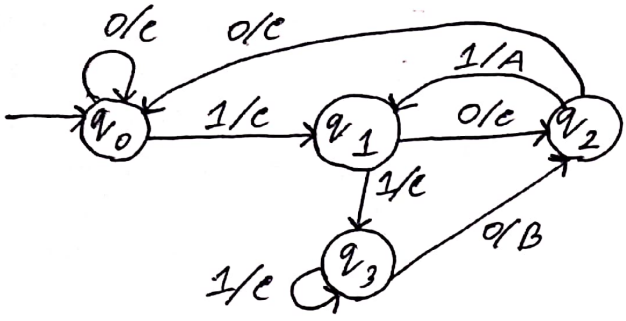
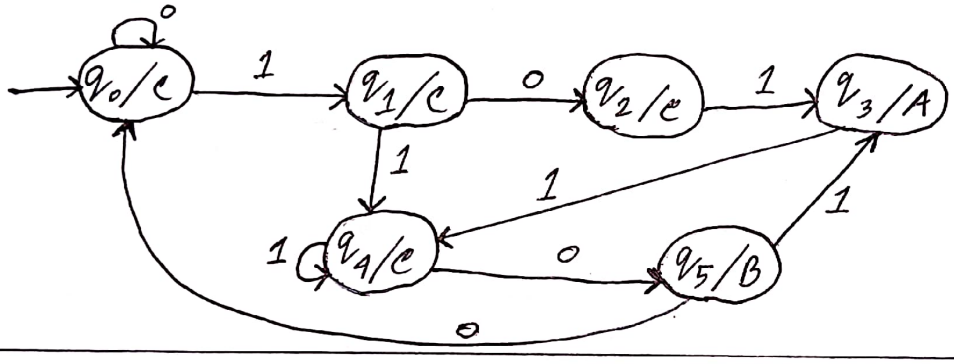
4] Design a Mealy machine for a binary input sequence such that if it has a substring 101, the machine output A, if the input has substring 110, it outputs B otherwise it outputs C.

Ans:



5] Design a Moore machine for a binary input sequence such that if it has a substring 101, the machine output A, if the input has substring 110, it outputs B otherwise it outputs C.

Ans:

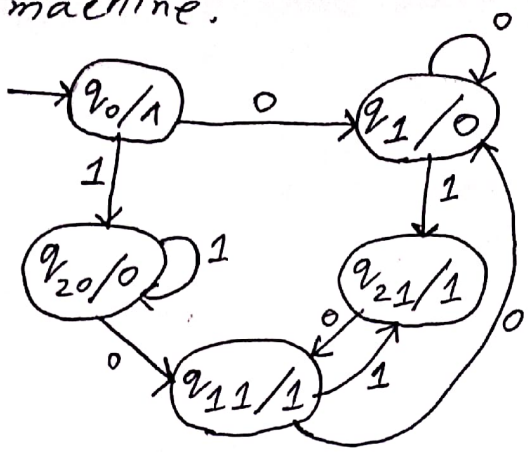


TOC

8] Convert Mealy to Moore machine.

	Input = 0		Input = 1	
Present State	Next State	Output	Next State	Output
q_0	q_1	0	q_2	0
q_1	q_1	0	q_2	1
q_2	q_1	1	q_2	0

Ans:



9] Given the context-free grammar G: $S \to AB$; $A \to aA|\varepsilon$; $B \to bB|\varepsilon$. Derive the string "aabb" ~~the string~~ using the grammer G.

Ans: $S \to AB \to aAB \to aaAB \to aaB \to aabB \to aabbB \to aabb$

10] Using the content-free grammer G: $S \to AaAb|\varepsilon$; $A \to Aa|\varepsilon$; $B \to Bb|\varepsilon$. Derive the string "aaabbbaaabbb" using the grammer G.

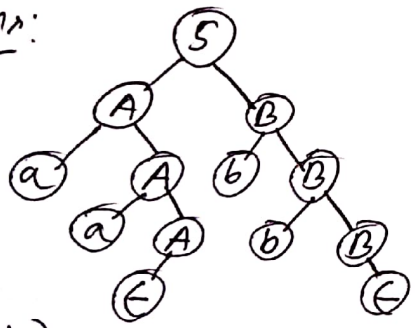
Ans: $S \to AaAb \to AaaAb \to AaaaAb$

∴ This grammer never generate the string.

11] For the CFG G: $S \to AB$; $A \to aA|\varepsilon$; $B \to bB|\varepsilon$.

a) Construct a derivation tree for string "aabb".

Ans:



b) Find all possible derivations for the same string using the given CFG.

Ans:

$S \to AB$
$\to aAB$ [$A \to aA$]
$\to aaAB$ [$A \to aA$]
$\to aaB$ [$A \to \epsilon$]
$\to aabB$ [$B \to bB$]
$\to aabbB$ [$B \to bB$]
$\to aabb$ [$B \to \epsilon$]

$S \to AB$
$\to AbB$
$\to Abbb$
$\to aaAbb$
$\to aabb$

TOC

13] Explain the concept of ambiguity in the content of content-free grammer. Give an example of a CFG with a ambiguous production.

Ans: A content free grammer is called ambiguous if there exists more than one left most derivation or more than one right most derivation for a sting which is generated by grammar. There will also be more than one derivation tree for a string in ambiguous grammer.

Example: $G = (\{S\}, \{a+b, +, *\}, P, S)$ where P consists of $S \to S+S \mid S*S \mid a \mid b$

$\therefore S \to S+S$
$\to a+S \quad [S \to a]$
$\to a+S*S \quad [S \to S*S]$
$\to a+a*S \quad [S \to a]$
$\to a+a*b \quad [S \to b]$

$\therefore S \to S*S$
$\to a*S \quad [S \to a]$
$\to a*S+S \quad [S \to S+S]$
$\to a*b+S \quad [S \to b]$
$\to a*b+a \quad [S \to a]$
$\to a+a*b$

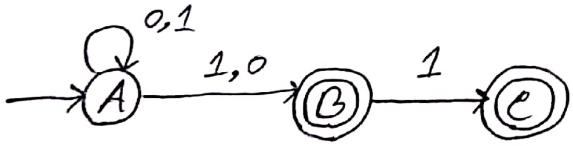
$\therefore$ So it is ambiguous grammer.

14] Convert the regular expression "ab*c" into a deterministic finite automation (DFA). Show the state transitions for the DFA.

Ans:

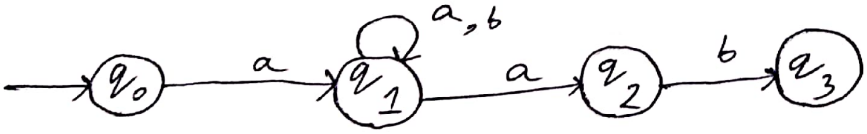
16] Starting with a regular expression for the language of all strings over $\{0,1\}$ that end with "01" or "10", create an NFA that recognizes the same language.

Ans:

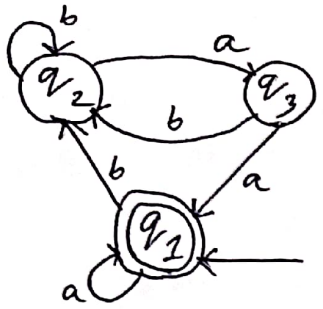


17] Draw NFA for the Regular Expression $a(a+b)^*ab$.

Ans:



18] Write the Regular expression for the following Finite Automata:



Ans:

$q_1 = \epsilon + q_1 a + q_3 a$ — (1)

$q_2 = q_1 b + q_2 b + q_3 b$ — (2)

$q_3 = q_2 a$ — (3)

$q_2 = q_1 b + q_2 b + q_3 b$

$= q_1 b + q_2 b + q_2 ab$

$= q_1 b + q_2 (b + ab)$

$[R = Q + RP = QP^*]$

$= q_1 b (b + ab)^*$ — (4)

$q_1 = \epsilon + q_1 a + q_3 a$

$= \epsilon + q_1 a + q_2 aa \quad [q_3 = q_2 a]$

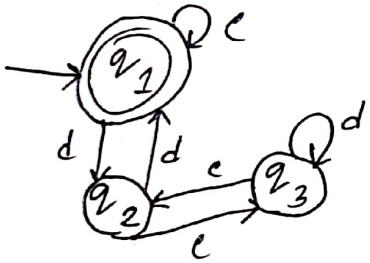
$= \epsilon + q_1 a + q_1 b (b+ab)^* aa \quad [\text{From } (4)]$

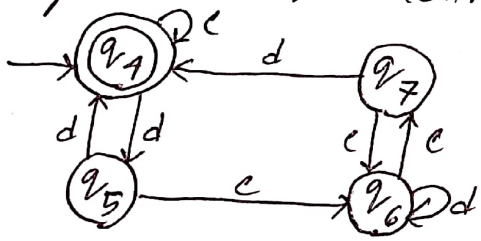
$= \epsilon + q_1 (a + b (b+ab)^* aa)$

$= \epsilon (a + b (b+ab)^* aa)^* \quad [R = Q + RP = QP^*]$

$= (a + b (b+ab)^* aa)^*$

17] Consider the two Deterministic Finite Automata (DFA) and check whether they are equivalent or not.





Ans:

States	c	d
$\{q_1, q_4\}$	$\{q_1, q_4\}$	$\{q_2, q_5\}$
$\{q_2, q_5\}$	$\{q_3, q_6\}$	$\{q_1, q_4\}$
$\{q_3, q_6\}$	$\{q_2, q_7\}$	$\{q_3, q_6\}$
$\{q_2, q_7\}$	$\{q_3, q_6\}$	$\{q_1, q_4\}$

∴ They are equivalent.

TOC

20] Define a pushdown automaton (PDA) and explain it, basic components, including the input alphabet, stack alphabet and transitions.

Ans: Pushdown Automata is a finite automata with extra memory called stack which helps pushdown automata to recognize content free language.

A PDA can remember an infinite amount of information.

∴ Finite state machine + a stack = PDA

A pushdown automaton has three components:

(i) an input tape (ii) a control unit (iii) a stack with infinite size.

It has seven tuples: $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$Q \rightarrow$ A finite set of states.

$\Sigma \rightarrow$ A finite input alphabet.

$\Gamma \rightarrow$ A finite stack alphabet.

$q_0 \rightarrow$ The starting state q_0 is in Q

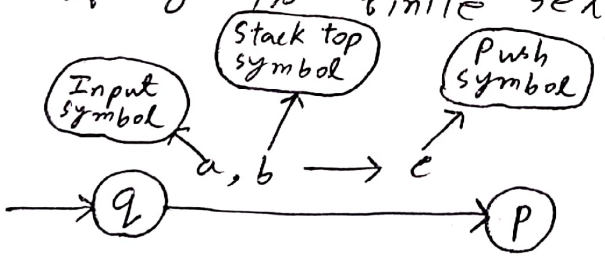
$Z_0 \rightarrow$ A starting stack symbol, is in Γ.

$F \rightarrow$ A set of final/accepting states, which is a subset of Q.

$\delta \rightarrow$ A transition function $\delta(q, a, x)$, where δ: $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ - finite subsets of $Q \times \Gamma^*$

The output of δ is finite set of pairs (P, Y).

Example:



21] What is the role of stack in a pushdown automaton and how does it differ from the tape in a Turing machine?

Ans: Basically a pushdown automaton is Finite state machine + a stack. A pushdown automaton has a stack with infinite size. The stack head scans the top symbol of the stack. A stack does two operations,

Push → A new symbol is added at the top.

Pop → The top symbol is read and removed.

For unlimited memory PDA has a stack. This is the role of the stack in a PDA.

A much more powerful abstract model of a computing device is a Turing Machine.

22] Describe the configuration of a PDA and how it changes during the computation of a string.

Ans: A pushdown Automaton (PDA) has a configuration defined by three components: three current state, the input tape, and the stack. During the computation of a string, the configuration changes as follows,

1. Initial Configuration: PDA starts in an initial state with an input string on the tape and an empty stack.
2. Stack Operations: Transitions involve popping from or pushing onto the stack.
3. Transition rules: Rules define state changes based on current state, input symbol and stack content.
4. Acceptance / Rejection: PDA accepts if it reaches an accepting state with an empty tape; rejects otherwise.
5. Non-Determinism: Multiple transitions may occur concurrently, exploring different possibilities.
6. Final Configuration: ~~Configuration~~ Computation ends when PDA reaches an accepting / rejecting state with an empty tape.

23] What is the significance of the PDA's initial stack symbol and how it used in PDA transitions?

Ans: The initial stack symbol often denoted as 'Z', marks the bottom of the stack, crucial for maintaining structure and aiding in proper stack operations during PDA computations. It is used in in ~~pda~~ PDA' transitions as,

Reference point: Used during transitions to signify the bottom of the stack.

Stack Operations: When popping, detecting the initial stack symbol indicates reaching the bottom.

Structural Role: Maintains stack structure, aiding in content-sensitive language recognition.

Enforces Patterns: Allows PDA, to recognize nested structures and patterns in input strings.

24] What is the role of the tape head in a Turing machine and how does it interact with the tape?

Ans: The tape head in a Turing machine plays a pivotal role in reading, writing and moving along the machine's tape. It interacts with the tape through the following mechanism.

1. Reading: The tape head reads the symbol under it on the tape.
2. Writing: After reading, the tape head can write a new symbol onto the tape.
3. Moving: The tape head moves one step left or right based on transition rules.
4. Infinite Tape: The tape is infinite, enabling unbounded computations.
5. Transition Rules: Interaction is governed by rules defining machine behavior with current state and tape symbol.

25] What is the significance of the acceptance state in a Turing machine and how does it determine whether a string is accepted or rejected?

Ans: The acceptance state in a Turing machine determines whether the machine accepts an input string. If the machine reaches an acceptance state after processing the string, it is accepted; otherwise, it is rejected, shaping the outcome of the computation.

1. Acceptance Criteria: The acceptance state in a Turing machine defines conditions for successful computation.
2. Rejection Criteria: If the machine does not reach an acceptance state, the input string is rejected.
3. Outcome Determination: Acceptance or rejection is determined by the final state the machine enters after processing the input.
4. Language Recognition: The acceptance state defines the language recognized by the Turing machine.
5. Marker of Success: Reaching the acceptance state signifies successful processing and acceptance of the input string.