✱ GREIBACK NORMAL FORM (GNF):

Every content free language without 'ϵ' can be generated by the grammar in which all productions are of the form -

$A \rightarrow a\alpha$

a: Terminal

α: Non-terminal

A: Non-terminal

Such type of grammar is said to be GNF. Thus, in this normal form the restrictions on the production are:

→ RHS should contain only one terminal symbol and that should be the leftmost symbol on the RHS followed by '0' or more non-terminal.

$A \rightarrow a$

$A \rightarrow aB$

$A \rightarrow aBC$

Let G be the content free grammar,

let A be the productions:

$A \rightarrow A\alpha_1 | A\alpha_2 | A\alpha_3 \ldots\ldots B_1 | B_2 | B_3 \ldots\ldots$

So we have some production such that left most symbol of RHS in A and few productions are not in A (i.e. $B_1, B_2, \ldots$). Thus, by introducing new variable B in the form $A \rightarrow B$: $A \rightarrow \beta_i$ AND $B \rightarrow \alpha_i$

$A \rightarrow B_i B$ $\quad B \rightarrow \alpha_i B$

EG:

$A \rightarrow \underset{\alpha_1}{Ae} | \underset{\alpha_2}{Ah} | \underset{\alpha_3}{Ak} | \underset{\beta_1}{bC} | \underset{\beta_2}{dE}$

$\therefore$ $A \rightarrow bC | dE | bCB | dEB$

$B \rightarrow e | h | k | eB | hB | kB$

1 Convert the following CFG to CNF,

$S \to AA|0$

$A \to SS|1$

We need to convert expression in form $A \to a\alpha$

$A \to SS|1$

$S \to \underbrace{AA}_{\alpha_1}S|\underbrace{0}_{B_1}S|\underbrace{1}_{B_2}$

$B \to AS|ASB$

$\therefore A \to 0S|1|0SB|1B$ ①

$\therefore B \to 0SS|1S|0SBS|1BS|0SSB|1BSB|1SB|0SBSB$ ②

$\therefore S \to 0SA|1A|0SBA|1BA|0$ ③

$\therefore$ Total GNF productions = 17

2 Convert following CFG to CNF,

$A_1 \to A_2A_3$

$A_2 \to A_3A_1|a$

$A_3 \to A_1A_2|b$

Substituting 'A_3' in 'A_1'

$A_3 \to A_2A_3A_2|b$

Now, substituting 'A_2' in 'A_3'

$A_3 \to \underbrace{A_3A_1A_3A_2}_{\alpha_1}|\underbrace{aA_3A_2}_{B_1}|\underbrace{b}_{B_2}$

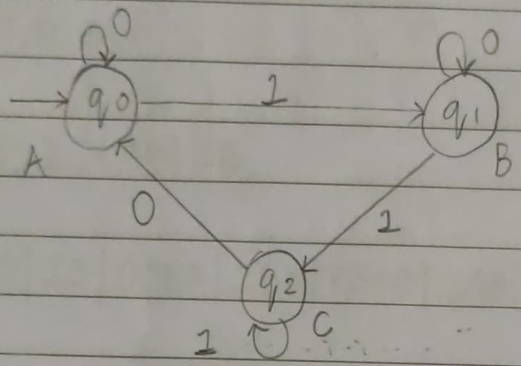
$A_3 \rightarrow aA_3A_2 \mid b \mid aA_3A_2B \mid bB$

* MOORE MACHINE:

It is a finite automata with no final state and it produces output sequence for the given input sequence. In moore machine, a symbol is associated with each state, such symbol is called output symbol.

$M = (Q, \Sigma, \delta, \Delta, \lambda, q_0)$

Q - finite states
Σ - i/p alphabet
δ - transition
Δ - o/p symbol
λ - mapping function
q_0 - initial state



Moore Machine

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$
$\lambda = Q \to \Delta$
$q_0 = \{q_0\}$
Δ = output symbol

Q \ Σ	0	1
q_0*	q_0	q_1
q_1	q_1	q_2
q_2	q_0	q_2

$\delta : Q \times \Sigma \to Q$

EG: $\{q_0, 0103$
$\{q_0, 103$
$\{q_1, 03$
$\{q_1 3$

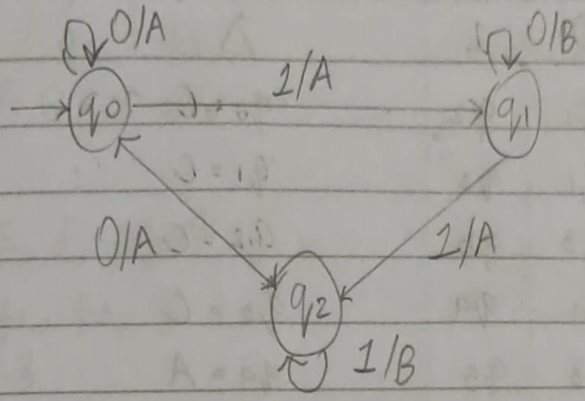
A
AA
AAB
AABB

} λ associated with state written as output

If i/p - "n" length, then o/p - "n+1" length
∴ Output: AABB

* MEALY MACHINE:

It is a finite automata with no final state, which produces output sequence for given input sequence. In mealy machine, a symbol is associated with each transition, such symbol is called output symbol.



Mealy Machine

Q \ Σ	0	1
q_0*	A	A
q_1	B	A
q_2	A	B

$\lambda : Q \times \Sigma \rightarrow \Delta$

EG: $\{q_0, 0103$
$\{q_0, 103$ A
$\{q_1, 03$ AA
$\{q_1 3$ ARB

If i/p = "n" length, then o/p = "n" length

∴ Output : AAB

Q1 Design a moore machine from

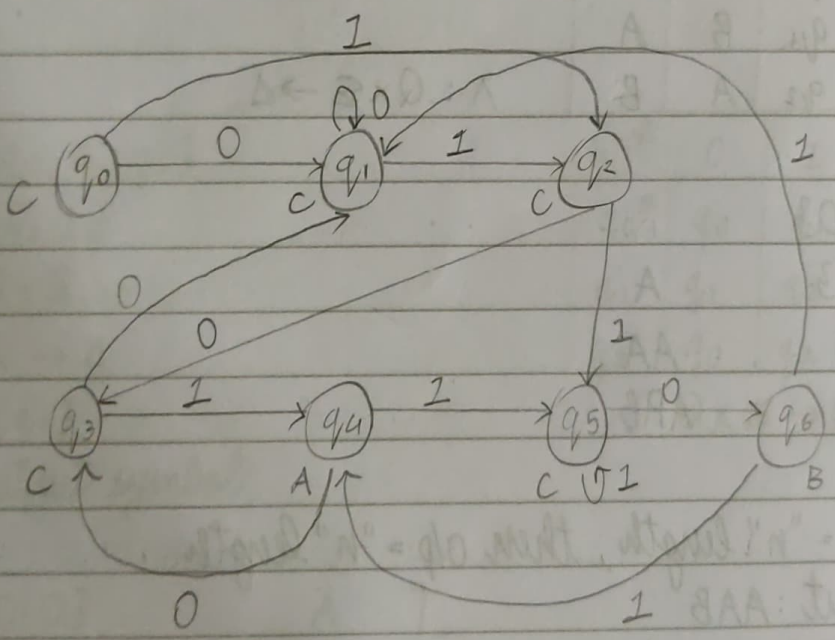
A → ends with '101'

B → ends with '110'

C → otherwise

Q \ Σ	0	1	Λ
→ q_0	q_1	q_2	$q_0 = C$
0 q_1	q_1	q_2	$q_1 = C$
1 q_2	q_3	q_5	$q_2 = C$
10 q_3	q_1	q_4	$q_3 = C$
101 q_4	q_3	q_5	$q_4 = A$
11 q_5	q_6	q_5	$q_5 = C$
110 q_6	q_1	q_4	$q_6 = B$

$\delta: Q \times \Sigma \to Q$



Q2 Design a mealy machine to output a remainder when a binary number is divisible by 4.

Q \ Σ	0	1
0	q_0	q_0	q_1
1	q_1	q_2	q_3
2	q_2	q_0	q_1
3	q_3	q_2	q_3

$\delta: Q \times \Sigma \to Q$

Q \ Σ	0	1
q_0	0	1
q_1	2	3
q_2	0	1
q_3	2	3

$\lambda: Q \times \Sigma \to \Delta$

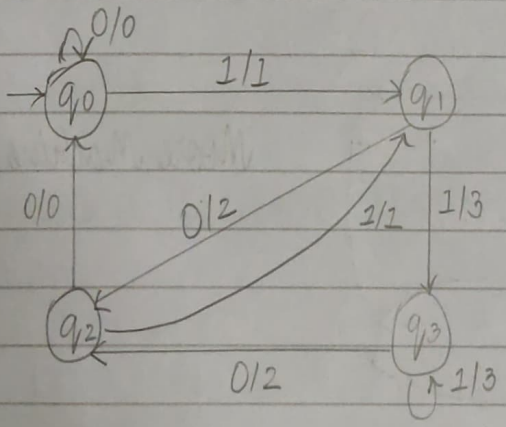
$M = (Q, \Sigma, \delta, \lambda, \Delta, q_0)$

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

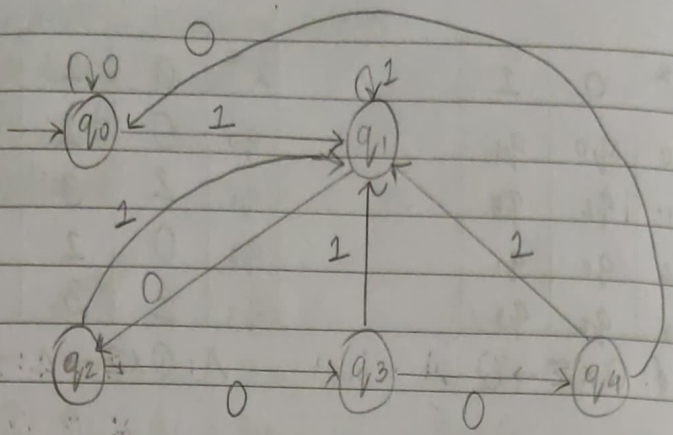
$\Delta = \{0, 1, 2, 3\}$

$q_0 = \{q_0\}$



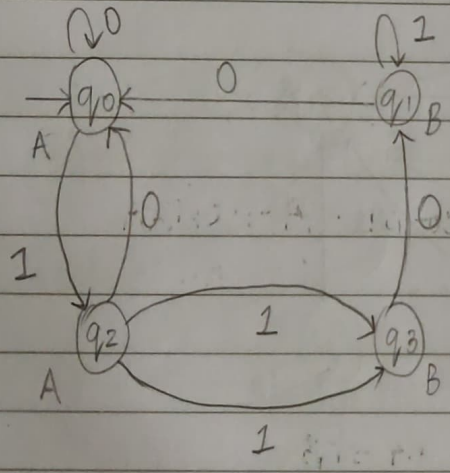
Q3 Design a moore machine to change each occurrence of "1000" to "1001".

		Q \ Σ	0	1
$\lambda(q_0) = 0$	0	q_0	q_0	q_1
$\lambda(q_1) = 1$	1	q_1	q_2	q_1
$\lambda(q_2) = 0$	10	q_2	q_3	q_1
$\lambda(q_3) = 0$	100	q_3	q_4	q_1
$\lambda(q_4) = 1$	1000	q_4	q_0	q_1

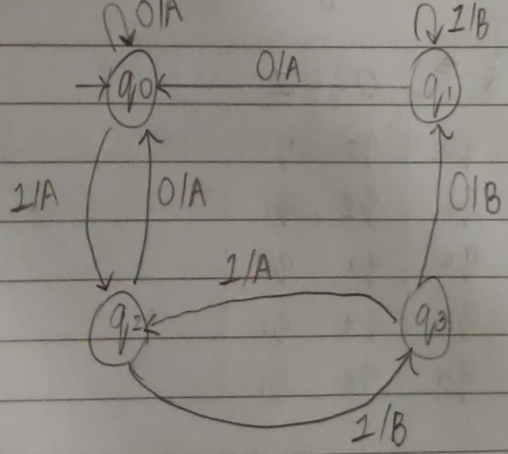


* MOORE TO MEALY MACHINE:

Q Assign output symbol associated with the state to all it's incoming transitions.



Moore Machine



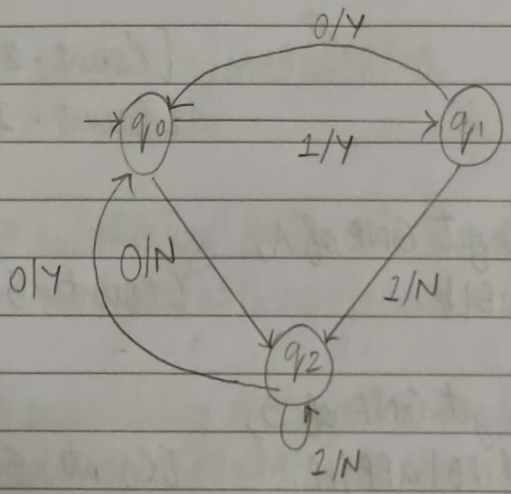
Mealy Machine

* MEALY TO MOORE MACHINE:

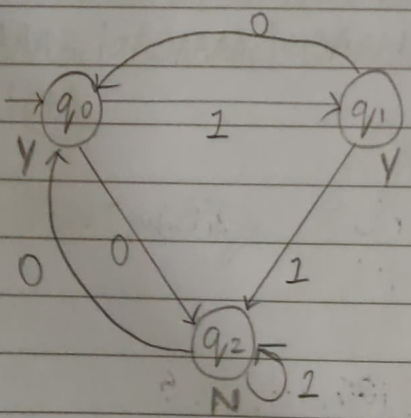
⇒ RULES:

1. If output symbol along with the incoming transition to a state are same then assign that output symbol to the state.
2. If output symbol along with the incoming transition to a state are not same then split that state as many times as output symbol with each producing a different output.

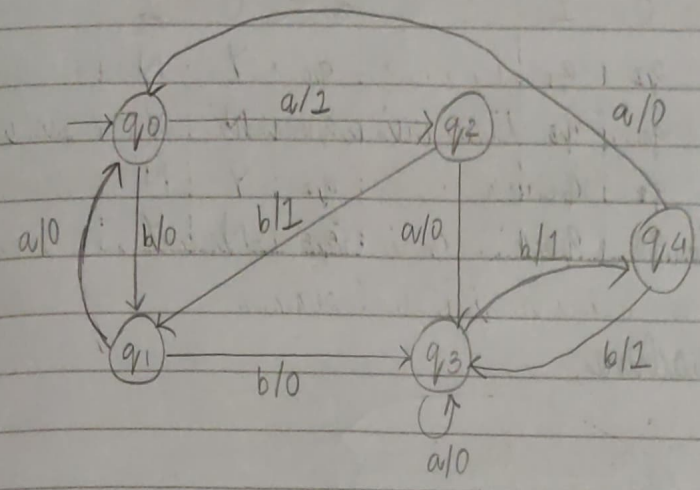
Q1 Given is a mealy machine.



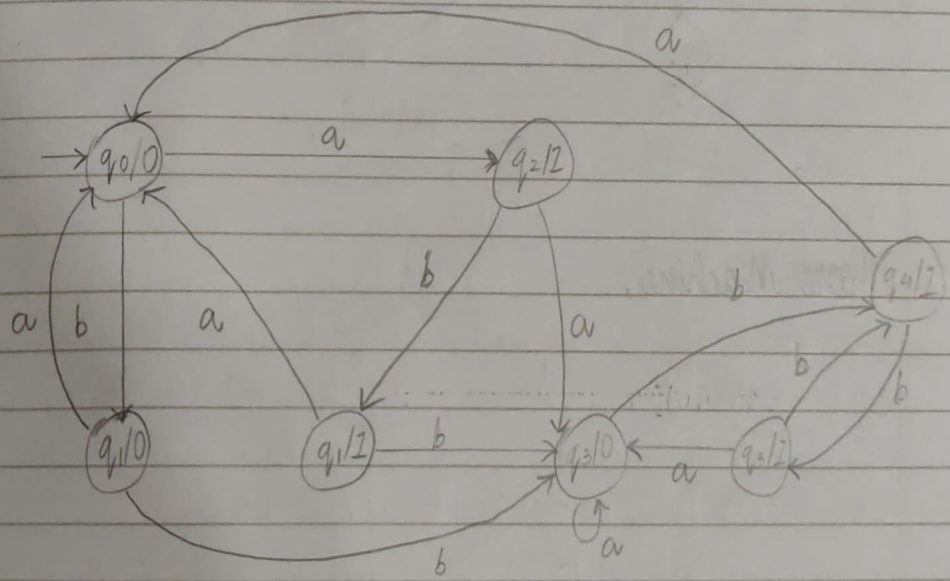
Now, the Moore Machine is:



Q2 Given is a Mealy Machine,



Now, the Moore Machine is:



Q3 Construct a Mealy Machine for given,

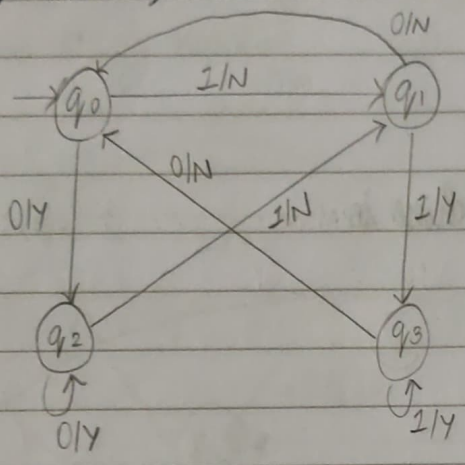
RE = (0+1)* (00+11)

L(G) = {00, 11, 000, 011, 100, 111, ... }

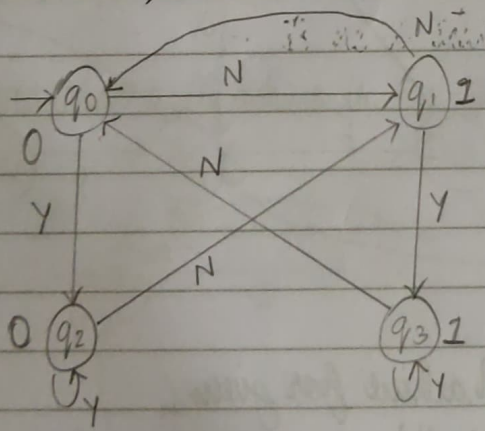
ε \ Q	0	1
0	q_0	q_2	q_1
1	q_1	q_0	q_3
00	q_2^*	q_2	q_1
11	q_3^*	q_0	q_3

Q	0	1
q_0	Y	N
q_1	N	Y
q_2	Y	N
q_3	N	Y

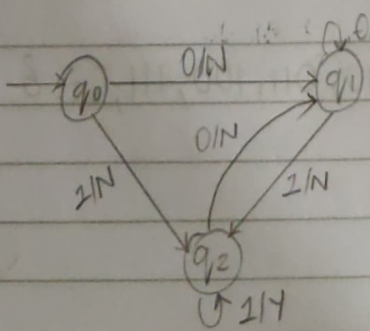
Mealy Machine,



Moore Machine,



ALTERNATE SOLN !!



* PUSH DOWN AUTOMATA:

→ It is a NFA with ε transition permitted with one additional capacity of the stack.

→ As the presence of stack, unlike finite automata, push down automata can remember infinite amount of information.

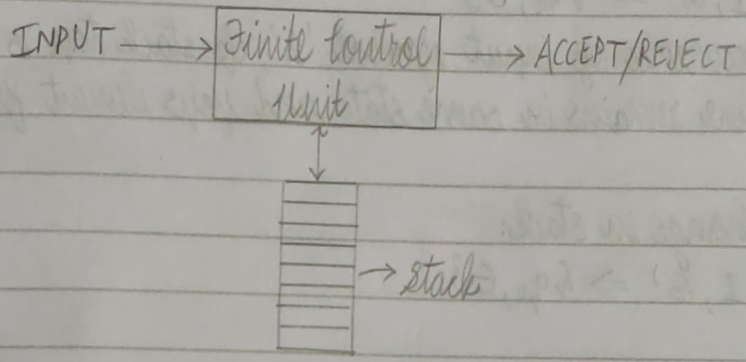
→ PDA has 3 basic components -

i) Input

ii) Finite control unit

iii) Stack with infinite size

→ PDA can be represented as -



Mathematical representation of PDA -

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

Q : finite set of state

Σ : input alphabet

Γ : finite set of start symbol

δ : transition function

q_0 : initial state

Z_0 : initial stack top symbol

F : final state

→ Some Possible Transitions in PDA:
δ(state, i/p, stack top symbol) ⇒ {state, stack top symbol}

1 Pushing element in stack while reading input alphabet:
$\delta(q_0, 0, a) \Rightarrow \{q_0, aa\}$
This indicates in a state 'q_0' if stack symbol is 'a' and '0' is the input string then machine will remain in same state and will push new element 'a' in stack.

2 Popping element from the stack:
$\delta(q_0, 1, a) \Rightarrow \{q_0, \in\}$
On occurence of input symbol '1', if stack top is 'a' then the machine remains in same state and pops element from stack.

3 No change in stack:
$\delta(q_0, 1, \in) \Rightarrow \{q_0, \in\}$

4 Change in state:
A machine will move to next state without performing any operation on stack.

Q Design PDA for language $L = \{a^n b^n \mid n \geq 1\}$.

⇒ Step ①
Let 'M' be required PDA.
Assume that 'M' is in 'q_0' state initially.
1 In 'q_0' state as long as input is 'a', push "1" in the stack.
2 In 'q_0' state if input is 'b' and top element is '1', pop "1" from stack and state will change to 'q_1'.
3 In 'q_1' state as long as input is 'b', & top element is '1', pop "1" from the stack.

4 When input is ours, if stack is empty, then accept the string or reject it.

⇒ Step ②

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

$Q = \{q_0, q_1\}$

$\Sigma = \{a, b\}$

$\Gamma = \{1\}$

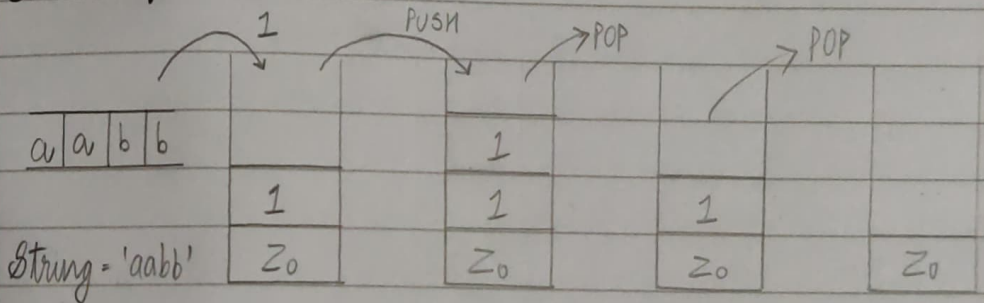
δ = step ③

$q_0 = \{q_0\}$

$F = \{q_1\}$

⇒ Step ③

Stack Operation



⇒ Step ④

i $\delta(q_0, a, z_0) = \{q_0, 1z_0\}$

ii $\delta(q_0, a, 1) = \{q_0, 11z_0\}$

iii $\delta(q_0, b, 1) = \{q_0, 1z_0\}$

iv $\delta(q_0, b, 1) = \{q_1, z_0\}$

v $\delta(q_1, \epsilon, z_0) = \{q_1, z_0\}$

⇒ Step ⑤

Simulation

$(q_0, aabb, z_0) \vdash (q_0, abb, 1z_0)$

$\vdash (q_0, bb, 11z_0)$

$\vdash (q_0, b, 1z_0)$

$\vdash (q_0, \epsilon, z_0)$