18/19

Module I

- Symbol.

  ↓

- Alphabets

  → finite set of symbols.

* In TOC, alphabet is represented as Σ

  ↓

- Strings

  → combined form of alphabets.

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

Powers of Alphabet.

Σ^1 = set of all string of length 1 over Σ

eg. $\Sigma^1 = \{a, b\}$

Σ^2 = set of all string of length 2 over Σ

eg. $\Sigma^2 = \{aa, ab, ba, bb\}$

⋮

Σ^n = set of all string of length n over Σ

Σ^0 = set of all string of length 0 over Σ

- $\{\underset{\text{epsilon}}{\epsilon}\}$ → empty string.

- $|\epsilon| = 0$

Kleene Closure (Σ^*)

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots \Sigma^n \dots$$

$$= \{\epsilon, a, b, aa, ab, ba, bb, \dots\}$$

= set of all strings of all lengths over Σ

or

= set of all possible strings over Σ

positive closure (Σ^+)

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \dots \Sigma^n \dots$$

$$= \Sigma^* - \{\epsilon\}$$

$|\Sigma| \to$ no. of symbols over Σ (length)

no of strings of length, n = $|\Sigma|^n$.

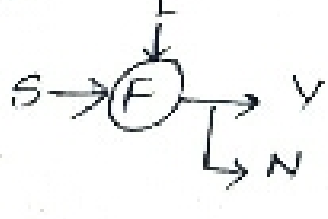
Language:

$$L \subseteq \Sigma^*$$

→ can be finite or infinite.

$\Sigma^* \to$ Mother of a language bcz all $L \subseteq \Sigma^*$

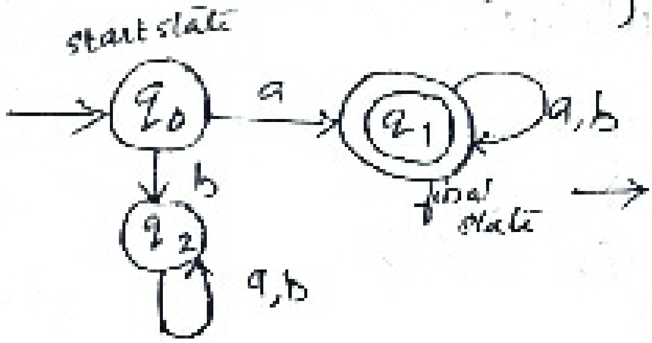
finite representation



$F \to$ finite string

Eg:- L = starts with a

= {a, ab, aa, ...}



→ Finite Automata

I/P S = aab

$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} \underline{q_1}$ (final state)   String is accepted.

I/P S = bab

$q_0 \xrightarrow{b} q_2 \xrightarrow{a} q_2 \xrightarrow{b} q_2$ String is rejected.

Finite Automata

- → FA with O/P
  - → Moore Machine   O/P fun: $\lambda : \delta \to \Delta$
  - → Mealy Machine
- → FA without O/P
  - → DFA
  - → NFA
  - → ε-NFA

Finite Automata

* Simplest model of computation.
* Limited amount of memory.

It is defined as $(Q, \Sigma, q_0, \delta, F)$

$Q \rightarrow$ set of states

$\Sigma \rightarrow$ i/p alphabet

$q_0 \rightarrow$ start state

$\delta \rightarrow$ transition function

$F \rightarrow$ set of final states.

:/19

DFA

→ Deterministic Finite Automata.

→ A DFA consists of

1. A finite set of states denoted by Q
2. A finite set of input symbols called alphabet denoted by Σ
3. A transition function that takes as arguments a state and an input symbol and returns a state denoted by δ. $\delta : Q \times \Sigma \rightarrow Q$
4. A start state, one of the states in Q
5. A set of final or accepting states F, $F \subseteq Q$

Therefore, a DFA is a five tuple notation.
If M is a DFA, then its five tuple notation is $M = (Q, \Sigma, \delta, q_0, F)$

The term 'Deterministic' refers to the fact that on each input symbol there is one and only one state to which the automata can transition from its current state.

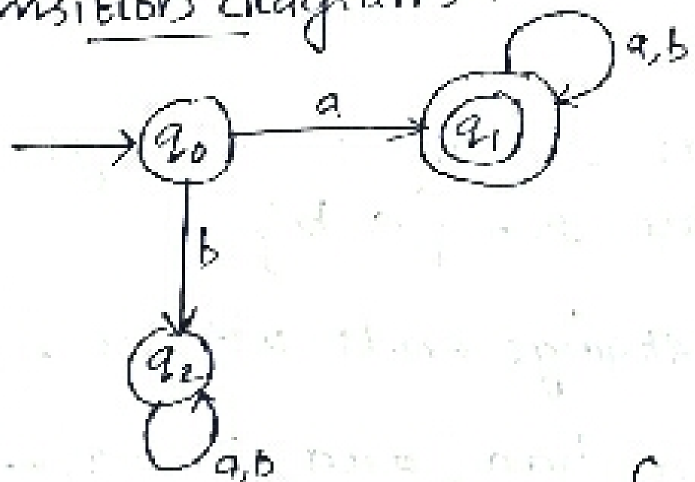
Q

Design a DFA which accepts all strings starts with 'a' over the alphabet $\Sigma = \{a, b\}$

L = set of all strings start with a...

= {a, aa, ab, aaa, aba, ...}

Transition diagram.



Transition table.

δ	a	b
$\rightarrow q_0$	q_1	q_2
$* q_1$	q_1	q_1
q_2	q_2	q_2

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

$\delta(q_0, a) = q_1$

$\delta(q_0, b) = q_2$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_1$

$\delta(q_2, a) = q_2$

$\delta(q_2, b) = q_2$

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

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

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

start state $= q_0$

$F = \{q_1\}$

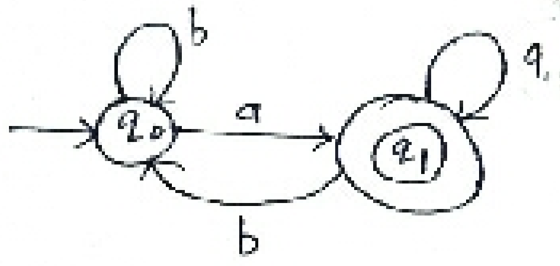
Q

Design a DFA that accepts all strings ends with 'a' over the $\Sigma = \{a, b\}$

L = set of all strings ends with a.

$= \{a, ba, aa, baa, aaa, bbaa, \ldots\}$

Transition diagram



	a	b
$\rightarrow q_0$	q_1	q_0
$* q_1$	q_1	q_0

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

$Q = \{q_0, q_1\}$

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

start state q_0

$F = \{q_1\}$

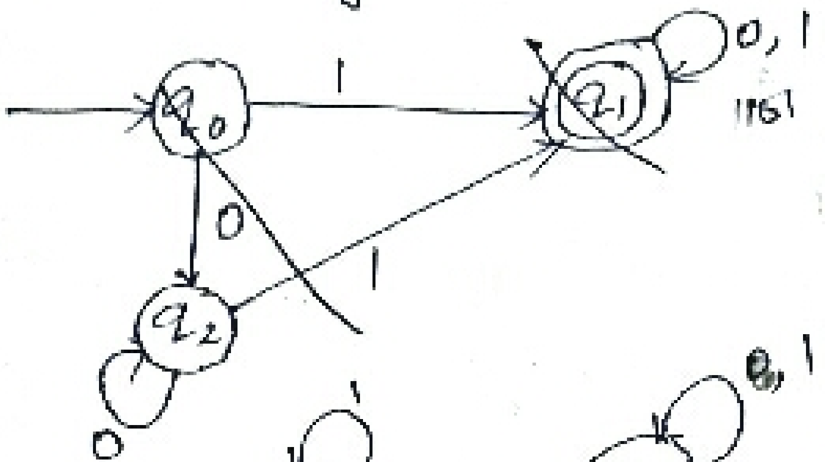
Q

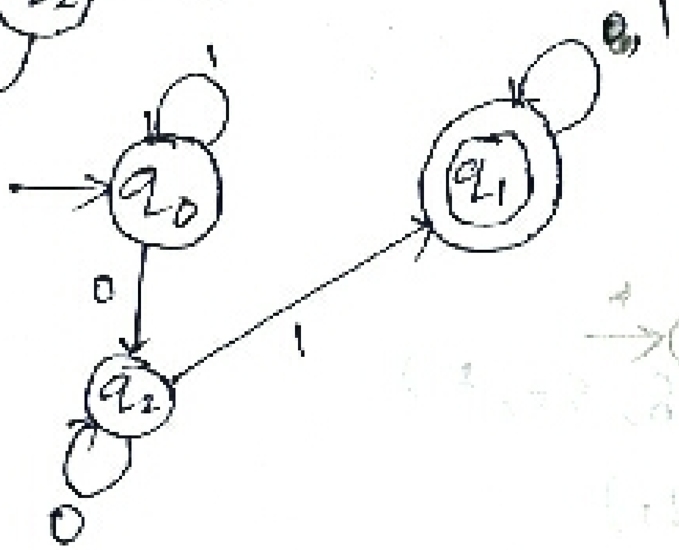
Design a DFA that accepts set of all strings end with ~~[illegible]~~ '01' over the alphabet $\Sigma = \{0, 1\}$

L = set of all strings ends with 01

$= \{01, 101, 001, 1001, \ldots\}$

Transition diagram





	0	1
$\rightarrow q_0$	q_2	q_0
$* q_1$	q_1	q_1
q_2	q_2	q_1

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

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

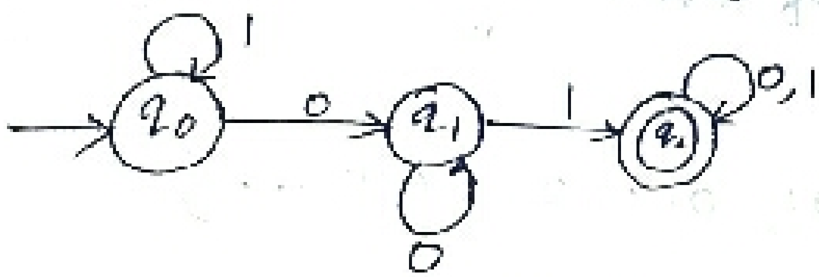
$\Sigma = \{0, 1\}$

Start state $= q_0$

$F = \{q_1\}$

Q) Construct a DFA accepting all strings with a substring 01 over the alphabet $\Sigma = \{0,1\}$

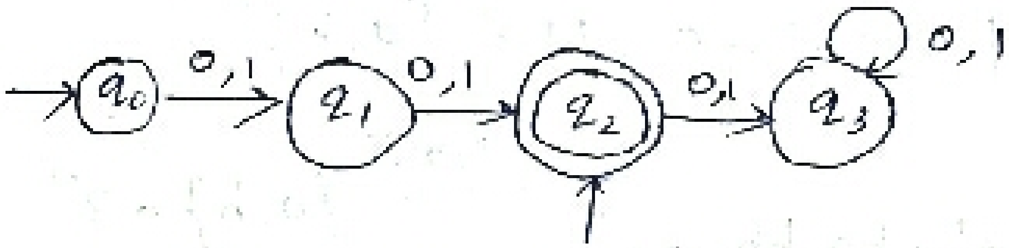
$L = \{01, 001, 1101, \dots\}$



Q) Construct a DFA to accept the following language over the alphabet $\Sigma = \{0,1\}$

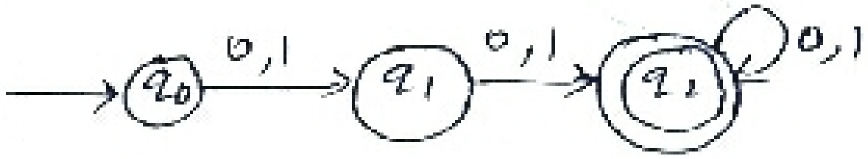
$L = \{ w \,/\, |w| = 2,\ w \text{ is a string} \}$.

$L = \{00, 01, 10, 11\}$



Q) $L = \{ w \,/\, |w| \geq 2,\ w \text{ is a string} \}$

set of all strings of length atleast 2.

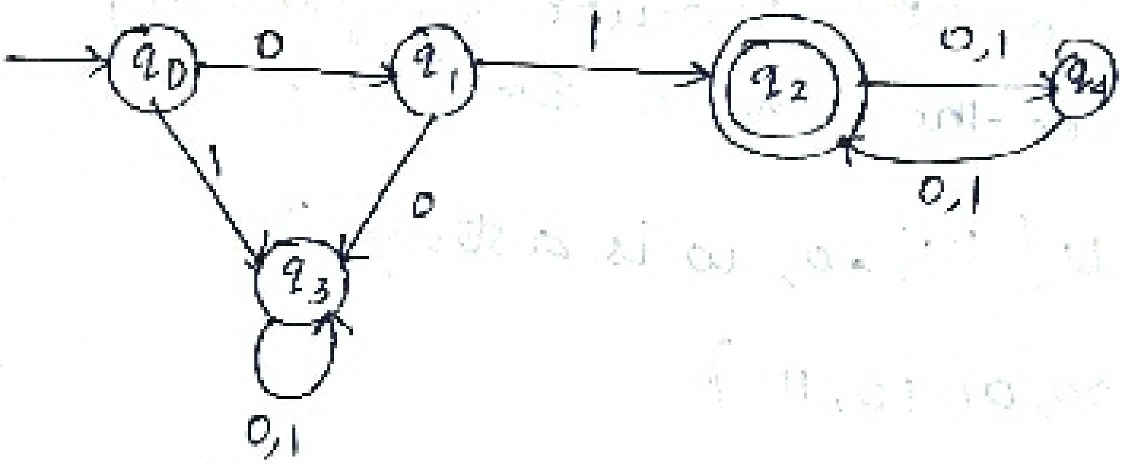


Q) $L = \{ w \,/\, |w| \leq 2,\ w \text{ is a string} \}$

$L = \{0, 1, 00, 01, 10, 11\}$

Q) $L = \{ w \mid w$ is of even length and begins with 01$\}$

$L = \{01, 0101, 010101, 0111, \ldots\}$

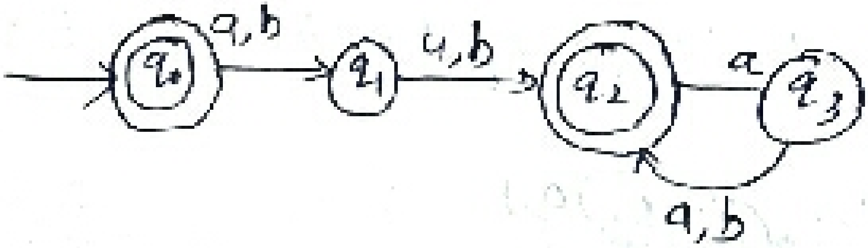


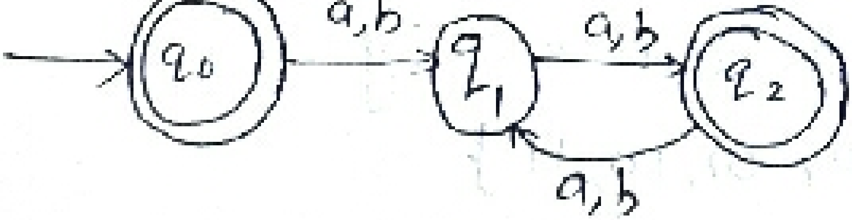
Q) $L = \{ w / |w| \bmod 2 = 0,\ w \in \{a, b\}^* \}$

$L = \{ \epsilon, aa, ab, ba, bb, \ldots \}$

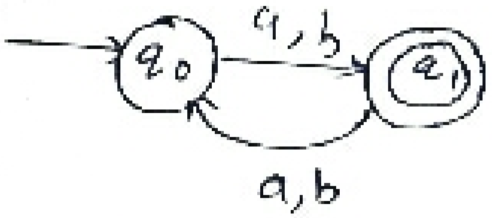
Note $\{a, b\}^* = \Sigma^*$

$L_2 = \Sigma^* - L$

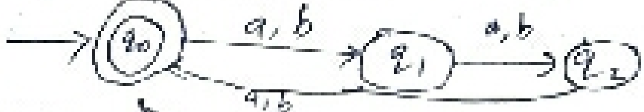




Q) $L = \{ w \mid |w| \bmod 2 = 1,\ w \in \{a,b\}^* \}$

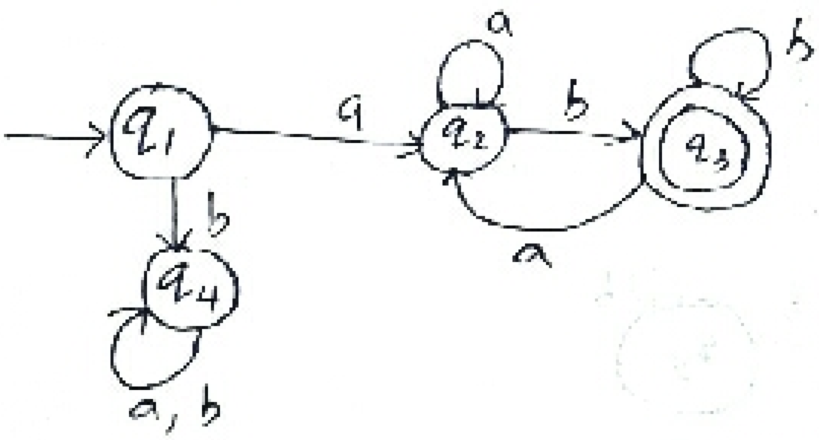


Q) $L = \{ w \mid |w| \bmod 3 = 0$ $L = \{ \epsilon, aaa, bbb, aaaaaa \}$



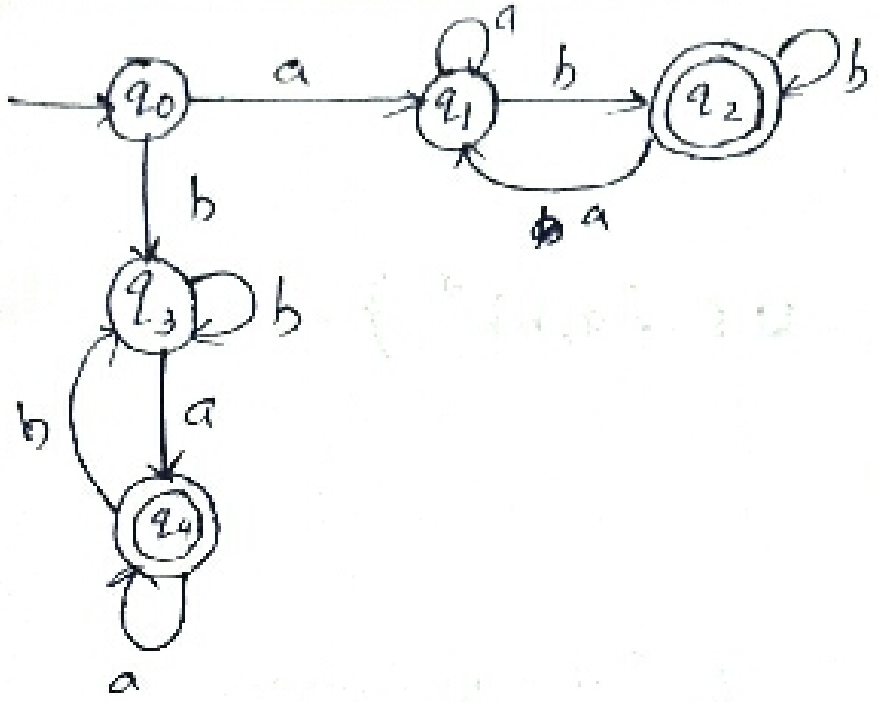
Q) Construct a DFA which accepts set of all strings over $\Sigma = \{a, b\}$ which starts with a and end with b.

$L = \{ab, abb, aabb, \dots\}$



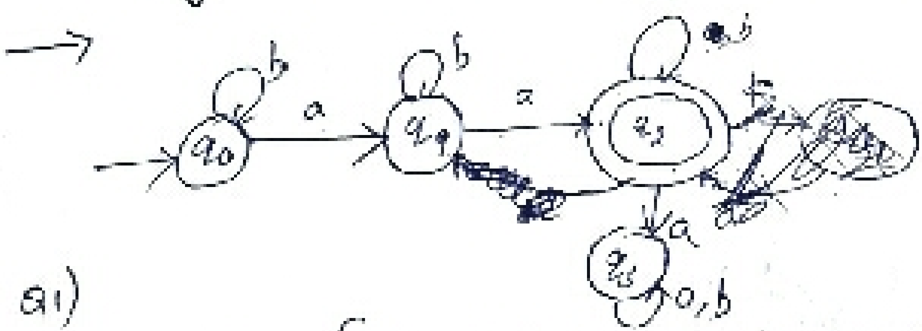
Q) Construct a DFA which accepts set of all strings which starts and ends with different symbol when $\Sigma = \{a, b\}$

$L = \{ab, ba, aab, baa, \dots\}$



Q) Starts and ends with same symbol.

Q) Construct a DFA which accepts set of all strings with exactly two a's



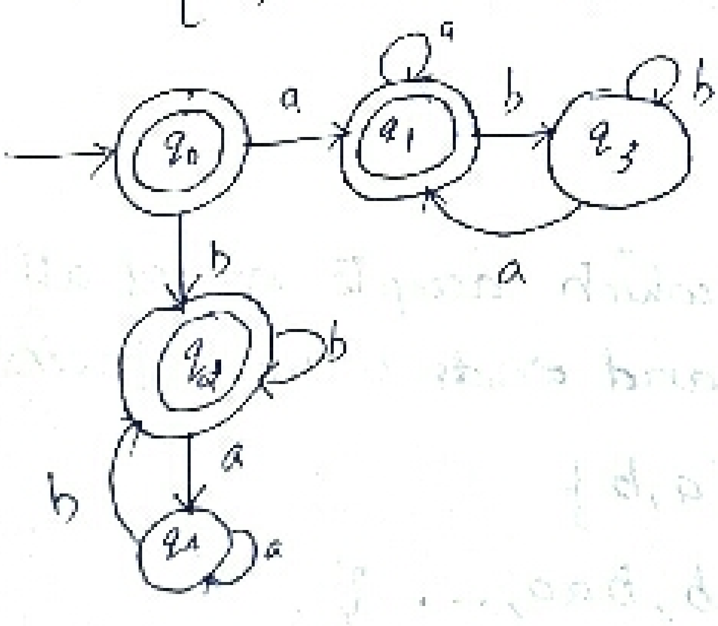
babab

bbbbaa

baba

Q1)

$L = \{\epsilon, a, b, aa, bb, aba, aaa, bab, \ldots\}$



16/8/17

1) DFA that accepts 101 as substring over $\Sigma = \{0,1\}$

00101 0 0
1 101

2) DFA that doesnot have 101 as substring over $\Sigma = \{0,1\}$

3) DFA that accepts exactly one 'a'. $\Sigma = \{a,b\}$.

4) DFA that accepts even no. of 'a's $\Sigma = \{a,b\}$.

5) Design a DFA that accepts all the strings in which even no. of 0's & 1's. $\Sigma = \{0,1\}$

$\{\epsilon, 0011, \ldots\}$

6) Design a DFA that accepts a language $L = \{a^n b, n \geq 0\}$ over $\Sigma = \{a,b\}$

$L = \{b, ab, aab, \ldots\}$

Extended Transition Function ($\hat{\delta}$)

It describes what happens when we start in any state and follow any sequence of inputs (ie a string).

The Extended transition function is a function that takes a state Q and a string w and returns a state P — the state that the automata reaches when starting in state Q and processing the string w.

1) $\hat{\delta}(q, \epsilon) = q$

2) if $w = xa$,

$$\hat{\delta}(q, w) = \hat{\delta}(q, xa)$$
$$= \hat{\delta}(\hat{\delta}(q, x), a)$$

3) A string s is to be accepted by a DFA ~~M = 5 tuples~~ $M = \{\Sigma, \delta, q_0, F, Q\}$ if $\hat{\delta}(q_0, s) = P$ where P is an element of F

We can define the language accepted by $M = (\Sigma, \delta, q_0, F, Q)$ ie $L(M) = \{ W / \hat{\delta}(q_0, W) \text{ is in } F \}$

The language accepted by finite automata is called regular language.

Non-Deterministic Finite Automata (NFA) (without ϵ-moves)

The NFA differs from the DFA in that the NFA can have any number of transitions (including 0) to next states from a given state on a given input symbol. That is, the difference between DFA and NFA is in the type of transition functions.

Formal definition:

An NFA 'M' is represented as a 5 tuple like a DFA i.e $M = (Q, \Sigma, \delta, q_0, F)$

$Q \to$ finite set of states

$\Sigma \to$ i/p alphabet

$q_0 \to$ start state

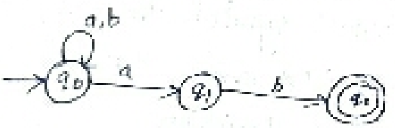
$F \to$ set of final state

$\delta \to$ transition function which is a function that takes a state in Q and an input symbol in Σ as argument and returns a subset of Q.

$\delta : Q \times \Sigma \to 2^Q$

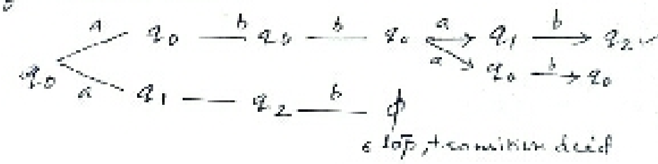
NFA accepting all strings ends with ab. over $\Sigma = \{a, b\}$

$L = \{ab, aab, bab, \ldots\}$



δ	a	b
$\to q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
$* q_2$	ϕ	ϕ

if $w = abbab$.



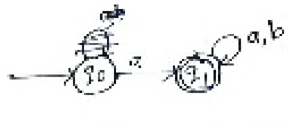
$Q = \{q_0, q_1, q_2\}$

$2^Q = \{\phi, \{q_1\}, \{q_0\}, \{q_2\}, \{q_0, q_2\}, \{q_0, q_1\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

2/1/19

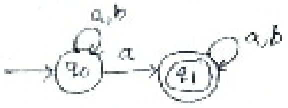
Q NFA accepting all strings starts with a over $\Sigma = \{a, b\}$

$L = \{a, ab, aab, \ldots\}$



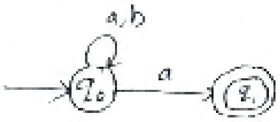
Q) accepting all strings contains a.

L = { a, ab, aa, bab, ... }



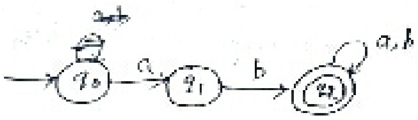
3) NFA accepting all strings ends with a

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



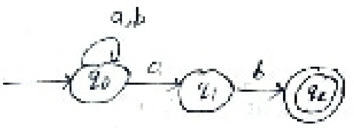
4) NFA accepting all strings
(i) starts with ab
(ii) contains ab
(iii) ends with ab.

(i)



(ii)

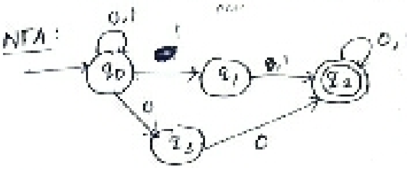
(iii)



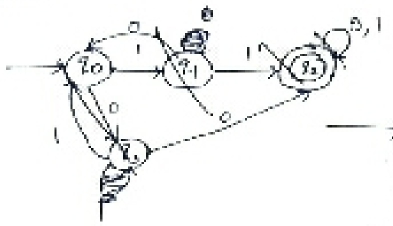
5) NFA to accept set of all strings with 2 consecutive 1's or 0's once. Σ = {0,1}

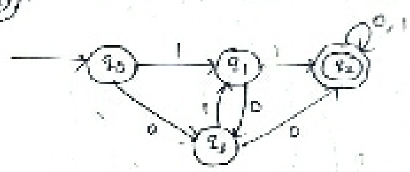
L = { 11, 00, 1001, 0011, ... }

NFA:



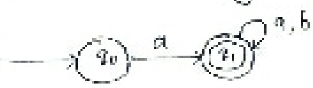
DFA:



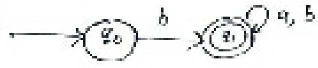


6) NFA accepting all strings
(i) starting with a
(ii) not starting with a.

(i)

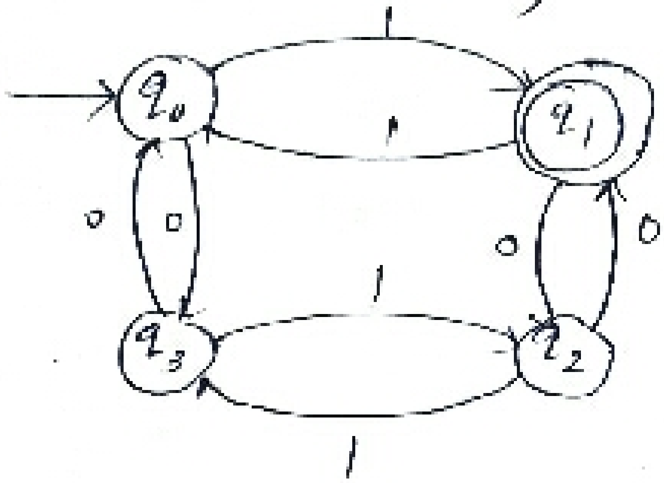


(ii)

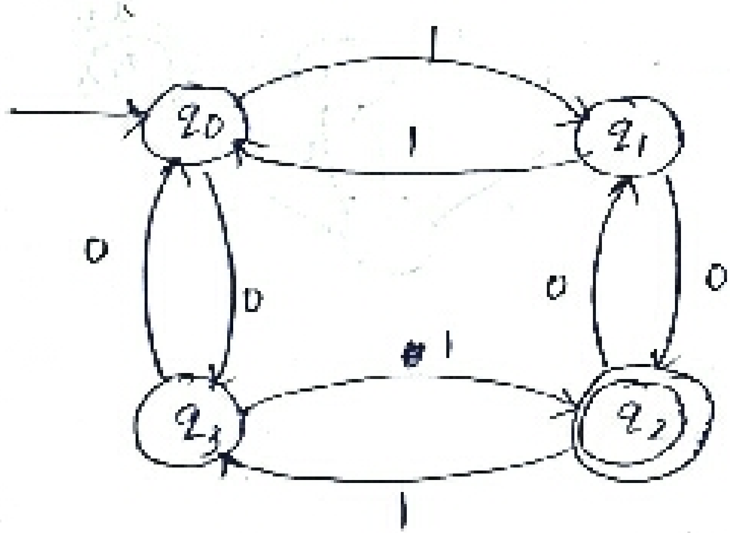


7) DFA even no of 0's & odd no. of 1's

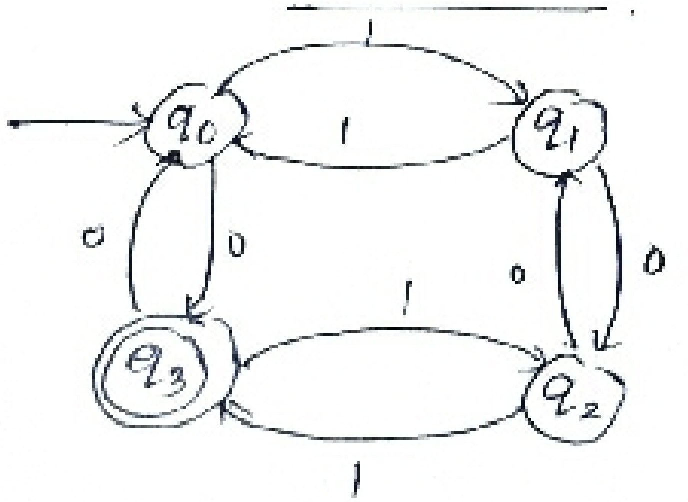
L = { 1, 001, ...}   00111



DFA odd no of 0's & 1's



DFA odd no. of 0's & even no of 1's



22/6/19

Language of an NFA.

If $M = (Q, \Sigma, \delta, q_0, F)$ is an NFA then

$L(M) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$

Equivalence of NFA and DFA.

1. NFA is easier to design than DFA
2. NFA has less no. of states when compared to the corresponding DFA
3. Every DFA is NFA but not viceversa.
4. Both NFA and DFA have same power and each NFA can be translated into a DFA
5. There can be multiple final states in both DFA and NFA.
6. NFA is more of a theoritical concept
7. DFA is used in lexical analysis in compiler.

1) Conversion of NFA to DFA

1) Convert the NFA $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ to DFA.

δ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$* q_1$	ϕ	$\{q_0, q_1\}$

Take $\{q_0\} - A$

$\delta(A,0) = \delta(\{q_0\},0) = \{q_0, q_1\} - B$

$\delta(A,1) = \delta(\{q_0\},1) = \{q_1\} - C$

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

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

$\delta(C,0) = \delta(\{q_1\},0) = \phi - D$

$\delta(C,1) = \delta(\{q_1\},1) = \{q_0, q_1\} - B$

$\delta(D,0) = \phi - D$

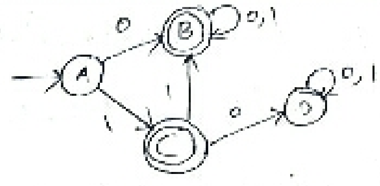
$\delta(D,1) = \phi - D$

DFA

$Q = \{A, B, C, D\}$, $\Sigma = \{0,1\}$

$q_0 = A$

$F = \{B, C\}$



Q) Convert the given NFA to DFA

δ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_1\}$
q_2	$\{q_3\}$	$\{q_3\}$
$* q_3$	ϕ	$\{q_2\}$

Take $\{q_0\} - A$

$\delta(A,a) = \delta(\{q_0\},a) = \{q_0, q_1\} - B$

$\delta(A,b) = \delta(\{q_0\},b) = \{q_0\} - A$

$\delta(B,a) = \delta(\{q_0, q_1, q_2\},a) = \{q_0, q_1, q_2, q_3\} - C$

$\delta(B,b) = \delta(\{q_0, q_1, q_2\},b) = \{q_0, q_1, q_3\} - D$

$\delta(C,a) = \delta(\{q_0, q_1, q_2, q_3\},a) = \{q_0, q_1, q_2, q_3\} - C$

$\delta(B,a) = \delta(\{q_0, q_1\},a) = \{q_0, q_1, q_2\} - C$

$\delta(B,b) = \delta(\{q_0, q_1\},b) = \{q_0, q_1\} - B$

$\delta(C,a) = \delta(\{q_0, q_1, q_2\},a) = \{q_0, q_1, q_2, q_3\}$ D

$\delta(C,b) = \delta(\{q_0, q_1, q_2\},b) = \{q_0, q_1, q_3\} - E$

$\delta(D,a) = \delta(\{q_0,q_1,q_2,q_3\}, a) = \{q_0,q_1,q_2,q_3\}$ - D

$\delta(D,b) = \delta(\{q_0,q_1,q_2,q_3\}, b) = \{q_0,q_1,q_2,q_3\}$ - D

$\delta(E,a) = \delta(\{q_0,q_1,q_3\}, a) = \{q_0,q_1,q_2\}$ - C

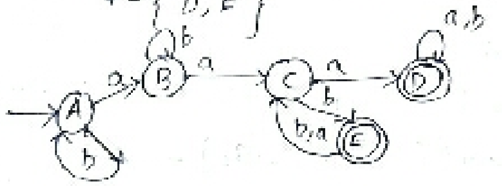
$\delta(E,b) = \delta(\{q_0,q_1,q_3\}, b) = \{q_0,q_1,q_2\}$ - C

DFA

$Q = \{A,B,C,D,E\}$ $\Sigma = \{a,b\}$

$q_0 = A$

$F = \{D,E\}$



5) Convert the NFA to DFA

δ	0	1
$\rightarrow p$	$\{p,q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	ϕ
$* s$	$\{s\}$	$\{s\}$

take $\{p\}$ - A

$\delta(A,0) = \delta(\{p\},0) = \{p,q\}$ — B

$\delta(A,1) = \delta(\{p\},1) = \{p\}$ — A

$\delta(B,0) = \delta(\{p,q\},0) = \{p,q,r\}$ — C

$\delta(B,1) = \delta(\{p,q\},1) = \{p,r\}$ — D

$\delta(C,0) = \delta(\{p,q,r\},0) = \{p,q,r,s\}$ - E

$\delta(C,1) = \delta(\{p,q,r\},1) = \{p,r\}$ - D

$\delta(D,0) = \delta(\{p,r\},0) = \{p,q,s\}$ - F

$\delta(D,1) = \delta(\{p,r\},1) = \{p\}$ - A

$\delta(E,0) = \delta(\{p,q,r,s\},0) = \{p,q,r,s\}$ - E

$\delta(E,1) = \delta(\{p,q,r,s\},1) = \{p,r,s\}$ - G

$\delta(F,0) = \delta(\{p,q,s\},0) = \{p,q,r,s\}$ - E

$\delta(F,1) = \delta(\{p,q,s\},1) = \{p,r,s\}$ - G

$\delta(G,0) = \delta(\{p,r,s\},0) = \{p,q,s\}$ - F

$\delta(G,1) = \delta(\{p,r,s\},1) = \{p,s\}$ - H

$\delta(H,0) = \delta(\{p,s\},0) = \{p,q,s\}$ - F

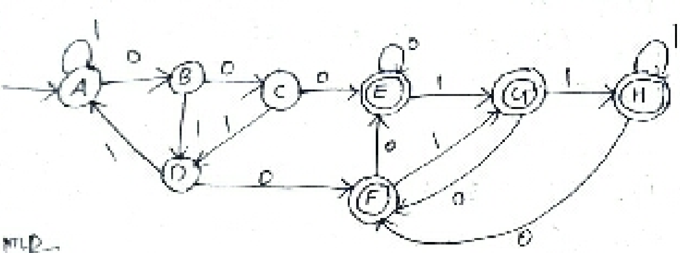
$\delta(H,1) = \delta(\{p,s\},1) = \{p,s\}$ - H

DFA

$Q = \{A,B,C,D,E,F,G,H\}$ $\Sigma = \{0,1\}$

$q_0 = A$

$F = \{E,F,G,H\}$



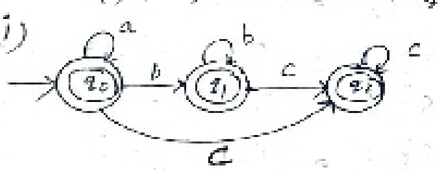
MLP

4) For the language $L = \{a^n b^m c^l \mid n, m, l \geq 0\}$ over $\Sigma = \{a, b, c\}$

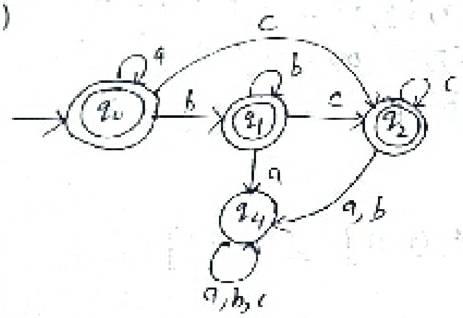
construct (i) NFA (ii) DFA

$L = \{\epsilon, abc, aabbcc, abbc, bc, ac, ab, abc\}$

(i)



(ii)



H.W.

5) Construct DFA over $\Sigma = \{0,1\}$

(i) Set of all strings that begins with 01 and ends with 11.

(ii) Set of all strings such that the no. of 1's is even and the no. of 0's is a multiple of 3

(iii) Set of all strings which when ~~interpreted~~ interpreted as binary integer which is a multiple of 3.

H.W.

6. Construct NFA

(i) Set of all strings such that containing either 101 or 110 as substring.

(ii) Set of all strings such that every 1 is followed immediately by 00.

(iii) Set of all strings containing exactly 2 occurrences of 10.

7. Convert NFA to DFA

δ	0	1
$\rightarrow p$	$\{p, q\}$	$\{q\}$
q	$\{r, s\}$	$\{p\}$
$*r$	$\{p, s\}$	$\{q\}$
$*s$	$\{q, r\}$	ϕ

ϵ-NFA :

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

$\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

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

ϵ - closure of a state

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

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

" " $(q_2) = \{q_2\}$

1) convert NFA with ϵ moves to NFA without ϵ - moves.

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

" $(q_1) = \{q_1, q_2\} \longrightarrow q_y$

" $(q_2) = \{q_2\} \longrightarrow q_z$

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

$\epsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$

$= \epsilon\text{-closure}(q_0 \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(q_0) \rightarrow q_x.$

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

$= \epsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$

$= \epsilon\text{-closure}(\phi \cup q_1 \cup \phi)$

$= \epsilon\text{-closure}(q_1) \rightarrow q_y.$

$\delta(q_y, a) = \delta(\{q_1, q_2\}, a) =$

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

$= \phi$

$\delta(q_y, b) = \delta(\{q_1, q_2\}, b) =$

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

$= q_y$

$\delta(q_z, a) = \delta(\{q_2\}, a) = \epsilon\text{-closure}(\delta(q_2, a))$

$= \phi$

$\delta(q_z, b) = \delta(\{q_2\}, b) = \epsilon\text{-closure}(\delta(q_2, b))$

$= \phi$

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

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

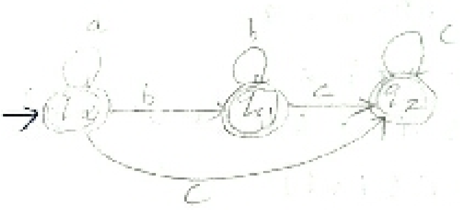
$\epsilon\text{-closure}(\phi \cup \phi \cup q_2) = \epsilon\text{-closure}(q_2) = q_z$

$\delta(q_y, c) = \delta(\{q_1, q_2\}, c) =$

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

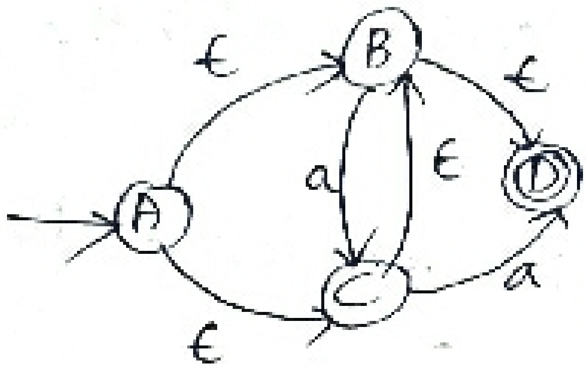
$= q_z$

$\delta(q_z, c) = \delta(\{q_2\}, c) = \epsilon\text{-closure}(\delta(q_2, c)) = q_z$



2)
29/8/19

$\Sigma = \{a\}$



$\epsilon\text{-closure}(A) = \{A, B, C, D\} \rightarrow q_0$

$\epsilon\text{-closure}(B) = \{B, D\} \rightarrow q_1$

$\epsilon\text{-closure}(C) = \{C, B, D\} \rightarrow q_2$

$\epsilon\text{-closure}(D) = \{D\} \rightarrow q_3$

$\delta(q_0, a) = \delta(\{A, B, C, D\}, a) =$

$\epsilon\text{-closure}(\delta(A, a) \cup \delta(B, a) \cup \delta(C, a) \cup \delta(D, a))$

$= \epsilon\text{-closure}(\phi \cup C \cup D \cup \phi)$

$= \epsilon\text{-closure}(C \cup D)$

$= \{C, B, D\} \rightarrow q_2$

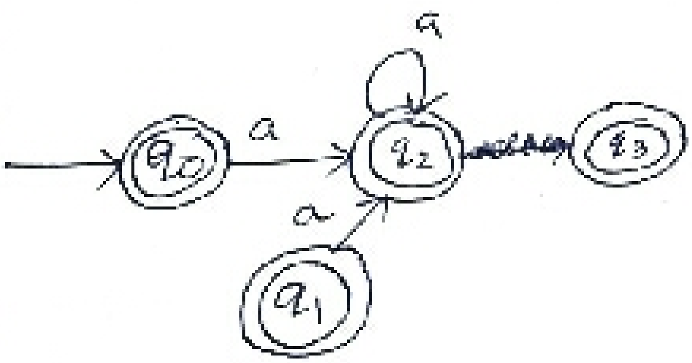
$\delta(q_1, a) = \delta(\{B, D\}, a) =$

$\epsilon\text{-closure}(\delta(B, a) \cup \delta(D, a)) =$

$\epsilon\text{-closure}(C \cup \phi) = q_2$

$\delta(q_2, a) = \delta(\{C, B, D\}, a) =$
$\epsilon\text{-closure}(\delta(C,a) \cup \delta(B,a) \cup \delta(D,a))$
$= \epsilon\text{-closure}(D \cup C \cup \phi)$
$= \{C, B, D\} \rightarrow q_2$

$\delta(q_3, a) = \delta(\{D\}, a) = \epsilon\text{-closure}(\delta(D,a))$
$= \phi$



... with and without ϵ-moves