FINITE AUTOMATA (FA)

Basic terminologies and definitions.

- **Alphabet** - An alphabet is any finite set of symbols $\Sigma = \{a, b, c, d\}$ is an alphabet set where 'a', 'b', 'c', 'd' are symbols.

- **String** - A string is finite sequence of symbols taken from Σ
  Ex - 'cabcad' is a valid string on the alphabet set $\Sigma = \{a, b, c, d\}$.

- **Length of a String** - It is the number of symbols present in a string
  Ex:- If S = 'cabcad' $|S| = 6$
  If $|S| = 0$ it is empty string (Denoted by λ or ε)

- **Language**:- A language is a subset of Σ^* for some alphabet Σ. It can be finite or infinite.
  Ex. - If the language takes all possible strings of length 2 over $\Sigma = \{a, b\}$, then $L = \{ab, aa, ba, bb\}$

- **Concatenation**:- $w = w_1 . w_2$
  Ex:- If $A = \{0, 1\}$ & $B = \{1, 2\}$ then $A.B = \{01, 02, 11, 12\}$

- **Prefix** - A prefix of a string is the string formed by taking any number of symbols that are added to the beginning of a string.
  Ex:- Let $w = 01011$ then prefixes are $\{\lambda, 0, 01, 010, 0101, 01011\}$.

leading & trailing

- Note:- In general, if the length of the string is n, there are n+1 number prefixes that are possible

- Proper prefix:- Any prefix of the string other than the string itself.
  Ex:- $\omega = 01011$ then $\{\lambda, 0, 01, 010, 0101\}$ are proper prefixes.

- ~~Suffix:-~~ Note:- In general, if the length of the string is n, ~~are~~ there are n number proper prefixes are possible.

- Suffix:- A suffix of a string by taking any number of symbols from the end of the string
  Example $\omega = 01011$. Now suffixes are $\{\lambda, 1, 11, 011, 1011, 01011\}$
  n, n+1.

- Proper suffix:- Any suffix of the string other than the string itself.
  Ex $\omega = 01011$ $\{\lambda, 1, 11, 011, 1011\}$ proper suffixes
  n, n

- Power set:- If A is the set then the power set of A, denoted as $2^A = \{P \mid P$ is a subset of $A\}$. i.e. set of all possible subsets of A.
  Ex:- $A = \{0, 1\}$
  $2^A = [\phi, \{0\}, \{1\}, \{0, 1\}]$ where $\phi = \{\}$ null set

* Relations and its properties

A set of ordered pairs is klas relation. The first component of the pair is from the set called domain and second component is from the set called range.

- If R is a relation & (a, b) is a pair in R, then we write aRb.

Properties

1] Reflexive, if aRa exists ∀ (for all) $a \subseteq S$.
2] Transitive, if aRb ~~&~~ & bRc imply aRc.
3] Symmetric, if aRb imply bRa.

Deterministic Finite Automata (DFA)

- If there is a single (unique) transition from one state to other with a single I/p symbol, then that automata is I/as Deterministic finite automata.

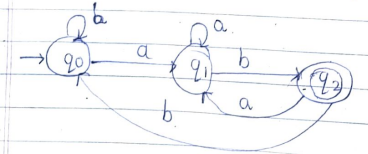
- DFA defined with 5 tuples $M = <Q, \Sigma, \delta, q_0, F>$
  - Q: Finite set of all states.
  - Σ: Finite set of input symbol
  - δ: Transition function
  - q_0: Initial state
  - F: Set of final state

Q. Design a DFA for input ~~sym~~ alphabet where every string ends with "ab".

Step 1] $L = \{ab, aab, aaab, abab, aabaab, ---\}$

Step 2] $M = <Q, \Sigma, \delta, q_0, F>$
where Q - Set of all states
Σ - Finite set of input symbol.
δ - Transition function
q_0 - Initial state
F - Finite set of final state.

Step 3]



Step 4] $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$

$\delta = Q \times \Sigma \to Q$.
$q_0 = \{q_0\}$
$F = \{q_2\}$

Step 5

Q/Σ	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0

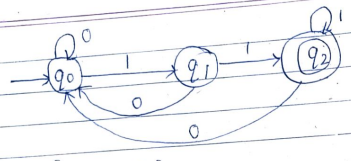
Step 6 Taking aaaab
$\delta(q_0, aaaab)$
$\delta(\{q_0, q\}\ aaab)$
$\delta(\{q_1, q\}\ aab)$
$\delta(\{q_1, q\}\ ab)$
$\delta(\{q_1, q\}\ b)$
$\delta(\{q_2, b\})$
$\delta(\{q_2\})$

Q. Design a DFA for input signal $\Sigma = \{0, 1\}$ where every string end with 11.

Step 1] $L = \{11, 011, 0011, 0111, 00011, 00110011, ---\}$

Step 2] $M = <Q, \Sigma, \delta, q_0, F>$.
where Q =
Σ -
δ -
q_0 -
F -

Step 3.



Step 4 $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_2\}$

Step 5

Q/Σ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_2

Step 6 Taking 00011
$\delta(q_0, 00011)$
$\delta(\delta(q_0,0)\ 0011)$
$\delta(\delta(q_0,0)\ 011)$
$\delta(\delta(q_0,0)\ 11)$
$\delta(\delta(q_0,1)\ 1)$
$\delta(\delta(q_2\ 1))$
$\delta(q_2)$

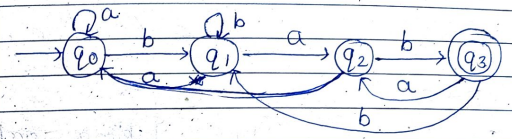
Q. Design a DFA for input signal $\Sigma = \{a, b\}$ where every string ends with bab.

Step 1 $L = \{bab, abab, aabab, bbbab, abbab, \ldots\}$

Step 2 $M = \langle Q, \Sigma, \delta, q_0, F \rangle$
where Q = Set of all states
Σ = Finite set of input symbol

δ - Transition function
q_0 - Initial state
F - Finite set of final state

Step 3



Step 4 $Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_3\}$

Step 5

Q/Σ	a	b
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_3
q_3	q_2	q_1

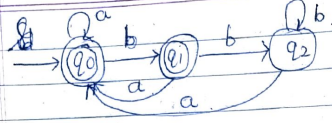
Step 6 Taking babab
$\delta(q_0, babab)$
$\delta(\delta(q_0,b)\ abab)$
$\delta(\delta(q_1,a)\ bab)$
$\delta(\delta(q_2,b)\ ab)$
$\delta(\delta(q_3,a)\ b)$
$\delta(\delta(q_2,b))$
$\delta(q_3)$

Q. Design a DFA for input signal $\Sigma = \{a, b\}$ where every string does not end with bb

Step 1 $L = \{ab, aa, bba, abab, aabaab, aaab \ldots\}$

Step 2 Here $M = \langle Q, \Sigma, \delta, q_0, F \rangle$
where Q =
Σ =
δ =
q_0 =
F =

Step 3:



Step 4 $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_0, q_1\}$

Step 5:

Q/Σ	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_2

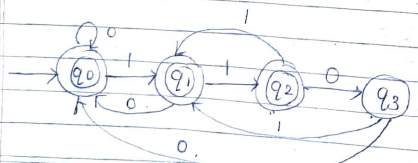
Step 6 Taking aaab
$\delta((q_0\, a)\, aab)$
$\delta(\delta(q_0\, a)\, ab)$
$\delta(\delta(\delta(q_0\, a)\, b)$
$\delta(\delta(q_0\, b))$
$\delta(q_1)$

Q. Design a DFA for input signal $\Sigma = \{0, 1\}$ where every string does not end with 110.

Step 1 $L = \{\varepsilon, 0, 1, 00, 11, 01, 011, 1011, 1010 \ldots\}$.

Step 2: $M = \langle Q, \Sigma, \delta, q_0, F \rangle$
where $Q =$
$\Sigma =$
$\delta =$
$q_0 =$
$F =$

Step 3:



Step 4: $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$.
$q_0 = \{q_0\}$
$F = \{q_0, q_1, q_2\}$

Step 5

Q/Σ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_0	q_1

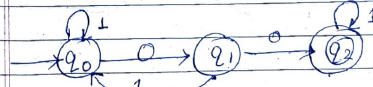
Step 6: Taking 10110
$\delta(q_0, 10110)$
$\delta(\delta(q_0, 1)\, 0110)$
$\delta(\delta(q_1, 0)\, 110)$
$\delta(\delta(q_0, 1)\, 10)$
$\delta(\delta(q_1, 1)\, 0)$
$\delta(\delta(q_2, 0))$
$\delta(q_3, 1)$
$\delta(q_1)$

Q. Design a DFA for input signal $\Sigma = \{0, 1\}$ where every string contain substring 00

Step 1 $L = \{\varepsilon, 100, 001, 1001, 000, 0000, 10011, 11001101001\}$

Step 2 $M = \langle Q, \Sigma, \delta, q_0, F \rangle$
where $Q =$
$\Sigma =$
$\delta =$
$q_0 =$
$F =$

Step 3



Step 4: $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \Rightarrow Q$
$q_0 = \{q_0\}$.

Step 5:

Q/Σ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

Step 6: Taking 110011

$\delta(q_0, 110011)$
$\delta(\delta(q_0,1)\ 10011)$
$\delta(\delta(q_0,1)\ 0011)$
$\delta(\delta(q_0,0)\ 011)$
$\delta(\delta(q_1,0)\ 11)$
$\delta(\delta(q_2,1)\ 1)$
$\delta(\delta(q_2,1))$
$\delta(q_2)$

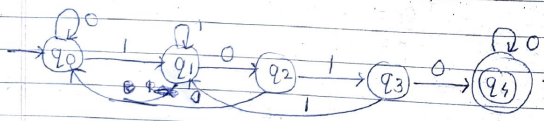
Q. Design a DFA for input signal $\Sigma = \{0,1\}$ where every string ~~does not~~ contain substring 1010

Step 1: $L = \{01010, 1010, 110101, 11010, 111010 \ldots\}$

Step 2: $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

where $Q =$
$\Sigma =$
$\delta =$
$q_0 =$
$F =$

Step 3:



Step 4: $Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{0,1\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_4\}$

Step 5:

Q/Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_3
q_3	q_4	q_4

Step 6: Taking 10101

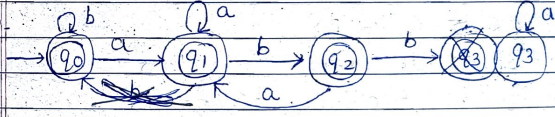
$\delta(q_0, 10101)$
$\delta(\delta(q_0,1)\ 0101)$
$\delta(\delta(q_1,0)\ 101)$
$\delta(\delta(q_2,1)\ 01)$
$\delta(\delta(q_3,0)\ 1)$
$\delta(q_4,1)$
$\delta(q_4)$

Q. Design a DFA for input signal $\Sigma = \{a,b\}$ where every string does not contain substring abb

Step 1: $L = \{aaba, abab, aabab, ababa, \ldots\}$

Step 2: $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

Step 3:



Step 4: $Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a,b\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_0, q_1, q_2\}$

Step 5:

Q/Σ	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

Step 6: Taking aabab

$\delta(q_0, aabab)$
$\delta(\delta(q_0,a)\ abab)$
$\delta(\delta(q_1,a)\ bab)$
$\delta(\delta(q_1,b)\ ab)$
$\delta(\delta(q_2,a)\ b)$
$\delta(\delta(q_1,b))$
$\delta(q_2)$

* Dead state

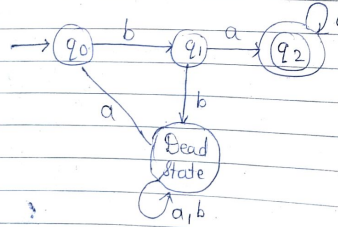
- We need to start a machine (DFA) to read a string. If machine reached successfully till its final string accepting state, then we say that string is accepted by our machine.
- Only one dead state & always non-final
- But if we reach on a state where machine can't move further to its final state, then this state is called dead state. It is also k/as dummy state

Q. Design a DFA $\Sigma = \{a, b\}$ where every string starts with ba

Step 1: $L = \{ba, baa, bab, baaa, babb, \ldots\}$

Step 2: $M = \langle Q, \Sigma, \delta, q_0, F \rangle$
- Q - Set of all states
- Σ - Finite set of input symbol
- δ - Transition function
- q_0 - Initial state
- F - Finite set of final state

Step 3:



Step 4: $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_2\}$

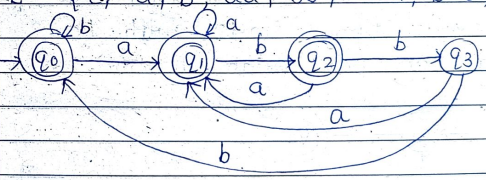
Step 5:

Q/Σ	a	b
q_0	DS	q_1
q_1	q_2	DS
q_2	q_2	q_2
Dead st	DS	DS

Step 6: Taking babb
$\delta(q_0, babb)$
$\delta(\delta(q_0, b), abb)$
$\delta(\delta(q_1, a), bb)$
$\delta(\delta(q_2, b), b)$
$\delta(\delta(q_2, b))$
$\delta(q_2)$

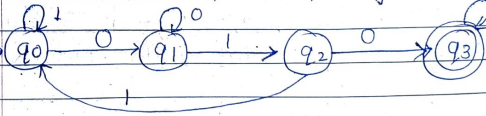
Q. Design a DFA $\Sigma = \{a, b\}$ where every string doesn't end with abb.

$L = \{\varepsilon, a, b, aa, bb, aaa, bbb, aaabbb, \ldots\}$



Q. Design a DFA $\Sigma = \{0, 1\}$ where every string contains substring 010.

$L = \{\varepsilon, 010, 1010, 0101, 0010, 0100, 1101011, 10101, 01011, \ldots\}$



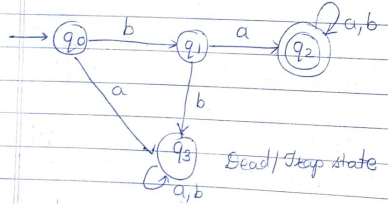
Q. Design a DFA for I/p signal $\Sigma = \{a, b\}$ where every string does not contain substring bbaa.

$L = \{\varepsilon, a, b, aa, bb, aaa, bbb, aaaba, abab, babab, \cdots\}$

Start with

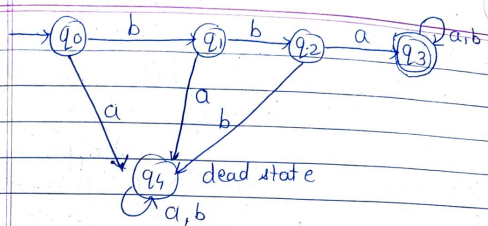
Q. Design a DFA with input signal $\Sigma = \{a, b\}$ where every string starts with ba.

$L = \{ba, baa, bab, baab, baaabb, \cdots\}$



Q. Design a DFA with input signal $\Sigma = \{a, b\}$ where every string starts with bba

$L = \{bba, bbab, bbaa, bbaab, bbaba, \cdots\}$



Q. Design a DFA for input signal where

i) $L = \{\omega \in (a,b)^* \mid |\omega| = 4\}$

Step 1: $L = \{aaaa, bbbb, abab, baba, aaab, aabb, abaa, baaa, bbaa, bbba, aaba\}$

Step 2: $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

where Q = Set of all states
Σ = Finite set of input symbol
δ = Transition function
q_0 = Initial state
F = Finite set of final state

Step 3:

Step 4: $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_4\}$

Step 5:

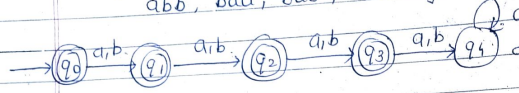
Q/Σ	a	b
q_0	q_1	q_1
q_1	q_2	q_2
q_2	q_3	q_3
q_3	q_4	q_4
q_4	q_5	q_5
q_5	q_5	q_5

Step 6: Taking aabb

$\delta(q_0, aabb)$
$\delta(\delta(q_0, a), abb)$
$\delta(\delta(q_1, a), bb)$
$\delta(\delta(q_2, b), b)$
$\delta(\delta(q_3, b))$
$\delta(q_4)$

ω - string
|ω| - length of string

iii) $L = \{\omega \in (a,b)^* \mid |\omega|\}$
$L = \{\varepsilon, a, b, ab, aa, bb, aaa, bbb, aab, bba, abb, baa, bab, aba \ldots\}$

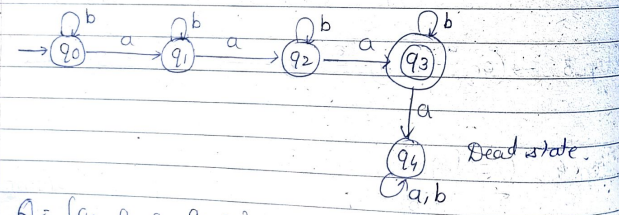


Q. Design a DFA over I/p signal $\Sigma = \{a,b\}$ where $L = \{\omega \in (a,b)^* \mid |\omega|_a = 3\}$ = 3a

Step 1 = $L = \{aaa, baaa, aaab, aaabb, bbaaa, abaaa, aaaba, \ldots\}$

Step 2 : → (q0) ℳ $M = \langle Q, \Sigma, \delta, q_0, F \rangle$
Q = Set of all states
Σ = Finite set of input symbol
δ = Transition function
q_0 = Initial state
F = Finite set of final state

Step 3:



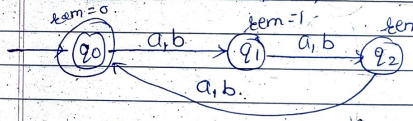
Step 4. $Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{a,b\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$q_f = \{q_3\}$

Step 5

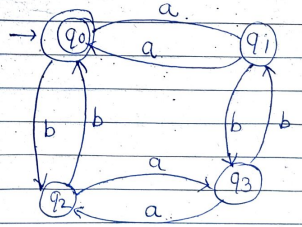
Θ/Σ	a	b
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_3
q_4	q_4	q_4

Step 6 Taking aaabb
$\delta(q_0, aaabb)$
$\delta(\delta(q_0,a), aabb)$
$\delta(\delta(q_1,a), abb)$
$\delta(\delta(q_2,a), bb)$
$\delta(\delta(q_3,b), b)$
$\delta(\delta(q_3,b))$
$\delta(q_3)$

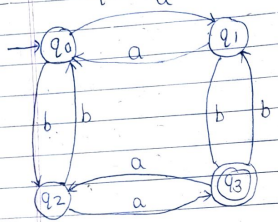
Q. Design a DFA signal where string contains -
$L = \{\omega \in (a,b)^* \mid |\omega| \bmod 3 = 0\}$
= $L = \{aaa, bbb, aba, bab, abb, baa, aaaaaa, bbbbbb, aaabbb, ababab, babbab \ldots\}$



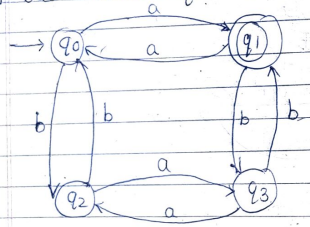
Q Design a DFA signal where string contains (even no of a & even no of b
= $L = \{\varepsilon, aa, bb, aabb, abab, baba, bbaa, ababab, bababa, aaaabbbb, \ldots\}$



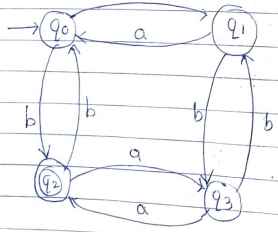
iii) Odd number of a & odd number of b.
L = {ε, aaa, bbb, aaabbb, aabbbb, ...}



iii) Odd number of a & even number of b.



iii) even number of a & odd number of b.



NFA → DFA.

Non-Deterministic Finite Automata (NFA).

If there are more than one transition from one state to another state with a single I/p symbol, then that automata is Clas Non-Deterministic Finite Automata.

- NFA stands for non-deterministic finite automata. It is easy to construct an NFA than DFA for a given regular language.
- The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.
- Every NFA is not DFA, but each NFA can be translated into DFA.
- NFA is defined in the same way as DFA but with the following two exceptions, it contains multiple next states & it contains ε transition.

There are total 5 tuples.
M = <Q, Σ, δ, q0, F>.

Q. Construct a DFA equivalent to NFA where
M = <{q0, q1}, {0, 1}, δ, {q0}, {q0}>.

Steps for Solving the problem:
Step I :- Define the tuples of NFA
M = <Q, Σ, δ, q0, F> and provide the value to the tuples.

Step II :- Draw the transition table of NFA.

Step III :- Define the tupples of DFA.

$M' = <Q', \Sigma, \delta', q_0', F'>$

where Q' - Finite set of all state in DFA.
Σ' = Finite set of input symbol
δ' = Transition function of DFA
q_0' = Initial state of DFA
F' = Set of all final state of DFA.

Step IV :- Construct the transition table of DFA by explaining the logic.

Step V :- Provides the values to the tuples of DFA.
$Q' \times \Sigma \rightarrow Q'$

Step VI :- Draw the transition diagram of DFA.

Equivalence Between NFA & DFA
Or
Conversion of NFA to DFA (NFA → DFA).

1. If in NFA there are n-states then in worst case DFA contain 2^n state.

2. Procedure of conversion of NFA to DFA.
Let $M = <Q, \Sigma, \delta, q_0, F>$ are the tuples of NFA & $M' = <Q', \Sigma', \delta', q_0', F'>$ are the tuples of DFA

3. The process of conversion of ~~NA~~ NFA into DFA is also 'SUBSET CONSTRUCTION'.

Step I :- Initial state
→ There is no change in initial state of NFA & DFA. It means that initial state of NFA and DFA is same.
$[q_0' = q_0]$

Step II :- Construction of δ' [Transition function of DFA]
- Starts the construction of δ' from initial state & continue the process for every new state which appears in the input columns & terminate the process whenever no new state appears in a input column.

Step III :- [Final state]
Every subset which contain final state of NFA is final state in DFA.

Construction of δ'.

i] Initial state of NFA & DFA is equal ie $[q_0 = q_0']$

ii] Last the construction of δ' from initial state & continue the process for every new state which appear in the input table & terminate the process whenever no new state appear in the input table.

iii] Every subset which contain final state of NFA is the final state in DFA.

$\delta'([q_0], 0) = \delta(\{q_0\}, 0)$
$[q_0]$

$\delta'([q_0], 1) = \delta(\{q_1\}, 1)$
$[q_1]$

$\delta'([q_1], 0) = \delta(\{q_1\}, 0)$
$[q_1]$

$\delta'([q_1], 1) = \delta(\{q_1\}, 1)$
$= \{q_0\} \cup \{q_1\}$
$= [q_0\ q_1]$

$\delta'([q_0 q_1], 0) = \delta(\{q_0\}, 0) \cup \delta(\{q_1\}, 0)$
$= [q_0] \cup [q_1]$
$= [q_0\ q_1]$

$\delta'([q_0, q_1], 1) = \delta(\{q_0\}, 1) \cup \delta(\{q_1\}, 1)$
$= [q_1] \cup [q_0\ q_1]$
$= [q_0\ q_1]$

Q $M = <\{q_0, q_1\}, \{0,1\}, \delta, \{q_0\}, \{q_0\}>$

Step I :- $M = <Q, \Sigma, \delta, q_0, F>$

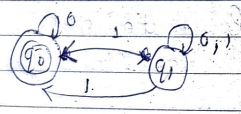
$Q = \{q_0, q_1\}$

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

$\delta = Q \times \Sigma = 2^Q$

$q_0 = \{q_0\}$

$F = \{q_0\}$



Step II :-

Q/Σ	0	1
q_0	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$

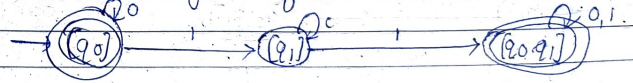
Step III :- Define the tuples of DFA i.e
$M = <Q', \Sigma', \delta, q_0', F'>$
where Q' = Finite set of all state in DFA
Σ' = Finite set of i/p symbol
δ = Transition function of DFA
q_0' = Initial state of DFA
F' = Set of all final state of DFA

Step IV: Initial state Transition table of DFA by explaining the logic.

Σ/Q	0	1
$[q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_0, q_1]$
$[q_0\ q_1]$	$[q_0\ q_1]$	$[q_0, q_1]$

Step V :- Tuples of DFA
$Q' \times \Sigma = Q'$
$M = <Q', \Sigma', \delta', q_0', F>$
$Q' = \{[q_0], [q_1], [q_0, q_1]\}$
$\Sigma' = \{0,1\}$
$\delta' = Q \times \Sigma = Q'$
$q_0' = \{[q_0]\}$
$F' = \{[q_0], [q_0, q_1]\}$

Step VI Transition diagram of DFA



Q $M = <\{q_0, q_1\}, \{0,1\}, \delta, \{q_0\}, \{q_1\}>$

Step I :- $M = <Q, \Sigma, \delta, q_0, F>$
$Q = \{q_0, q_1\}$
$\Sigma = \{0,1\}$
$\delta = Q \times \Sigma = 2^Q$
$q_0 = \{q_0\}$
$F = \{q_1\}$

~~Step II~~ :-

Q/Σ	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	$\emptyset$	$\{q_0, q_1\}$

Q

Step III :- Tupples of DFA

Step IV :- Transition table of DFA.

Q/Σ	0	1
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_0 q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$[q_1]$	$[q_2]$	$[q_0, q_1]$
$[q_2]$	$[q_2]$	$[q_2]$

Step V :- Tuples of DFA

$M' = \langle Q', \Sigma', \delta', q_0', F' \rangle$

$Q' = \{[q_0], [q_0, q_1], [q_1], [q_2]\}$

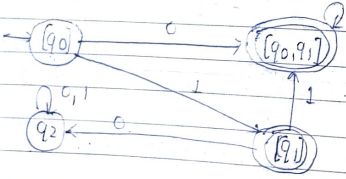
$\Sigma' = \{0, 1\}$

$\delta' = Q' \times \Sigma' = 2^{Q'}$.

$q_0' = \{[q_0]\}$

$F' = \{[q_1], [q_0, q_1]\}$

Step VI



Q. Construct a DFA equivalent to.

$M = \langle \{q_0, q_1, q_2\}, \{a, b\}, \delta, \{q_0\}, \{q_2\} \rangle$

Q/Σ	a	b
q_0	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_0, q_2\}$
q_2	-	$\{q_0, q_1\}$

Step I :- $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

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

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

$\delta = Q \times \Sigma = 2^Q$.

$q_0 = \{q_0\}$

$F = \{q_2\}$

Step II

Q/Σ	a	b
q_0	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_0, q_2\}$
q_2	-	$\{q_0, q_1\}$

Step III :- Define Tupples of DFA

Step IV :- Transition table of DFA

Q/Σ	a	b
$[q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_0, q_1]$	$[q_0 \Sigma, q_2]$
$[q_0 q_1 q_2]$	$[q_0 q_1]$	$[q_0 q_1 q_2]$
$[q_2]$	$[q_3]$	$[q_0 q_1]$
$[q_3]$	$[q_3]$	$[q_3]$

Step V :- Tuples of DFA

$M' = \langle Q', \Sigma', \delta', q_0', F' \rangle$.

$Q' = \{[q_0], [q_0, q_1], [q_0 q_2], [q_0 q_1 q_2], [q_3]\}$.

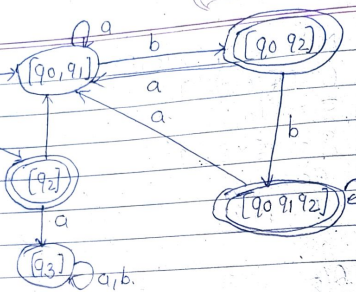
$\Sigma' = \{a, b\}$

$\delta' = Q' \times \Sigma' = 2^{Q'}$.

$q_0' = \{[q_0]\}$

$F' = \{[q_0 q_2], [q_0 q_1 q_2], [q_2]\}$

Step VI →



Q Convert a NFA into DFA

$M = \langle \{p,q,r,s\}, \{0,1\}, \delta, \{p\}, \{s\} \rangle$

Q/Σ	0	1
p	{p,q}	{p}
q	{r}	{s}
r	{s}	-
s	{s}	{s}

Step I :- $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

$Q = \{p,q,r,s\}$

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

$\delta = Q \times \Sigma = Q$

$q_0 = \{p\}$

$F = \{s\}$

Step II :-

Q/Σ	0	1
p	{p,q}	{p}
q	{r}	{s}
r	{s}	-
s	{s}	{s}

Step III Define tuples of DFA

Step IV :-

Q/Σ	0	1
[p]	[p,q]	[p]
[p q]	[p q r]	[p s]
[p q r]	[p q r s]	[p s]
[p q r s]	[p q r s]	[p s]
[p s]	[p q s]	[p s]
[p q s]	[p q r s]	[p s]

Step V Tuples of DFA

$M = \langle Q', \Sigma', \delta', q_0', F' \rangle$

$Q' = \{ [p], [p q], [p q r], [p q r s], [p s], [p q s] \}$

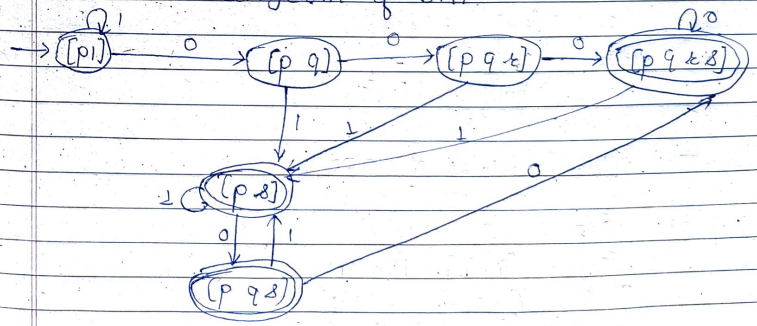
$\Sigma' = \{0,1\}$

$\delta' = Q' \times \Sigma' = 2^{Q'}$

$q_0' = \{[p]\}$

$F' = \{ [p q r s], [p s], [p q s] \}$

Step VI Transition diagram of DFA



Q.

Θ/Σ	0	1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	–	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	–
q_4	$\{q_4\}$	$\{q_4\}$

Step I :- $M = <Q^*, \Sigma^*, \delta^*, q_0^*, F^*>$

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$\delta = Q \times \Sigma = Q$

$q_0 = \{q_0\}$

$F = \{q_4, q_2\}$

Step II :-

Q/Σ	0	1
$\{q_0\}$	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	–	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	–
q_4	$\{q_4\}$	$\{q_4\}$

Step III. Define tuples of DFA

Step IV :-

Θ/Σ	0	1
$[q_0]$	$[q_0 q_3]$	$[q_0 q_1]$
$[q_0 q_3]$	$[q_0 q_3 q_4]$	$[q_0 q_1]$
$[q_0 q_3 q_4]$	$[q_0 q_3 q_4]$	$[q_0 q_1 q_4]$
$[q_0 q_1 q_4]$	$[q_0 q_3 q_4]$	$[q_0 q_1 q_2 q_4]$
✓ $[q_0 q_1 q_2 q_4]$	$[q_0 q_2 q_3 q_4]$	$[q_0 q_1 q_2 q_4]$
✓ $[q_0 q_2 q_3 q_4]$	$[q_0 q_2 q_3 q_4]$	$[q_0 q_1 q_2 q_4]$
$[q_0 q_1]$	$[q_0 q_3]$	$[q_0 q_1 q_2]$
$[q_0 q_2 q_3]$	$[q_0 q_3 q_2 q_4]$	$[q_0 q_1 q_2]$

Step V Tuples of DFA :

$M = <Q', \Sigma', \delta', q_0', F'>$

$Q' = \{[q_0], [q_0 q_3], [q_0 q_1], [q_0 q_3 q_4], [q_0 q_1 q_4], [q_0 q_1 q_2 q_4], [q_0 q_3 q_3 q_4], [q_0 q_1 q_2], [q_0 q_2 q_3]\}$,

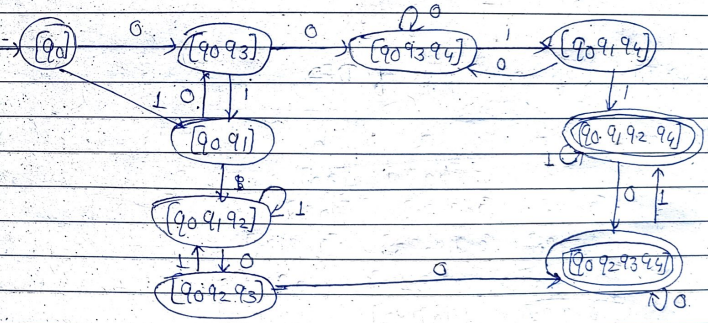
$\Sigma' = \{0, 1\}$

$\delta' = Q' \times \Sigma' = 2^{Q'}$

$q_0' = \{[q_0]\}$

$F' = \{[q_0 q_1 q_2 q_4], [q_0 q_2 q_3 q_4]\}$

Step VI →



Q. $M = <\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, \{q_0\}, \{q_3\}>$

where δ is defined by

Q/Σ	a	b
q_0	$\{q_0 q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_1\}$
q_2	$\{q_3\}$	$\{q_3\}$
q_3	–	$\{q_2\}$

Step I :- $M = <Q, \Sigma, \delta, q_0, F>$

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

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

$\delta = Q \times \Sigma = Q$

$q_0 = \{q_0\}$

$F = \{q_3\}$

Step II :-

Q/Σ	a	b
q_0	$\{q_0 q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_1\}$
q_2	$\{q_3\}$	$\{q_3\}$
$\{q_3\}$	-	$\{q_2\}$

Step III :- Define tuples of DFA

Step IV :-

Q/Σ	a	b
$[q_0]$	$[q_0 q_1]$	$[q_0]$
$[q_0 q_1]$	$[q_0 q_1 q_2]$	$[q_0 q_1]$
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_3]$
$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_2 q_3]$
$[q_0 q_1 q_3]$	$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$

Step V :- Tuples of DFA

$M = <Q', \Sigma', \delta', q_0', F'>$

$Q' = \{[q_0], [q_0 q_1], [q_0 q_1 q_2], [q_0 q_1 q_2 q_3], [q_0 q_1 q_3]\}$

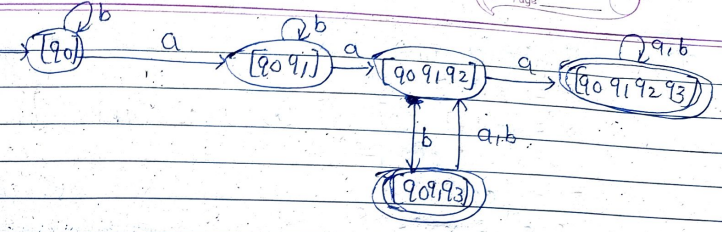
$\Sigma' = \{a, b\}$

$\delta' = Q' \times \Sigma' = 2^{Q'}$

$q_0' = \{[q_0]\}$

$F' = \{[q_0 q_1 q_2 q_3], [q_0 q_1 q_3]\}$

Step VI



Q	Q/Σ	0	1
p		$\{q, s\}$	$\{q\}$
q		$\{r\}$	$\{q, r\}$
r		$\{s\}$	$\{p\}$
s		-	$\{p\}$

Step I :- $M = <Q, \Sigma, \delta, q_0, F>$

$Q = \{p, q, r, s\}$

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

$\delta = Q \times \Sigma$

$q_0 = \{p\}$

$F = \{s\}$

Step II :-

Q/Σ	0	1
p	$\{q, s\}$	$\{q\}$
q	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
s	-	$\{p\}$

Step III :- Define tuples of DFA

Step IV :- Transition table of DFA

Q/Σ	0	1
$[p]$	$[q s]$	$[q]$
$[q s]$	$[r]$	$[p q r]$
$[q]$	$[r]$	$[q r]$
$[r]$	$[s]$	$[p]$
$[p q r]$	$[q r s]$	$[p q r]$
$[q r]$	$[r s]$	$[p q r]$
$[s]$	$[t]$	$[p]$
$[q r s]$	$[r s]$	$[p q r]$
$[r s]$	$[s]$	$[p]$
$[t]$	$[t]$	$[t]$

Step V Tuples of DFA

$M = \langle Q', \Sigma', \delta', q_0', F' \rangle$

$Q' = \{[q_0], [p], [q], [r], [pq_0], [q_0], [s], [q_0 s], [r s], [t]\}$

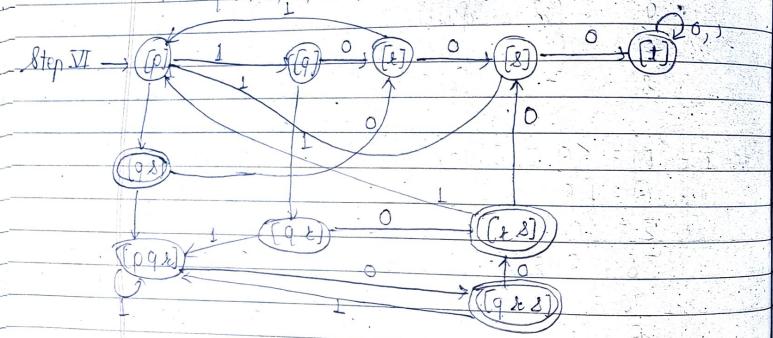
$\Sigma' = \{0, 1\}$

$\delta' = Q' \times \Sigma' = 2^{Q'}$

$q_0' = \{[p]\}$

$F' = \{[q_0 s], [q_0 s], [r s], [s]\}$

Step VI



MOORE MACHINE

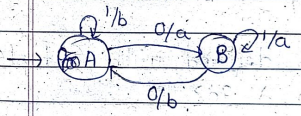
Definition:- It is defined as a finite automata where o/p is generated or associated on state itself.

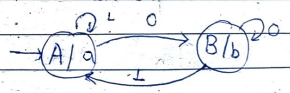
Moore Machine is defined with 6 tuples.

$M = \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$

Finite Automata With Outputs

MEALY MACHINE	MOORE MACHINE
$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$	$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$
where	where
Q = Finite set of states.	Q = Finite set of states
Σ = Finite non-empty set of I/p alphabets	Σ = Finite non-empty set of I/p alphabets
Δ = The set of o/p alphabets	Δ = The set of o/p alphabets
δ = Transition function. $Q \times \Sigma \to Q$	δ = Transition function $Q \times \Sigma \to Q$
λ = Output function $\Sigma \times Q \to \Delta$	λ = Output function: $Q \to \Delta$
q_0 = Initial state / Start state	q_0 = Initial state / Start state





FINITE AUTOMATA WITH OUTPUT
(Output Generator)

- Important points on Moore & Mealy Machine
- Both Moore & Mealy machine are output generators rather than language acceptors.
- There is no need to define ant final state in Moore & Mealy machine i.e. $F = \phi$
- There is no concept of dead state in Moore & Mealy machine.

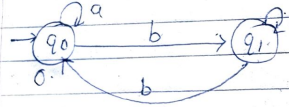
- Both Moore & Mealy machine follows the principle of DFA i.e. from each other state for each input symbol there will be exactly one transition.

Representation of Moore Machine

- Moore Machine can be represented in two ways.

1. Transition Diagram
2. Transition table

Ex:-



Steps for solving the problem.

Step 1: Moore Machine is defined with 6 tuples.

$M = <Q, \Sigma, \delta, \Delta, \lambda, q_0>$

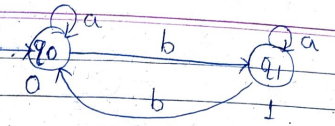
Q: Finite set of all states.
Σ: Finite set of I/p symbol.
δ: Transition function
Δ: Finite set of O/p symbol.
λ: Output Mapping
q_0: Initial state

Step 2: Explain the logic in detail.

Step 3: Draw transition of Moore Machine.

Step 4: Provide the value to the tuples of Moore Machine.

Example:-



$M = <Q, \Sigma, \delta, \Delta, \lambda, q_0>$
$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$
$\Delta = \{0, 1\}$
$\lambda \Rightarrow Q \to \Delta$
$q_0 = \{q_0\}$

δ Mapping

Q/Σ	a	b
q_0	q_0	q_1
q_1	q_1	q_0

λ Mapping

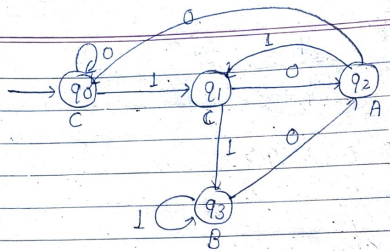
Q	Δ
q_0	0
q_1	1

Combine mapping

Q/Σ	a	b	Δ
q_0	q_0	q_1	0
q_1	q_1	q_0	1

Q. Construct a Moore machine which takes set of all string as input & generate 'A' as O/p if string ends with 10 & generate 'B' as O/p if string ends with 11 & otherwise 'C'.

Transition diagram of Moore machine



Transition table

δ : Mapping

Q/Σ	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3

λ : Mapping

Q	Δ
q_0	C
q_1	C
q_2	A
q_3	B

Combine mapping

Q/Σ	0	1	Δ
q_0	q_0	q_1	C
q_1	q_2	q_3	C
q_2	q_0	q_1	A
q_3	q_2	q_3	B

Tuples of Moore machine

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

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

$\delta = Q \times \Sigma \to Q$

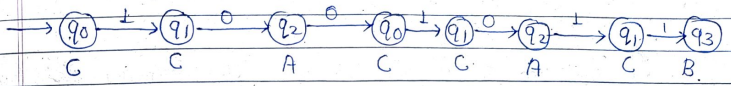
$\Delta = \{A, B, C\}$

$\lambda = Q \to \Delta$

$q_0 = \{q_0\}$

Processing of input string 1001011

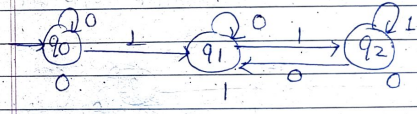
$\lambda(1001011) = CCACCACB$



Q Construct a Moore machine for 0/p 2's complement for every binary o/p string

=

Transition diagram of Moore machine



Transition table

δ : Mapping

Q/Σ	0	1
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_2

λ : Mapping

Q	Δ
q_0	0
q_1	1
q_2	0

Combine mapping

Q/Σ	0	1	Δ
q_0	q_0	q_1	0
q_1	q_1	q_2	1
q_2	q_1	q_2	0

Tupples of Moore machine

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

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

$\delta = Q \times \Sigma \to Q$

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

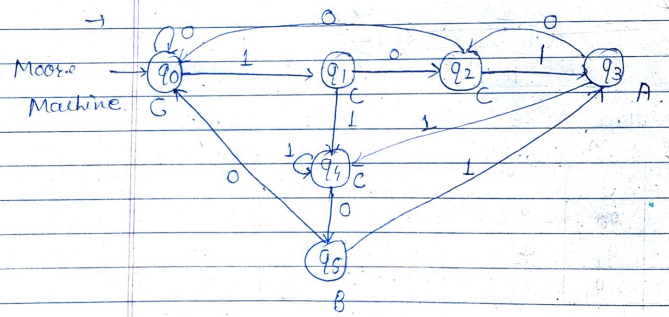
$\lambda = Q \to \Delta$

$q_0 = \{q_0\}$

Q. Design a Moore machine for binary i/p sequence if it ends in 101 o/p is 'A' if ends in 110 o/p is 'B'

- Since we have to generate o/p A for binary i/p string which is ends with 101
- Whenever string ends with 110 it procedure o/p 'B' & otherwise it procedure o/p 'C'

→



Transition table

δ Mapping		
Q/Σ	0	1
q_0	q_0	q_1
q_1	q_2	q_4
q_2	q_0	q_3
q_3	q_2	q_4
q_4	q_5	q_4
q_5	q_0	q_3

λ Mapping	
Q	Δ
q_0	C
q_1	C
q_2	C
q_3	A
q_4	C
q_5	B

Combine mapping

Q/Σ	0	1	Δ
q_0	q_0	q_1	C
q_1	q_2	q_4	C
q_2	q_0	q_3	C
q_3	q_2	q_4	A
q_4	q_5	q_4	C
q_5	q_0	q_3	B

Tupples of Moore machine

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

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

$\delta = Q \times \Sigma \to Q$

$\Delta = \{A, B, C\}$

$\lambda = Q \to \Delta$

$q_0 = \{q_0\}$

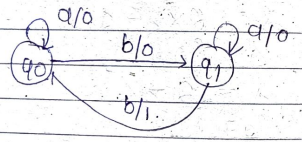
MEALY MACHINE

- It is defined as finite automata where output depends or associated on the transition. Hence it is called as Mealy machine.
- In Mealy machine, current state & current input symbol will decide o/p.
- Mealy Machine is defined with 6 tuples
  $M = \langle Q, \Sigma, \delta, \lambda, \Delta, q_0 \rangle$
- Q = Finite set of all state
  Σ = Finite set of I/p Symbol
  δ : Transition function

λ : Output Mapping. $Q * \Sigma \to \Delta$.
Δ : Finite set of o/p symbol.
q_0 : Initial state.

Note :- In Mealy machine the length of input & output symbol is same because from each state & for each input symbol it represent the output.

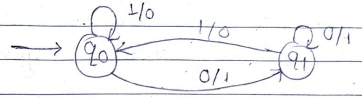
Representation of Mealy Machine
Mealy machine can be represented in two way
a. Transition Diagram
b. Transition Table



Q. Design a Mealy machine to generate the output 1's complement of any binary number.

→ Mealy machine is defined with 6 tuples.
$M = <Q, \Sigma, \delta, q_0, \Delta, \lambda>$
Q = Finite set of all state.
Σ = Finite set of i/p symbol.
δ - Transition function
Δ - Finite set of o/p symbol.
λ = O/p mapping
q_0 = Initial state.

Logic : From every state & for each i/p symbol we have to generate the o/p whenever the '0' input comes we have to generate output '1' & vice versa.



$Q = \{q_0, q_1\}$.
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$.
$\lambda = Q \times \Sigma \to \Delta$
$\Delta = \{0, 1\}$
$q_0 = \{q_0\}$

δ: Mapping

Q/Σ	0	1
→ q_0	q_1	q_0
q_1	q_1	q_0

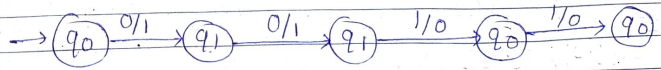
λ : Mapping

Q/Σ	0	1
→ q_0	1	0
q_1	1	0

Combine mapping

Q/Σ	0	Δ	1	Δ
→ q_0	q_1	1	q_0	0
q_1	q_1	1	q_0	0

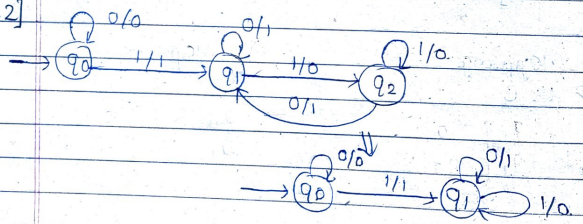
λ (0011) = 1100.



Steps to Design a Mealey Machine

- Define the tuples
- Logic: From every state & for each i/p symbol we have to generate the output whenever the '0' i/p comes we have to generate o/p '1' & vice versa
- Name the tuples.
- Draw δ: Mapping
- Draw λ: Mapping
- Combine Mapping
- Verify the random string.

Q.2]



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

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

$\delta = Q \times \Sigma \to Q$

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

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

$q_0 = \{q_0\}$.

- Logic - ~~From every~~
- Mapping

δ: Mapping

Q/Σ	0	1
→ q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_2

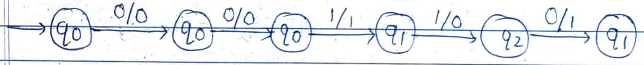
λ: Mapping

Q/Σ	0	1
→ q_0	0	1
q_2	1	0
q_2	1	0

Combine Mapping

→ q_0 Q/Σ	0	Δ	1	Δ
→ q_0	q_0	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_1	1	q_2	0

Verifying λ (00110) = 00101



CONVERSION OF MOORE MACHINE TO MEALY MACHINE

i] While converting a given Moore machine to Mealy machine the total number of state remain & total number of transition are also same.

ii] Only change we have to perform transfer the state o/p on transition.

11061 + 1 = 11010 → 00110

11001 + 1 = 11010 → 00101

Steps for solving Moore machine to Mealey Machine

Step 1 :- Moore Machine has 6 tuples
$M = <Q, \Sigma, \delta, \lambda, \Delta, q_0>$

Step 2 :- In the problem transition diagram is given then construct transition table & vice versa

Step 3 :- Mealy machine is defining 6 tuples where $M = <Q, \Sigma, \Delta, \delta, \lambda, q_0>$
where Q = Finite sets of all states
Σ = Finite set of I/p
δ - Transition function
Δ - Finite set of O/p Symbol
λ = O/p mapping
q_0 - Initial state

Step 4 :- Logic explanation - a] While constructing Moore machine to Mealey machine, the total no of state remain same

b] Only the logic that we have to apply is to transfer the state O/p to transition

Step 5 :- Draw transition diagram of Mealey machine by applying above logic

Step 6 :- Provide value to tupples of Mealey machine

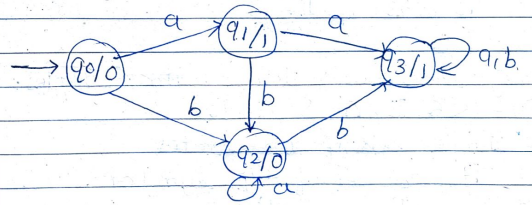
Step 7 :- Draw the transition diagram of Mealey machine.

Q. Construct a Moore machine equivalent to Mealey machine.

Present State	Input state	NEXT State a=0 O/p	Input state	a=1 O/p
q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Ans :- Mealey Machine is defined with 6 tuples
$M = <Q, \Sigma, \delta, \Delta, \lambda, q_0>$
$Q = \{q_1, q_2, q_3, q_4\}$
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$.
$\Delta = \{0, 1\}$.
$\lambda = Q \times \Sigma \to \Delta$
$q_0 = \{q_1\}$

Q. Construct an equivalent Mealey machine from the given Moorey machine.



Step 1. Moore machine has 6 tuples.
$M = <Q, \Sigma, \delta, \lambda, \Delta, q_0>$

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$
$\Delta = \{0, 1\}$
$\lambda = Q \times \Sigma \to \Delta$
$q_0 = \{q_0\}$

Step 2:- Transition table

δ : Mapping

Q/Σ	a	b
→ q_0	q_1	q_2
q_1	q_3	q_2
q_2	q_2	q_3
q_3	q_3	q_3

λ : Mapping

Q/Σ	Δ
→ q_0	0
q_1	1
q_2	0
q_3	1

Combine Mapping

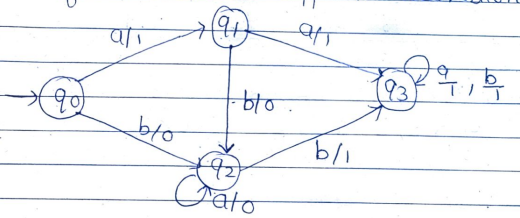
Q/Σ	a	b	Δ
→ q_0	q_1	q_2	0
q_1	q_3	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_3	1

Step 3:- Mealey Machine is define with 6 tuples
$M = <Q, \Sigma, \Delta, \delta, \lambda, q_0>$
Q = Finite sets of all states.
Σ = Finite set of I/p symbol.
δ = Transition function
Δ - Finite set of O/p symbol
λ - O/p mapping.
q_0 - Initial state

Step 4:- Logic explaination :- a) While constructing Moore machine to Mealey machine, the total no of state remain same

b) Only the logic that we have to apply is to transfer the state O/p to transition.

Step 5



Step 6:- Tupples of Mealey machine
$M = <Q, \Sigma, \Delta, \delta, \lambda, q_0>$
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\Delta = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$
$\lambda = Q \times \Sigma \to \Delta$
$q_0 = \{q_0\}$

Step 7. Transition diagram of Mealey machine

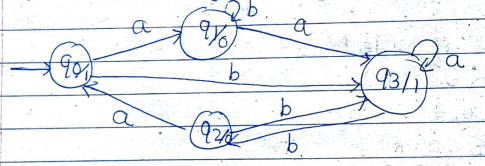
Q/Σ	a	Δ	b	Δ
q_0	q_1	1	q_2	0
q_1	q_3	1	q_2	0
q_2	q_2	0	q_3	1
q_3	q_3	1	q_3	1

Q. Convert Moorey to Mealey Machine

State	Input a	Input b	Output Δ
q_0	q_1	q_3	1
q_1	q_3	q_1	0
q_2	q_0	q_3	0
q_3	q_3	q_2	1

Step 1:- Moorey Machine has 6 tuples
$M = \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$

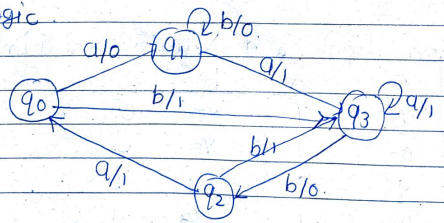
Step 2:- Transition diagram



Step 3:- Mealey machine has 6 tuples
$M = \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$

Step 4:- Logic

Step 5:-



Step 6:- Tupples of Mealey machine

$M = \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$
$\Delta = \{0, 1\}$
$\lambda = Q \times \Sigma \to \Delta$
$q_0 = \{q_0\}$

Step 7:- Transition table of Mealey machine

Q/Σ	a	Δ	b	Δ
q_0	q_1	0	q_3	1
q_1	q_3	1	q_1	0
q_2	q_0	1	q_3	1
q_3	q_3	1	q_2	0

CONVERSION FROM MEALEY MACHINE TO MOORE MACHINE

1. While converting mealy machine to moore machine the total number of states may change
2. Initial state of mealey machine & Moore machine remain same
3. While constructing moore machine from Mealy machine transfer the transition output on state and start the construction from initial state and continue the process for every new state appear in the input column

and terminate the process whenever no new state appear in the input column

4. While converting mealey machine to moore machine transfer the transition output on state if multiple output is traversing on the state the create its multiple instance.

STEPS FOR CONVERSION OF MEALEY MACHINE TO MOORE MACHINE

Step 1:- Mealey Machine is defined with 6 tuples.
$M = \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$ and provide the values to the tuples

Step 2:- If δ in the problem transition diagram is given then construct the transition table and vice versa

Step 3:- Moore machine is defined with 6 tuples.
$M' = \langle Q', \Sigma, \delta', \Delta, \lambda', q_0' \rangle$
Where Q' = Finite set of all states in Mealey machine
Σ = Finite set of input symbol.
δ' = Transition function of Mealey
Δ = Output mapping.
λ = Finite set if output symbol
q_0' = intial state

Step 4:- Explanation of logic of converting Mealey to Moore machine.

Step 5:- Draw the transition table of Moore Machine

Step 6:- Provide the values to the tuples of Moore machine.

Step 7:- Draw the transition Diagram of Moore machine

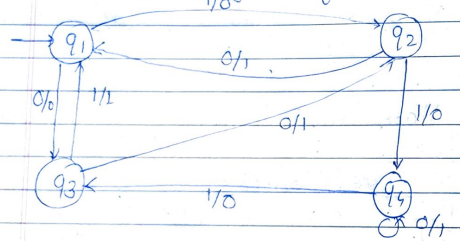
Q. Construct a Moore machine equivalent to Mealy machine.

Present state	NEXT STATE. Input state (a=0)	o/p	Input state (a=1)	o/p
q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Step1. Mealey machine is defined with 6 tuples.
$M = \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$
$Q = \{q_1, q_2, q_3, q_4\}$
$\Sigma = \{0, 1\}$.
$\delta : Q \times \Sigma \to Q$.
$\Delta = \{0, 1\}$.
$\lambda : Q \times \Sigma \to \Delta$.
$q_0 = \{q_1\}$.

Present state	Next state: Input = 0	o/p	Input = 1	o/p
	Input state		Input state	
q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0.

Step 2. Transition diagram of Mealey Machine



Step 3:- Moore Machine is defined with 6 tuples.

$M' = <Q', \Sigma, \delta', \Delta, \lambda', q_0'>$

Q' : Finite set of all state of Moore machine

Σ : Finite set of input symbol

δ' : Transition function of Moore Machine

Δ : Finite set of output symbol.

λ' : Output mapping of Moore Machine.

q_0' : Initial state of Moore Machine

Step 4:- logic : 1] While converting Mealey Machine to Moore machine the total number of state may change

2] Initial state of Moore machine remain same

3] While constructing Moore Machine from Mealey Machine transfer the transition output in state and start the construction from initial state and contain the process for every new state appear in the input column and terminate the process whenever us new state appear in the input column

4] While converting Mealey machine to Moore machine transfer the transition output on state of multiple output is transversing on that state then create its multiple instances.

Step 5:- Transition table of Moore Machine

Θ/Σ	0	1	Δ
→ q_11	q_30	q_20	1
q_20	q_11	q_40	0
q_21	q_11	q_40	1
q_30	q_21	q_11	0
q_40	q_41	q_30	0
q_41	q_41	q_30	1

Step 6:- Tuples of Moore Machine

$Q' = \{q_11, q_20, q_21, q_30, q_40, q_41\}$

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

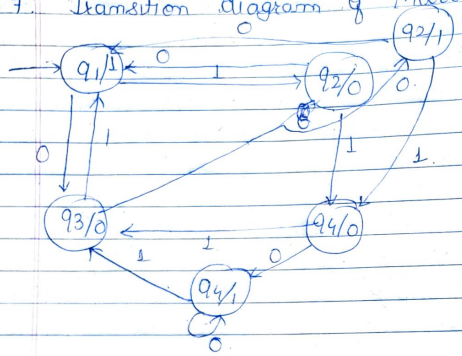
$\delta' = \Theta' \times \Sigma \rightarrow \Theta'$

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

$\lambda = Q' \rightarrow \Delta$

$q_0 = \{q_11\}$

Step 7: Transition diagram of Moore machine



Q.

Present state	NEXT STATE a	Δ	b	Δ
q_0	q_5	0	q_3	0
q_1	q_4	1	q_4	1
q_2	q_3	0	q_5	1
q_3	q_2	1	q_1	1
q_4	q_1	0	q_0	0
q_5	q_0	0	q_2	0

Step 1:- Mealey machine is defined with 6 tuples.

M: $<Q, \Sigma, \delta, \Delta, \lambda, q_0>$

Q: $\{q_0, q_1, q_2, q_3, q_4, q_5\}$

Σ: $\{a, b\}$

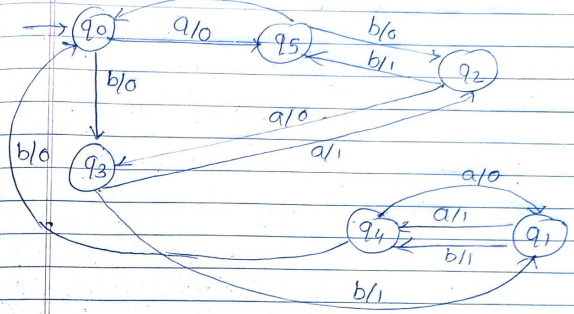
δ: $Q \times \Sigma \to Q$

Δ: $\{0, 1\}$

λ: $Q \times \Sigma \to \Delta$

q_0: $\{q_0\}$

Step 2: Transition diagram of Mealey machine.



Step 3: Moore machine is defined with 6 tuples.

$M' = <Q', \Sigma, \delta', \Delta, \lambda', q_0'>$

Q': Finite set of all state of Moore machine

Σ: Finite set of input symbol

δ': Transition function of Moore Machine

Δ: Finite set of output symbol

λ': Output mapping of Moore Machine

q_0': Initial state of Moore Machine

Step 4: Logic

Step 5:

Q/Σ	a	b	Δ
q_0 0	q_5 0	q_3 0	0
q_1 0	q_4 1	q_4 1	0
q_1 1	q_4 1	q_4 1	1
q_2 1	q_3 0	q_5 1	1
q_2 0	q_3 0	q_5 1	0
q_3 0	q_2 1	q_1 1	0
q_4 1	q_1 0	q_0 0	1
q_5 0	q_0 0	q_2 0	0
q_5 1	q_0 0	q_2 0	1

Step 6: Tupples of Moore Machine

$M' = \langle Q', \Sigma, \delta', \Delta, \lambda', q_0' \rangle$

$Q' = \{q_0^0, q_0^1, q_1^1, \cancel{q_1^0}, q_2^1, q_2^0, q_3^0, q_4^1, q_5^0, q_5^1\}$

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

$\delta' = Q' \times \Sigma \Rightarrow Q'$

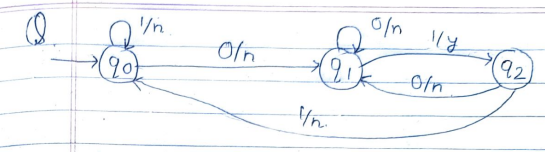
$\Delta = \{0, 1\}$

$\lambda = Q' \times \Sigma \rightarrow \Delta$

$q_0 = \{q_0^0\}$

Step 7:

Q.



Step 1:- Mealey machine is defined with 6 tuples

$M : \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$

$Q : \{q_0, q_1, q_2\}$

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

$\delta : Q \times \Sigma \rightarrow Q$

$\Delta : \{n, y\}$

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

$q_0 : \{q_0\}$

Step 2:- Transition table of Mealey Machine

Present state	NEXT STATE Input state	a=1 O/P	Input state	a=0 O/P
q_0	q_0	n	q_1	n
q_1	q_2	y	q_1	n
q_2	q_0	n	q_1	n

Step 3:- Moore machine is defined with 6 tuples.

$M' = \langle Q', \Sigma, \delta', \Delta, \lambda', q_0' \rangle$

Q' : Finite set of all state of Moore machine.

Σ : Finite set of input symbol

δ' : Transition function of Moore Machine.

Δ : Finite set of output symbol.

λ' : Output mapping of Moore Machine.

q_0' : Initial state of Moore Machine.

Step 4: Logic.

Step 5:-

Q/Σ	0	1	Δ
$q_0 n$	$q_1 n$	$q_0 n$	n
$q_1 n$	$q_1 n$	$q_2 y$	n
$q_2 y$	$q_1 n$	$q_0 n$	y

Step 6:- Tupples of Moore Machine

$M' = \langle Q', \Sigma, \delta', \Delta, \lambda', q_0' \rangle$

$Q' = \{q_0 n, q_1 n, q_2 y\}$

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

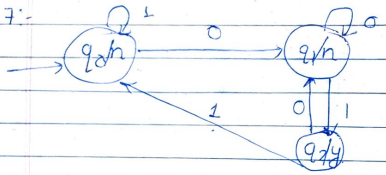
$\delta' = Q \times \Sigma \to Q'$

$\Delta = \{n, y\}$

$\lambda = Q' \times \Sigma \to \Delta$

$q_0 = \{q_0 n\}$

Step 7:-



✱ NFA WITH ∈-MOVE OR ∈-NFA

Definition

It is defined as NFA which has transition for empty string ∈ is called as ∈-NFA, ∈ NFA is defined with 5 tuples.

$M = \langle Q, \Sigma, \delta, q_0, F \rangle$

where

Q :- Finite set of all state.

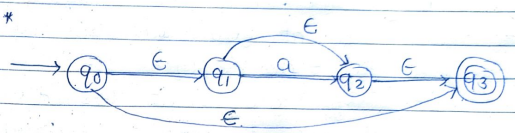
Σ :- Finite set of input symbol.

δ: Transition function $(Q * \{\Sigma \cup \in\} \to 2^{Q})$

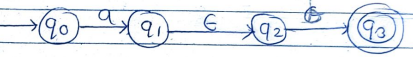
q_0 : Initial State

F : Start of all final state

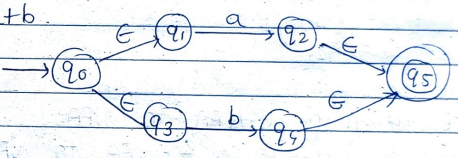
1] a^*



ii] a.b



iii] a+b



Note :-

i] Excepting power of capabilities of ∈ NFA, NFA and DFA are same

ii] ∈-NFA can be converted into NFA and NFA can be converted into DFA but reverse mechanism is not possible.

iii] The inclusion or exclusion transition of ∈-NFA will not affect the language to NFA.

Defination of ϵ - closure

ϵ - closure (q) :-

i) If q is any state in ϵ NFA then the set of all state which are at zero distance from state q is called as ϵ-closure (q)

iii) By default every state is at zero distance from itself therefore it cannot be null set.

ex :-

ϵ - closure $(q_0) = \{q_0, q_1, q_2\}$

$(q_1) = \{q_1, q_2\}$

$(q_2) = \{q_2\}$

Logic of conversion of ϵ-NFA to NFA

i) The process of conversion of ϵ-NFA to NFA is called Thomson conversion.

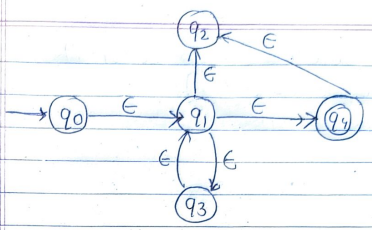
ii) There is no change in total number of state while converting ϵ NFA to NFA. Initial state is also same

iii) Every state whose ϵ - closure contain the final state of ϵ NFA. Final state may be change in NFA

iv) Transition function of NFA is denoted by δ'

$\delta'(q, x) = \epsilon$ - closure $(\delta(\epsilon\text{-closure}(q), x))$.

Q



ϵ - closure $(q_0) = \{q_0, q_1, q_2, q_3, q_4\}$

ϵ - closure $(q_1) = \{q_1, q_2, q_3, q_4\}$

$(q_2) = \{q_2\}$ ~~q_1, q_3, q_4~~

$(q_3) = \{q_3, q_1, q_2, q_4\}$

$(q_4) = \{q_4, q_2\}$

Q.

ϵ - closure of $(q_0) = \{q_0, q_1\}$

$(q_1) = \{q_1\}$

$\delta'(q_0, 0) = \epsilon$ - closure $(\delta(\epsilon\text{-closure of } q) x)$

$= \epsilon$ - closure $(\delta(\epsilon\text{-closure of } q_0) 0)$

$= \epsilon$ - closure $(\delta(q_0, q_1) 0)$

$= \epsilon$ - closure $(q_0 \cup \phi)$

$= \{q_0, q_1\}$.

$\delta'(q_0, 1) = \epsilon$ - closure $(\delta(\epsilon\text{-closure of } q_0) 1)$

$= \epsilon$ - closure $(\delta\{q_0, q_1\} 1)$

$= \epsilon$ - closure $(\phi \cup q_1)$

$= \epsilon$ - closure $\{q_1\}$.

$= \{q_1\}$.

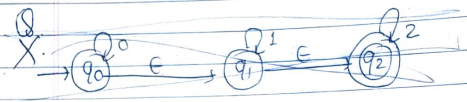
$\delta'(q_1, 0) = \epsilon$ - closure $(\delta(\epsilon\text{-closure of } q_1) 0)$

$= \epsilon$ - closure $(\delta\{q_1\} 0)$

$= \epsilon$ - closure ϕ

$= \phi$

$\delta'(q_1, 1) = \epsilon$ - closure $(\delta(\epsilon\text{-closure } q_1), 1)$
$= \epsilon$ - closure $(\delta(q_1), 1)$
$= \epsilon$ - closure (q_1)
$= \{q_1\}$



STEPS FOR SOLVING PROBLEM

Step 1: ϵ-NFA is defined with 5 tuples.
$M = <Q, \Sigma, \delta, q_0, F>$ and also provide the values to the tuples.
~~Q: Finite set of all state in NFA~~
Σ: Finite set of input symbol.
δ: Transition fun^n of NFA.
q_0': Initial state in NFA.
~~F: Set of~~

Step 2: Determine the ϵ-closure of every state.

Step 3: let $M = <Q', \Sigma, \delta', q_0', F'>$ are the tuples of NFA where
Q': Finite set of all state in NFA.
Σ: Finite set of i/p symbol
δ': Transition fun^n of NFA
q_0': Initial ~~set~~ state in NFA
F': Set of all final state in NFA.

Step 4: Define the construction of δ' & explain the logic.

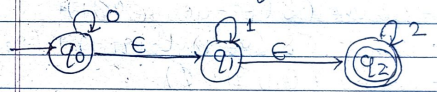
Step 5: Draw the transition diagram of NFA.

Step 6: Provide the value to the tupples of NFA.

Step 7: Draw the transmission table of NFA.

Logic (Explain):
i While converting ϵ-NFA to NFA the total no of state remain same & initial state is also same i.e. $q_0' = q_0$.
ii Every state whose ϵ-closure contains the final state of ϵ-NFA is the final state of NFA.

Q. Convert NFA from ϵ NFA.



Step 1: ϵ-NFA is defined with 5 tuples.
$M = <Q, \Sigma, \delta, q_0, F>$.
$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$.
$q_0 = \{q_0\}$
$F = \{q_2\}$.

Step 2: Determine ϵ-closure.
ϵ closure of $(q_0) = \{q_0, q_1, q_2\}$
ϵ closure of $(q_1) = \{q_1, q_2\}$
ϵ closure of $(q_2) = \{q_2\}$.

Step 3: Let $M = <Q', \Sigma, \delta', q_0', F'>$ are the tuples of NFA where
Q': Finite set of all states in NFA
Σ: Finite set of i/p symbol
δ': Transition fun^n of NFA
q_0': Initial state in NFA
F: Set of all final state in NFA

Step 4: Construction of δ'

$\delta'(q_1, x) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q) x)$

$\delta'(q_0, 0) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_0) 0)$
$= \epsilon$ - closure $(\delta(q_0 q_1 q_2) 0)$
$= \epsilon$ - closure $(\delta(q_0,0) \cup (q_1,0) \cup (q_2,0))$
$= \epsilon$ - closure $(q_0 \cup \phi \cup \phi)$
$= \{q_0, q_1, q_2\}$

$\delta'(q_0, 1) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_0) 1)$
$= \epsilon$ - closure $(\delta(q_0,1) \cup (q_1,1) \cup (q_2))$
$= \epsilon$ - closure $(\phi \cup q_1 \cup \phi)$
$= \{q_1, q_2\}$

$\delta'(q_1, 0) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1), 0)$
$= \epsilon$ - closure $(\delta(q_1,0) \cup (q_2,0))$
$= \epsilon$ - closure $(\phi \cup \phi)$
$= \{\phi\}$

$\delta'(q_1, 1) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1), 1)$
$= \epsilon$ - closure $(\delta(\epsilon$ ~~- closure~~ $q_1 q_2) 1)$
$= \epsilon$ - closure $(\delta(q_1,1) \cup (q_2,1)$
$= \epsilon$ - closure $(q_1 \cup q_2)$
$= \{q_1, q_2\}$

$\delta'(q_0, 2) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_0), 2)$
$= \epsilon$ - closure $(\delta(q_0,2) \cup (q_1,2) \cup (q_2,2))$
$= \epsilon$ - closure $(\phi \cup \phi \cup q_2)$
$= \{q_2\}$

$\delta'(q_1, 2) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1), 2)$
$= \epsilon$ - closure $(\delta(q_1,2) \cup (q_2,2))$
$= \epsilon$ - closure $(q_2 \cup \phi)$
$= \{q_2\}$

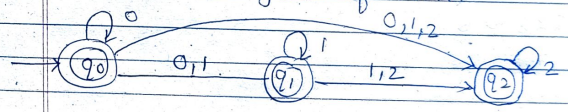
$\delta'(q_2, 0) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_2) 0)$
$= \epsilon$ - closure $(\delta(\epsilon$ - closure $(q_2)) 0)$
$= \epsilon$ - closure $(\delta(q_2, 0)$
$= \{\phi\}$

$\delta'(q_2, 1) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_2), 1)$
$= \epsilon$ - closure $(\delta(q_2, 1))$
$= \{\phi\}$

$\delta'(q_2, 2) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_2), 2)$
$= \epsilon$ - closure $(\delta(q_2, 2))$
$= \{q_2\}$

Logic :-

Step 5: Transition diagram of NFA.

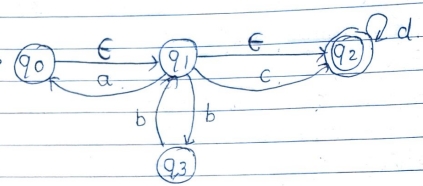


Step 6: Tuples of NFA.
$M = <Q', \Sigma, \delta', q_0', F'>$
$Q' = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\delta' = Q \times \Sigma \to_2 Q'$
$q_0' = \{q_0\}$
$F = \{q_0, q_1, q_2\}$.

Step 7: Transmission table of NFA.

Q	0	1	2
→ q_0	$[q_0 q_1 q_2]$	$[q_1 q_2]$	$[q_2]$
q_1	ϕ	$[q_1 q_2]$	$[q_2]$
q_2	ϕ	ϕ	$[q_2]$

Q.



Step 1:- NFA has 5 tuples
$M = \langle Q, \Sigma, \delta, q_0, F \rangle$
$Q := \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, c, d\}$
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_2\}$

Step 2:- Determine ε - closure
ε - closure of $(q_0) = \{q_0, q_1, q_2\}$.
$(q_1) = \{q_1, q_2\}$.
$(q_2) = \{q_2\}$
$(q_3) = \{q_3\}$.

Step 3:- let $M = \langle Q', \Sigma, \delta', q_0', F' \rangle$ are the tuples of NFA.
where Q' = Finite set of all state in NFA.
Σ := Finite set of i/p symbol.
δ': Transition function of NFA.
q_0': Initial state of NFA
F': Final state of NFA.

Step 4: Construction of δ'

$\delta'(q, x) = \epsilon$ - closure $(\delta(\epsilon$ - closure of $q), x)$.

$\delta'(q_0, a) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_0), a)$
$= \epsilon$ - closure $(\delta(q_0, a) \cup (q_1, a) \cup (q_2, a))$
$= \epsilon$ - closure $(\phi \cup q_0 \cup \phi)$
$= \{q_0, q_1, q_2\}$.

$\delta'(q_0, b) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1), a\ b)$
$= \epsilon$ - closure $(\delta(q_1, b) \cup (q_2, b) \cup (q_0, b))$
$= \epsilon$ - closure $(\phi \cup \phi \cup q_3)$
$= \{q_3\}$.

$\delta'(q_0, c) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_0), c)$.
$= \epsilon$ - closure $(\delta(q_0, c) \cup (q_1, c) \cup (q_2, c))$
$= \epsilon$ - closure $(\phi \cup q_2 \cup \phi)$
$= \{q_2\}$

$\delta'(q_0, d) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_0)\ d)$
$= \epsilon$ - closure $(\phi \cup \phi \cup q_2)$
$= \{q_2\}$

$\delta'(q_1, a) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1), a)$.
$= \epsilon$ - closure $(q_2 \cup \phi)$
$= \{q_0, q_1, q_2\}$

$\delta'(q_1, b) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1)\ b)$
$= \epsilon$ - closure $(q_3 \cup \phi)$
$= \{q_3\}$

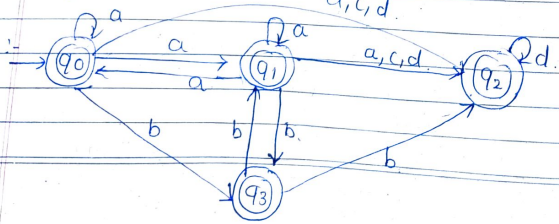
$\delta'(q_1, c) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1)\ c)$
$= \epsilon$ - closure $(q_2 \cup \phi)$
$= \{q_2\}$

$\delta'(q_1, d) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_1)d)$
$= \epsilon$ - closure $(\phi \cup q_2)$
$= \{q_2\}$

$\delta'(q_2, a) = \epsilon$ - closure $(\delta(\epsilon\text{- closure } q_2), a)$.
$= \epsilon$ - closure $(\delta \emptyset)$
$= \{\emptyset\}$
$\delta'(q_2, b) = \epsilon$ - closure $(\delta(\epsilon\text{- closure } q_2), b)$.
$= \epsilon$ - closure $(\delta \emptyset)$
$= \{\emptyset\}$
$\delta'(q_2, c) = \{\emptyset\}$
$\delta'(q_2, d) = \epsilon$ - closure $(\delta(q_2))$
$= \{q_2\}$
$\delta'(q_3, a) = \epsilon$ - closure $(\delta(\epsilon\text{- closure } q_3), a)$.
$= \epsilon$ - closure $(\delta\emptyset)$
$= \{\emptyset\}$
$\delta'(q_3, b) = \epsilon$ - closure $(\delta\emptyset)$. $(\delta(q_1, b)$.
$= \epsilon$ - closure (q_1).
$= \{q_1, q_2\}$
$\delta'(q_3, c) = \epsilon$ - closure $(\delta(q_3, c))$
$= \{\emptyset\}$
$\delta'(q_3, d) = \{\emptyset\}$

Logic :- i) While converting ϵ-NFA to NFA the total number of state remain same & initial state is also same i.e. $q_0' = q_0$.
ii) Every state whose ϵ - closure contains the final state of ϵ - NFA is the final state of NFA.

Step 5 :-

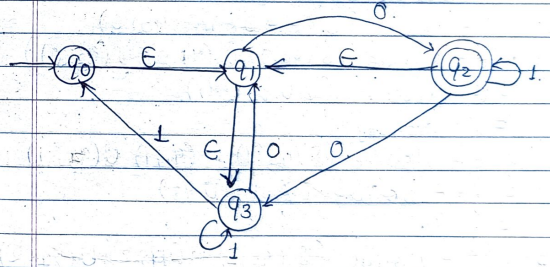


Step 6 :- Tupples of NFA.
$M = \langle Q', \Sigma, \delta', q_0', F' \rangle$
$Q' = \{q_0, q_1, q_2\}$.
$\Sigma = \{a, b, c, d\}$
$\delta' : Q \times \Sigma \to 2^{Q'}$
$q_0' = \{q_0\}$.
$F' = \{q_0, q_1, q_2, q_3\}$.

Step 7 :- Transmission table of NFA

Q	a	b	c	d
$\to q_0$	$[q_0 q_1 q_2]$	$[q_3]$	$[q_2]$	$[q_2]$
q_1	$[q_0, q_1 q_2]$	$[q_3]$	$[q_2]$	$[q_2]$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$[q_2]$
q_3	$\emptyset$	$[q_1 q_2]$	$\emptyset$	$\emptyset$

(8)



Step 1 :- NFA has 5 tuples.
$M = \langle Q, \Sigma, \delta, q_0, F \rangle$
$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{0, 1\}$
$\delta = Q \times \Sigma \to Q$.
$q_0 = \{q_0\}$
$F = \{q_2\}$.

Step 2:- Determine ϵ - closure

ϵ - closure of $(q_0) = \{q_0, q_1, q_3\}$

$(q_1) = \{q_1, q_3\}$

$(q_2) = \{q_1, q_2, q_3\}$

$(q_3) = \{q_3\}$

Step 3:- Let $M = <Q', \Sigma, \delta', q_0', F'>$ are the tupples of NFA

Q': Finite set of all state.

Σ : Finite set of i/p symbol.

δ': Transition function of NFA.

q_0': Initial state of NFA.

F': Final state of NFA

Step 4: Construction of δ'

$\delta'(q, x) = \epsilon$ - closure$(\delta(\epsilon$ - closure $q), x)$

$\delta'(q_0, 0) = \epsilon$ - closure $(\delta(\epsilon$ - closure $q_0), 0)$.

$= \epsilon$ - closure $(\delta(q_0, 0) \cup (q_1, 0) \cup (q_3, 0))$

$= \epsilon$ - closure $(\phi \cup q_2 \cup q_1)$

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

$\delta'(q_0, 1) = \epsilon$ - closure $(\delta(q_0, 1) \cup (q_1, 1) \cup (q_3, 1)$

$= \epsilon$ - closure $(\phi \cup \phi \cup q_3)$.

$= \{q_3\}$.

~~$\delta'(q_0, 0) = \epsilon$ - closure $(\delta(q_0, 2) \cup (q_1, 2) \cup (q_2, 2))$~~

~~$= \epsilon$ - closure~~

$\delta'(q_1, 0) = \epsilon$ - closure $(\delta(q_1, 0) \cup (q_3, 0))$.

$= \epsilon$ - closure $(q_2 \cup q_1)$

$= \{q_1, q_2, q_3\}$.

$\delta'(q_1, 1) = \epsilon$ - closure $(\delta(q_1, 1) \cup (q_3, 1))$

$= \epsilon$ - closure $(\phi \cup q_3)$

$= \{q_3\}$

$\delta'(q_2, 0) = \epsilon$ - closure $(\delta(q_1, 0) \cup (q_2, 0) \cup (q_3, 0))$.

$= \epsilon$ - closure $(q_2 \cup q_3 \cup q_1)$.

$= \{q_1, q_2, q_3\}$.

$\delta'(q_2, 1) = \epsilon$ - closure $(\delta(q_1, 1) \cup (q_2, 1) \cup (q_3, 1))$.

$= \epsilon$ - closure $(\phi \cup q_2 \cup q_3)$

$= \{q_1, q_2, q_3\}$.

$\delta'(q_3, 0) = \epsilon$ - closure $(\delta(q_3, 0))$.

$= \epsilon$ - closure (q_1).

$= \{q_1, q_3\}$.

$\delta'(q_3, 1) = \epsilon$ - closure $(\delta(q_3, 1))$

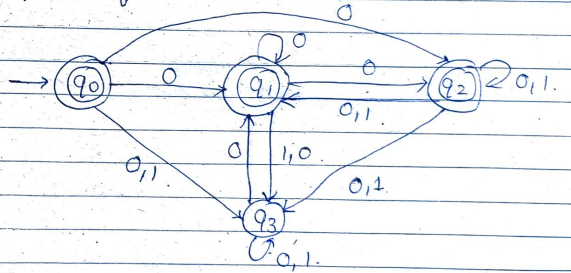
$= \epsilon$ - closure (q_3)

$= \{q_3\}$.

Logic :- i] While converting ϵ-NFA to NFA the total number of state remain same & initial state is also same i.e. $q_0' = q_0$

ii] Every state whose ϵ - closure contains the final state of ϵ - NFA is the final state of NFA.

Step 5:-



Step 6:- Tupples of NFA

$M = <Q', \Sigma, \delta', q_0', F'>$

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

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

$\delta' = Q \times \Sigma \to 2^{Q'}$

$q_0' = \{q_0\}$

$F' = \{q_0, q_1, q_2\}$.

Step 7:- Transmission table of NFA

δ	0	1
→ q_0	$[q_1 q_2 q_3]$	$[q_3]$
q_1	$[q_1 q_2 q_3]$	$[q_3]$
q_2	$[q_1 q_2 q_3]$	$[q_1 q_2 q_3]$
q_3	$[q_1 q_3]$	$[q_3]$

Unit 3:

Definition of Regular Expression.

- The mathematical or algebraic representation of Regular language in term of expression is called as Regular expression.

- It is an expression which is generated or constructed over input symbol from the alphabet (Σ) by using regular expression operators
  '*'
  '.'
  '|'
  '+'

Sr No.	Regular language	Regular expression
1.	$L = \{\}$	$r = 0$
2.	$L = \{\epsilon\}$	$r = \epsilon$
3. $\|w\| = 1$	$L = \{a, b\}$	$r = a + b$
4. $\|w\| = 2$	$L = \{aa, bb, ab, ba\}$	$r = aa + bb + ab + ba$ $= (a+b)(a+b)$
5. $L = \{a^n \mid n >= 0\}$	$L = \{\epsilon, a, aa, aaa, aaaa, \ldots\}$	$r = a^*$
6. $L = \{a^n \mid n >= 1\}$	$L = \{a, aa, aaa, aaaa, \ldots\}$	$r = a^+$
7. $L = \{a^n b^n \mid n >= 1\}$	$L = \{ab, abab, \ldots\}$	$r = (a+b)^*$

Q. Construct a regular expression over input symbol $\Sigma = \{a, b\}$ where

i) Every string starts with ab [abx]
L = {ab, aba, abb, abab, abba, abbbaaa, ...}
Regular expression :
$r = ab\,(a+b)^*$

ii) Every string ends with ba [xba]
$r = (a+b)^* ba$

iii) Every string contains substring aa [xaax]
$r = (a+b)^* aa (a+b)^*$

iv) Every string whose length is exactly three $|\omega| = 3$
$r = (a+b)(a+b)(a+b)$

v) Every string whose length is ~~at least~~ three $|\omega| >= 3$
$r = (a+b)(a+b)(a+b)(a+b)^*$.

vi) At most three $|\omega| <= 3$.
$r = (a+b+\epsilon)(a+b+\epsilon)(a+b+\epsilon)$

vii) Every string starts & ends with a -
$r = a + a.(a+b)^* a$

ix) Every string starts & ends with same symbol. (axa, bxb).
$r = (a+b+\epsilon) + a(a+b)^* a + b(a+b)^* b$

x. Every string start and end with different symbol (axb, bxa).
$r = (a(a+b)^* b) + (b(a+b)^* a)$

xi. length of string is even $|\omega| \bmod 2 = 0$.
$r = [(a+b)(a+b)]^*$

xii. length of string is odd.
$r = [(a+b)(a+b)]^* (a+b)$

xiii. $|\omega| \bmod 3 = 0$, length of string is divisible by 3.
$r = [(a+b)(a+b)(a+b)]^*$

xiv. $|\omega|_a = 2$.
$r = b^* a b^* a b^*$

xv. $|\omega| <= 2$ (at most two a)
$r = b^* (a+\epsilon) b^* (a+\epsilon) b^*$

xvi. $|\omega| >= 3$ (at least 3)
$r = b^* a b^* a b^* a (a+b)^*$.

$R = Q + RP$

$q_1 = \epsilon + q_1 0 = \epsilon 0^*$

CONVERSION OF FINITE AUTOMATA TO REGULAR EXPRESSION OR Equivalence between finite automata and regular expression.

1. Arden's theorem or Arden's lemma is used for converting DFA & NFA into equivalent regular expression
2. ϵ-NFA is not used in these mechanism because Arden's theorem is application to only FA and NFA.
   ϵ - transition are not allow.

CLOSURE PROPERTIES & IDENTITIES OF REGULAR SATE.

Properties

1. $\phi + R = R$
2. $\epsilon . R = R . \epsilon = R$
3. $R + R = R$
4. $R^* + R^* = R^*$
5. $(R^*)^* = R^*$
6. $R^* . R^+ = R^+$
7. $\epsilon + R . R^* = R^*$
8. $(R_1 + R_2)^* = (R_1^* + R_2^*)^*$
   $= (R_1^* + R_2^*)^*$
   $= (R_1 + R_2^*)^*$
   $= (R_1^* . R_2^*)^*$

ARDEN'S LEMMA.

Let P, Q & R are the three regular expression and having form

$R = Q + RP$

then according to Arden's lemma its unique solution is $R = QP^*$.

$R = Q + RP$

$R = Q + QP^* P$

$R = Q (\epsilon + P^* P)$

$R = QP^*$

Steps for Solving the problem.

Step 1: From the given finite automata construct the regular expression for each and every state by observing incoming arrows.

Step: While writing the regular expression of initial state and its related must be added in the expression.

Step 2: Solve the regular expression for final step state & its related state by applying Arden's Theorem

Step 3: Apply Arden's lemma & simplify the expression in the form $R = Q + RP$ by using substitution rule and then apply its theorem

$R = QP^*$

Define : TURING MACHINE.

i. The mathematical representation of "Recursive Enumerable language is called as 'TURING MACHINE'.

ii. Turing machine is also defined as FA with infinite tape with reading and writing capabilities is also Turing machine.

iii) Turing machine contain all the properties therefore it act as Transducer

iv) Turing machine has capabilities.
a. Language Accept
b. Language Generator
c. I/O device (Transducer).

With all these capabilities Turing machine is also called Abstract Model of Computer

TURING MACHINE is defined with 7 tuples.
$\langle Q, \Sigma, \delta, q_0, \Gamma, F, B \rangle$
where Q: Finite set of all state.
Σ: Finite set of i/p symbol
δ: Transition fun^n $Q^* \Gamma \rightarrow Q^* \Gamma^* \{L, R\}$
q_0: Initial state
Γ: Finite set of tape symbol.
F: Set of final state
B: Blank symbol

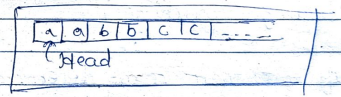
Block diagram of Turing Machine
Turing machine consist of three components.
1. Imifinite tape
2. Read - write header
3. FCU

1. Infinite tape:
A. Infinite tape is divided into number of cells, where each cell hold exactly one input symbol

B. This infinite tape is further divided into 2 types
a. Two - way infinite tape
b. One - way infinite tape
Depends upon the logic and condition we can use the tape.

C. One-way infinite tape is extended upto infinite number of cell in only one direction and that direction is "Right Direction."

the input string is present in the tape from left hand side.



2. Read - Write header.
a. The header reader or scan the i/p symbol form the infinite tape and give that i/p symbol to finite control unit. The finite control unit apply an appropriate and modified that i/p symbol in the tape and perform write or left direction over the input tape.

b. The read - write header is always pointing only one cell at a time

c. After scanning the i/p symbol from infinite tape the header can move left or right direction exactly one cell along with the tape.

d. The header moment is bidirectional with reading & writing capabilities.

Important points.

1. The capabilities of Turing machine is equal to capabilities of computer.

2. Turing machine can be constructed in two mode.
   i. Deterministic Turing Machine (DTM).
   ii. Non-Deterministic Turing Machine &

3. By default a Turing machine is constructed deterministic Turing machine.

4. DTM is more effisient than non-deterministic Turing machine.

5. Turing machine is more powerful than FA & PDA because it accept regular language, context free language, context sensitive language and recursive enumerable language.

3. Finite Control Unit (FCU)

a. It is an imp part of architecture of Turing machine where the processing of i/p string is done.

b. FCU perform all the computation just like a CPU.

c. All the business logic are important in FCU.

Representation of Turing Machine.

- Turing machine can representation in two different ways

1. Transition diagram.
2. Transition table

Instantaneous Description.

The next moment of Turing machine is depend on two entities.

a. Current state.
b. Current Input Symbol (Tape Symbol)

Halt (Termination)

1. The state from which transition is not defined for the i/p symbol is also HALT.
2. Halt must occur on final state then the string is accepted by Turing machine & if halt occurs on non-final state then the string is rejected by Turing machine.
3. The halting problem of Turing machine is undecidable no appropriate algorithm is present.

Instantaneous Description

$$\delta(q_i, d) = (q_j, x, L/R)$$

Conversion of Regular Expression to Finite Automata

Here we have to study two methods

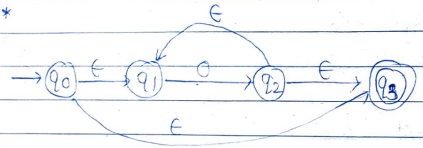
1. Method by Synthesis
2. Method by Decomposition

1 METHOD BY SYNTHESIS

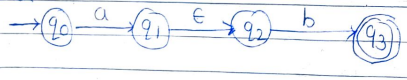
- This method is used for the conversion of regular expression to ϵ-NFA.
- Since we have three regular operators
  - i. *
  - ii. +
  - iii. .

FORMULAE

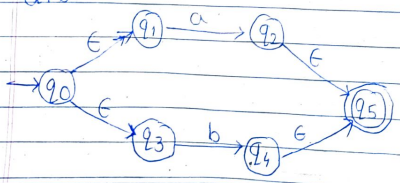
- a^*



- a.b



- a+b

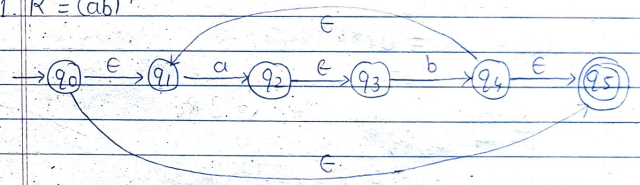


Basic Examples

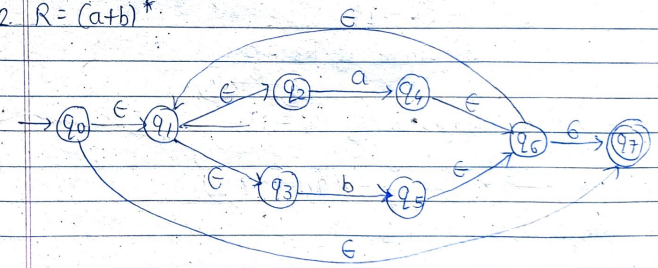
1. $R = (ab)^*$
2. $R = (a+b)^*$
3. $R = (a^*.b^*)$
4. $R = (a^*+b^*)$

Note :- While converting regular expression into ϵ NFA divide the given expression into smallest part and apply the method of synthesis to each part of regular expression and in final transition diagram combine all parts of regular expression

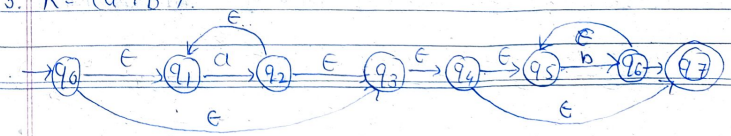
1. $R = (ab)^*$



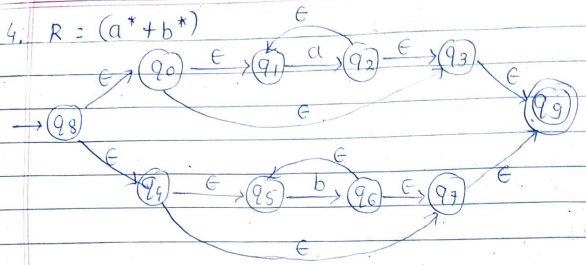
2. $R = (a+b)^*$



3. $R = (a^*.b^*)$



4. $R = (a^* + b^*)$



Construct NFA. ϵ-move for the R.E. 1] $(01^* + 1)$ 2] $1 + (0+1)^* 1^*$

⇒ Let R be the given R.E

$R = (01^* + 1)$.

Let R_1 & R_2 be the parts of the given R.E.

$R = R_1 + R_2$

where $R_1 = 01^*$.

$R_2 = 1$.

R_1 is further divided in R_3 & R_4

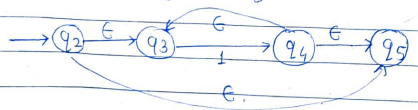
$R_1 \rightarrow R_3, R_4$

Where $R_3 \rightarrow 0$

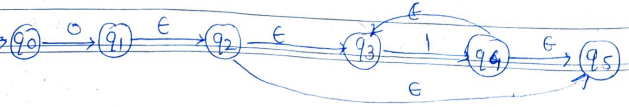
$R_4 \rightarrow 1^*$

Transition diagram of $R_3 = 0$

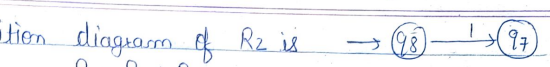
Transition diagram of $R_4 = 1^*$.



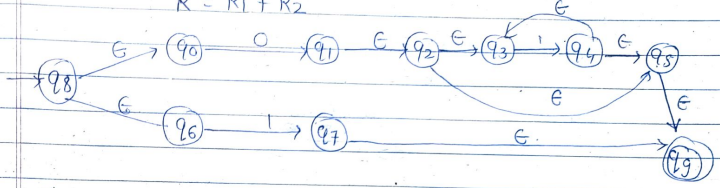
Transition diagram of R_1 is $R_1 \rightarrow R_3 . R_4$



Transition diagram of R_2 is



$R = R_1 + R_2$



Total state = 10

ϵ-NFA is defined with 5 tuples

$M = \langle Q, \Sigma, \delta, q_0, F \rangle$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$.

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

$\delta = Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$

$q_0 = \{q_8\}$

$F = \{q_9\}$

2] $1 + (0+1)^* 1^*$

⇒ Let R be the given R.E

$R = 1 + (0+1)^* 1^*$

Let R_1 & R_2 be the parts of the given R.E.

$R = R_1 + R_2$

Where $R_1 = 1$

$R_2 = (0+1)^* 1^*$

Transition diagram of R_1

R_2 is further divided into R_3 and R_4.

$R_2 = R_3 . R_4$

Where $R_3 = (0+1)^*$.

$R_4 = 1^*$.

Since R_3 is further divided into R_5 & R_6.

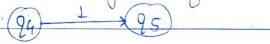
$R_3 = (R_5 + R_6)^*$

Where $R_5 = 0$

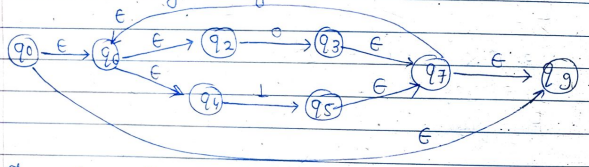
$R_6 = 1$

Transition diagram of R_5

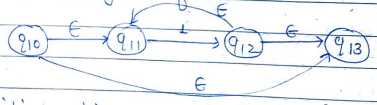
Transition diagram of R_6



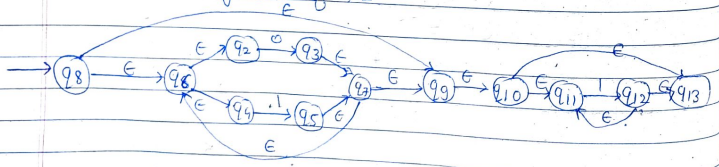
Transition diagram of $R_3 = (R_5 + R_6)^*$.



Transition diagram of R_4

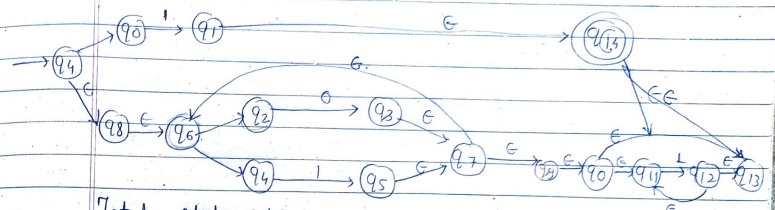


Transition diagram of R_2



Now final transition diagram of given R.E. is

$R = R_1 + R_2$



Total state = 16.

ϵ - NFA is defined with 5 tupples

$M = \langle Q, \Sigma, \delta, q_0, F \rangle$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}\}$

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

$\delta = Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$

$q_0 = \{q_4\}$

$F = \{q_{15}\}$

[Q 2] Construct NFA with ϵ - transition for

$L = (a+b)^* \; b \; (a+bb)^*$.

= Since L be the given R.E.

$L = (a+b)^* \; b \; (a+bb)^*$

Let L is divided into 3 part ie R_1, R_2 & R_3

$L = R_1 R_2 R_3$

Where

$R_1 = (a+b)^*$.

$R_2 = b$.

& $R_3 = (a+bb)^*$.

R_1 is further divided into two parts R_4 & R_5

$R_1 = (R_4 + R_5)^*$.

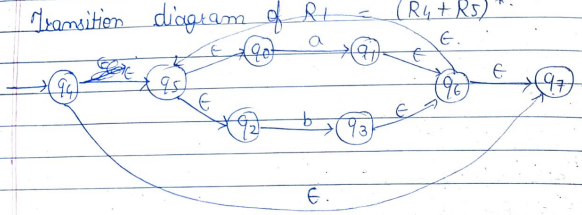
Where $R_4 = a$

$R_5 = b$.

Transition diagram of R_4 $(q_0) \xrightarrow{a} (q_1)$

Transition diagram of R_5 $(q_2) \xrightarrow{b} (q_3)$

Transition diagram of $R_1 = (R_4 + R_5)^*$.



Transition diagram of R_2.

$(q_8) \xrightarrow{b} (q_9)$

Since R_3 is further divided into two parts.

$R_3 = (R_6 + R_7)^*$

Where $R_6 = a$.

$R_7 = bb$

Transition diagram of R_6.

$(q_{10}) \xrightarrow{a} (q_{11})$

R_7 is further divided into two parts i.e

R_8 & R_9.

$R_7 = R_8 R_9$

Where $R_8 = b$.

$R_9 = b$.

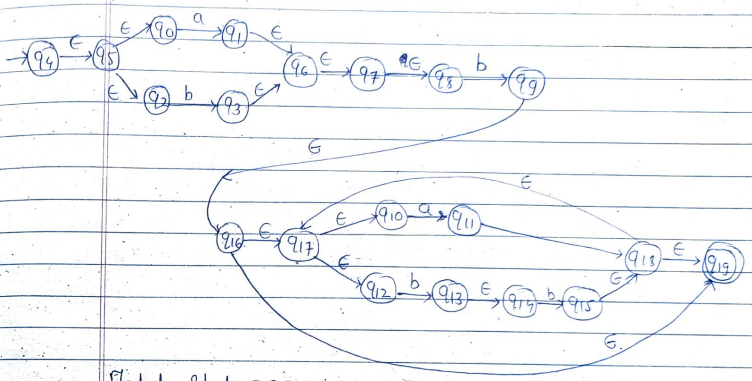
Transition diagram of R_8 $(q_{12}) \xrightarrow{b} (q_{13})$

Transition diagram of R_9 $(q_{14}) \xrightarrow{b} (q_{15})$

Transition diagram of $R_7 = R_8 R_9$.

$(q_{12}) \xrightarrow{b} (q_{13}) \xrightarrow{\epsilon} (q_{14}) \xrightarrow{b} (q_{15})$

Transition diagram of $R_3 = (R_6 + R_7)^*$



Total State = 20

Now the final transition diagram of R.E is $L = R_1 . R_2 . R_3$

ϵ - NFA is defined with 5 tuples

$M = \langle Q, \Sigma, \delta, q_0, F \rangle$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}, q_{18}, q_{19}\}$.

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

$\delta = Q \times (\Sigma \cup \epsilon) \longrightarrow 2^Q$

$q_0 = \{q_4\}$

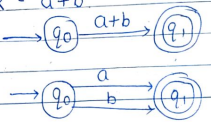
$F = \{q_{19}\}$

Q3 $[00 + 11 + (11 + 0)^* + (1 + 00)^*]^*$

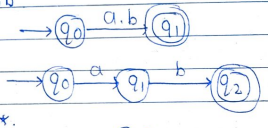
Q.4 $(ab)^* ba^* b(a^* b^*) + a^* b^* (aa^* + b^* b)$

2] METHOD BY DECOMPOSITION.

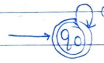
i] $R = a+b$



ii] $R = a.b$

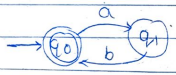


iii] $R = a^*$

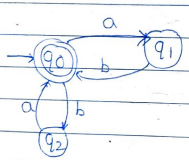


iv] $R = (a+b)^*$

v] $R = (a.b)^*$

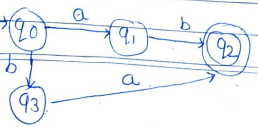


vi] $(a.b + b.a)^*$

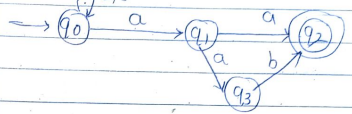


vii] $R = t_1 t_2{}^* t_3$

viii] $R = ab + ba$

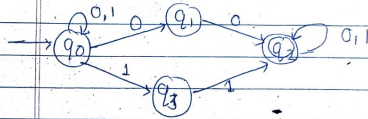


ix] $R = (a+b)^* a.(ab+a)$.



Q Construct FA for RE

i] $(0+1)^* (00+11)(0+1)^*$



For NFA $\Rightarrow$ $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

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

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

$\delta = Q \times \Sigma \longrightarrow 2^Q$

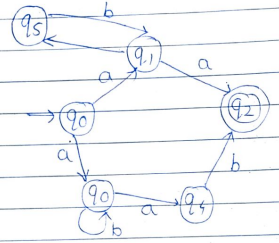
$q_0 = \{q_0\}$

$F = \{q_2\}$

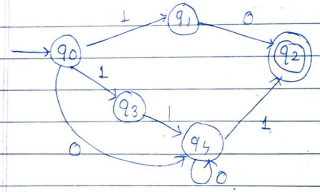
Transition table

Q/Σ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_3\}$	$\{\phi\}$
(q_2)	$\{q_2\}$	$\{q_2\}$
q_3	ϕ	$\{q_2\}$

iii) $R = a(ab)^* a + ab^* ab$



iii) $10 + (0+11)0^* 1$



$M = \langle Q, \Sigma, \delta, q_0, F \rangle$

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$q_0 = \{q_0\}$

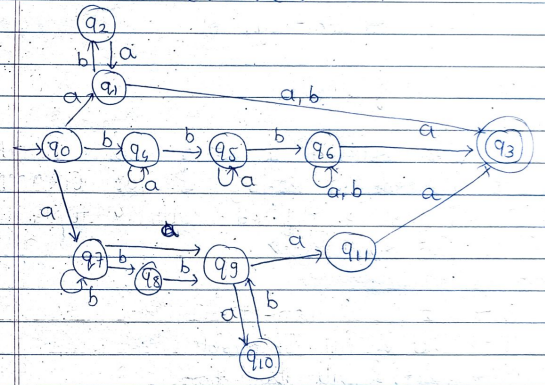
$F = \{q_2\}$

Q/Σ	0	1
→ q_0	$\{q_4\}$	$\{q_1, q_3\}$
q_1	$\{q_2\}$	$\emptyset$
(*q_2)	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\{q_4\}$
q_4	$\{q_4\}$	$\{q_2\}$

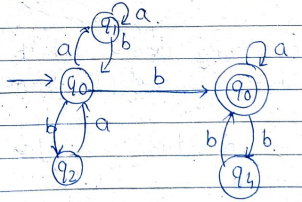
DFA $= M' = \langle Q', \Sigma', \delta', q_0', F' \rangle$

Q/Σ	0	1
→ $[q_0]$	$[q_4]$	$[q_1, q_3]$
$[q_1, q_3]$	$[q_2]$	$[q_4]$
$[q_4]$	$[q_4]$	$[q_2]$
(*$[q_2]$)	$\emptyset$	$\emptyset$

Q. $R = a(ba)^*(a+b) + ba^* ba^* b(a+b)^* a + ab^*(a+bb)(ab)^* aa$



Q. $R = [a^* b + (ba)^*]^* \; b(a+bb)^*$



Capital letter: Set of Non-terminals

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

$|w| = 2$

$S \Rightarrow aa/ab/ba/bb$

$S \rightarrow AA$

$A \rightarrow a/b$

$V = \{S, A\}$

$T = \{a, b\}$

$P = S \rightarrow AA$

$A \rightarrow a/b$

$S = S$ → Start symbol

GRAMMER (Set of Rules).

- Definition.

This collection of set of rules which are used to generate a string is c/as GRAMMAR. Every Grammer is defined with 4 tuples.

$G = \langle V, T, P, S \rangle$

Where

V → Set of all non-terminals.
T → Set of all Terminals
P → Set of all Production.
S → Set Symbol

Note:-

i. Grammer is an generating device which generate the string with the help of Production (Set of rules)

ii. The process of deriving a string is k/as Derivation. The geometrical representation of a derivation is c/as Parse tree & Derivation Tree

iii. Every grammar contain only one start symbol.

iv. Graphically, we can represent the derivation of string by using derivation tree or Parse tree.

v. Every Grammar generates only one language but a language can be generated by more than one form of Grammar.

vi. G is the Grammar the L(G) is the set of all string generated by the Grammar c/as language generated by the Grammar.

vii. Grammar is further divided into two types: 1. RLG = Right linear Grammar
2. LLG = left linear Grammar.

Recursive Grammar

- The grammar 'G' is said to be recursive if & only if in at least one product has the same non-terminal on the left hand side and right hand side than such type of Grammar is c/as Recursive Grammar.

- Ex $S \rightarrow AB \mid BA$
$A \rightarrow Aa/a$
$b \rightarrow bB \mid b$
$S \rightarrow aS/\epsilon$
$S \rightarrow aS \mid bS \mid \epsilon$

left linear Grammar (LLG)

If a regular Grammar G having the production in the form

$\alpha \rightarrow \beta w \mid w$ where

α & $\beta \in V$ (single non terminal)

& $w \in T^*$ (Arbitary string of terminals).
This such type of regular grammar is Cas Left linear Grammar.

Ex:- $S \to Sa \mid Sb \mid Aab$
$A \to Aa \mid Ab \mid \epsilon$

Right linear Grammar (RLG).
If a regular Grammar Gr having the production in the form
$\alpha \to w\beta \mid w$ where
α & $\beta \in V$ (Single non terminal)
& $w \in T^*$ (arbitary string of terminals)
This such type of regular grammar is Cas Right linear Grammar
Ex:-
$S \to aS \mid bS \mid abA$.
$A \to aA \mid bA \mid \epsilon$

~~Regular Grammar (Type 3)~~
- ~~A grammar~~
Non-Recursive Grammar.
- The Grammar 'Gr' is said to be recursive if & only if in at least one product has the same non-terminal on the left hand side & right hand side than such type of Grammar is Cas Recursive Grammar.

Ex. $S \to AB \mid BA$
$A \to a$.
$B \to b$.

- Every non-recursive grammar represent the finite language

Regular Grammar (Type 3)

- A grammar Gr is said to be a regular grammar if and only if it having the production in the form
$\alpha \to \beta$
Where $|\alpha| <= |\beta|$ at
$\alpha \in V$ [single non-terminal] &
$\beta \to w\beta \mid w$.
or $\beta \to \beta w \mid w$
Where $\beta \in V$ [single non-terminal].
And $w \in T^*$ [Arbitary string of terminals].
Then such type of grammar is Cas Regular Grammar.

Q] Start with a
ax
R.E. $= a(a+b)^*$.
L = {a, aa, ab, aaa, ---}.

$S \to Aa$
$A \to aA \mid bA \mid \epsilon$.

Substring ab

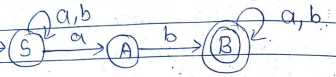
R.E. = $(a+b)^* ab (a+b)^*$

L = {ab, aba, aaba, aabb, ...}

S → aS | bS | aA

A → bB

B → aB | bB | ∈



Q. S → aS | bS | abA

A → aA | bA | ∈.

(1)

6) Flowlines / Flowarrows - It indicates the exact sequence in which information instructions are executed. Arrows represent the direction of flowcontrol & relationship among diff symbols of flowchart.

Memory allocation

Static & Dynamic.

Space complexity - 1) Instruction space 2) Data space
3) Environment space.

V.I.Q. Performance analysis of algorithm ?

Different way of analysing algorithm.

Two criteria

Unit - 2. 29/03/23

S.P An assembler is a program that accepts as input an assembly lang program & produces its machine lang equivalent along with info for the loader.

Assembly lang program → [Assembler] → Machine language & other info for the loader.

[Databases]

Fun^n of an assembler

General Design procedure of an assembler.

- There are 6 steps that should be followed during the design of an assembler.

1. Specify the problem.
2. Specify data structures
3. Define format of data structures
4. Specify algorithm
5. Loop for modularity ie. capability of one program to be sub-divided into independent programming units.

6. Repeat steps 1 to 5 on modules.

1) Specify the problem.

① USING - Using is a pseudo op that indicates to the assembler which general register to use as a base register & what its contents will be. This is necessary because no special registers are set aside for addressing thus the programmer must informed the assembler which registers to use & how to use them.

② BALR - BALR is the instruct^n to the comp to load a register with the next address & branch to the address in the second field.

③ START - Start is a pseudo op that tells the assembler where the program begins & allows the user to give a name to the program.

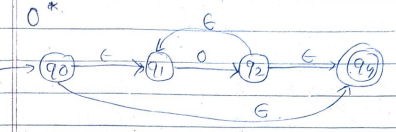
④ END - End is a pseudo op that tells the assembler that the last card of the program has been reached.

⑤ DS - Define storage - DS is a pseudo op that instructs the assembler the size of the memory allotted to the label.

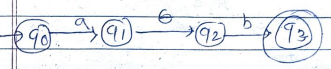
⑥ DC (Define const) - DC is a pseudo op assigns a constant value & memory storage to the label.

		First Pass			Second Pass	
		Relative Add.		Machine Instruct^n	Relative Address	Machine Instruction
TEST	START 0					
USING	*, 15.					
	L 1, FIVE	0	L	1, -(0,15)	0	L 1, 16(0,15)
	A 1, FOUR	4	A	1, -(0,15)	4	A 1, 12(0,15)
	ST 1, TEMP	8	ST	1, -(0,15)	8	ST 1, 20(0,15)
FOUR	DC F'4'	12	4		12	4
FIVE	DC F'5'	16	5		16	5
TEMP	DS 1F	20	-		20	-
	END					

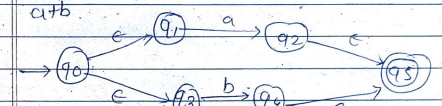
- 0^*



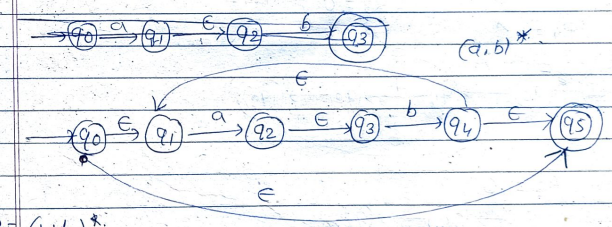
- a.b



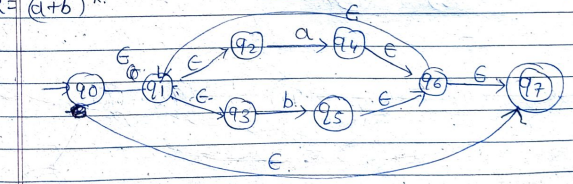
- a+b



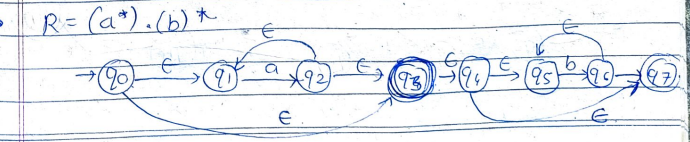
- $R.(a|b)^*$



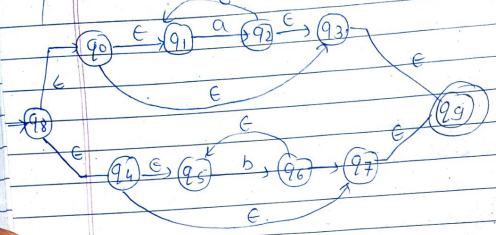
- $R = (a+b)^*$



- $R = (a^*).(b)^*$



$R = (a^* + b^*)$



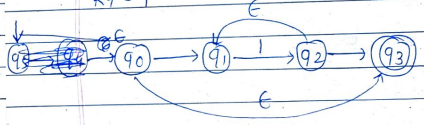
Note: while converting it

1] $(01^* + 1)$

R_1 R_1 $\quad R_1 = 01^*$.

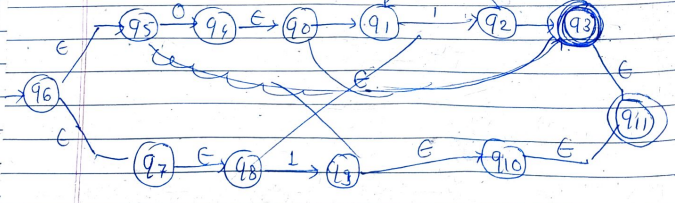
$R_3 = 0 \quad R_4 = 1^*$

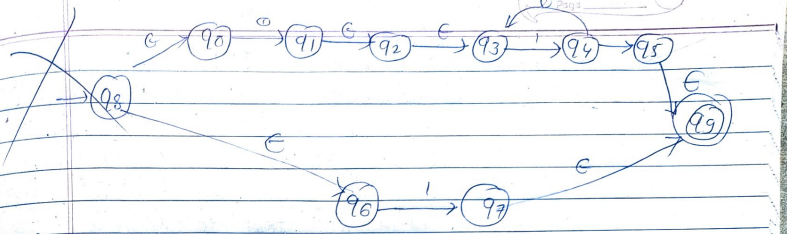
$R_4 = 1^*$



$R_3 = 0$

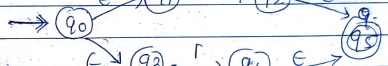
$R_3 . R_4$:-





[Q2] $1 + (0+1)^* 1^*$.

R_1 $\quad R_2$

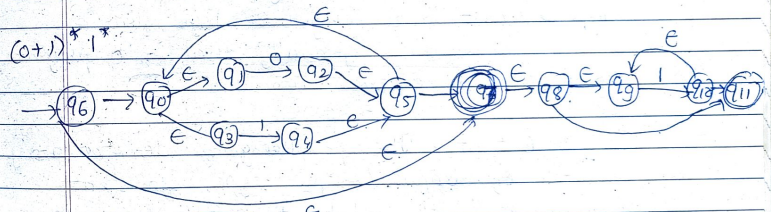


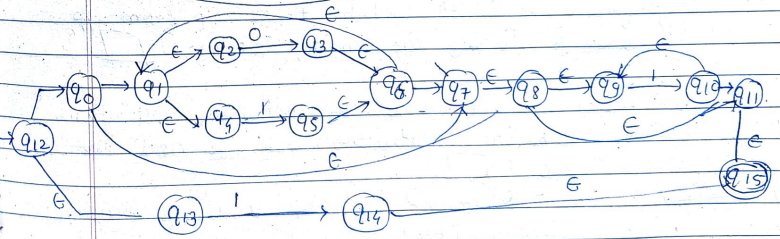
$(0+1)$

$R_2 = (0+1)^* 1^*$

R_3 $\quad R_4$

$(0+1)^* 1^*$.





Flowchart - A graphical representation of any Program

1) Terminal - It indicates start & stop
2) Input and o/p
3) Condition ◇ 3) Processing. □
5) Connector 6) Flowlines

Flowchart is the graphical representation of an algorithm. Programmer often use it as a programming planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of info & processing

Basic symbol

1) Terminal - It indicates as a oval symbol with start, stop & halt in a program's logic flow. A pause or halt is generally used in a program logic under some error condition. ⬭

Input/Output - The parallelogram denotes any function of input/output type. In that program instruct^n which takes i/p from i/p device & display o/p on o/p device. ▱

3) Processing - This box represents the arithmetic instructions. All arithmetic instruction such as add^n, sub^n mult^n etc. □

4) ~~Condition~~/ Decision - Diamond symbol represents decision point. decision based on such yes or no que or true or false.

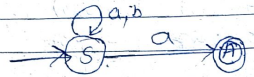
5) Connector - This ⊙ symbol whenever flowchart becomes complex & its spread over more page. It is useful to use connectors to avoid confusion in any complex program. ⊙

			FIRST PASS		
			~~Relative~~ Address	~~Relative~~ Machine Instruct^n	
MAIN	START	10			
	USING	*,14			
	L	1,ONE	10	L	1,-(0,14)
	A	1,TWO	14	A	1,-(0,14)
	ST	1,DATA	18	ST	1,-(0,14)
	L	2,THREE	22	L	
	S	2,FOUR			
	ST	2,TEMP			
ONE	DC	F'1'			
TWO	DC	F'2'			
THREE	DC	F'3'			
FOUR	DC	F'4'			
DATA	DS	1F			
TEMP	DS	1F			

Q. Ends with a.

RE = $(a+b)^* a$

L = {a, ba, aa, aaa, aaaa ....}



S →

Two types of analysis: Lexical & Syntax Analysis

Syntax → Intermediate Code → Code Optimization → Code generation

Error handling | Table Management | Two block

(1) L = {$a^n | n >= 0$}

L = {ϵ, a, aa, aaa, aaaa, ...}

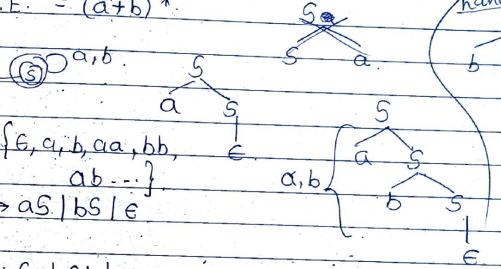
RE = a^*

S → aS | ϵ (RRG)

Or

S → Sa | ϵ (LRG)

(2) R.E. = $(a+b)^*$



L = {ϵ, a, b, aa, bb, ab ...}

S → aS | bS | ϵ

Or

S → Sa | Sb | ϵ

Conversion of right grammar to left linear grammar

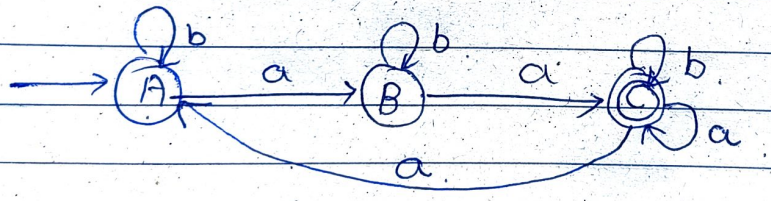
Step 1: Interchange initial & final state of FA.
Step 2: Reverse dir^n of transition (the reverse of loop is loop.

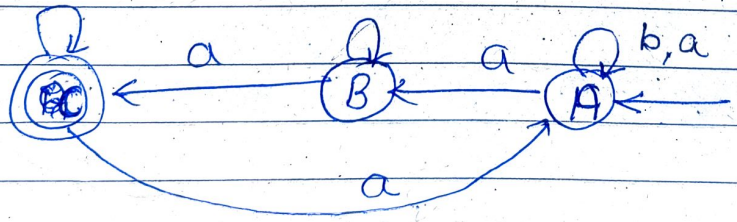
RLG: $S \rightarrow aA|a$
$A \rightarrow aA|bA|ab$ or $S \rightarrow aA$

LLG: $S \rightarrow Sa|Sb|a$
$A \rightarrow \epsilon$
or $S \rightarrow Sa|Sb|a$

Converting a RLG into finite automata.

$A \rightarrow aB|bA|b$
$B \rightarrow aC|bB$
$C \rightarrow aA|bC|a$.

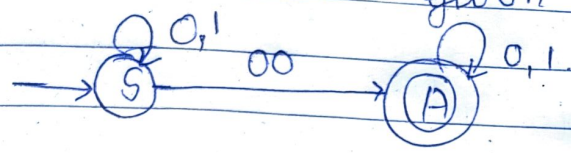




Construct an RLG or LLG for the given regular expression $(0+1)^*00(0+1)^*$.

$(0+1)^*00(0+1)^*$.

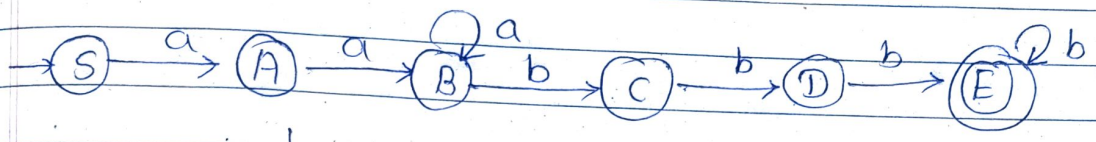
Let 'R' be the given R.E.



Right linear grammar → A REgular grammar & having the production in the form of $\alpha \to \omega\beta / \omega$ where, α & $\beta \in V$ (Single non

Find LLG & RLG for the Regular language
$L = \{a^n b^m \mid n >= 2, m >= 3\}$.

$L = \{aabbb$

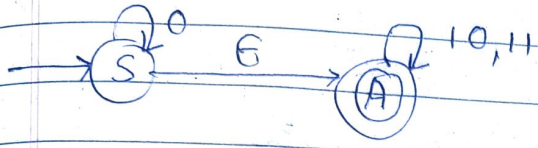


$S \rightarrow aA$	$C \rightarrow bD$
$A \rightarrow aB$	$D \rightarrow bE$
$B \rightarrow bC \mid a$	$E \rightarrow b$

Construct a RLG & LLG for the given Regular expression $0^*[1(0+1)]^*$.

$0^*[1(0+1)]^*$.

$0^*[10+11]^*$.



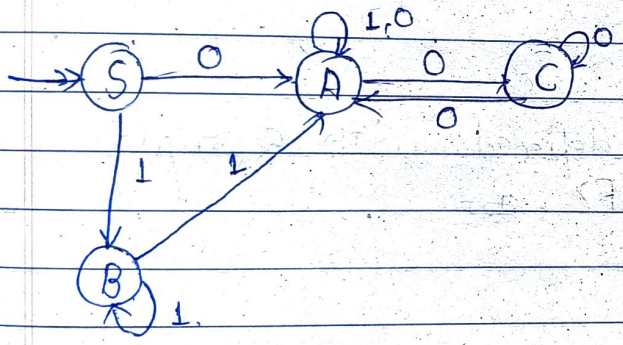
Given an equivalent LLG for the following RLG

$S \rightarrow 0A|1B$.

$A \rightarrow 0C|1A|0$.

$B \rightarrow 1B|1A|1$.

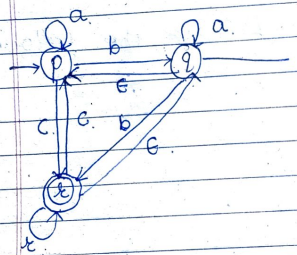
$C \rightarrow 0|0A$



Q.1 Consider ϵ-NFA.

Q/Σ	ϵ	a	b	c
→p	--	p	q	r
q	p	q	r	ϕ
*r	q	r	ϕ	p

i Compute ϵ-closure of each state



Step 1: ϵ-NFA is defined with 5 tuples.
M = <Q, Σ, δ, q0, F>
Q = {p, q, r}
Σ = {a, b, c}
δ = Q × Σ = Q
q0 = {p}
F = {r}

Step 2 - Determine ϵ-closure of each state.
ϵ-closure of p = {p}
ϵ-closure of q = {p, q, r}
ϵ-closure of r = {r, q}

Step 3: Let M = <Q', Σ, δ', q0', F'>

Step 4: Construction of δ'
δ'(q, x) = ϵ-closure (δ (ϵ-closure of q) x)
δ'(p, a) = ϵ-closure (δ (ϵ-closure p) a)
= ϵ-closure (δ (p) a)
= ϵ-closure (δ (p))
= {p}
δ'(p, b) = ϵ-closure (δ (ϵ-closure p) b)
= ϵ-closure (δ (p, b))
= ϵ-closure (q)
= {p, q}
δ'(p, c) = ϵ-closure (δ (ϵ-closure of p) c)
= ϵ-closure (δ (p, c))
= ϵ-closure (r)
= {r, q}
δ'(q, a) = ϵ-closure (δ (ϵ-closure q) a)
= ϵ-closure (δ (p, q) a)
= {p, q} ϵ-closure (δ (p, a) ∪ (q, a))
δ'(q, b) = ϵ-closure (p ∪ q)
= {p, q}
δ'(q, b) = ϵ-closure (δ (ϵ-closure q) b)
= ϵ-closure (δ (p, q) b)
= ϵ-closure (δ (p, b) ∪ (q, b)))
= ϵ-closure (q ∪ r)
= {p, q, r}
δ'(q, c) = ϵ-closure (δ (ϵ-closure q) c)
= ϵ-closure (δ (p, q) c)
= ϵ-closure (δ (p, c) ∪ δ (q, c))
= ϵ-closure (r ∪ ϕ)
= {r, q}

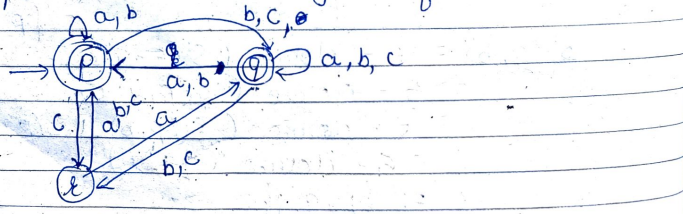
* $\delta'(r, a) = \epsilon$ - closure $(\delta(\epsilon$ - closure $r), a)$.
$= \epsilon$ - closure $(\delta(r, q), a)$.
$= \epsilon$ - closure $(\delta(r, a) \cup \delta(q, a)$
$= \epsilon$ - closure $(\phi \cup q)$
$= \{p, q\}$.

* $\delta'(r, b) = \epsilon$ - closure $(\delta(\epsilon$ - closure $r), b)$.
$= \epsilon$ - closure $(\delta(r, q), c)$.
$= \epsilon$ - closure $(\delta(r, c) \cup \delta(q, c)$.
$= \epsilon$ - closure $(p \cup \phi)$.
$= \{p\}$.

$\delta'(r, c) = \epsilon$ - closure $(\delta(\epsilon$ - closure $r), c)$.
$= \epsilon$ - closure $(\delta(r, q), c)$
$= \epsilon$ - closure $(\delta(r, c) \cup \delta(q, c))$
$= \epsilon$ - closure $(p \cup \phi)$
$= \{p\}$.

Logic :-

Step 5 :- Transition diagram of NFA.



Step 6: Tupples of NFA.
$M = \langle Q', \Sigma, \delta', q_0', F' \rangle$.
$Q = \{p, q, r\}$
$\Sigma = \{a, b, c\}$
$\delta' = Q \times \Sigma \to 2^Q$.
$q_0' = \{p\}$
$F' = \{p, q\}$.

Step 7: Transition table of NFA

δ	a	b	c
→ p	[p]	[p, q]	[q, r]
q	[p, q]	[p, q, r]	[q, r]
r	[p, q]	[p]	[p]

iii] Compute all the strings of length 3 or less accepted by the automata.

$L = \{a, b, c, ab, bc, ac, abc\}$.

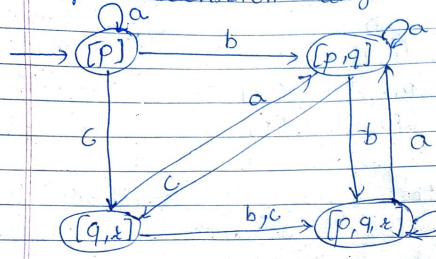
iii] Convert to DFA

Step 4:- Transition table of DFA

δ/Σ	a	b	c
→ [p]	[p]	[p, q]	[q, r]
~~[q]~~	[p, q]	[p, q, r]	[q, r]
~~[r]~~	[p, q]	[p]	[p]
[p, q]	[p, q]	[p, q, r]	[q, r]
[q, r]	[p, q]	[p, q, r]	[p, q, r]
[p, q, r]	[p, q]	[p, q, r]	[p, q, r]
~~[p]~~	[p]	[p, q]	~~[q, r]~~

Step 5: Tuples of DFA.
$M' = \langle Q', \Sigma', \delta', q_0', F' \rangle$
$Q' = \{[p], [q], [r], [p, q], [q, r], [p, q, r]\}$.
$\Sigma' = \{a, b, c\}$
$\delta' = Q' \times \Sigma' = 2^{Q'}$
$q_0' = \{[p]\}$
$F' = \{[p], [q], [p, q], [q, r], [p, q, r]\}$.

Step 6: Transition diagram of DFA

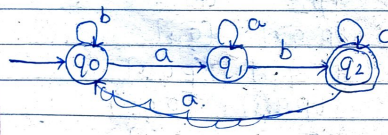


Q2] Design a DFA for string which should contain 'ab' as substring over $\Sigma = (a,b)^*$.

Step 1:- $L = \{\epsilon, ab, aab, aaab, abab, ababab, bbabbb, aaabbb, \ldots\}$

Step 2:- $M = \langle Q, \Sigma, \delta, q_0, F \rangle$

Step 3:-



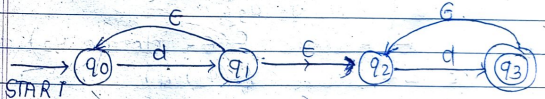
Step 4:-
$M = \langle Q, \Sigma, \delta, q_0, F \rangle$
$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$\delta = Q \times \Sigma \to Q$.
$q_0 = \{q_0\}$
$F = \{q_2\}$

Step 5:-

δ/Σ	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

Step 6:- Taking ababab
$\delta(q_0, ababab)$
$\delta(\delta(q_0,a)\ babab)$
$\delta(\delta(q_1,b)\ abab)$
$\delta(\delta(q_2,a)\ bab)$
$\delta(\delta(q_2,b)\ ab)$
$\delta(\delta(q_2,a)\ b)$
$\delta(\delta(q_2,b))$
$\delta(q_2)$

Q3] Obtain a DFA for



Step 1:- ϵ-NFA is defined with 5 tuples
$M = \langle Q, \Sigma, \delta, q_0, F \rangle$
$Q = \{q_0, q_1, q_2, q_3\}$.
$\Sigma = \{d\}$.
$\delta = Q \times \Sigma \to Q$
$q_0 = \{q_0\}$
$F = \{q_3\}$.

Step 2:- Determine ϵ-closure
ϵ-closure of $(q_0) = \{q_0\}$
ϵ-closure of $(q_1) = \{q_0, q_1, q_2\}$.

ϵ- closure of $(q_2) = \{q_2\}$
ϵ - closure of $(q_3) = \{q_2, q_3\}$

Step3:- $M = <Q', \Sigma', \delta', q_0', F'>$ Define.

Step 4:- Construction of δ'

$\delta'(q, x) = \epsilon$-closure $(\delta(\epsilon\text{- closure } q)\, x)$

$\delta'(q_0, d) = \epsilon$-closure $(\delta(\epsilon\text{- closure } q_0)\, d)$

$= \epsilon$-closure $(\delta(q_0, d))$

$= \epsilon$ - closure (q_1)

$= \{q_0, q_1, q_2\}$.

$\delta'(q_1, d) = \epsilon$ - closure $(\delta(\epsilon\text{- closure } q_1)\, d)$.

$= \epsilon$ - closure $(\delta(q_0, q_1, q_2)\, d)$

$= \epsilon$ - closure $(\delta(q_0, d) \cup (q_\phi, d) \cup (q_2, d))$

$= \epsilon$ - closure $(q_1 \cup \phi \cup q_3)$

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

$\delta'(q_2, d) = \epsilon$ - closure $(\delta(\epsilon\text{- closure } q_2)\, d)$.

$= \epsilon$- closure $(\delta(q_2)\, d)$.

$= \epsilon$ - closure (q_3).

$= \{q_2, q_3\}$

$\delta'(q_3, d) = \epsilon$ - closure $(\delta(\epsilon\text{- closure } q_3)\, d)$.

$= \epsilon$ - closure $(\delta(q_2, q_3)\, d)$

$= \epsilon$ - closure $(\delta(q_2, d) \cup (q_3, d))$

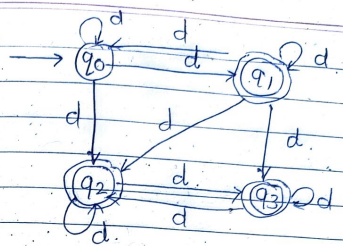
$= \epsilon$ - closure $(q_3 \cup \phi)$

$= \epsilon$ - closure (q_3)

$= \{q_2, q_3\}$

Logic :-

Step 5:-



Step 6:- Tupple of NFA

$M = <Q', \Sigma, \delta', q_0', F'>$

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

$\delta' = Q \times \Sigma \to 2^{Q'}$

$q_0' = \{q_0\}$

$F' = \{q_1, q_2, q_3\}$

Step 7:- Transmission table of NFA

Q	$\Sigma/Q\Sigma$	d
~~q_0~~	$\to q_0$	$[q_0, q_1, q_2]$
	q_1	$[q_0, q_1, q_2, q_3]$
	q_2	$[q_2, q_3]$
	q_3	$[q_2, q_3]$

& Conversion of NFA to DFA

Step 4:- Transition table of DFA

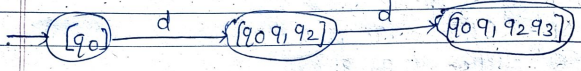
Q/Σ	d
$[q_0]$	$[q_0, q_1, q_2]$
$[q_0 q_1, q_2]$	$[q_0, q_1, q_2\ q_3]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

Step 5:- Tuples of DFA

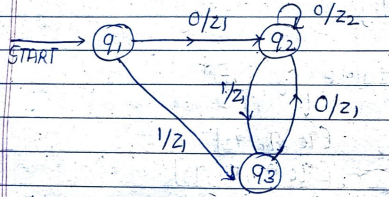
$M' = <Q', \Sigma', \delta', q_0', F'>$

$Q' = \{[q_0], [q_0 q_1 q_2], [q_0 q_1 q_2 q_3]\}$.
$\Sigma' = \{d\}$
$\delta' = Q' \times \Sigma' \to 2^{Q'}$.
$q_0' = \{q_0\}$
$F' = \{[q_0 q_1 q_2], [q_0 q_1 q_2 q_3]\}$.

Step 6: Transition diagram of DFA



Q.4 Convert the given Mealey machine into equivalent Moore machine



Step 1:- ~~Trans~~ Mealey machine is defined with 6 tuples
$M: \langle Q, \Sigma, \delta, \Delta, \lambda, q_0 \rangle$
$Q: \{q_1, q_2, q_3\}$.
$\Sigma = \{0, 1\}$.
$\delta = Q \times \Sigma \to Q$.
$\Delta = \{z_1, z_2\}$.
$\lambda = Q \times \Sigma \to \Delta$
$q_0 = \{q_1\}$

Step 2:- Transition table of Mealey machine.

Present state	Input state a=1	O/P	Input state a=0	O/P
q_1	q_3	z_1	q_2	z_1
q_2	q_3	z_1	q_2	z_2
q_3	ϕ	ϕ	q_2	z_1

Step 3:- Moore machine is defined with 6 tuples
$M' = \langle Q', \Sigma, \delta', \Delta, \lambda', q_0' \rangle$

Step 4:- Logic

Step 5:-

Q/Σ	0	1	Δ
$q_1 z_1$			

TURING MACHINE

also d/as FA with infinite steps for solving

$M = \langle Q, \Sigma, \delta, q_0, \sigma, F, B \rangle$

F = Final

where

Q = Set of all finite states

Σ = Finite set i/p states.

$\delta = Q \times \sigma \rightarrow Q \times \sigma \times \{L, R\}$ Modified states.

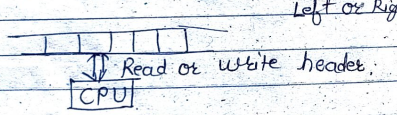
q_0 = Input initial state

σ = Finite set of tape symbol.

F = Set of finite state.

B = Blank symbol

① Infinite tape symbols.
② Read or Write header.
③ Processing unit.



$\delta(q_i, d) = (q_j, x, L/R)$

q_i: Current state; d: Current tape symbol; q_j: Next state; x: Modified symbol; L/R: Direction.

STEP FOR SOLVING THE PROBLEM

Step I :- Define the i/p string from the given language which is accepted by Turing Machine.

Q Construct a Turing Machine for following language.

$L = \{0^n 1^n \mid n \geq 0\}$.

Step I $L = \{\underset{\varepsilon}{0^0 1^0}, 0011, 000111, 00001111, 0000011111, \ldots\}$

Step 2 :- $M = \langle Q, \Sigma, \delta, q_0, P, \sigma, B \rangle$

where

Q = Finite set of all state

Σ = Finite set of i/p symbol

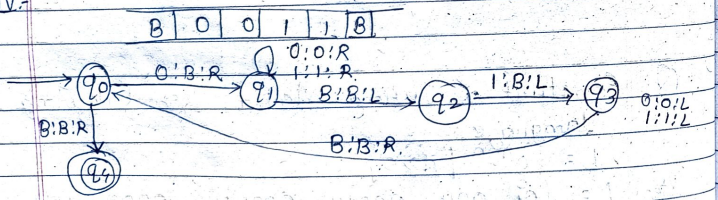
δ = Transition fun^n.

q_0 = Initial state.

F = Set of all final state.
Γ = Finite set of tape symbol.
B = Blank symbol

Step III :- Logic explanation :- Since we have to construct a Turing Machine for a where equal no. of zeros are followed by equal no. of ones.
Whenever, the 1st i/p symbol 0 comes make it blank(B) and move towards right after that for every zero keep it as a zero only & move towards right for every i/p symbol one by

Step IV :-



Context Free Grammar & PDA

$\Sigma = \{a, b\}$
$L = \{a^n, b^n \mid n = 0, 1, 2 \ldots\}$ - Non-Regular.
$L = \{w \mid w$ is a palindrome word$\}$.

* Standard notation for formal grammar

1. Capital letters near the beginning of alphabets like A, B, C, D denotes non-terminals of variables.

2. Small case letters near the beginning of alphabet like a, b, c, d denotes terminals. So also identifier, keyword, operator symbols punctuation mark, constants

3. Capital letter near the end of alphabet like X, Y, denotes single symbol it may terminal or non-terminal.

4. Small case letter near the end of alphabet like u, v, w denotes string of terminals only.

5. Greek letters α, β, γ denotes string of terminals and non-terminals both.

6. Unless otherwise stated left hand side non terminal of the first product state is treated as start symbol.

Type '0' $\alpha \to \beta$

Type '1' $\alpha \to \beta$, $|\beta| \geq |\alpha|$

Type '2' $\alpha \to \beta$, $|\beta| \geq |\alpha|$, $\alpha \in V$

Type '3' $\alpha \to \beta$, $|\beta| \geq |\alpha|$, $\alpha \in V$,
β contains atmost one non-terminal.
V - set of variables.

$S \to bA$
$A \to aA$
$A \to b$
$\Sigma = \{a, b\}$
Elements in Σ are terminals
S & A are variables.

* CFG is a quadruple (V, T, P, S)
V: set of variables or non-terminals (Capital)
T: set of terminals $(V \cap T = \phi)$
P: a set of rules $(P: V \to (V \cup T)^*)$
S: $S \in V$, a start symbol

$V = \{A, B\}$
$T = \{0, 1\}$
$P = \{ A \to 11A,$
$A \to 00B,$
$B \to 11B,$
$B \to \epsilon \}$
$S = A$

$(P = \{A \to 11A | 00B \ , \ B \to 11B | \epsilon\})$

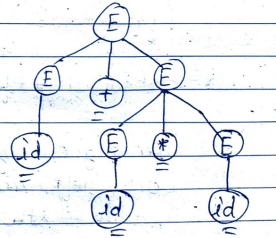
- If $A \to B$, then $xAy \to xBy$ & we say that xAy derivates xBy.
- If $S \to \dots \to t$, then we write $S \xrightarrow{*} t$. (in any no of states.)
if $S \xrightarrow{*} w$
$L(G) = \{w \text{ in } \Sigma^* \mid S \to w^*\}$.

- A string w in Σ^* is generated by $G = (V, T, P, S)$
if $S \to w^*$.
$L(G) = \{w \text{ in } \Sigma^* \mid S \to w^*\}$.

Defn - A language L is context-free if there is a context free grammar $G = (V, T, P, S)$ such that L is generated from G.

Ex:- $G = (\{E\}, \{+, *, (,), id\}, P, E)$ where
$P = \{E \to E+E \mid E*E \mid (E) \mid id$
$W = id + id * id$.

- left Most Derivation.
$E \to E+E$
$\to id+E$
$\to id + E*E$
$\to id + id*E$
$\to id + id*id$
Sentence.
(Sentential form.)



Parse tree or derivation tree

- Right most derivation.
$E \to E+E$
$\to E + E*E$
$\to E + E*id$
$\to E + id*id$
$\to id + id*id$

Leftmost derivation tree / parse tree

All intermediate strings that are generated during the formation of sentence is c/as sentential form.
The final statement is c/as sentence

Derivation - Process

Ambiguity of grammar

Right Most derivation (2):

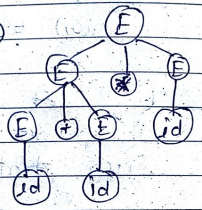
$E \to E * E$

By Precedence $\to E * id$

Right most $\to E + E * id$

derivation (1) $\to E + id * id$

is correct $\to id + id * id$



Whenever there exists two parse tree for same string then it is ambiguity of grammar.

Q. Show that following grammar is ambiguous.

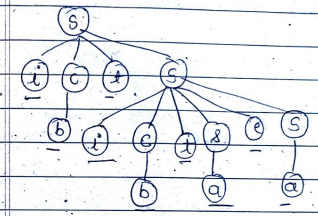
Ambiguous grammar - If there exist more than one left most / right most derivation for the same string then the grammar is said to be ambiguous grammar.

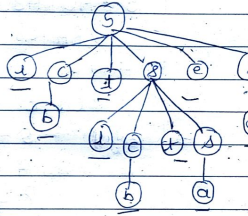
Ambiguous grammar is not stable for programming languages.

Prove that the following grammar is ambiguous

1) $S \to iCts / iCtses / a$, $C \to b$
2) $S \to aSbS / bSaS / \epsilon$

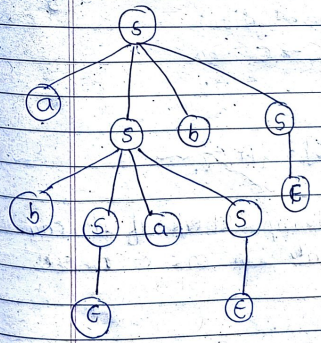
1) $S \to iCts / iCtses / a$,
$C \to b$

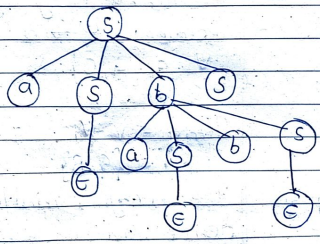




For $w = ibtibtaea$, there exists two parse trees. Therefore the given grammar is ambiguous.

2) $S \to aSbS / bSaS / \epsilon$





Terminal (Small).

Reduction of Grammar Symbols

Reduct^{n} of Grammar : Includes elimination of
1. Use-less grammar symbol
2. Null able Non-terminal
3. Unit Production.

Reduce the grammar : Elimination of useless symbol

✓ $S \to AC \mid AB$

✗ $B \to BA \mid CB$

✓ $A \to a$

✓ $C \to cC \mid \epsilon$

= Useful $S \to AC \mid AB$

$A \to a$

$C \to cC \mid c$

$V = \{S, A, C\}$

$T = \{a, c\}$

Step 1: Find Non-terminals that derives to string of terminals only (direct production)

Because $A \to a$ and $C \to c$

$V_1 = \{A, C\}$

$Temp = V - V_1 = \{S, B, A, C\} - \{A, C\}$

$= \{S, B\}$

Step 2] For every variable in Temp, check if it can derive string of terminals in one or more steps.

$S \to AC \mid AB$

$V_1 = V_1 \cup \{S\}$ B'coz in prod. $S \to AC$, A & C both are in V_1

$B \to BA \mid CB$, Here is not in V_1 ∴ B is not added in V_1

Therefore grammar reduce to.

$S \to AC$

$A \to a$

$C \to cC \mid c$

Algo 2. $S \to AC$

$A \to a$

$C \to cC \mid c$.

$N_2 = \{S\}$, $T_2 = \phi$ $P_2 = \phi$

Consider $S \to AC$ A & C are Non-terminal

$N_2 = \{S\} \cup \{A, C\} = \{S, A, C\}$, $T_2 = \phi$, $P_2 = \{S \to AC\}$

Consider $A \to a$.

$N_2 = \{S, A, C\}$, $T_2 = \{a\}$, $P_2 = \{S \to AC, A \to a\}$.

Consider $C \to cC$.

$N_2 = \{S, A, C\}$, $T_2 = \{a, c\}$ $P_2 = \{S \to AC, A \to a, C \to cC\}$

Consider $C \to c$.

$N_2 = \{S, A, C\}$, $T_2 = \{a, c\}$ $P_2 = \{S \to AC, A \to a, C \to cC \mid c\}$

∴ Reduced grammar is

$S \to AC$

$A \to a$

$C \to cC \mid a$.

Ex 2

$A \to xyz \mid Xyzz$ ✓

$X \to Xz \mid xYx$ ✗

$Y \to yYy \mid XZ$ ✗

$Z \to Zy \mid z$ ✓

Exercise

1. $S \to AC \mid SB$
   $A \to bASC \mid a$
   $B \to Asb \mid bbC$
   $C \to Bc \mid ad$

2. $S \to aC \mid SB$
   $A \to bSCa$
   $B \to aSB \mid bBC$
   $C \to aBC \mid ad$

3. $S \to aA \mid aBB$
   $A \to aaA \mid \epsilon$
   $B \to bB \mid bbC$
   $C \to B$

★ Elimination of Null able Non terminal / ϵ prod.
If $A \to \epsilon$, then A is nullable non-terminal we eliminate all ϵ except one.

- Replace each occurrence of A by ϵ in every production & add newly generated string as production.
  $S \to aAbA$
  $S \to aAbA \mid abA \mid aAb \mid ab$
- Repeat this for all ϵ-production.

Ex1. $S \to aA \mid bB$
$A \to c \mid \epsilon$
$B \to d \mid \epsilon$

Put $A = \epsilon$ in S production & delete $A \to \epsilon$
$S \to aA \mid bB \mid a$
$A \to c$, $B \to d \mid \epsilon$

Now put $B = \epsilon$ in S production & delete $B \to \epsilon$
$S \to aA \mid bB \mid a \mid b$
$A \to c$, $B \to d$

Ex2. $S \to ABAC$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$
$C \to c \quad \to C \to \epsilon$ Not $C \to \epsilon$

- First eliminate $A \to \epsilon$
  $S \to ABAC \mid BAC \mid ABC \mid BC$
  $A \to aA \mid a$
  $B \to bB \mid \epsilon$
  $C \to c$

Not possible

$S \to ABAC \mid BAC \mid ABC \mid BC \mid AC \mid C \mid AAC \mid ABA \mid BA \mid B \mid A$

- Now eliminate $B \to \epsilon$.
  $S \to ABAC \mid BAC \mid ABC \mid BC \mid AAC \mid AC \mid C$
  $A \to aA \mid a$
  $B \to bB \mid b$
  $C \to c$

$S \to ABAC \mid BAC \mid ABC \mid BC \mid AAC \mid AB$
$A \to aA \mid a$
$B \to bB \mid b$
$C \to \epsilon$

Eliminate $C \to \epsilon$

Elimination of Unit production:
- A production of form $A \to B$ (a unit production)
- It un-necessarily increases cost of derivation.
- To eliminate unit production $A \to B$, search for non unit production $B \to \alpha$.
- Then write $A \to \alpha$ & delete $B \to \alpha$.

Ex1.
$S \to AB$
$A \to a$
$B \to C \mid b$
$C \to D$
$D \to E$
$E \to a$

- Eliminate $D \to E$
  $S \to AB$
  $A \to a$
  $B \to c \mid b$
  $C \to D$
  $D \to a$
- Eliminate $C \to D$
  $S \to AB$
  $A \to a$
  $B \to c \mid b$
  $C \to a$
- Eliminate $B \to C$
  $S \to AB$
  $A \to a$
  $B \to a \mid b$

★ Elimination of left Recursion
$G_1 = \{V, T, P, S\}$ is said to be left recursive if it contains at least one pair of production of the form $A \to A\alpha \mid \beta$

- Left recursive grammar is not suitable for Top-Down parse.
- To eliminate left recursion:
  If $A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \mid \beta_1, \beta_2 \ldots \beta_n$
  Where all β's are not starting from A, then we write
  $A \to \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A'$
  $A' \to \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_n A' \mid \epsilon$
  Collect all β & append with new A then collect all α & append with new A & ϵ.

(1) Eliminate
$E \to E+T \mid T$ — β
$T \to T * F \mid F$
$F \to (E) \mid id$ — as it is
= (1) $E \to E+T \mid T$
$E \to TE'$
$E' \to +TE' \mid \epsilon$
(2) $T \to T*F \mid F$
$T \to FT'$
$T' \to *FT' \mid \epsilon$
Finaly
$E \to TE'$
$E' \to +TE' \mid \epsilon$
$T \to FT'$
$T' \to *FT' \mid \epsilon$
$F \to (E) \mid id$

Q. $D \to TL$
$T \to int \mid float$
$L \to L, id \mid id$

either two non-terminal or single terminal

★ Normal Forms

1. Chomsky Normal Form (CNF)
   $A \to BC | a$
2. Greibach Normal Form (GNF)
   $A \to a\alpha$
   where α is string of non-terminals only.

$A \to$ must start with non-t
α - have any other terminal & non-t

Q. Bring grammar in CNF.
$S \to aB | bA$
$A \to a | aS | bAA$
$B \to b | bS | aBB$

Sol:
- Add new non-terminal C_a & C_b with production
  $C_a \to a$ & $C_b \to b$.

Rewrite as
$S \to C_aB | C_bA$
$A \to a | C_aS | C_bAA$
$B \to C_b | C_bS | C_aBB$

- Introduce two more N.T.D & E with production as $D \to AA$ & $E \to BB$.

∴ Rewriting the grammar
$S \to C_aB | C_bA$
$A \to a | C_aS | C_bD$
$B \to b | C_bS | C_aE$
$C_a \to a$
$C_b \to b$
$D \to AA$
$E \to BB$

$A \to A\alpha | \beta$
$A \to \beta A'$
$A' \to \alpha A' | \epsilon$

★ Greibach Normal Form (GNF)
$S \to AA | 0$ — ①
$A \to SS | 1$ — ②
Put ② in ①
$S \to SSA | 1A | 0$ — ③

SSA: α = SA, β = 1A | 0

Eliminate left recursion from 3
$S \to 1AS' | 0S'$
$S' \to SAS' | \epsilon$ } ④

Eliminate $\alpha \in$ product^n from ④.
Put $S' = \epsilon$ in 4
$S \to 1AS' | 0S' | 1A | 0$ } ⑤
$S' \to SAS' | SA$
Put value of S in S' production
$S' \to 1AS'AS' | 0S'AS' | 1AAS' | 0AS'$
$S' \to 1AS'A | 0S'A | 1AA | 0A$
All the production of S & S' are in GNF.

Put value of S in A-product^n eq. 2.
$A \to SS | 1$
$A \to 1AS'S | 0S'S | 1AS | 0S | 1$
∴ Grammar in GNF is
$S \to 1AS' | 0S' | 1A | 0$
$S' \to 1AS'AS' | 0S'AS' | 1AAS' | 0AS'$
$S' \to 1AS'A | 0S'A | 1AA | 0A$
$A \to 1AS'S | 0S'S | 1AS | 0S | 1$

Ex.1 Reduce the grammar to CNF
$S \rightarrow \sim S \mid [S \supset S] \mid p \mid q$.

Ex.1 Reduce the grammar to GNF
$S \rightarrow SS$
$S \rightarrow 0S1 \mid 01$

2. $S \rightarrow AB$
$A \rightarrow BS \mid b$
$B \rightarrow SA \mid a$.

Ex. Let G_1 is a CFG & $W \in L(G_1)$ such that $|w|=l$ How long will be derivation of w.
i) G_1 is in CNF ii) G_1 is in GNF.

Step 1) i) If any production have RHS more than one symbol, convert terminals to Non-terminals introducing new N.T.
$S \rightarrow TS \mid LSNSR \mid p \mid q$
$T \rightarrow \sim$
$L \rightarrow [$
$N \rightarrow \supset$
$R \rightarrow]$

Step 2) $L[SNSR]$
LA
$A \rightarrow [\,|\,]\,|\,BC$
$B \rightarrow SN$
$C \rightarrow SR$

If RHS of any product have more than one NT, Divide NT into group to make only TWO NTs

Ex: $S \rightarrow SS \mid OS1A \mid OA$
$A \rightarrow 1$

Step 1: Eliminate left recursion
$S \rightarrow OSAS' \mid OAS'$ —— (1)
$S' \rightarrow SS' \mid \epsilon$ —— (2)

Step 2: Eliminate ϵ-production.
$S \rightarrow OSAS' \mid OAS' \mid OSA \mid OA$ —— (3)
$S' \rightarrow SS' \mid S$

Ex: a) G_1 is in CNF $A \rightarrow BC \mid a$
To derive string of NT of length l: $l-1$ steps
To derive string of terminals of length
Total $2l-1$ steps

b) G_1 is in GNF $A \rightarrow a\alpha$
To derive string of terminals of length l: l steps

Push Down Automata.

PDA M is defined in 7 tuple.

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

Q: Finite set of States

Σ: Finite set of i/p symbols

Γ: Finite set of stack symbols

δ: Transition funct^n

$$Q \times \Sigma \cup \{\in\} \times \Gamma \to Q \times \Gamma^*$$

q_0: is in Q, is a start state

z_0: z_0 is in Γ, is a initial stack symbol

F: is set of final state.

$F = \phi$ for EMPTY (NULL) stack acceptance.

Language accepted by PDA

→ Two ways of acceptance.

① Empty Stack or Null Stack Acceptance.
The language accepted to be the set of all i/ps. for which some sequence of moves causes PDA to empty its stack. This language referred to as language accepted by empty stack or Null stack.

② Final State Acceptance.
The lang accepted to be set of all i/ps for which some sequence of moves causes PDA to enter into one of its final state. This language referred to as lang accepted by final state

3 i/p - 2 o/p

Current State
I/p symb
Stack symbo

Moves of PDA.

$\delta(q, a, z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \dots (p_m, \gamma_m)\}$

p & q for $1 <= i <= m$, are states in Q.
a is in Σ, z is in Γ, & γ_i is in Γ^*

PDA is q, i/p sym a & z is on top of stacks, enteem in state p_i, replace stack symbol z by γ_i for any i, & advance i/p head one symbol

$\delta(q, \in, z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \dots (p_m, \gamma_m)\}$

PDA is in q, independent of i/p symbol being scanned & z on stack top, enters into state p_i, replacing stack z by γ_i. The read head is NOT advanced.

Current a
Next p
a-i/p
z replaced

Graphical representation.

$\delta(q, a, z) = (p, \gamma)$ — q, z/γ → p

Instaneous Description (ID):
We define ID to be triple (q, w, γ) where q is state, w is string of i/p symbols & γ a string of stack symbols

we say $(q, aw, z\alpha) \vdash^M (p, w, \beta\alpha)$
if $\delta(q, a, z) = (p, \beta)$ is defined in PDA.

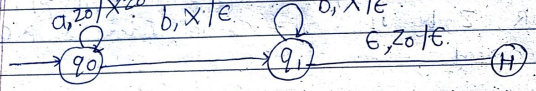
★ Accepted language $M = (Q, \Sigma, F, \delta, q_0, z_0, F)$.
lang accepted by final state, to be
$L(M) = \{w \mid (q_0, w, z_0)^n \vdash (p, w, \gamma)$ for some p in F : & γ in $\Gamma^*\}$.

Lang. accepted by Empty Stack (Null Stack) to be
$N(M) = \{w \mid (q_0, w, z_0) \overset{M}{\vdash} (p, \epsilon, \epsilon)$ for some p in $Q\}$.

Q. Construct PDA for $L = \{a^n b^n \mid n >= 1\}$

1. Push all a's
2. For every b, pop up a
3. When string ends & stack empty simultaneously ~~empty~~ accept

Transition diagram



OR

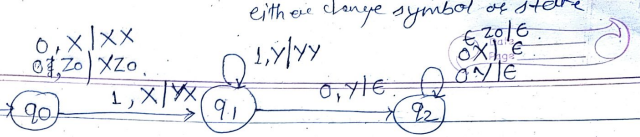
Transition Function

$\delta(q_0, a, z_0) = (q_0, XZ_0)$
$\delta(q_0, a, X) = (q_0, XX)$
$\delta(q_0, b, X) = (q_1, \epsilon)$
$\delta(q_1, b, X) = (q_1, \epsilon)$
$M = \{\{q_0, q_1\}, \{a, b\}, \{X, z_0\}, \delta, q_0, z_0, \phi\}$
↑ stack symbol.

Q. Construct PDA for $L = \{0^m 1^n 0^{m+n} \mid m, n >= 1\}$.

1. Push all m 0's
2. Push n 1's
3. For every 0 to the right of 1 pop one symbol from the stack (0 or 1 whatever)
4. If string ends and stack empty simultaneously accept

either change symbol or store



$\delta(q_0, 0, z_0) = (q_0, XZ_0)$
$\delta(q_0, 0, X) = (q_0, XX)$
$\delta(q_0, 1, X) = (q_1, YX)$
$\delta(q_1, 1, Y) = (q_1, YY)$
$\delta(q_1, 0, Y) = (q_2, \epsilon)$
$\delta(q_2, 0, Y) = (q_2, \epsilon)$. $\delta(q_2, 0, X) = (q_2, \epsilon)$
$\delta(q_0, \epsilon, z_0) = (q_2, \epsilon)$

$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{z_0, X, Y\}, \delta, q_0, z_0, \phi)$

Q $L = \{0^{m+n} 1^n 0^m \mid m, n >= 1\}$.
Q $L = \{0^m 1^{m+n} 0^n \mid m, n >= 1\}$.

Ex- Construct PDA for $L = \{wcw^R \mid w \in \{a, b\}^*$, & w^R represents reverse of $w\}$.
- odd palindrome

=> Push all w's

$\delta(q_0, a, z_0) = (q_0, Xz_0)$
$\delta(q_0, b, z_0) = (q_0, Yz_0)$
$\delta(q_0, a, X) = (q_0, XX)$
$\delta(q_0, b, X) = (q_0, YX)$
$\delta(q_0, a, Y) = (q_0 \epsilon, XY)$
$\delta(q_0, b, Y) = (q_0, YY)$
$\delta(q_0, c, X) = (q_1, X)$
$\delta(q_0, c, Y) = (q_1, Y)$

c is centre.
No

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

Ex: Construct PDA for $L = \{a^m b^n / m, n > 1 \ \& \ m \neq n\}$

1. Push all as
2. For each b, popup one a from the stack.
3. If I/p is b & stack top is z_0 OR.
   If i/p ends and at least one a on stack, Accept
   Such PDA is clas Final state PDA.

$\delta(q_0, a, z_0) = (q_0, Xz_0)$
$\delta(q_0, a, X) = (q_0, XX)$
$\delta(q_0, b, X) = (q_1, \epsilon)$
$\delta(q_1, b, X) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, X) = (q_2, \epsilon)$
$\delta(q_1, b, z_0) = (q_2, \epsilon)$
$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{z_0, X, Y\}, \delta, q_0, z_0, \{q_2\})$ Final state

Ex. Construct PDA for $L = \{w w^R / w \in \{a, b\}^*$, and w^R represents reverse of $w\}$

aaabbb = a

$M = (\{q_0, q_1\}, \{a, b\}, \{z_0, X, Y\}, \delta, q_0, z_0, \phi)$
$\delta(q_0, a, z_0) = (q_0, Xz_0)$
$\delta(q_0, b, z_0) = (q_0, Yz_0)$
$\delta(q_0, a, X) = \{(q_0, XX), (q_1, \epsilon)\}$ Non-Determinism
$\delta(q_0, b, Y) = \{(q_0, YY), (q_1, \epsilon)\}$
$\delta(q_0, b, X) = \{(q_0, YX)$
$\delta(q_0, a, Y) = (q_0, XY)$
$\delta(q_1, b, Y) = (q_1, \epsilon)$
$\delta(q_1, a, X) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

Such PDA is clas Non-Deterministic PDA (NDPDA).