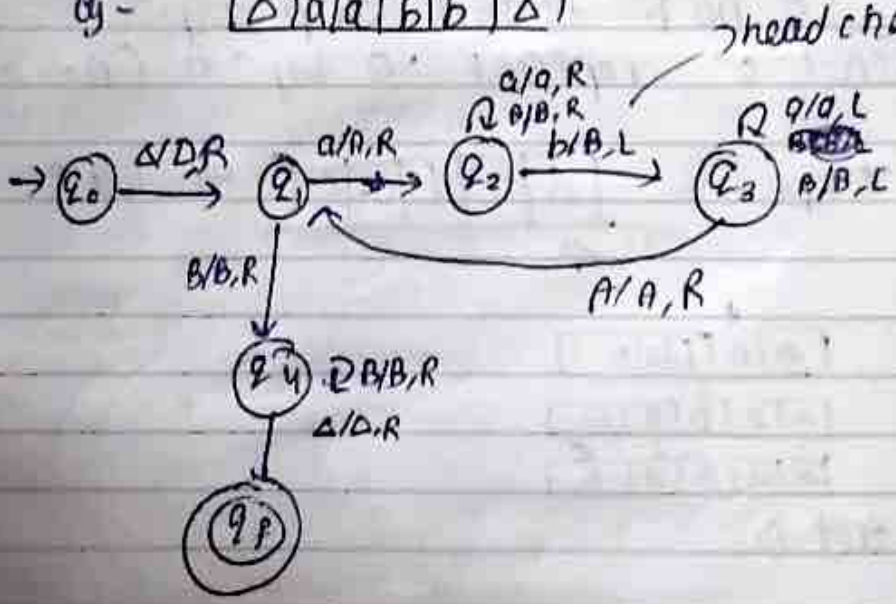
ip 0101
o/p 1010



Q- $\{a^n b^n, n \geq 1\}$

aa bb

Q- | Δ | a | a | b | b | Δ |



Types of Turing machine ⇒ . Acceptor (lang. acceptor)
. Generator (mealy or moore machine)
. Transducer

Q- Turing machine for ab^*a

step1) min string to be accepted

strings accepted ⇒ aa, aba, abba, ...

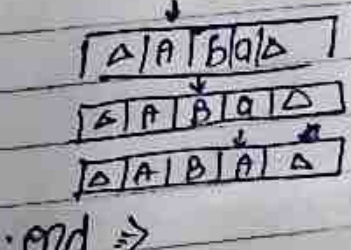
Logic

1) Read 'a', replace 'a' by 'A' (a → A)
2) Read all 'b' one by one & replace it by 'B' (b → B)

If no 'b' found, skip step 2

3) Read 'a', replace 'a' by 'A' (a → A)

Tape ⇒ | Δ | a | b | a | Δ | Eg - Table



⇒ end ⇒

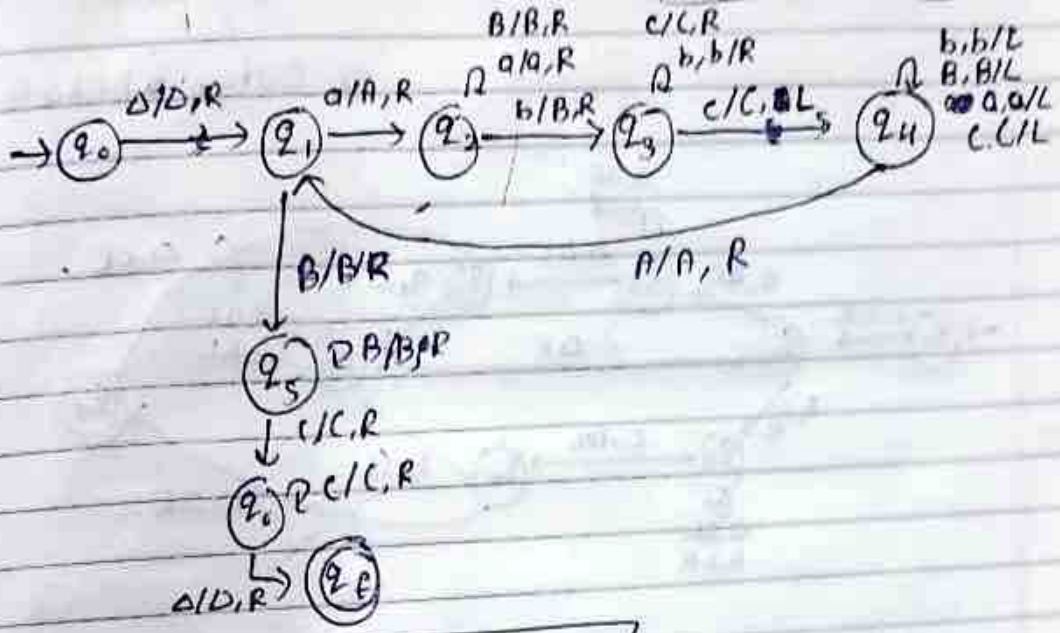
similarly for ~~ABC~~ abba,

'ΔABBAΔ'

for aa :-

'ΔAAΔ'

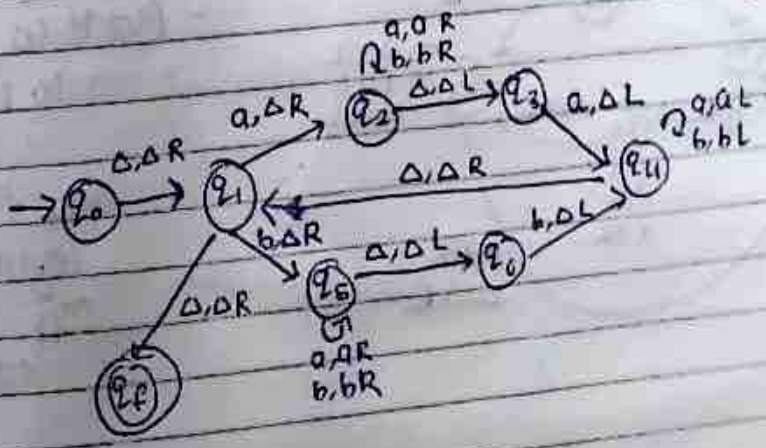
3. $\{a^n b^n c^n, n \geq 1\}$ | $\Delta|ab|bb|cc|\Delta$



check ⇒	Aa Bb Cc
(Trick)	aabb cc

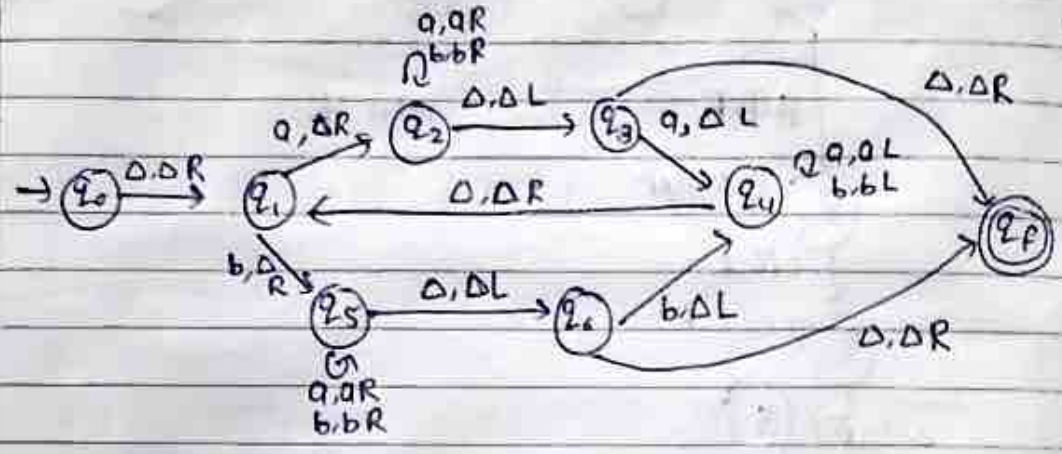
- Turing machine accepting even length palindrome.

eg- ΔabaabbaabaΔ



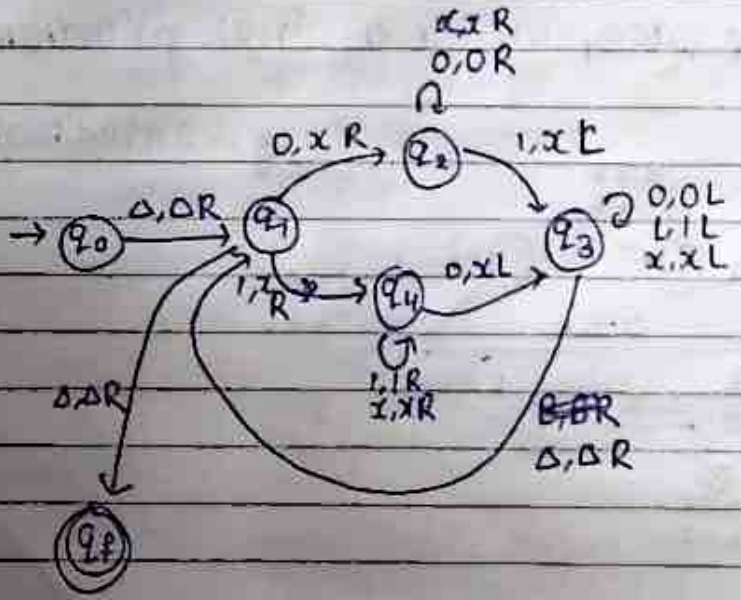
Q- Turing machine accepting odd length palindromes

eg- Δabab ; a,baba Δ



Q- Turing machine to accept equal no. of 0's & 1's

eg- Δ 0110100Δ



Logic

- Find Leftmost 1 or 0
- Put it as x
- Go to beginning (Δ)
- Now find Leftmost 0 or 1 (opp. to the letter found earlier)
- Put it x
- Again go to beginning
- Repeat

→ Recursive Language

- By reading symbol x turing machine halts in non final states, then given string is rejected and by reading some shift symbol, Turing machine always halts in final states then the lang. is accepted.

→ Recursive Enumerable Lang. → superset of Recursive lang.

By reading symbol x, turing machine may halt, may not halt & go in ∞ state

→ Halting problem → whether Turing machine halts or not cannot be decided.

- Undecidable problem
- we cannot design a generalized program which takes any string as i/p & says. if Turing machine halts or not.

Proof

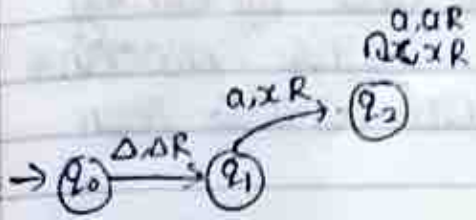
Let H(P, I) (P → program, I → input) be decidable → either Halts or doesn't halt
- Final or non-final state

Let C(x) be a program.

```
C(x)
  if (H(x,x) == Halt)
        Loop forever.
  else Halt (return)
```

Q- Turing machine to accept equal no. of a, b & c

eg- Δ abaccb Δ



If we run C on itself
then in C(C),
H(C,C) ~~= Halt~~ == Halt } contradictory.
then H(C,C) = Not Halt.

It means we checked the condition as Halt, but inside we observe that H(C,C) is not Halt.

. So, proof by contradict^n.

→ Universal Turing Machine

- It accepts some of the Turing machine.
$A_{TM} = \{ \langle M, w \rangle \mid M$ is a turing machine & M accepts $w \}$

. Turing machine to check a Turing machine.

Q- given desc. of a TM, & some i/p string, can we decide if machine accepts it?
A- run the TM on i/p.

possible results - M accepts w - halt & accept
- M rejects w - halt & reject
- M loops on w - not halt

i/p → · M - desc. of TM
w - i/p string for M.

action - run M on w.

→ Church Turing Hachine

`Any algo. procedure that can be carried out by a human computer or a team of computers can be carried out by a TM'.
·This is Church TM thesis.

Summary

· Humans need 2-D sheet of paper. TM tape can simulate 2D, but it would req. more moves than human.
· Various enhancements of TM - eg - multiple TM (multiple tapes)
· other theoritical models - abstract machines with 2 stacks or with a queue
since introduct^n of TM, no one suggested any type of comput^n that ought to be included in category of `algorithmic procedure' & cannot be implemented on TM.

- - - -

→ We have 2 types of o/p generators
-1) Mealy - o/p dep. on both - present i/p & present st
-2) Moore - o/p depends on present state only

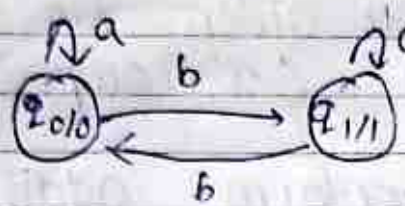
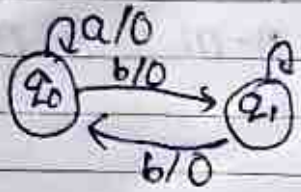


abb ⇒ o/p = 0
aba ⇒ o/p = 1

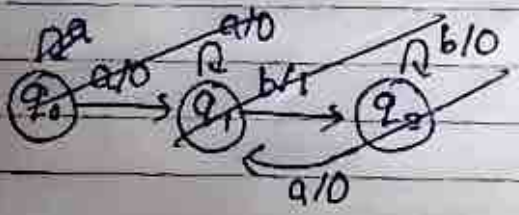
↳ Moore machine

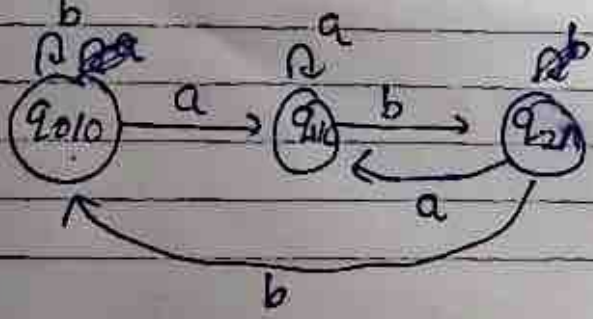


a/1 ⇒ i/p = a
o/p = 1

↳ mealy machine.

Q1- Const. Moore machine that gen. all strings of a & b & counts total no. of ~~a & b~~ 'ab' substring.





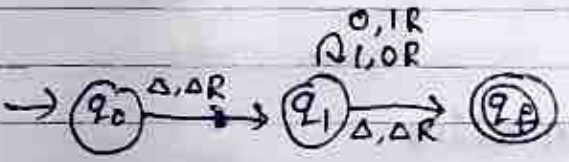
Q-1 Const. of TM for 1's comp. of given binary number.

A-2 Q-2 Const. a TM that produces 2's comp. of given bin. string as o/p (2's comp.)

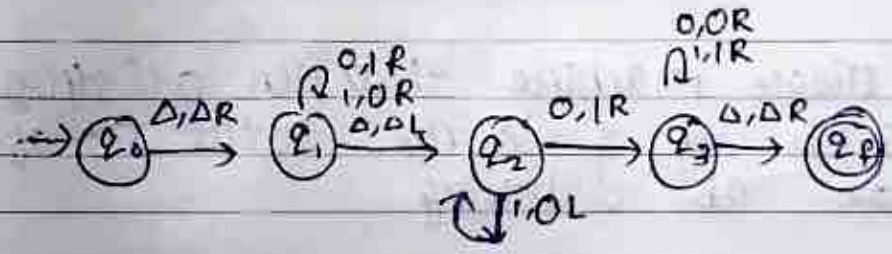
Q-3 Const. TM that performs addition of 2 binary nos.

Q-4 Const. a TM that performs subtract^n of 2 +ve nos where n-m, & n>m

A-1

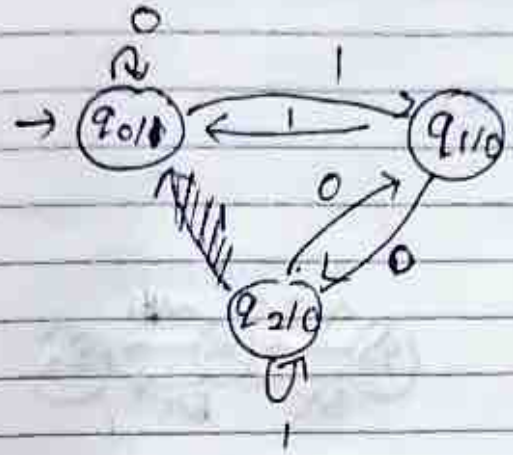


A-2



A-3

Q Design moore machine taking all bin. no. as I/p & produce modulo 3 as o/p.

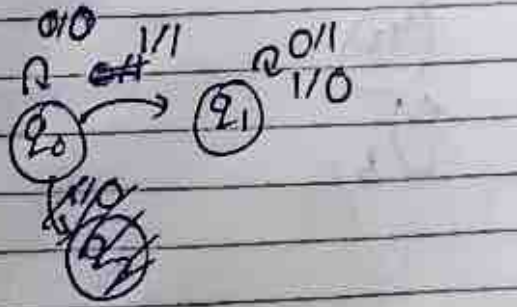


Q- Design mealy machine of 1's comp. of given bin. no.

A-

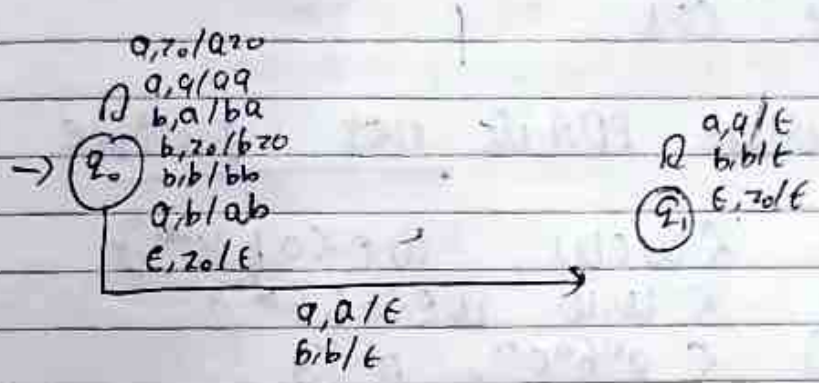
Q Design mealy machine for 2's comp. reading LSB to MSB

A-



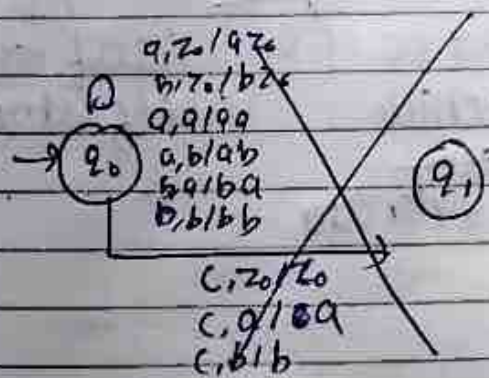
$L = \{ w w^R \mid w \in (a,b)^* \}$

- By default - PDA is N-PDA.
- every N-PDA may or may not have a PDA.
  - ↓ DCF = deterministic context free



Q. $L = \{ wcw \mid w \in (a,b)^* \}$

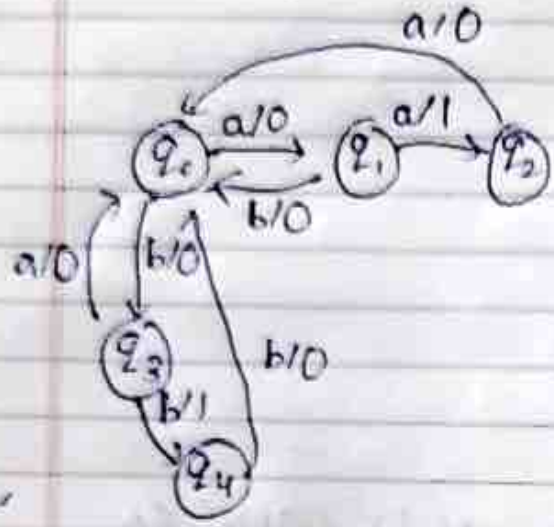
↳ not possible



$L = \{ ww \mid w \in (a,b)^* \}$

↳ not possible.

Q- Const. mealy machine taking all i/p of a & b & produce 1 as o/p iff last 2 symbols of o/p are same.



1