FLAT Syllabus

Unit-I

Finite Automata & Regular Expressions:

Basic concepts of finite state system, determanistic & non determanistic autometa finite automata with e-moves Regular expressions and mealy & moove relations

→ Introduction to automata theory & language computation by D. Ubnan. 3rd edition

→ Theory of comp. sc. Mishra & chandra sekharan.

Formal languages:

Basic concepts
- symbols
- Alphabets
- strings
- languages

* Symbols: letters/ numericals

a, b, c .. 0, 1, 2 ... 9

bits: {0, 1}

English words: systematic concepts of programming language.

for, begin, end → i/p alphabet

ii) Alphabets : An alphabet Σ is a finite set of symbols

Ex: $\Sigma = \{a, b, c \ldots z\}$

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

keywords, strings, identifiers etc

iii) String : is a finite set of symbols from Σ

Ex : $\Sigma = \{a, b, c, d\}$

abcd

string notations :

1. length of string is given by $|u|$

Ex : $u = abac$

$|u| = 4$

2. for concatination $u = abc$

$v = aba$

$u.v = abcaba$

3. substring : $u \times v$

$u = abc$

$v = aba$

$x = bab$

abc bab aba

4. Prefix:

x uv bab abc aba

5. Suffix:

uvx

abc aba bab

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

$\Sigma = \{a, b, c\}$ length 4 begin with a; end with b

aabb
aacb
abcb
abbb
aaab
accb
acab
acbb
abab

Empty string: length is 0

$\varepsilon, \lambda, \varepsilon$

$\varepsilon u = u = u\varepsilon$

* Language

If starts with alphabet Σ. The set of all strings over Σ is denoted by Σ^*

A language over Σ is any subset of Σ^*

→ Construct a language of string length 4 that begin with a and end with b.

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

$L = \{$ aaab, aabb, aacb, abbb, acbb, abcb, accb, acab, abab $\}$

ϕ = length is 0 $\quad \phi = \{\varepsilon\}$

$\{\varepsilon\}$ = length is 1

Automata Theory (AT)

In theoritical computer science, it is a study of abstract machines & computational problems that can be solved using these machines. These abstract machines are called automata.

Deterministic finite Automata (DFA)

It is a 5 tuple notation

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

Q = finite set of states

Σ = the input alphabets

$\delta = Q \times \Sigma \to Q$ is a transition func

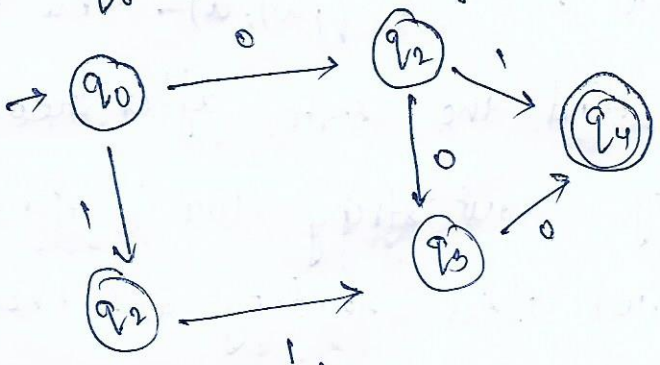
$q_0 \in Q$ is the start state

$F \in Q$ is the set of final or Acception state

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

$\delta(q_0, 0) \to q_1$

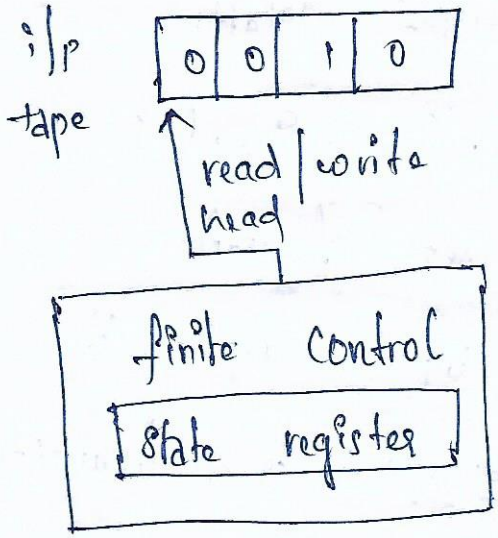
q_0 on 0 changes to q_1



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

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

The model for DFA



Properties of transition function

1. $\delta(q, \epsilon) = q$ is a state that without reading an input symbol.

The Finite Automata cannot change the state

2. For all strings w and input symbol a.

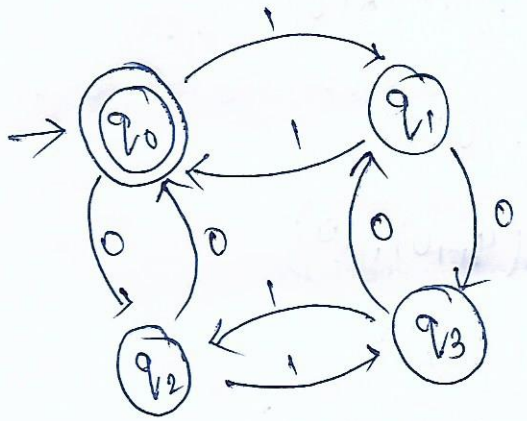
$\delta(q, wa) = \delta(\delta(q, w), a)$. This tells us how to find the state after reading the non empty input string wa is find the state $p = \delta(q, w)$ after reading w. then

compute $\delta(p, a)$

Similarly

$\delta(q, aw) = \delta(\delta(q, a), w)$

* checks given finite automata accepts even number of 0's & 1's if $\Sigma = \{0,1\}$



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

$= (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, q_0)$

Transition Function

q	0	1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$x = 1001$

$\delta(q_0, x) = \delta(q_0, 1001)$

$= \delta(\delta(q_0,1), 001)$

$= \delta(q_1, 001)$

$= \delta(\delta(q_1,0), 01)$

$= \delta(q_3, 01)$

$= \delta(\delta(q_3,0), 1)$

$= \delta(q_1, 1)$

$= q_0 \in F$

$x = 1010$

$\delta(q_0, x) = \delta(q_0, 1010)$

$= \delta(\delta(q_0, 1), 010)$

$= \delta(q_1, 010)$

$= \delta(\delta(q_1, 0), 10)$

$= \delta(q_3, 10)$

$= \delta(\delta(q_3, 1), 0)$

$= \delta(q_2, 0)$

$= q_0 \in F$

Is FS

$q_0 \to q_0$ = $\in$ 0's & $\in$ 1's

$q_0 \to q_1$ = $\in$ 0's & 0 1's

$q_0 \to q_2$ = 0 0's & $\in$ 1's

$q_0 \to q_3$ = (0 0's & 0 1's) & (0 0's & $\in$ 1's) & ($\in$ 0's & 0 1's).

Non-Deterministic Finite Automata (NFA)

NFA is a 5 tuple

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

Q = Number of states

Σ = number of finite non empty set of i/p symbols

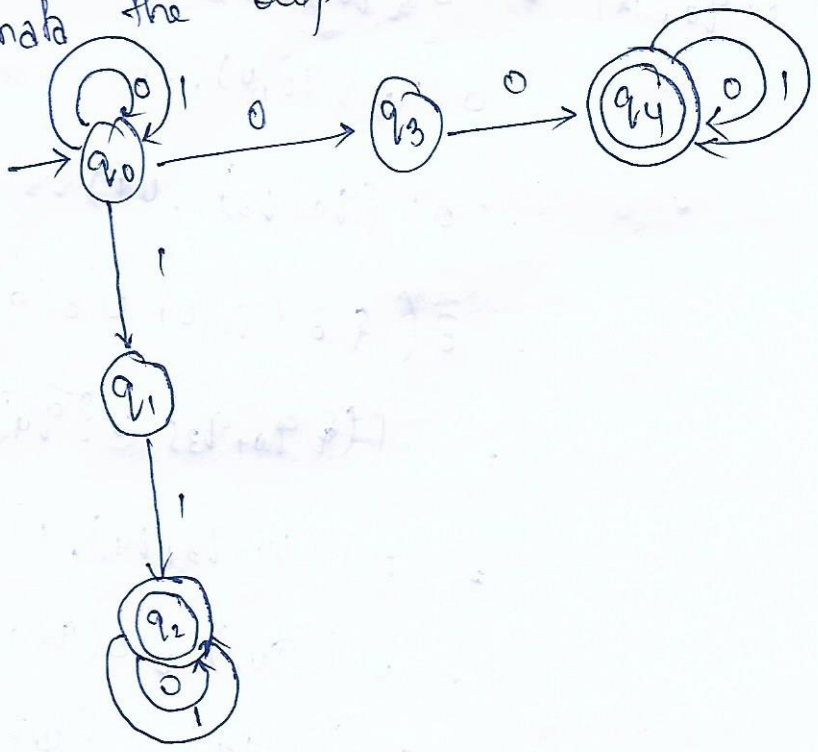
δ = transition function ($\delta = Q \times \Sigma \to 2^Q$)

$q_0 \in Q$ is starting state

$F \subseteq Q$ is final state

The difference between the deterministic & non deterministic automata is only δ. for deterministic automata the outcome is a state i.e an element of Q. for Non deterministic automata the output is subset of Q.

Ex:



From the transition system

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

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2$

Transition table

	0	1
→ q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	ϕ	q_2
(*q_2)	q_2	q_2
q_3	q_4	ϕ
(*q_4)	q_4	q_4

Check i/p symbol $x = 001$ whether the machine is accepted or not

$\delta(q_0, x) = \delta(q_0, 001)$

$= \delta(\delta(q_0, 0), 01)$

$= \delta(\{q_0, q_3\}, 01)$

$= \delta(\{\delta(q_0, 0) \cup \delta(q_3, 0)\}, 1)$

$= \delta((\{q_0, q_3\} \cup \{q_4\}), 1)$

$= \delta((q_0, q_3, q_4), 1)$

$= \delta(q_0, 1) \cup \delta(q_3, 1) \cup \delta(q_4$

$= \{q_0, q_3\} \cup \{\phi\} \cup \{q_4\}$

$= \{q_0, q_1, q_4\}$

∴ The set consists of atleast one final state so the given string 001 is accepted. this is NFA.

→ 110

$\delta(q_0, x) = \delta(q_0, 110)$

$= \delta(\delta(q_0, 1), 10)$

$= \delta((q_0, q_1), 10)$

$= \delta((\delta(q_0, 1) \cup \delta(q_1, 1)), 0)$

$= \delta(\{q_0, q_1\} \cup \{q_2\}, 0)$

$= \delta(\{q_0, q_1, q_2\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$

$= \{q_0, q_3\} \cup \delta\{\phi\} \cup \{q_2\}$

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

This is NFA.

→ 101

$\delta(q_0, x) = \delta(q_0, 101)$

$= \delta(\delta(q_0, 1), 01)$

$= \delta(\{q_0, q_1\}, 01)$

$= \delta((\delta(q_0, 0) \cup \delta(q_1, 0)), 1)$

$= \delta(\{q_0, q_3\} \cup \{\phi\}), 1)$

$= \delta(\{q_0, q_3\}, 1)$

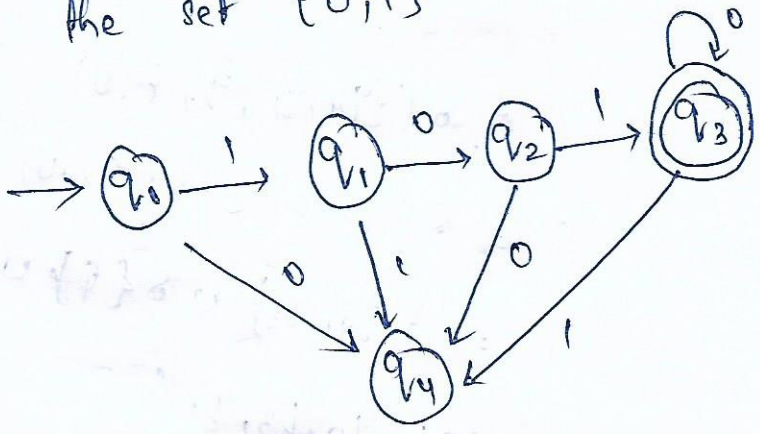
$= \delta(q_0, 1) \cup \delta(q_3, 1)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

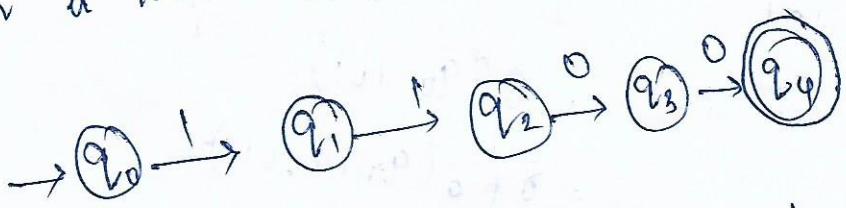
Since this string 101 doesn't contain any final state. Therefore it is not accepted by this NFA.

* Design DFA that accepts only input 101 over the set $\{0,1\}$



$M = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, q_3)$

* Design a NFA that accepts 1100 only.



$M = (\{q_0, q_1, q_2, q_3, q_4\}, \Sigma, \delta, q_0, q_4)$

Conversion of NFA to DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA, Now we construct its equivalent DFA as follows:

Let $M' = (Q', \Sigma, \delta', [q_0], F')$ where

① $Q' = 2^Q$ (Power set elements) denote that a state in Q' by $[q_1, q_2 \ldots q_j]$ which is a single state in Q' and several states in Q.

② Σ is same.

③ δ is mapping from $\delta' : Q' \times \Sigma \longrightarrow Q'$ and is defined as $\delta'([q_0], x) = [q_1, q_2 \ldots q_j]$ iff $\delta(\{q_0\}, x) = \{q_1, q_2 \ldots q_j\}$.

④ The initial state of NFA becomes the initial state of DFA.

⑤ F' is a set of those states of 2^Q which contain atleast one element of F.

→ Construct DFA equivalent to NFA

$(\{P, Q, R, S\}, \{0, 1\}, \delta, \{P\}, \{Q, S\})$

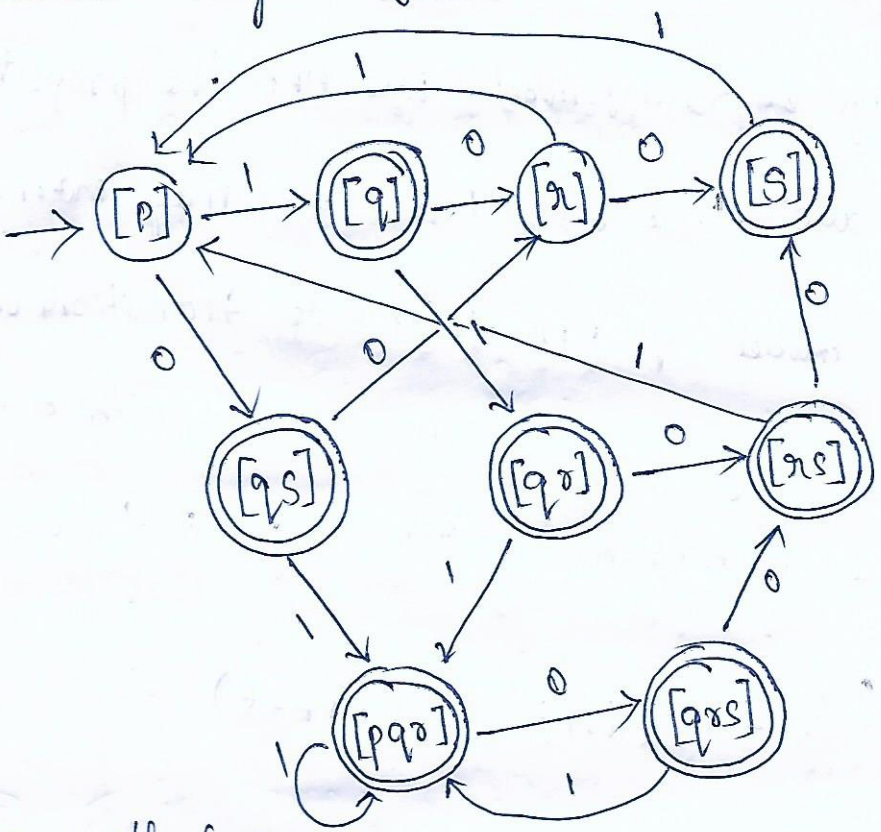
	0	1
→ p	{q, s}	{q}
(q)	{r}	{q, r}
r	{s}	{p}
(s)	ϕ	{p}

Sol: DFA = $(Q', \Sigma, \delta', [q_0], F')$

(Transition Table)

	0	1
→ [p]	[q, s]	[q]
[q]	[r]	[q, r]
[r]	[s]	[p]
[s]	{ϕ}	[p]
[qs]	[r]	[pqr]
[qr]	[rs]	[pqs]
[rs]	[s]	[p]
[pqr]	[qrs]	[pqr]
[qrs]	[rs]	[pqr]

Transition diagram for DFA



NFA with ε moves

In automata theory, a NFA with ε-moves (NFA-ε) is a extension of NFA which allows a transformation of usecase without consuming any input symbols. The transition without consuming an input symbols are called ε transitions. In the state diagrams they are usually labelled with a greek letter ε. ε transitions doesn't act any extra capacity of recognising formal languages. NFA ε are defined because certain properties can be more easily proved on them as

Compared to NFA. Since Theorems, A NFA ε can always be transformed into NFA the properties are also true for NFA's we may extend our model of NFA to include transitions on the empty input ε. An NFA with ε move is a mathematical model or 5-tuple

$$(Q, \Sigma, \delta, q_0, F) \quad (NFA-\epsilon)$$

where Q = finite non empty set of states

Σ = " " " " of i/p symbols

δ = transition function which is mapping from

$$\delta = Q \times \{\Sigma \cup \{\epsilon\}\} \longrightarrow 2^Q$$

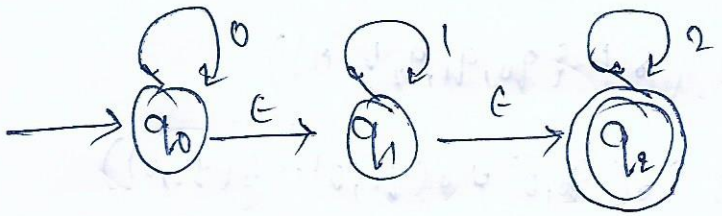
$q_0 \in Q$ is defined as $\delta(q, a) = p$ where $a = \epsilon$ or a = any input symbol.

$q_0 \rightarrow$ Initial state

$F \rightarrow$ Final state

We say that a NFA with ε moves accepts a string 'w' if there is some path from

initial state to final state.



Convert it in to NFA without ϵ moves

Transition table

	0	1	2	ϵ
q_0	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$	ϕ

In this content, quantity called ϵ-closure of q denoted by $\epsilon\text{-cl}(q) = p$

There is a transition labelled on ϵ from q to p

$$\epsilon\text{-cl}(q_0) = \{q_0, q_1, q_2\}$$

$$\epsilon\text{-cl}(q_1) = \{q_1, q_2\}$$

$$\epsilon\text{-cl}(q_2) = \{q_2\}$$

In order to eliminate ϵ moves we define δ as follows

$\hat{\delta}$ as follows

$$\hat{\delta}(q, a) = \epsilon\text{-cl}\left(\delta(\epsilon\text{-cl}(q), a)\right)$$

$\hat{\delta}(q_0,0) = e\text{-}cl(\delta(e\text{-}cl(q_0),0))$

$= e\text{-}cl(\delta(\{q_0,q_1,q_2\},0))$

$= e\text{-}cl(\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0))$

$= e\text{-}cl(\{q_0\} \cup \phi \cup \phi)$

$= e\text{-}cl(q_0)$

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

$\hat{\delta}(q_0,1) = e\text{-}cl(\delta(e\text{-}cl(q_0),1))$

$= e\text{-}cl(\delta\{q_0,q_1,q_2\},1))$

$= e\text{-}cl(\delta(q_1,1) \cup \delta(q_2,1)) \cup \delta(q_0,1))$

$= e\text{-}cl(q_1 \cup \phi \cup \phi)$

$= e\text{-}cl(q_1)$

$= \{q_1, q_2\}$

$\hat{\delta}(q_0,2) = e\text{-}cl(\delta(e\text{-}cl(q_0),2))$

$= e\text{-}cl(\delta(q_0,q_1,q_2),2))$

$= e\text{-}cl(\delta(q_0,2) \cup \delta(q_1,2) \cup \delta(q_2,$

$= e\text{-}cl(\phi \cup \phi \cup q_2)$

$= e\text{-}cl(q_2)$

$= q_2$

$\hat{\delta}(q_1,0) = e\text{-}cl(\delta(e\text{-}cl(q_1),0))$

$= e\text{-}cl(\delta(q_1,q_2),0))$

$= e\text{-}cl(\delta(q_1,0) \cup \delta(q_2,0))$

$= e\text{-}cl(\phi)$

$= \phi$

$\hat{\delta}(q_1,1) = e\text{-}cl(\delta(e\text{-}cl(q_1),1))$

$= e\text{-}cl(\delta(q_1,q_2),1))$

$= e\text{-}cl(\delta(q_1,1) \cup \delta(q_2,1))$

$= e\text{-}cl(q_1 \cup \phi)$

$= e\text{-}cl(q_1) = \{q_1,q_2\}$

$\hat{\delta}(q_1,2) = e\text{-}cl(\delta(e\text{-}cl(q_1),2))$

$= e\text{-}cl(\delta(q_1,q_2),2))$

$= e\text{-}cl(\delta(q_1,2) \cup \delta(q_2,2))$

$= e\text{-}cl(\phi \cup q_2)$

$= q_2$

$\hat{\delta}(q_2,0) = e\text{-}cl(\delta(e\text{-}cl(q_2),0))$

$= e\text{-}cl(\delta(q_2,0))$

$= e\text{-}cl(\phi) = \phi$

$\hat{\delta}(q_2,1) = e\text{-}cl(\delta(e\text{-}cl(q_2),1))$

$= e\text{-}cl(\delta(q_2,1))$

$= e\text{-}cl(\phi) = \phi$

$\hat{\delta}(q_2,2) = e\text{-}cl(\delta(q_2,2))$

$= q_2$

$\hat{\delta}$ =

	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	$\{q_2\}$

27/6/16

Now we define NFA without ε moves

$M = (Q, \Sigma, \hat{\delta}, q_0, F)$

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

$\Sigma = \{0, 1, 2\}$

$\hat{\delta}$ = as shown above

* Initial state: ε-cl (initial state of NFA with ε moves)

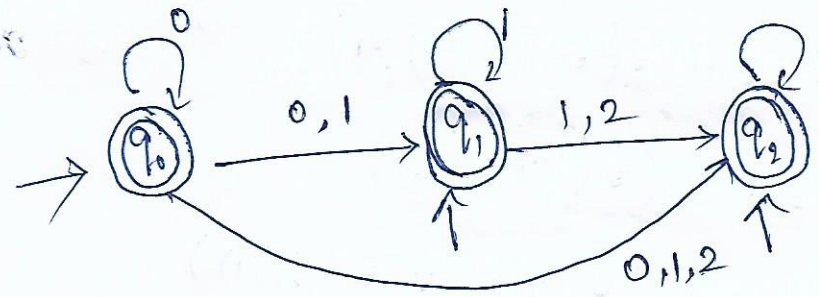
∴ ε-cl(q_0) = $\{q_0, q_1, q_2\}$

* Final state: ε-cl(Q) Containing a state F then it becomes final state

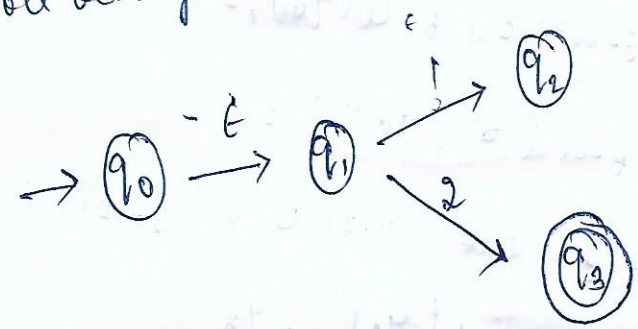
ε-cl(q_0) = $\{q_0, q_1, \underline{q_2}\}$

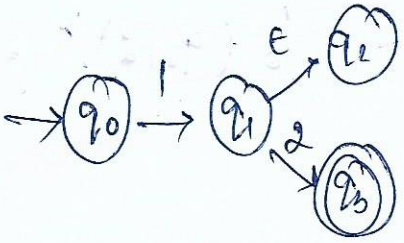
ε-cl(q_1) = $\{q_1, \underline{q_2}\}$

ε-cl(q_2) = $\{\underline{q_2}\}$



→ Construct NFA without ϵ moves from the following NFA with ϵ moves





$\epsilon\text{-cl}(q_0) = \{q_0\}$

$\epsilon\text{-cl}(q_1) = \{q_1, q_3\}$

$\epsilon\text{-cl}(q_2) = \{q_2\}$

$\epsilon\text{-cl}(q_3) = \{q_3\}$

$\delta =$

	1	2	ϵ
q_0	ϕ	ϕ	$\{q_1\}$
q_1	$\{q_2\}$	$\{q_3\}$	ϕ
q_2	ϕ	ϕ	ϕ
q_3	ϕ	ϕ	ϕ

$\epsilon\text{-cl}(q_0) = \{q_0, q_1\}$

$\epsilon\text{-cl}(q_1) = \{q_1\}$

$\epsilon\text{-cl}(q_2) = \{q_2\}$

$\epsilon\text{-cl}(q_3) = \{q_3\}$

$\hat{\delta}(q_0,1) = e\text{-}cl(\delta(e\text{-}cl(q_0),1)$

$= e\text{-}cl(\delta(q_0,q_1),1)$

$= e\text{-}cl(\delta(q_0,1) \cup \delta(q_1,1))$

$= e\text{-}cl(q_2)$

$= q_2$

$\hat{\delta}(q_0,2) = e\text{-}cl(\delta(e\text{-}cl(q_0),2)$

$= e\text{-}cl(\delta(q_0,q_1),2)$

$= e\text{-}cl(\delta(q_0,2) \cup \delta(q_1,2))$

$= e\text{-}cl(q_3) = q_3$

$\hat{\delta}(q_1,1) = e\text{-}cl(\delta(e\text{-}cl(q_1),1)$

$= e\text{-}cl(\delta(q_1,1))$

$= e\text{-}cl(q_2) = q_2$

$\hat{\delta}(q_1,2) = e\text{-}cl(\delta(e\text{-}cl(q_1),2)$

$= e\text{-}cl(\delta(q_1,2))$

$= e\text{-}cl(q_3) = q_3$

$\hat{\delta}(q_2,1) = e\text{-}cl(\delta(e\text{-}cl(q_2),1)$

$= e\text{-}cl(\delta(q_2),1))$

$= e\text{-}cl(\phi) = \phi$

$\hat{\delta}(q_2,2) = e\text{-}cl(\delta(q_2,2))$

$= e\text{-}cl(\phi) = \phi$

$\hat{\delta}(q_3,1) = e\text{-}cl(\delta(e\text{-}cl(q_3),1))$

$= e\text{-}cl(\delta(q_3,1)$

$= \phi$

$\hat{\delta}(q_3,2) = e\text{-}cl(\delta(q_3,2) = \phi.$

$\hat{\delta}$ =

	1	2
q_0	q_2	q_3
q_1	q_2	q_3
q_2	ϕ	ϕ
q_3	ϕ	ϕ

* Initial state = e-cl (initial state of NFA with ε moves)

$\therefore e\text{-}cl(q_0) = \{q_0, q_1\}$

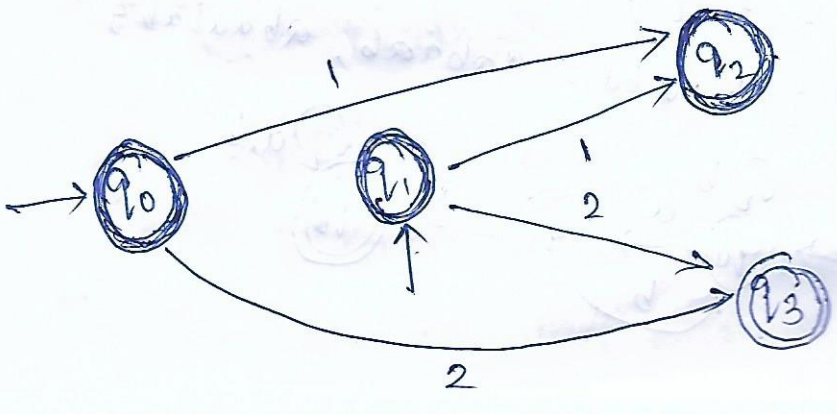
* final state : e-cl(q)

$e\text{-}cl(q_0) = \{q_0, q_1\}$ ✓

$e\text{-}cl(q_1) = \{q_1\}$ ✗

$e\text{-}cl(q_2) = \{q_2\}$ ✗

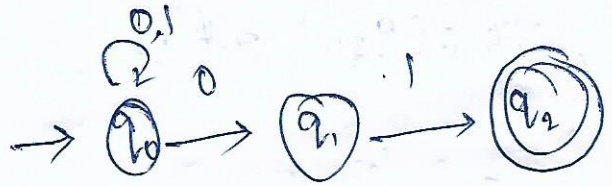
$e\text{-}cl(q_3) = \{q_3\}$ – final state



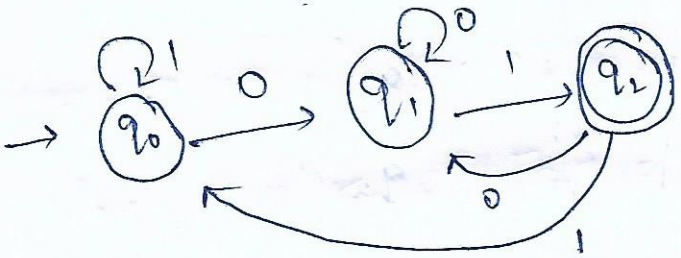
28/6/16

Design NFA that accepting all strings ending with 01 over sig $\Sigma = \{0,1\}$

NFA:

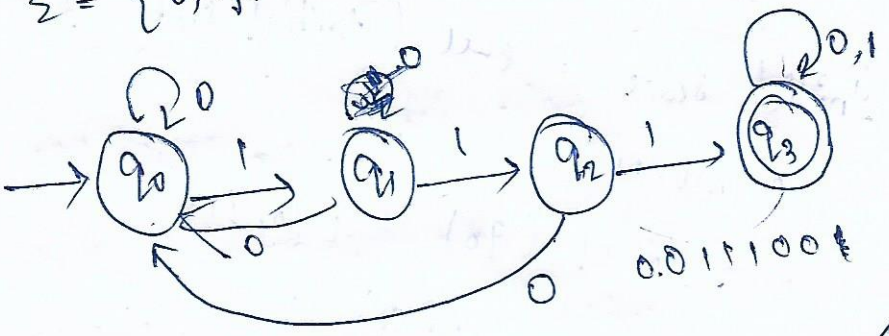


DFA:



⇒ A design DFA which accepts 111 as substring

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



0.0111001

0101101

0110

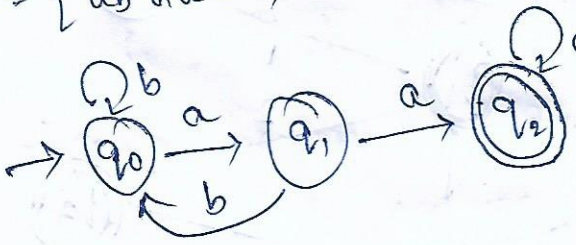
01110

29/6/16

* DFA with aa as a sub string

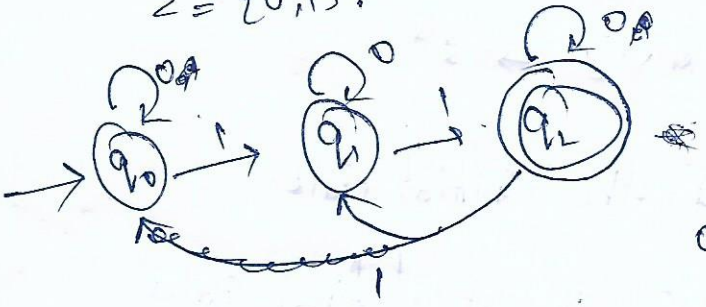
$\Sigma = \{a,b\}$

$L = \{abaaba, aabaab, abaabab\}$

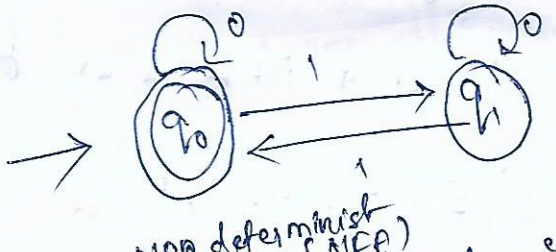


* Design a DFA with even number of 1's.

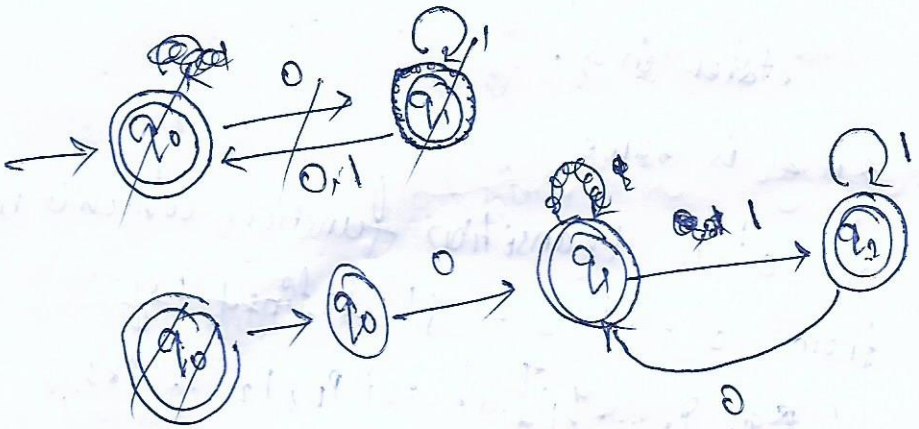
$\Sigma = \{0, 1\}$.



01010101



Design a non deterministic (NFA) finite automata in which every string start with 0 followed by any no. of 1's but having no two consecutive 0.



Theorem: If L is accepted by an NFA then there exists DFA that accepts L.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA where Q is finite non empty set of states

Σ is finite non empty set of i/p symbols.

δ is transition function which is mapping from : $Q \times \Sigma \rightarrow 2^Q$

q_0 is the initial state

$F \subseteq Q$ is set of final states

Now we construct a DFA is as follows

$$M' = (Q', \Sigma, \delta', [q_0], F')$$

where

1) $Q' = 2^Q$ (power set of elements)

$[q_1, q_2, \ldots q_j]$ is a single state for several states in Q.

2) Σ is same

3) δ' is transition function which is mapping from $\delta' = Q' \times \Sigma \rightarrow Q'$ defined by

$\delta'([q_1, q_2 \ldots q_j], a) = \{p_1, p_2 \ldots p_k\}$

if and only if

$$\delta(\{q_1, q_2 \ldots q_j\}, a) = \{p_1, p_2 \ldots p_k\}$$

4) q_0 is initial state (The initial state of NFA becomes initial state of DFA)

5) $F' \subseteq Q'$ which contains atleast one

element of F we have to show that

$$\delta'([q_0], a) = [p_1, p_2 \dots p_j]$$

if

$$\delta(\{q_0\}, x) = \{p_1, p_2 \dots p_j\}$$

we have to prove this by mathematical induction.

let $x' = a'$ of length 1 then

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

$$\delta'([q_0], a) = \delta(\{q_0\}, a)$$ this is true for length of string $|x| = 1$

we assume that it is true for x' of length n i.e $\delta'([q_0], x) = \delta(\{q_0\}, x)$.

Let the string be 'xa' which is of length $n+1$, we have to prove that.

$$\delta'([q_0], xa) = \delta'(\delta([q_0], x), a)$$

by the inductive hypothesis

$$\delta'([q_0], x) = [p_1, p_2 \dots p_j]$$

if and only if

$$\delta(q_0, x) = \{p_1, p_2 \dots p_j\}$$

but by defn of δ'

$\delta'([p_1, p_2 \ldots p_j], a) = [r_1, r_2 \ldots r_k]$

if and only if

$\delta(\{p_1, p_2 \ldots p_j\}, a) = \{r_1, r_2 \ldots r_k\}$

thus

$\delta'([q_0], xa) = [r_1, r_2 \ldots r_k]$

if and only if

$\delta(q_0, xa) = \{r_1, r_2 \ldots r_k\}$

which establish as hypothesis

$\delta'([q_0], xa) \in F'$

iff $\delta(\{q_0\}, xa\}$ should contain atle

one final state of F

Regular Expression

Regular expression consists of constants, ope

symbols that denote sets of strings a

operation.

Regular expression describe regular langu

in formal language theory.

Let L be a language that we de

$L^0 = \epsilon$ and $L^i = LL^{i-1}$ then the closure

L^* is defined by L^* and is defined as $L^* = \bigcup_{i=0}^{\infty} L^i$.

Let $\Sigma = \{0\}$

$L^0 = \epsilon$

$L^1 = L \cdot L^0 = 0\epsilon$

$L^2 = L \cdot L^1 = 00\epsilon$

$L^3 = L \cdot L^2 = 000\epsilon$

$\vdots$

$L = L^0 + L^1 + L^2 + \ldots + L^{10} + \ldots$

$= \{\epsilon, 0\epsilon, 00\epsilon, 000\epsilon \ldots\}$

$0^* = \{\epsilon, 0, 00, 000, 0000 \ldots\}$

→ The positive closure of L is denoted by L^+ and is defined as $L^+ = \bigcup_{i=1}^{\infty} L^i$

The positive closure of L is

$\Sigma = \{0\}$

$L^1 = L = 0$

$L^2 = L \cdot L^1 = 00$

$L^3 = L \cdot L^2 = 000$

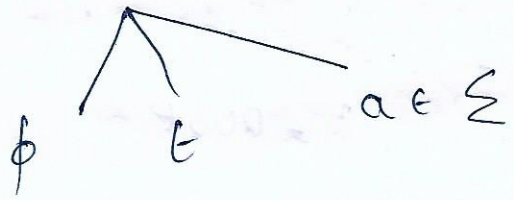
$\vdots$

$L^+ = L^0 + L^1 + L^2 + \ldots$

$= \{0, 00, 000 \ldots\}$

$0^+ = \{0, 00, 000 \ldots\}$

Regular expression



→ If r and s are two regular expressions denoting the languages R & S then r+s, rs, r^* are regular expressions.

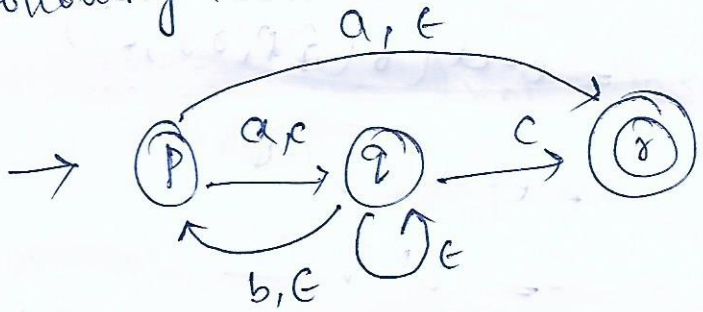
* design a finite automata (DFA) that accepts exa two 0's.

design a finite automata for the language that accepts the string containing at least 2 0's over $\Sigma = \{0, 1\}$

design a finite automata for the langua the string contains double 0's alwa over $\Sigma = \{0, 1\}$

design a finite automata to accept set of all strings not containing 101 substring.

Construct NFA without ε moves from the following NFA with ε moves.



δ =	a	b	c	ε
p	{q, r}	ϕ	q	r
q	ϕ	p	r	{p, q}
r	ϕ	ϕ	ϕ	ϕ

$\epsilon\text{-cl}(p) = \{p, r\}$

$\epsilon\text{-cl}(q) = \{p, q, r\}$

$\epsilon\text{-cl}(r) = \{r\}$

$\hat{\delta}(p, a) = \epsilon\text{-cl}(\delta(\epsilon\text{-cl}(p), a))$

$= \epsilon\text{-cl}(\delta(\{p, r\}, a))$

$= \epsilon\text{-cl}(\delta(p, a) \cup \delta(r, a))$

$= \epsilon\text{-cl}(\{q, r\} \cup \phi)$

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

$\hat{\delta}(p,b) = e\text{-}cl(\delta(e\text{-}cl(p),b))$

$= e\text{-}cl(\delta(\{p,q\},b))$

$= e\text{-}cl(\delta(p,b) \cup \delta(q,b))$

$= e\text{-}cl(\phi \cup \phi) = \phi$

$\hat{\delta}(p,c) = e\text{-}cl(\delta(\{p,q\},c))$

$= e\text{-}cl(\phi \cup \{b,d\}) =$

$= \{b,d\}$

$\hat{\delta}(q,a) = e\text{-}cl(\delta(\{p,q,r\},a))$

$= e\text{-}cl(\phi \cup \phi \cup \{r,b\}) =$

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

$\hat{\delta}(q,b) = e\text{-}cl(\delta(\{p,q,r\},b))$

$= e\text{-}cl(\phi \cup d \cup \phi) = \{p,r\}$

$\hat{\delta}(q,c) = e\text{-}cl(\delta(\{p,q,r\},c))$

$= e\text{-}cl(\phi \cup r \cup b) =$

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

$\hat{\delta}(r,a) = e\text{-}cl(\delta\{\{r\},a))$

$= \phi$

$\hat{\delta}(r,b) = e\text{-}cl(\delta(r,b)) = \phi$

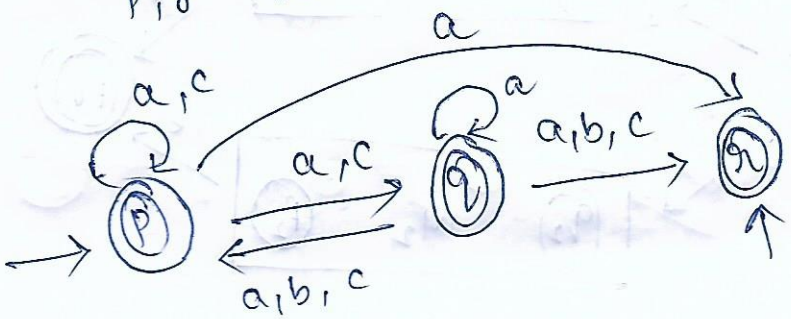
$\hat{\delta}(r,c) = \phi$

$\hat{\delta}$ =

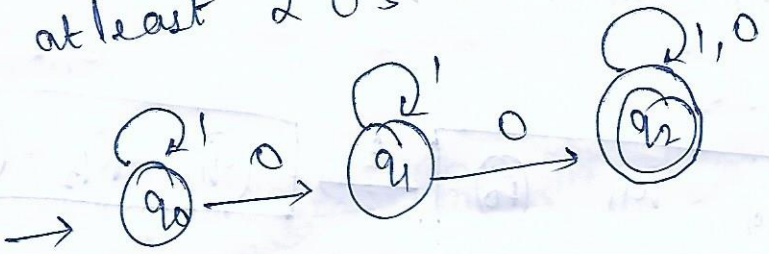
	a	b	c
p	{p, q, r}	ϕ	{p, q}
q	{p, q, r}	{p, r}	{p, q, r}
r	ϕ	ϕ	ϕ

p, q, r are final states

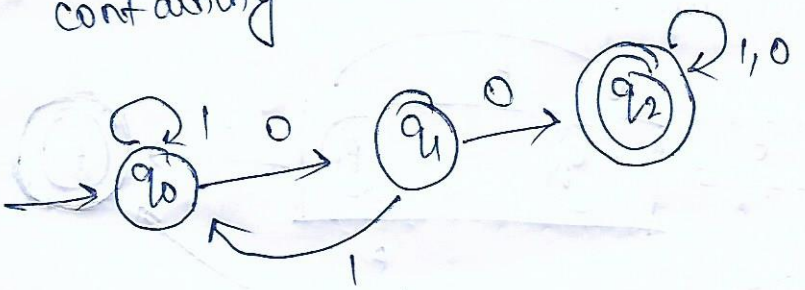
p, r are initial states



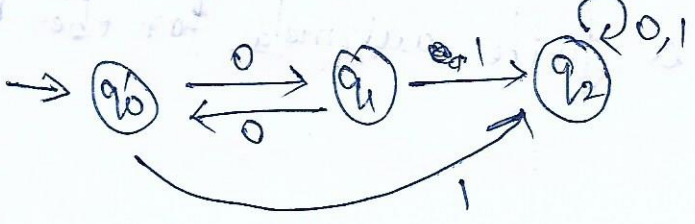
2) atleast 2 0's



3) containing 00's

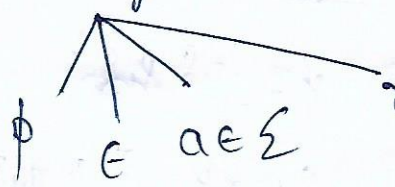


1) enactly 2 0's



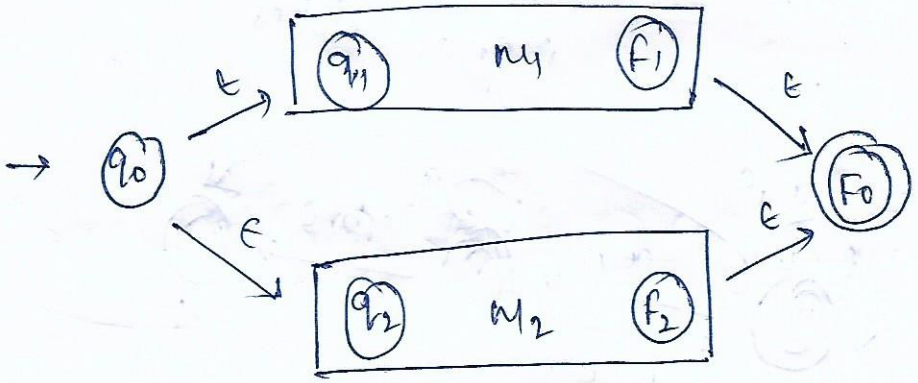
2/7/16

Regular Expression

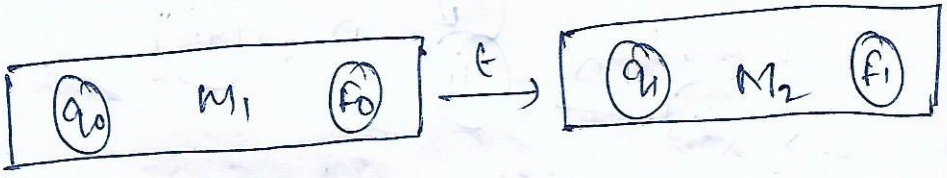


r & s two RES denoting the language R & S then r+s, rs are RES

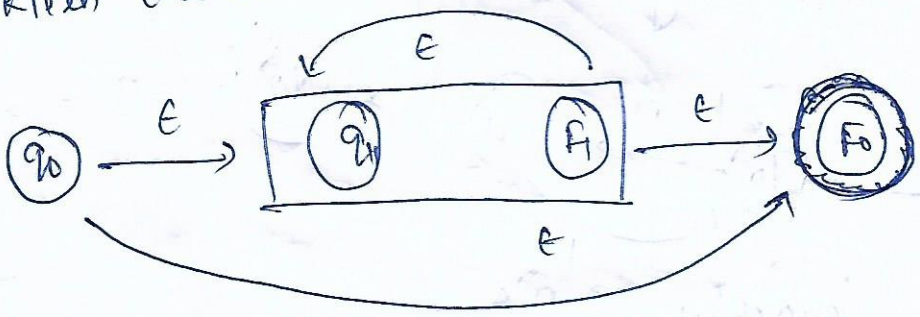
UNION : R+S : $M_1 + M_2$



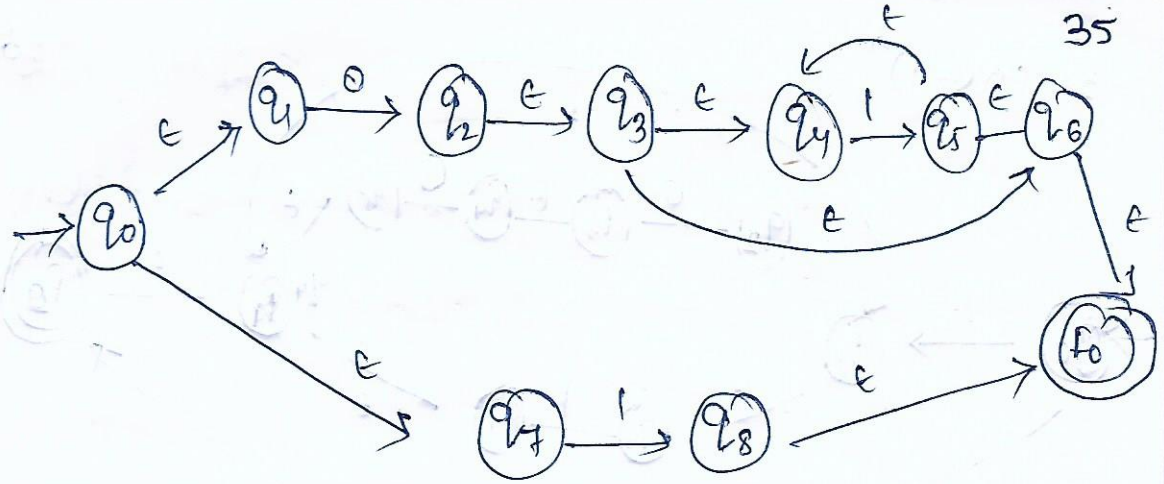
Concatination : $M_1 M_2 = RS$



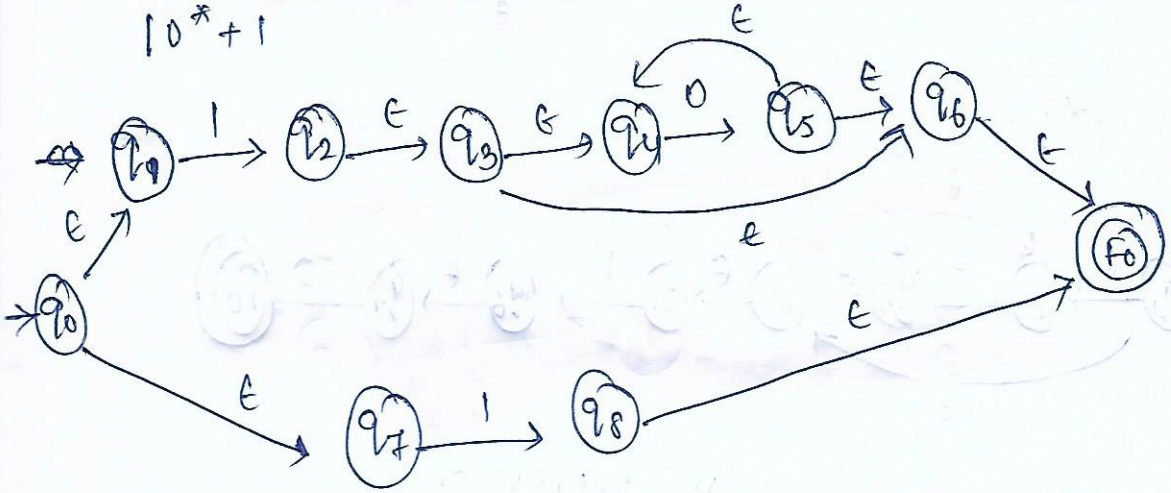
Kleen closure : $R^* = M^*$



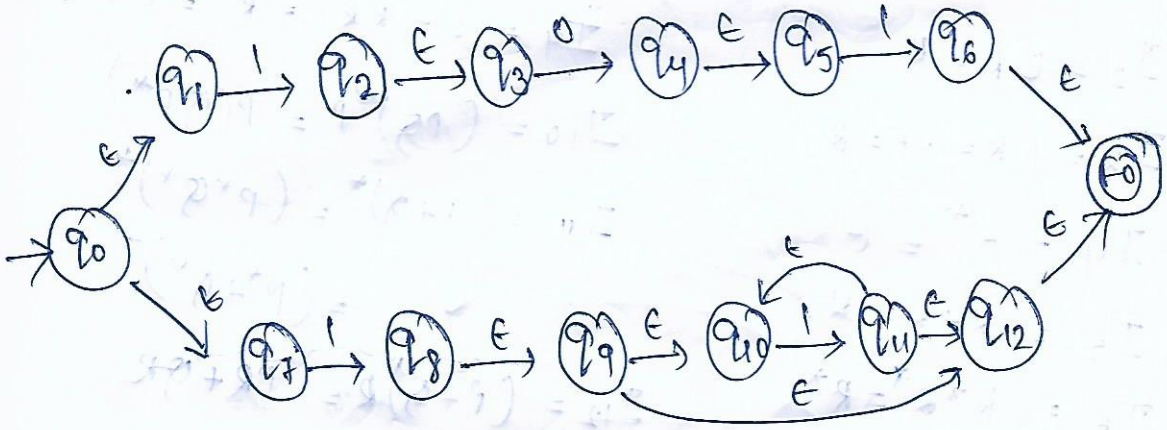
Construct a finite automata for the RE



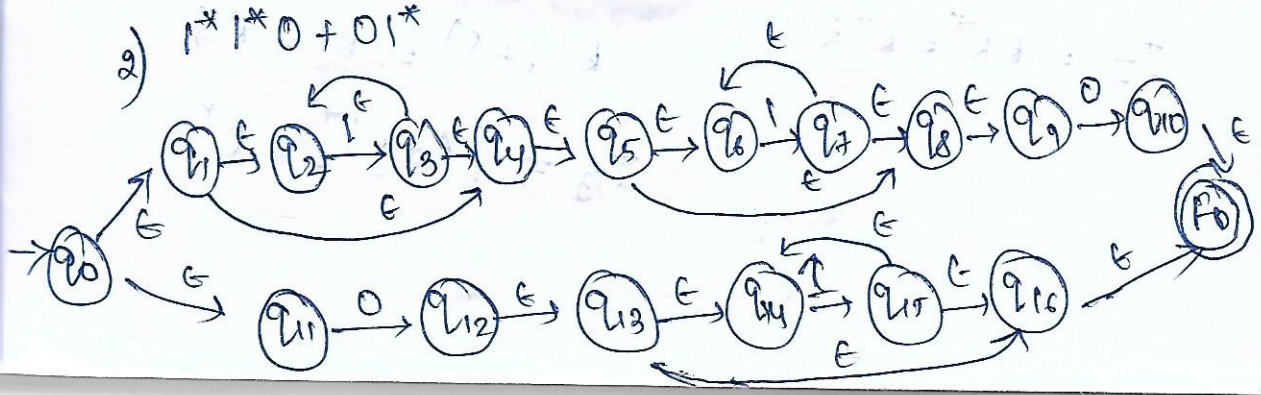
$10^* + 1$



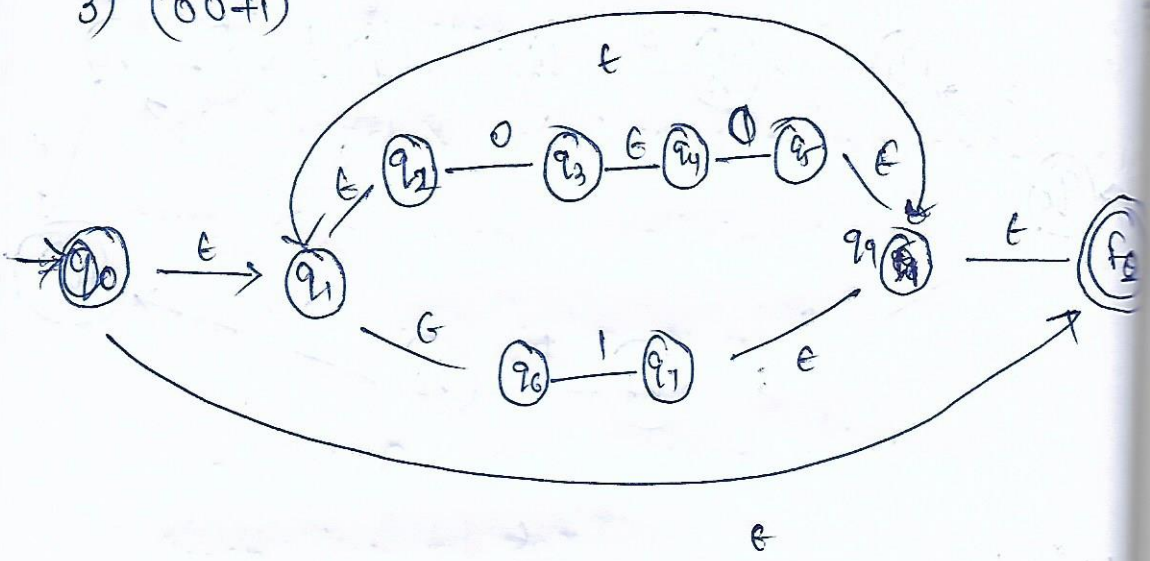
1) $101 + 11^*$



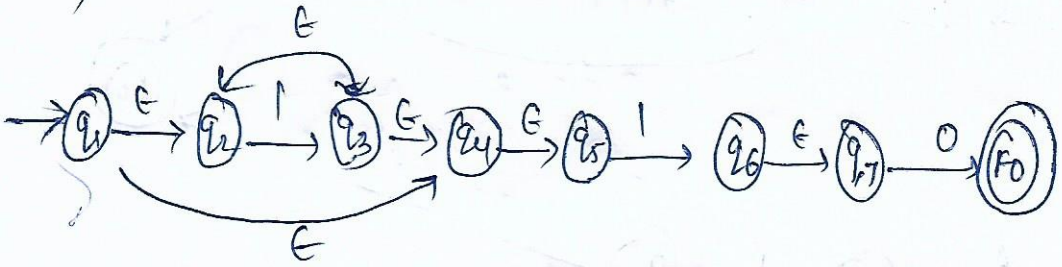
2) $1^*1^*0 + 01^*$



3) $(00+1)^*$



4) 1^*10



Identities of regular expression

$I_1 = \phi + R = R$

$I_2 = \phi R = R$

$I_3 = \epsilon R = R\epsilon = R$

$I_4 = \epsilon^* = \epsilon$

$I_5 = R + R = R$

$I_6 = R^* R^* = R^*$

$I_7 = RR^* = R^* R$

$I_8 = (R^*)^* = R^*$

$I_9 = \epsilon + RR^* = R^* = \epsilon +$

$I_{10} = (PQ)^* P = P(QP)^*$

$I_{11} = (P+Q)^* = (P^* Q^*)^*$

$= (P^* + Q^*)^*$

$I_{12} = (P+Q)R = PR + QR$

$R(P+Q) = RP + RQ$

$I_{13} = (\epsilon + R)^* = R^*$

show that

$(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$

$=0^*1(0+10^*1)^*$

$1(\epsilon+00^*)+1(\epsilon+0$

$=(1+00^*1)(\epsilon+(0+10^*1)^*(0+10^*1)$

$=(1+00^*1)(0+10^*1)^*$

$=1(\epsilon+00^*)(\epsilon+10^*1)^*$

$=100^*(0+10^*1)^*$

$=100^*(0^*+(0^*1)^*)^*$

$=100^*0^*+$

P^*R

$*\ \epsilon+(1^*(011)^*)(1^*(011)^*)^*$

$=(1^*(011)^*)^*$

$=1^*(011)^*(1+011)^*$

* Show that $0^*(1(\epsilon+(0+1)0^*1)^*(\epsilon+(0+1)0^*1)$

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

$=0^*1[(\epsilon+(0+1)0^*1)^*(\epsilon+(0+1)0^*1)+\epsilon)$

$=0^*1(\epsilon+(0+1)0^*1)^*$

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

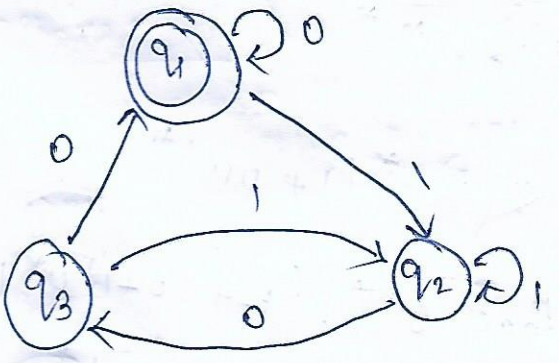
Arden's theorem:

Let P and Q be two regular expression over Σ, P doesn't contain ϵ then the following eqn in R w.r.t.a

$R = Q + RP$ has unique solution given by $R = QP^*$

$$R = Q + RP$$
$$= Q + (Q + RP)P$$
$$= Q + QP + RP^2$$
$$= Q + QP + QP^2 + QP^3 + \cdots$$
$$= Q(\epsilon + P + P^2 + P^3 + \cdots)$$
$$= QP^*$$

*



$q_1 = \epsilon + q_1 0 + q_3 0 \rightarrow (1)$

$q_2 = q_1 1 + q_2 1 + q_3 1 \rightarrow (2)$

$q_3 = q_2 0 \rightarrow (3)$

$q_2 = q_1 + q_2(1+01)$

$R = Q + RP$

$q_2 = q_1 1(1+01)^* \quad -(4)$

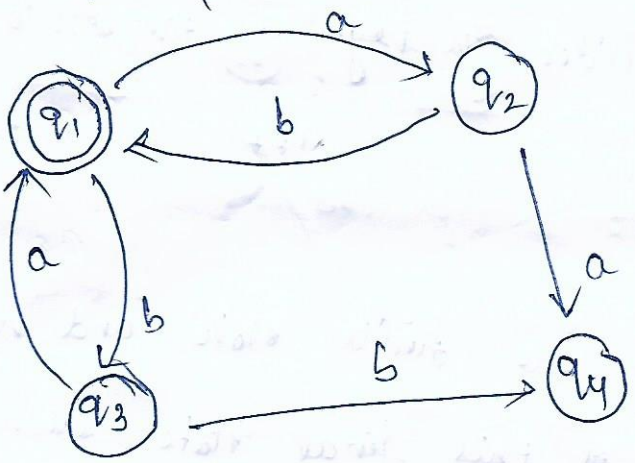
$q_1 = \varepsilon + q_1 0 + q_1 1(1+01)^* 00$

$= \varepsilon + q_1 0 + q_1 1(1+01)^* 00$

$q_1 = \varepsilon(0+1(1+01)^* 00)^*$

$= (0+1(1+01)^* 00)^*$

*



$q_1 = \varepsilon + q_2 b + q_3 a$

$q_2 = q_1 a$

$q_3 = q_1 b$

$q_4 = q_2 a + q_3 b$

$q_1 = \varepsilon + q_1 ab + q_1 ba$

$q_1 = \varepsilon + q_1 ab$

$R = Q + RP$

$q_1 = \epsilon(ab)^*$

05/07/16

Theorem:

Let r be a regular expression then there exists a NFA with ϵ transition that accepts L.

We prove the theorem by induction no. of operations.

Consider a regular expression 'r' then there is an NFA 'N' with transition having one finite state and no. of transitions out of this final state such that

$$L(N) = L(r)$$

(a) **0 operators**

The expression r must be ϵ, ϕ, from some symbol in Σ

$r = \epsilon$

$r = a$

$r = \phi$

(b) One or more operators :

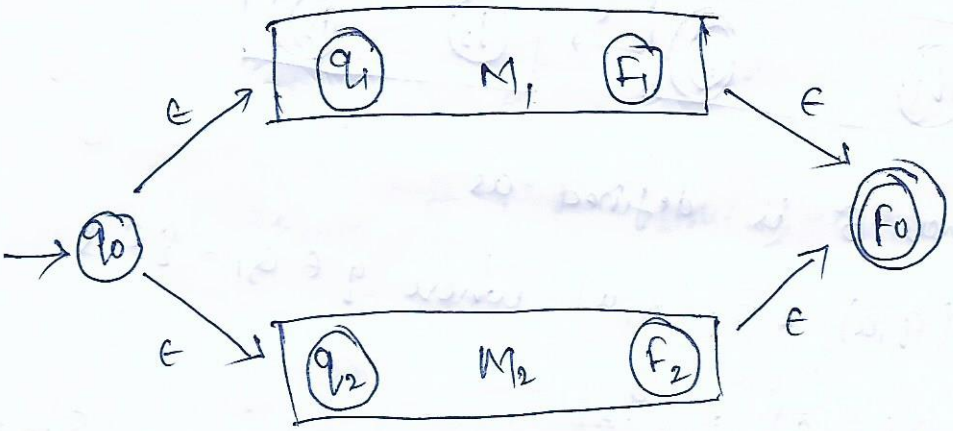
There are 3 cases depends on form of r

Case 1 : $r = r_1 + r_2$.

with r_1 & r_2 exits a NFA

$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, f_1)$ and

$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, f_2)$



1) $\delta(q_0, \epsilon) = \{q_1, q_2\}$

2) $\delta(q, a) = \delta_1(q_1, a)$ for q in $Q - \{f_1\}$ and a in $\Sigma_1 \cup \{\epsilon\}$

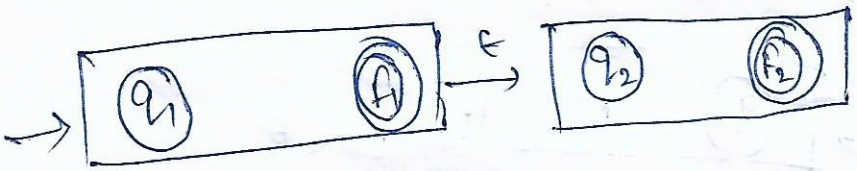
3) $\delta(q_1, a) = \delta_2(q_2, a)$ for q in $Q - \{f_2\}$ and a in $\Sigma_2 \cup \{\epsilon\}$

4) $\delta_1(f_1, \epsilon) = \delta_2(f_2, \epsilon) = \{f_0\}$

$\Rightarrow$ Any path from in M from q_0 to f_0 should begin either from q_1 or q_2 or ϵ.

$$L(M) = L(M_1) + L(M_2)$$

case 2: Let $r = r_1 r_2$ where r_1 & r_2 mu
have $< i$ operators then there exists an
NFA $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, f_1)$ and
$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, f_2)$.



where δ is defined as

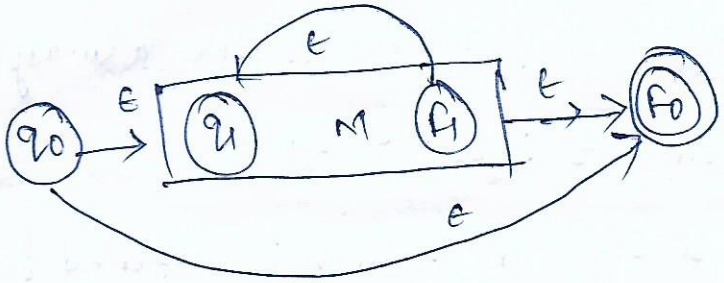
$\delta(q,a) = \delta_1(q,a)$ where $q \in Q_1 - \{f_1\}$

$\delta(f_1, \epsilon) = \{q_2\}$

$\delta(q,a) = \delta_2(q,a)$ where $q \in Q_2 - \{f_2$
$a \in$

Every path in M from q_1 & f_2 is a path
labelled by the string x from q_1 to f_1
followed by an edge f_1 to q_2 on ϵ
then followed by some string y fro
q_2 to f_2.

Case 3: $r = r_1^*$ with r_1 must have $< i$ oper
then there exists an NFA $M_1 = (Q$
$q, F_1)$ such that $L(M_1) = L(r_1^*)$



$M = (Q \cup \{q_0, f_0\}, \Sigma, \delta, q_0, f_0)$

where δ is defined as

$\delta(q_0, \epsilon) = \{q_1, f_0\}$

$\delta(f_1, \epsilon) = \{q_1, f_0\}$

$\delta(q, a) = \delta_1(q, a)$ where $q \in Q - \{f_1\}$

Any path from q_0 to f_0 consists either a path from q_0 to f_0 on ϵ or any other transition on M_1 to reach f_1 followed by reading ϵ, we can get back to q_1 or go to f_0. Hence $L(M) = L(r^*)$

08/07/16

Moore Machine

A moore machine is six tuple notation where $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Q = finite non empty set of states

Σ = finite non empty set of i/p symbols.

Δ = the output alphabet.

δ = transition function which always ma

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

λ = is a o/p fun^c which is defined from

$\lambda = Q \to \Delta$

q_0 = Initial state.

In moore machine the value of output depends on the present state.

Mealy machine

A Mealy machine is a six tuple

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

Q = finite non empty set of states.

Σ = finite non empty set of i/p symb

Δ = the o/p alphabet.

δ = transition fun^c which always

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

λ = is a o/p fun^c which is defined

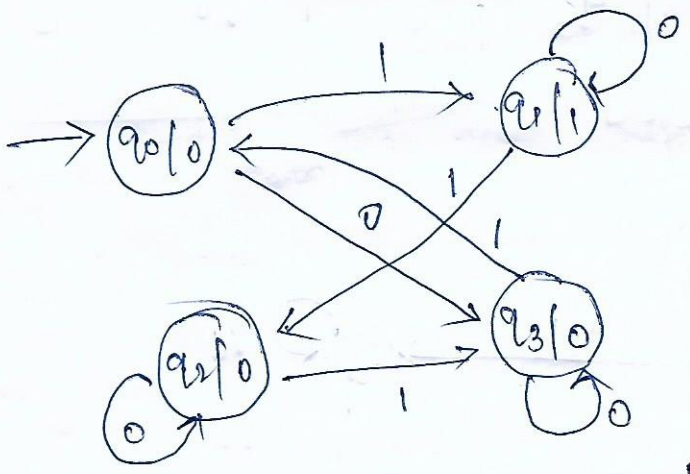
$Q \times \Sigma \to \Delta$

q_0 = Initial state.

Moore machine (Transition Table)

Present state	Next state a=0	Next state a=1	output
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Transition diagram



n i/p

i/p string 0111

$q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2$

o/p 0 0 0 1 0

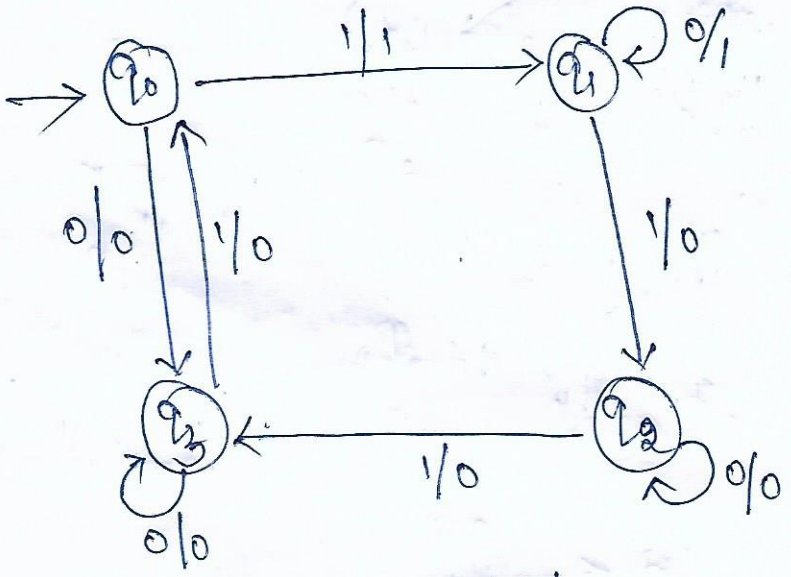
n+1 o/p

i/p string 0101

o/p: 00000

Mealy Machine

Present state	Next state			
	a=0		a=1	
	state	o/p	state	o/p
→ q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0



i/p string 0111

$q_0 \xrightarrow{0} q_3 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2$

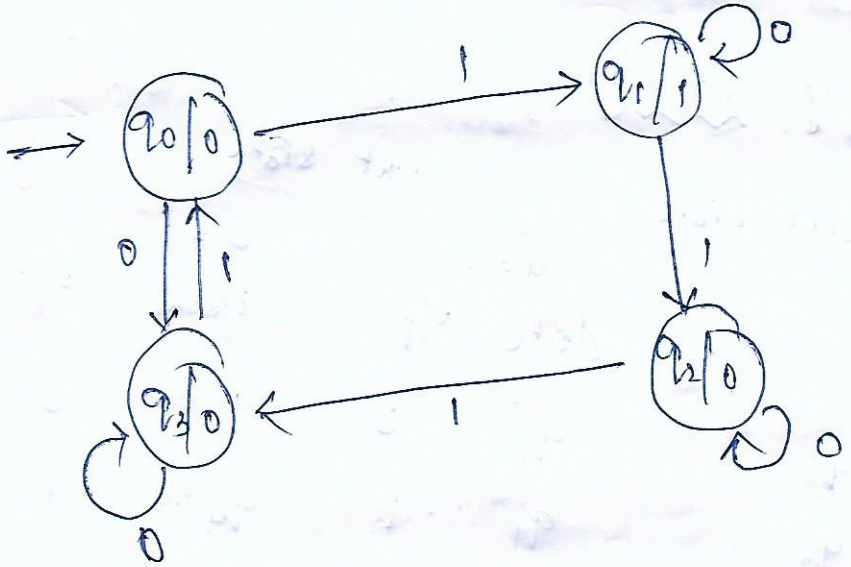
o/p: 0 0 1 0

n i/p's gives n o/p's

Moore to Mealy conversion 9/07/16

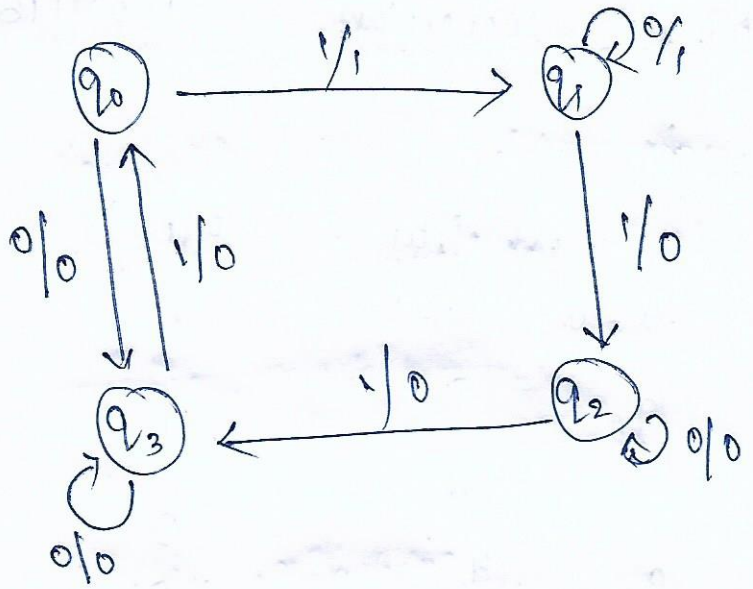
Moore machine

Present state	Next states $a=0$	Next states $a=1$	O/P (b)
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0



Mealy Machine

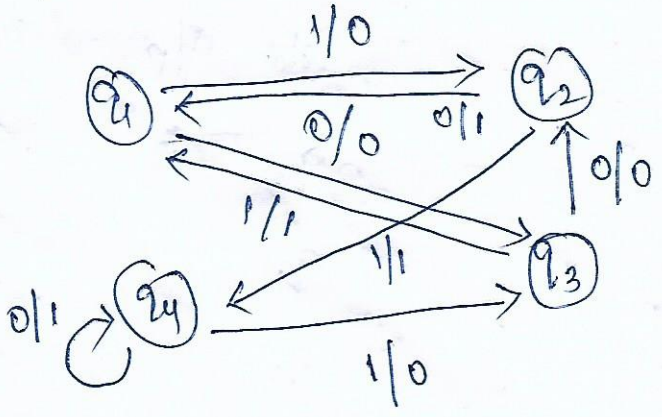
Present state	Next state $a=0$ state	Next state $a=0$ O/P	Next state $a=1$ state	Next state $a=1$ O/P
→ q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0



Mealy to Moore Conversion

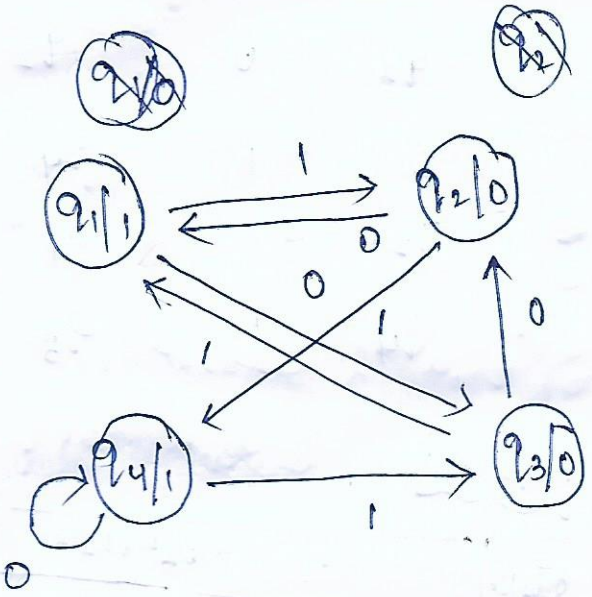
Mealy machine

Present state	Next state a=0		Next state a=1	
	state	o/p	state	o/p
→ q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	1
q_3	q_2	0	q_4	1
q_4	q_4	1	q_3	0



Moore Machine

Present state	Next state		o/p(z)
	a=0	a=1	
→ q_1	q_3	q_2	1
q_2	q_1	q_4	0
q_3	q_2	q_1	0
q_4	q_4	q_3	1



i) We look into the next state column for any state q_i and determine the number of different output associated with q_i in that column.

Conversion of given Mealy to moore.

Consider the Mealy machine described by the transition table. Construct moore machine which is equivalent the Mealy machine.

Mealy Machine

Present state	Next state			
	a=0		a=1	
	state	o/p	state	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

Modified Mealy Machine

Present state	Next state			
	a=0		a=1	
	state	o/p	state	o/p
→ q_1	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

Moore Machine

Present state	Next state $a=0$	Next state $a=1$	O/P
$\rightarrow q_1$	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

We look into the next state column for any state say q_i and determine the number of different outputs associated with q_i in that column. We split q_i into several different states. The number of such states being equal to the number of different outputs associated with q_i.