Copy 2 Date: Page:

Automata Theory :-

- The study of abstract computing devices or "machines"
- Before computers (1930), Alan Turing studied an abstract machine (Turing machine) that had all the capabilities of today's computers (concerning what they could compute). His goal was to describe precisely the boundary between what a computing machines could do and what it could not do.

- Automata came from the Greek word, which means "self acting"
- Automation :- (Automatically controlled machines)

It is defined as a system where energy, material & information are transformed, transmitted and used for performing some functions without direct participation of man.

eg:- Automatic machine tools

Automatic packaging machines.

- ~~And the plural of automation is Automata.~~
- Finite automata are a useful model for many important kinds of software and hardware.
  - For designing and checking the behavior of digital circuits eg: Counters, Sequence detectors.
  - For compiler designing. i.e. the compiler components breaks the input text into logical units.

- Software for scanning large bodies of text, such as collections of webpages to find occurrences of words, phrases, or other patterns. etc.

Other applications, parity checkers, vending machines, security

Finite state machine :-

⇒ A finite-state machine (FSM) or finite-state automation (plural: automata), or simply a state machine, is a mathematical model used to design computer programs and digital logic circuits.

- It is conceived as an abstract machine that can be in one of a finite number of states
- The machine is in only one state at a time, the state it is in ~~at~~ any given time is called current state.
- It can change from one state to another when initiated by a triggering event or condition, this is called a transition

* In short "It is a mathematical model to design state relation or transition relation.

state, i/p ⟶ next state.

- It has finite i/ps

Date: Page:

Deterministic finite Automata (DFA):-

→ Can't be in more than one state at any time.

⇒ Strings are fed into the device by means of input tape, which is divided into squares, with one symbol inscribed in each tape square.

- Initially, the reading head is placed at the leftmost square of the tape and the initial state.
- At regular intervals the automata reads one symbol from the input tape and then enters a new state that depends only on the current state and the symbol just read.
- After reading an input symbol, the reading head moves one square to the right on the input tape. This process is repeated again & again
- Eventually the reading head reaches the end of the input string.
- Reading head is unidirectional, either right to left or left to right.
- Finite control consists of finite number of states of a given FA.

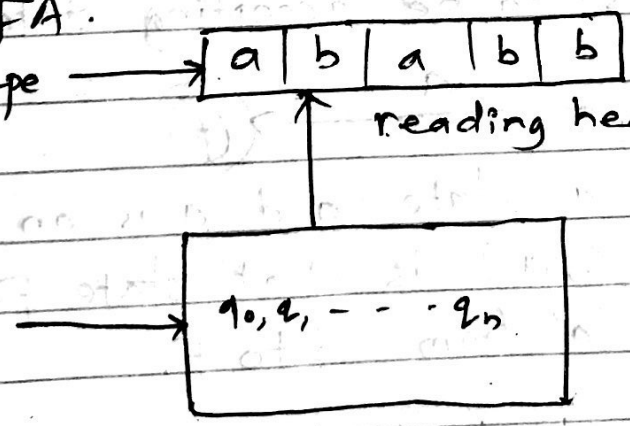


fig:- block diagram of FA.

- During scanning the symbols, when the string terminates, i.e. empty string is encountered and if finite control gives any of the defined final states is next state, the string is said to be accepted otherwise rejected.

DFA

A formal definition :- (applies to FA, DFA, NFA)

A deterministic finite automata is defined by a quintuple (5-tuple) as

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

where,

Q - Finite set of states

Σ - Finite set of input symbols

δ - Transition function that maps

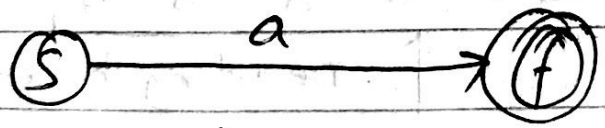
$(Q \times \Sigma \longrightarrow Q)$ → unique path follows only 1

q_0 - initial state or start state, $q_0 \in Q$

F - Set of final states : $F \subseteq Q$

also called accepting state

eg:



If S is a state and a is an input symbol then $\delta(S, a)$ is that state F. The arc is labelled 'a' from S to F.

Date: Page:

General Notations of FA.

- Transition Table
- Transition Diagram

Transition Table:-

- Transitional table is a tabular representation of the transition function 'δ', that takes the arguements from Q × Σ & returns a value which is one of the states of the automata.
- The row of the table corresponds to state while column corresponds to the input symbols.
- The starting state in the table is represented by → followed by the state i.e. → q_1 for q being start state.
- Whereas final state as * q_1 for q being final state.

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

Where

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

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

δ - consists of

$\delta(q_0, 0) \to q_0 \mid \delta(q_1, 0) \to q_2$

$\delta(q_0, 1) \to q_1 \mid \delta(q_1, 1) \to q_0$

$\delta(q_2, 0) \to q_2$

$\delta(q_2, 1) \to q_1$; q_0 = initial state, q_1 = final state.

state table (transition table)

Q \ Σ (δ)	0	1
→ q_0	q_0	q_1
* q_1	q_2	q_0
q_2	q_2	q_0

Date: Page:

Transition Diagram:-

- A transition diagram of a DFA is a graphical representation where,
  - for each state in Q, there is a node represented by circle.
  - The start state is labeled by an arrow written with "start" on the node.
  - The final or accepting state is marked by double circle.

eg

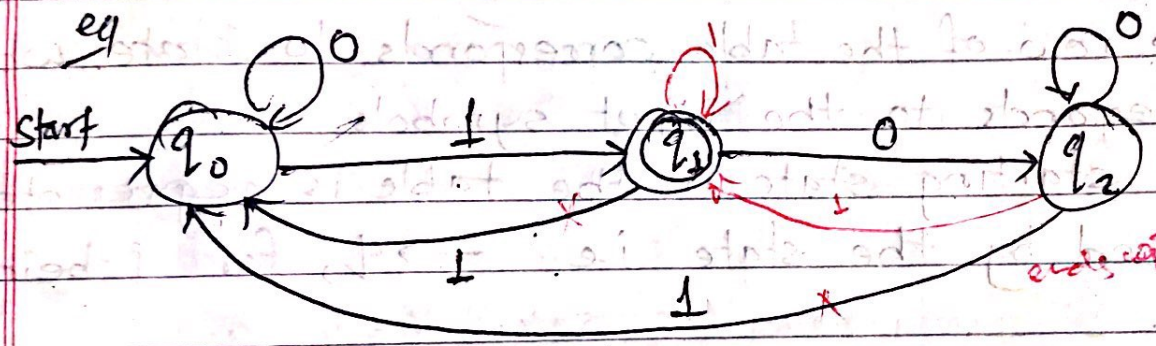


fig:- Transition diagram.

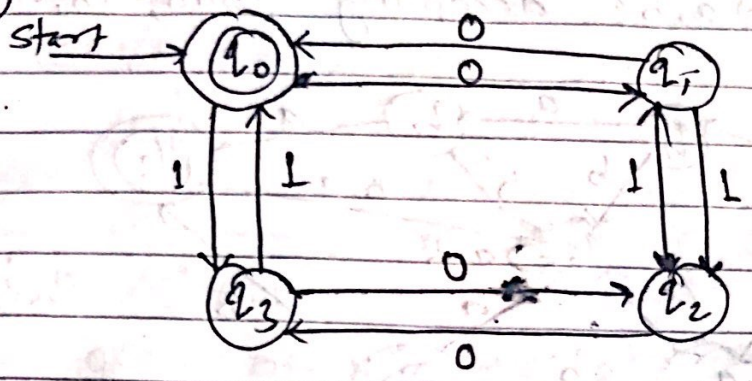
Design:-

Q1 Draw DFA for the following languages over {0, 1}?

a) All strings with even no of 0's & even no of 1's

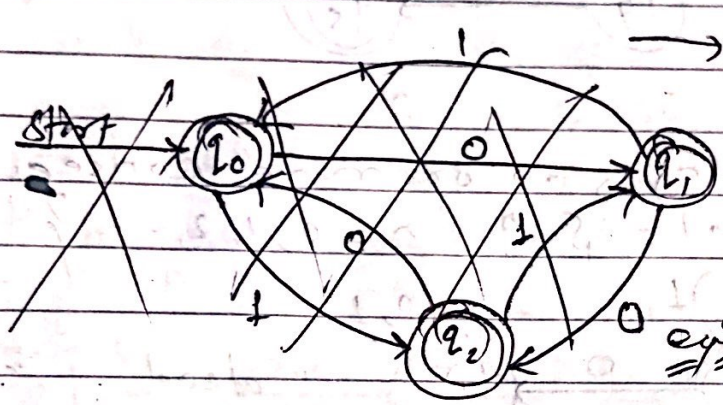
b) All string of length at most 4.

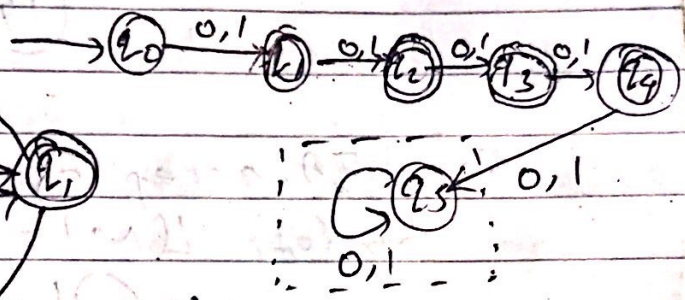
⇒ a)



strings
eg 1100, 1010, 0011, 1100

b)

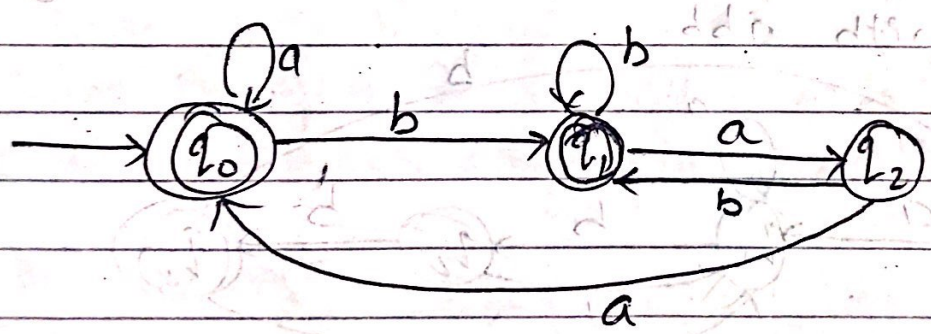




eg strings 0000, 0001, 0010.

2. Construct a DFA, that accepts all the string over $\Sigma = \{a, b\}$ that <u>do not</u> end with ba.

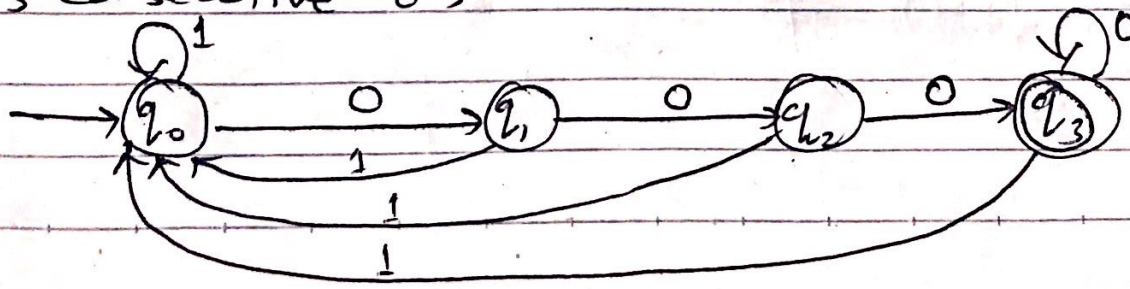
⇒



hint.
flip of final & non final state.

dfa s.c

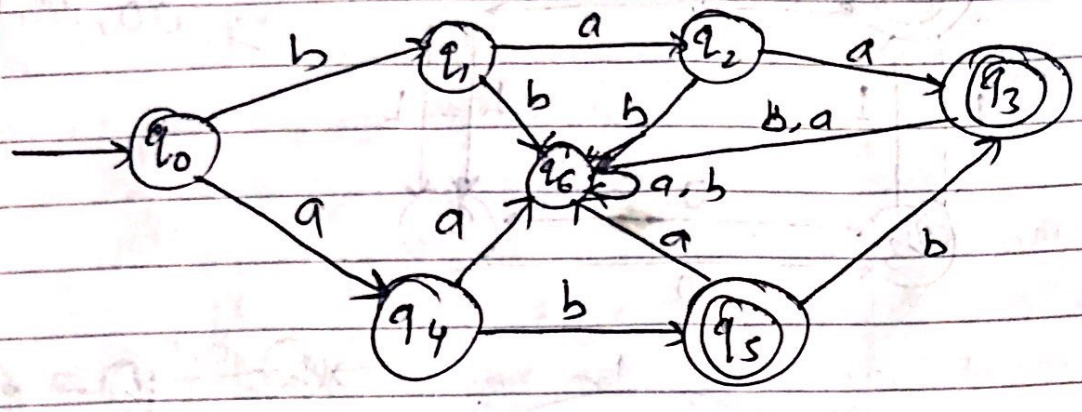
3. DFA accepting all strings over $\Sigma = \{0, 1\}$ ending with 3 consecutive 0's



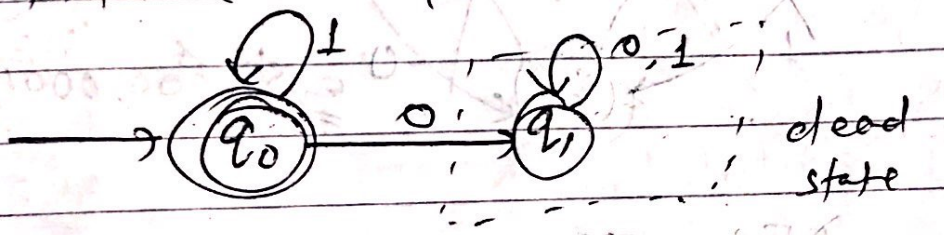
Date: Page:

4. DFA over {a, b} accepting {baa, ab, abb}

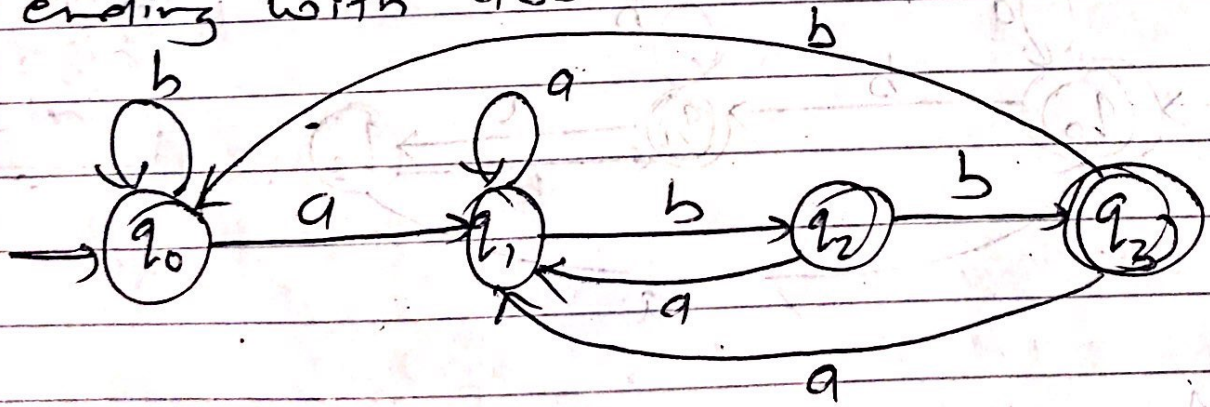
⇒



5. DFA accepting zero or more consecutive 1's over {0,1} $L(M) = \{1^n \mid n = 0, 1, 2, \ldots\}$



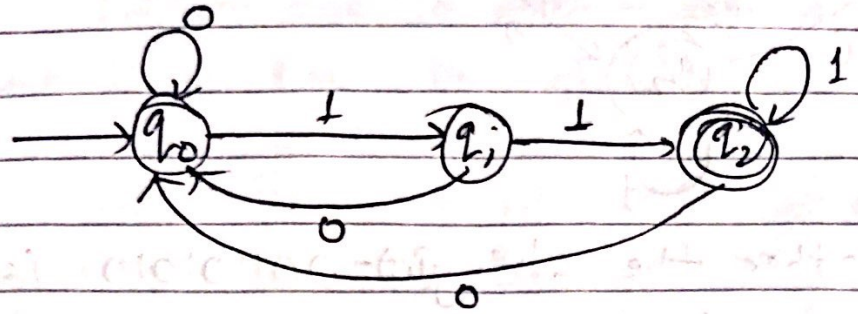
6. DFA over {a, b} that accepts the strings ending with abb.



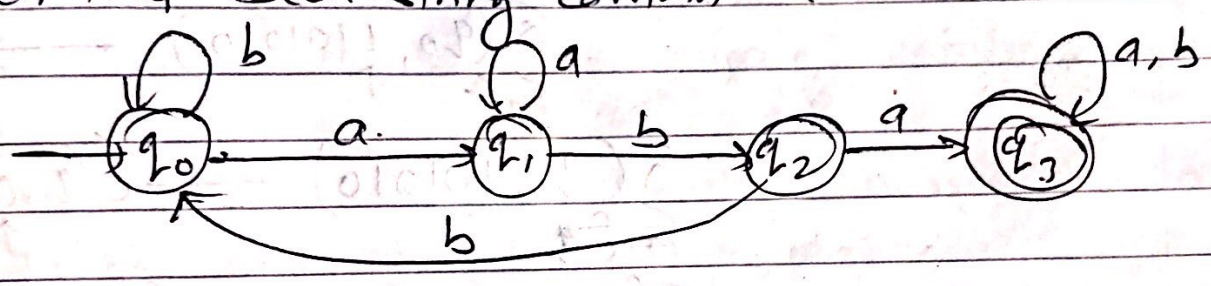
Date: Page:

7. Give the DFA for the language of string over {0,1} in which each string end with 11.

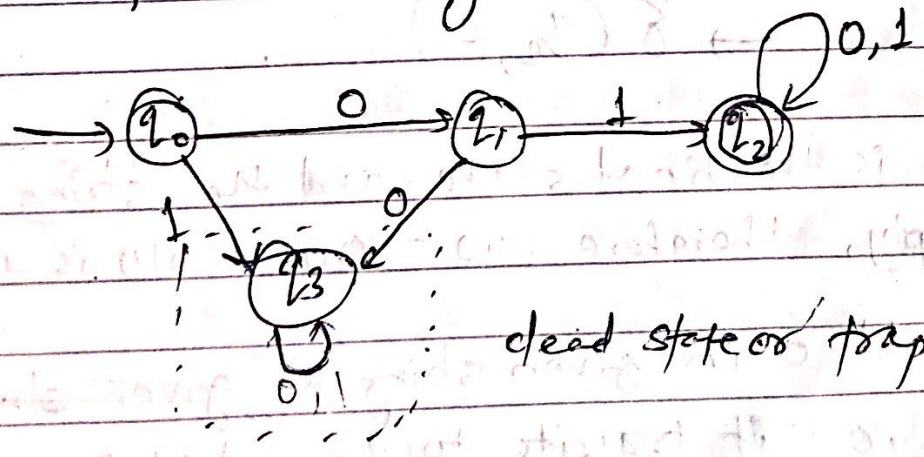
⇒



8. Give the DFA for the Language of strings over {a,b} such that each string contain aba as substring

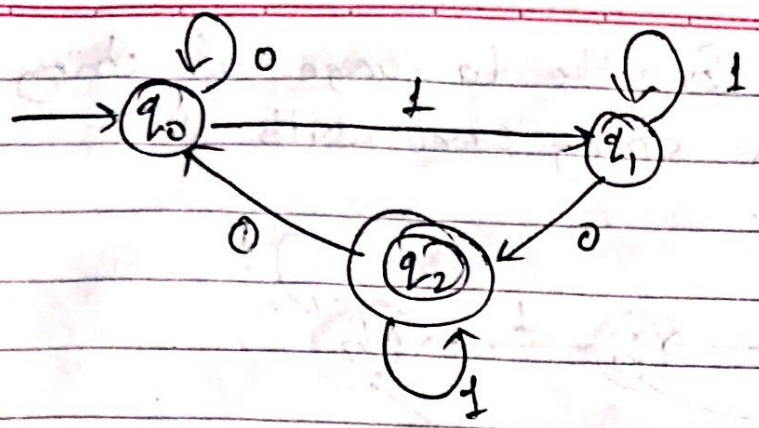


9. Give the DFA for the Language of string over {0,1} such that each string start with 0 1



FA is a Language accepter or recognizer.

Date: Page:



Check whether the string(w)=00110101010 is accepted or rejected.

$\delta(q_0, 00110101010) \longrightarrow \delta(q_0, 0110101010) \longrightarrow$

$\delta(q_0, 110101010) \longrightarrow$

~~δf~~

$\delta(q_1, 10101010) \longrightarrow \delta(q_1, 0101010)$

~~$\delta f q$~~

$\longrightarrow \delta(q_2, 101010) \longrightarrow \delta(q_2, 01010)$

$\longrightarrow \delta(q_0, 1010) \longrightarrow \delta(q_1, 010)$

$\longrightarrow \delta(q_2, 10) \longrightarrow \delta(q_2, 0)$

$\longrightarrow \delta(q_2, \epsilon)$.

∴ q_2 is the final state and the string becomes empty, ~~therefore~~ $w = 00110101010$ is accepted

* If we add '0' in given string 'w', ~~given str~~ then after $\delta(q_2, 0)$ it transits to q_0 which is not the final state so the string will not be accepted.

To formally describe the behavior of an FA on a string, we must extend the transition function δ to apply to the state and a string(w) rather than a state & symbol.

Date: Page:

Extended Transition function of DFA ($\hat{\delta}$)

⇒ In terms of the transition diagram, the Language of a DFA is the set of labels along all the paths that lead from the start state to any accepting state

⇒ In extended transition function, it describes what happens when we start in any state and follow any sequence of inputs.

⇒ The extended transition function of DFA denoted by $\hat{\delta}$ is a transition function is a function that takes q and a string w and returns a state p.

$$\hat{\delta}(q, w) \longrightarrow p$$

⇒ This state p is that automation reaches when starting in state q & processing the sequence of input w.

Basis step: $\hat{\delta}(q, \epsilon) = q$

i.e. if we are in state q and read no inputs, then we are still in state q.

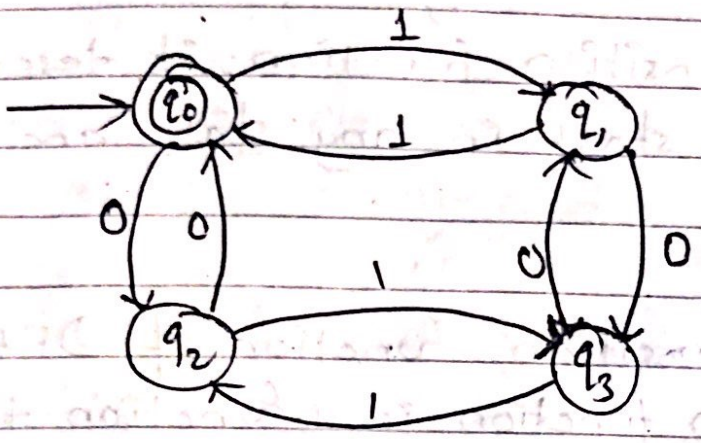
Induction: let w be a string from Σ^* such that $w = xa$, where:

'x' is a substring of 'w' without last symbo & 'a' is a last symbol of w, then.

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

Date: Page:

eg:- Design a FA which accepts the language
$L = \{w / w$ has both an even number of 0's and an even number of 1's over alphabet $\Sigma = \{0,1\}\}$.



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

Check 11010'1 string on above FA ie. it is accepted by FA or rejected!

Suppose extended transition function is $\hat{\delta}$,

$\hat{\delta}(q_0, \epsilon) = q_0$

$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_1$ (w=x, a)

$\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$

$\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$

$\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$

$\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$

$\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$ (x, a)

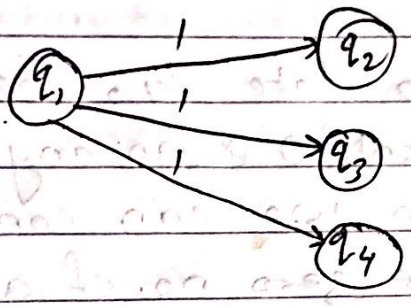
and q_0 is final state so string 110101 is accepted by FA. ✱

Date: Page:

Non Deterministic Finite Automata (NFA):

⇒ In the automata theory, a nondeterministic finite-automation (NFA) or non-deterministic finite state machine is a Finite state machine, where from each state and a given input symbol the automation may jump or transits to several possible next states.

- This distinguishes it from the deterministic finite automata (DFA), where the next possible state is uniquely determined.



- As shown in figure, the state from q_1 can change to the possible states q_2, q_3 & q_4 on reading the same input 1.
- As there are multiple state for the same input so the system transits non-deterministically ~~sometime may not reach to the final sta~~ ~~te~~, hence termed as the non-deterministic automata.
- A NFA is a quin-tuple,

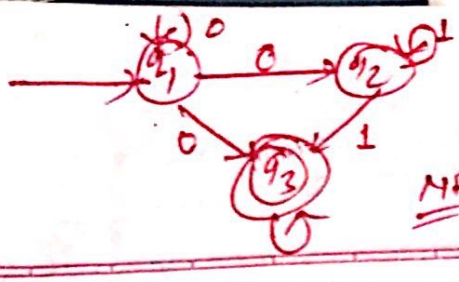
$M = (Q, \Sigma, \delta, q_0, F)$ which consists of

Q = Finite non-empty set of states

Σ - finite set of input symbols

δ - Transition function that maps

$(Q \times \Sigma \longrightarrow Q)$ where $Q = 2^Q$ (power set) ~~subsets~~



$q = (q_1, q_2, q_3)$

NFA $\{\{\emptyset\}, \{q_1\}, \{q_2\}, \{q_3\}\} \Rightarrow$ DFA

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

Date: Page:

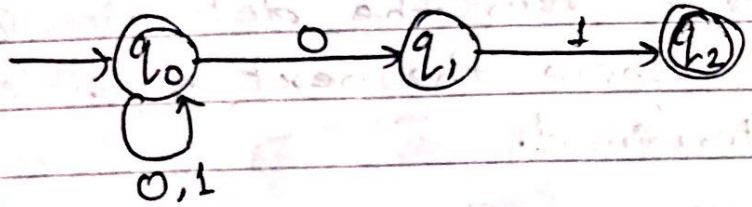
q_0 - Initial state or start state, $q_0 \in Q$

F - Set of final states : $F \subseteq Q$

also called accepting state.

Examples:-

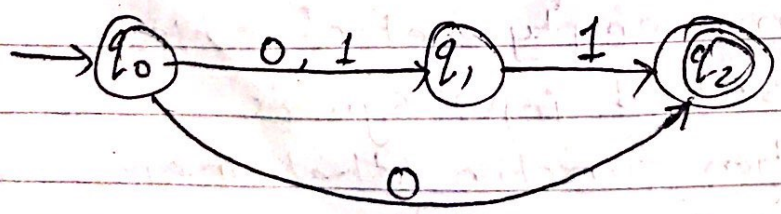
Q1 Construct a NFA to accept all strings terminating 01 over $\{0,1\}$



- Here, from state q_1, there is no any arc for input symbol '0' & no any arc out of q_2 for '0' & '1'. So we can conclude in a NFA, there may be zero no. of arcs out of each state for each input symbol.
- While in DFA, it has exactly one arc out of each ~~symbol~~ state for each input symbol.

Q2 Construct NFA over $\{0,1\}$ accepting strings $\{0, 01, 11\}$.

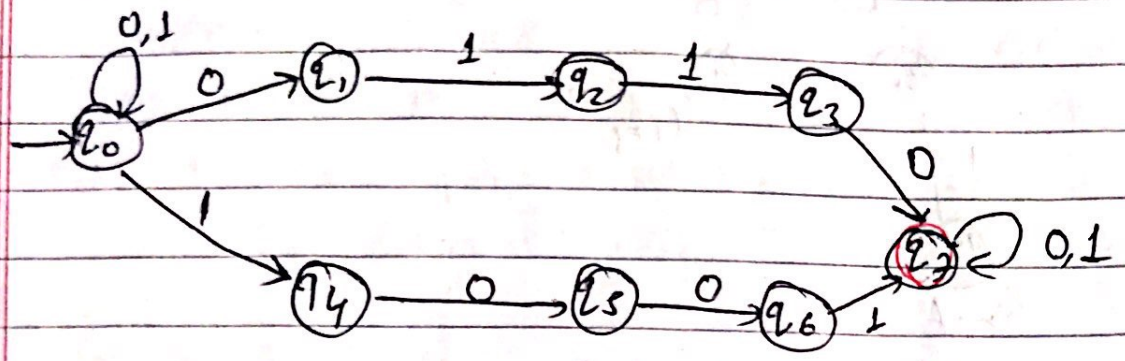
⇒



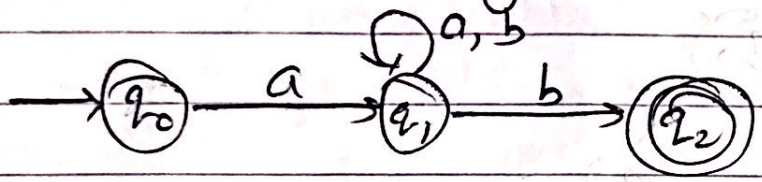
Date: Page:

Q4 NFA for strings over {0,1} that contain substring 0110 or 1001

⇒



Q5 NFA over {a,b} that accepts strings starting with a and ending with b.



The Extended transition function of NFA :-

⇒ The extended transition function $\hat{\delta}$ that takes a state q and a string of input symbol w and returns the set of states.

Definition by induction.

Basis step : $\hat{\delta}(q, \epsilon) = \{q\}$

Induction : let w be a string from Σ^*

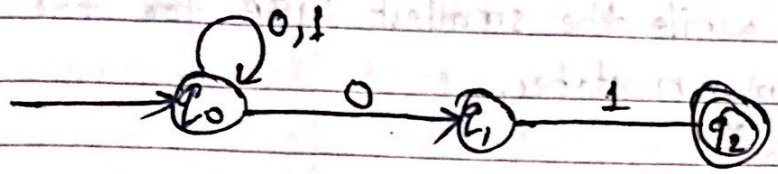
such that $w = xa$, where x is a substring

Date: Page:

let $\hat{\delta}(q, x) = \{p_1, p_2, p_3 \dots p_k\}$ set of states

$$\bigcup_{t=1}^{k} \delta(p_t, a) = \{r_1, r_2, r_3 \dots r_m\}$$

eg



Now compute for $\hat{\delta}(q_0, 01101)$

$\Rightarrow \hat{\delta}(q_0, 01101)$

we have,

$\hat{\delta}(q_0, \epsilon) = \{q_0\}$

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

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$

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

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

$= \{q_0\} \cup \{q_2\}$

$= \{q_0, q_2\}$

$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1) = \delta((q_0, q_2), 1) = \delta(q_0, 1) \cup \delta(q_2, 1)$

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

$\hat{\delta}(q_0, 0110) = \delta(\hat{\delta}(q_0, 011), 0) = \delta(q_0, 0) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 01101) = \delta(\hat{\delta}(q_0, 0110), 1) = \delta(q_0, q_1), 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

$= \{q_0, q_2\}$

∴ in the result q_2 is one of the final state the above string is accept

Equivalance of NFA and DFA:-

⇒ for each non-deterministic automata (NFA), there is an equivalent DFA that accepts the same language. The DFA ~~can have 2^n state while~~ has more transition than NFA. ~~and~~ In worst case the smallest DFA can have 2^n state while the smallest NFA for the same languages has only n states.

⇒ To show that DFA and NFA accepts exactly the same class of language;
We show that any language accepted by a NFA can also be accepted by some DFA.

⇒ for this we describe an algorithm that takes an NFA and converts it into a DFA that accepts the same language. This algorithm is called subset construction algorithm.

⇒ Each state of DFA corresponds to a subset of the set of states of the NFA.

⇒ If NFA has n-states, the DFA can have 2^n states (at most).

The steps are:-

⇒ To convert a NFA, $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ into an equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ we have following steps.

Date: Page:

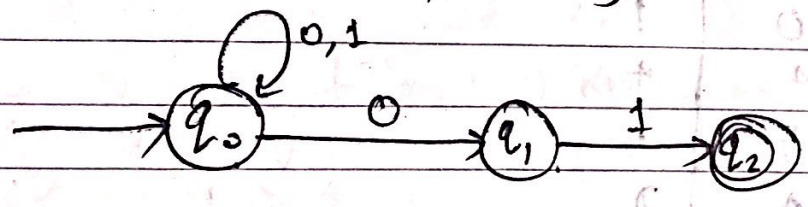
1. ⇒ The start state of D is the set of start states of N i.e. if q_0 is start state of N, then D has start state as $\{q_0\}$

2. ⇒ Q_D is set of subset of Q_N i.e. $Q_D = 2^{Q_N}$.
So, if Q_N has n states then Q_D will have 2^n states. However, all of these states may not be accessible from the start state of Q_D so they can be eliminated. So Q_D will have less than 2^n states.

3. ⇒ F_D is set of subset S of Q_N such that $S \cap F_N \neq \emptyset$ (empty set symbol)
i.e. F_D is all sets of N's state that includes at least one final state of N.

eg:- Convert the following NFA to equivalent DFA.

⇒



Soln:-

Subset construction method.

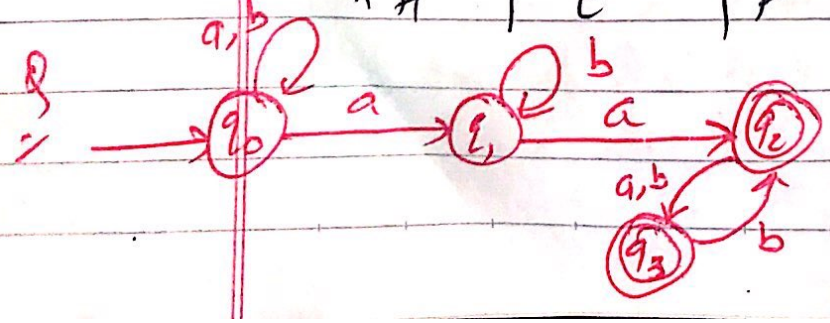
$\{\emptyset\}$, $\{q_0\}$, $\{q_1\}$, $\{q_2\}$, $\{q_0, q_1\}$, $\{q_0, q_2\}$, $\{q_1, q_2\}$, $\{q_0, q_1, q_2\}$.

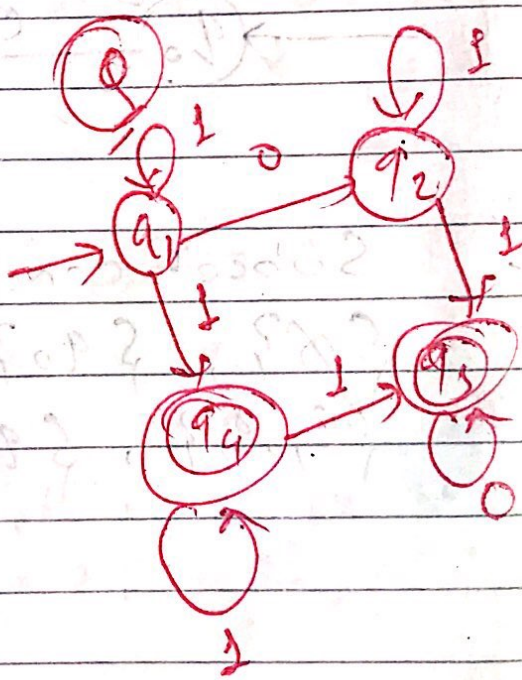
Transition table :-

δ'	0	1	δ'
$\emptyset$	$\emptyset$	$\emptyset$	
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$	
$\{q_1\}$	$\emptyset$	$\{q_2\}$	⇒
* $\{q_2\}$	$\emptyset$	$\emptyset$	
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$	
* $\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$	
* $\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$	
* $\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$	

The same table can be represented with renaming the state on table entry as

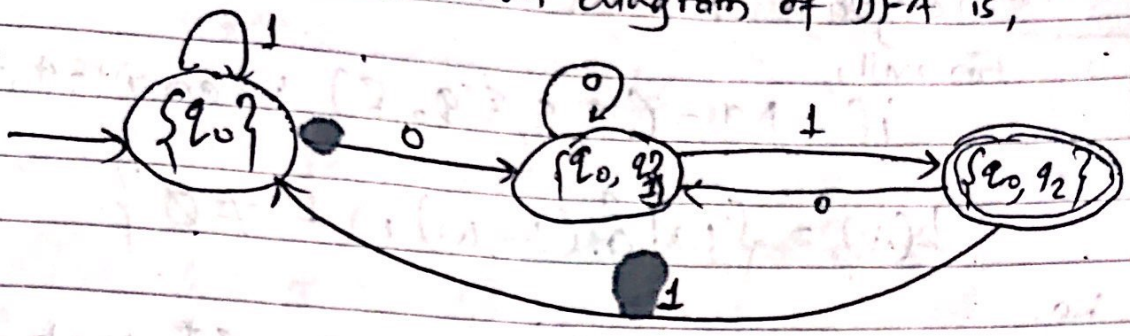
δ'	0	1
A	A	A
→ B	E	B
C	A	D
* D	A	A
E	E	F
* F	E	B
* G	A	D
* H	E	F



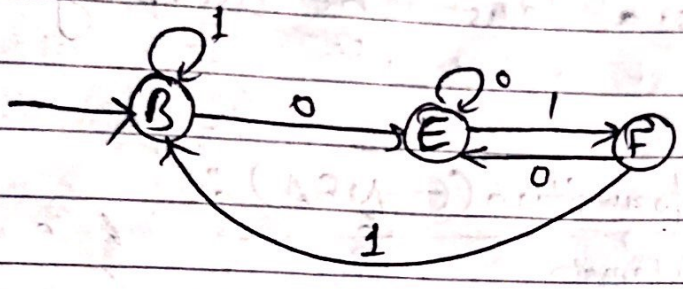


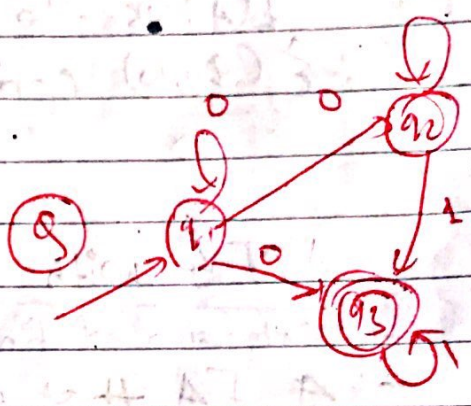
Date: Page:

Equivalent transition diagram of DFA is,



OR



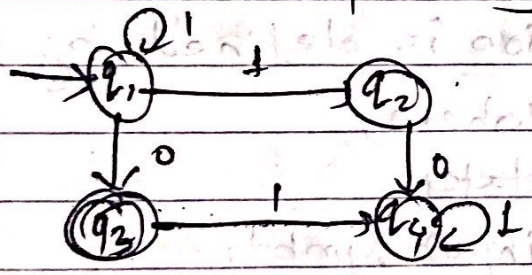


⇒ All the other states that are not reachable from the start state are removed.

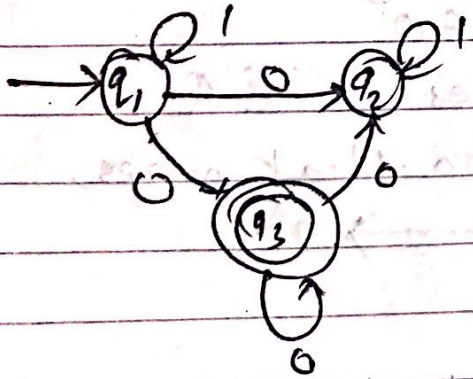
Assignment

Q Convert the following NFA to DFA.

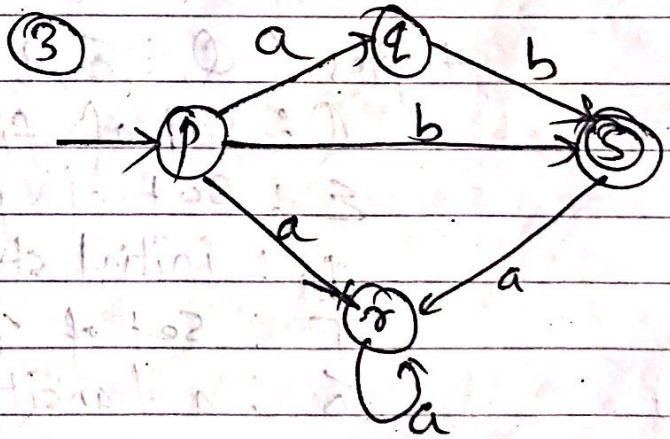
(1)



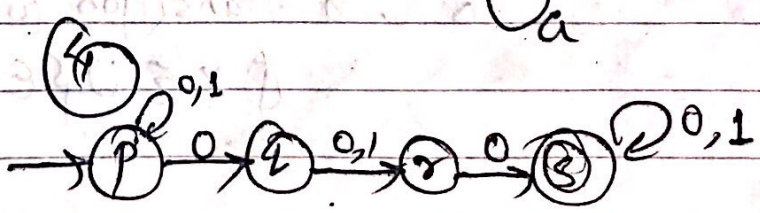
(2)



(3)



(4)



The Language of NFA:-

Formally,

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

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

i.e.

$L(A)$ is the set of strings w in Σ^* such that $\hat{\delta}(q_0, w)$ contains at least one accepting state.

NFA with ϵ-transition (ϵ-NFA):-

- Extension of finite automata
- A FA that changes the state without consuming any input symbol is called ϵ-NFA.
- To transit from one state to another state, it is labelled with epsilon(ϵ).
- A NFA with ϵ-transition is defined by five tuples $(Q, \Sigma, \delta, q_0, F)$ where,

  Q: set of finite states.

  Σ: set of finite input symbols

  q_0: initial state, $q_0 \in Q$

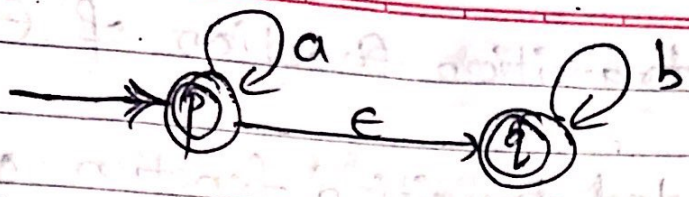
  F: set of final states, $F \subseteq Q$

  δ: a transition function that maps,

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

Date: Page:

eg:-



This accepts the language
{a, aa, ab, abb, b, - - - bbb, - - - }

eg:-

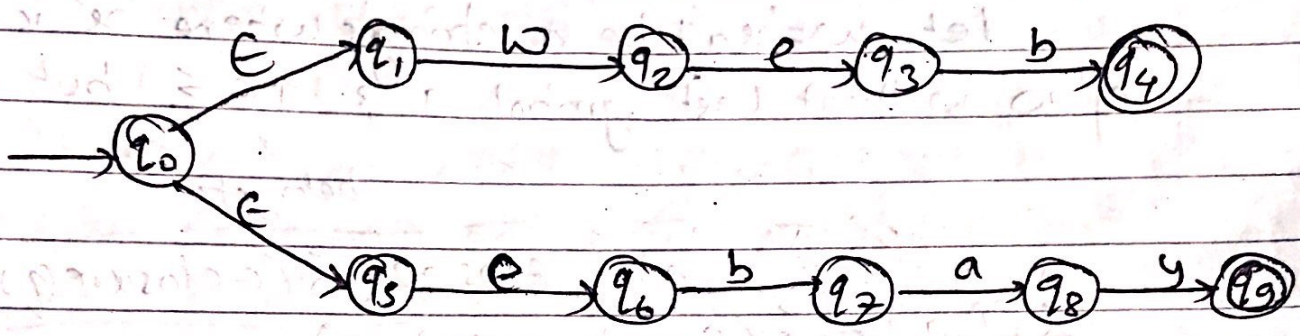
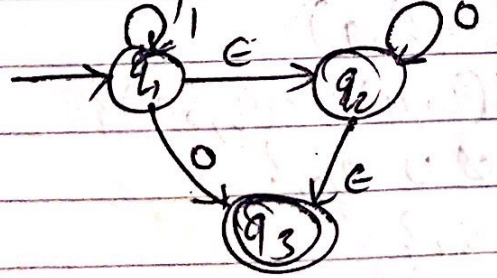


fig:- ϵ-NFA that accepts keyword web and ebay.

ϵ-closure of a state:-

- ϵ-closure of a particular state is the continious path from this state to the state in which all path is labelled epsilon (ϵ)

eg



In the figure,
ϵ-closure(q_1) = {q_1, q_2, q_3}
ϵ-closure(q_2) = {q_2, q_3}
ϵ-closure(q_3) = {q_3}

Extended transition function of ϵ-NFA :-

⇒ The extended transition function of ϵ-NFA denoted by $\hat{\delta}$, is defined as;

a) Basic step :

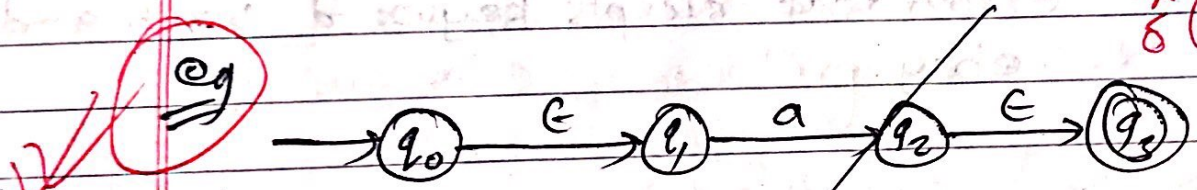
$$\hat{\delta}(q, \epsilon) = \epsilon\text{-closure}(q)$$

b) Induction step :

let $w = xa$ be a string where x is substring of w without last symbol a, & $a \in \Sigma$ but $a \neq \epsilon$ (belongs to, not empty)

$$\hat{\delta}(q, a) = \epsilon\text{closure}(\delta(\epsilon\text{-closure}(q), a))$$

~~$\hat{\delta}(q, a) = \delta((\epsilon\text{-closure}(q), a)$~~



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

$\hat{\delta}(q_0, a) = ?$

~~ϵ-closure $(q_0, \epsilon) = q_1$~~

$$\hat{\delta}(q_0, a) = \epsilon\text{closure}(\delta(\epsilon\text{closure}(q_0), a))$$

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

$\hat{\delta}(q_0, a) = \epsilon\text{closure}(\delta(\epsilon\text{closure}(q_0), a))$

$\therefore \hat{\delta}(q_0, a) = \epsilon\text{closure}(\delta(q_0, q_1), a)$

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

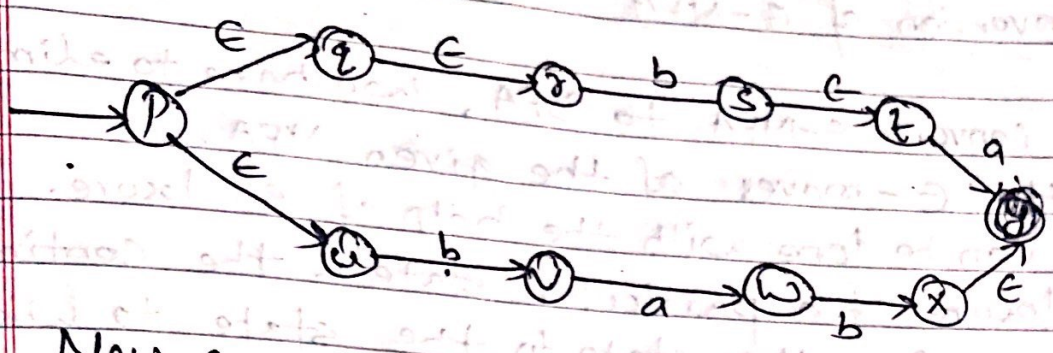
$= \epsilon\text{-closure}\{\emptyset \cup q_2\}$

$= \epsilon\text{-closure}\{q_2\}$

$= \{q_2, q_3\}$

Date: Page:

example:-



Now, compute for string ba.

we know,

$\hat{\delta}(p, \epsilon) = \epsilon\text{-closure}(p) = \{p, q, r, u\}$

Compute for 'b',

$\hat{\delta}(p, b) = \epsilon\text{-closure}(\delta(\epsilon\text{closure}(p), b))$

$\Rightarrow \delta(p,b) \cup \delta(q,b) \cup \delta(r,b) \cup \delta(u,b)$

$= \{\phi \cup \phi \cup s \cup v\}$

$= \{s, v\}$

again ϵ-closure $\{s, v\}$

$\epsilon\text{-closure}(s) \cup \epsilon\text{closure}(v)$

$= \{s, t\} \cup \{v\}$

$= \{s, t, v\}$

$\therefore \hat{\delta}(p, b) = \{s, t, v\}$

Compute for next input 'a'

$\Rightarrow \delta(s,a) \cup \delta(t,a) \cup \delta(v,a)$

$= \{\phi \cup y \cup w\}$

$= \{y, w\}$

$\epsilon\text{-closure}(y) \cup \epsilon\text{-closure}(w)$

$= \{y, w\}$ ∴ The final result contains one of final state the string is accepted

$\hat{\delta}(p, ba) = \epsilon\text{cl}(\delta(\epsilon\text{closure}(p), b), a)$

$= \epsilon\text{closure}(\delta(\{s,t,v\}, a)$

$= \epsilon\text{closure}(\delta(s,a) \cup \delta(t,a) \cup \delta(v,a))$

$= \epsilon\text{closure}(\phi \cup y \cup w)$

$= \epsilon\text{closure}(y) \cup \epsilon\text{closure}(w)$

$= \{y, w\}$

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

$\delta((\epsilon\text{-closure}(p)), b)$

$\epsilon\text{cl}(\epsilon\text{cl}(\epsilon\text{cl}(p), b), a))$

$(\delta(p, q, r, u), b), a))$

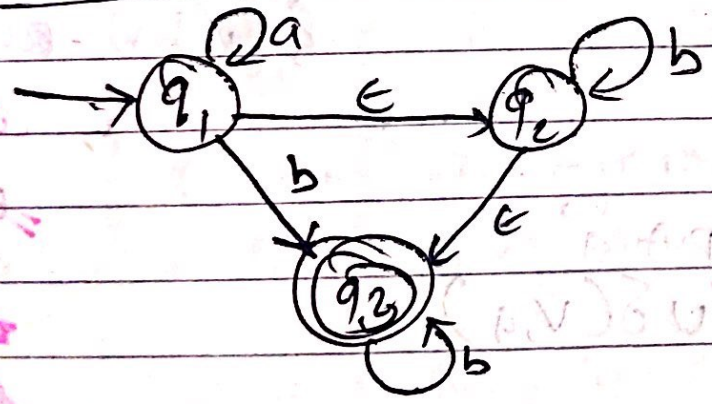
Date: Page:

Conversion of ϵ-NFA to DFA:-

- To convert ϵ-NFA to DFA, we have to eliminate all the ϵ-moves of the given NFA.
- This can be done with the help of ϵ-closure.
- ϵ-closure of a particular state is the continuous path from this state to the state to which all path is labelled with epsilon (ϵ).
- To eliminate ϵ-moves, we first find ϵ-closure of all the states of the given ϵ-NFA.
- This ϵ-closure gives the new set of states.
- We then find the transition for all these new states and finally draw a diagram with respect to the transitions of these new states. Which is resulting equivalent DFA.

eg Convert ϵ-NFA to equivalent DFA:-

⇒



Soln:

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

ϵ-closure $(q_2) = \{q_2, q_3\}$: y

ϵ-closure $(q_3) = \{q_3\}$: z

$\delta'(\{q_1, q_2, q_3\}, a)$ Date: Page:

$= \epsilon\text{-closure}(\delta(\{q_1, q_2, q_3\}, a))$

$= \epsilon\text{-closure}(\delta(q_1, a)) \cup \epsilon\text{closure}(\delta(q_2, a)) \cup \epsilon\text{closure}(\delta(q_3, a))$

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

$= \epsilon\text{closure}(q_1)$

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

$\delta'(\{q_1, q_2, q_3\}, b)$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2, q_3\}, b)) = \{q_2, q_3\} : y$

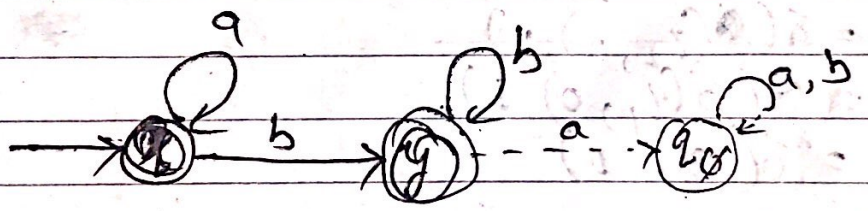
$\delta'(\{q_2, q_3\}, a)$

$= \epsilon\text{-closure}(\delta(\{q_2, q_3\}, a) = \{\emptyset\}$

$\delta'(\{q_2, q_3\}, b) = \epsilon\text{-closure}(\delta(\{q_2, q_3\}, b) = \{q_2, q_3\} : y$

$\delta'(\{q_3\}, a) = \epsilon\text{-closure}(\delta(\{q_3\}, a)) = \{\emptyset\}$

$\delta'(\{q_3\}, b) = \epsilon\text{-closure}(\delta(\{q_3\}, b) = \{q_3\} : z$



Which is the required DFA.

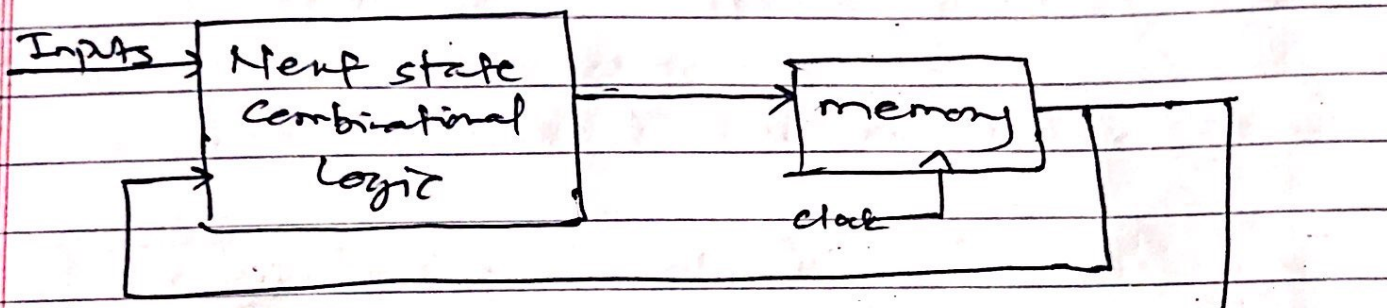
Date: Page:

Finite state machine with o/p:-

- 2 models for representing fsm with o/p.
- They differ in the way o/p is generated.
  1) Moore machine
  2) Mealy "

Moore machine:-

- It is a Fsm whose o/p depends only on the present state only.



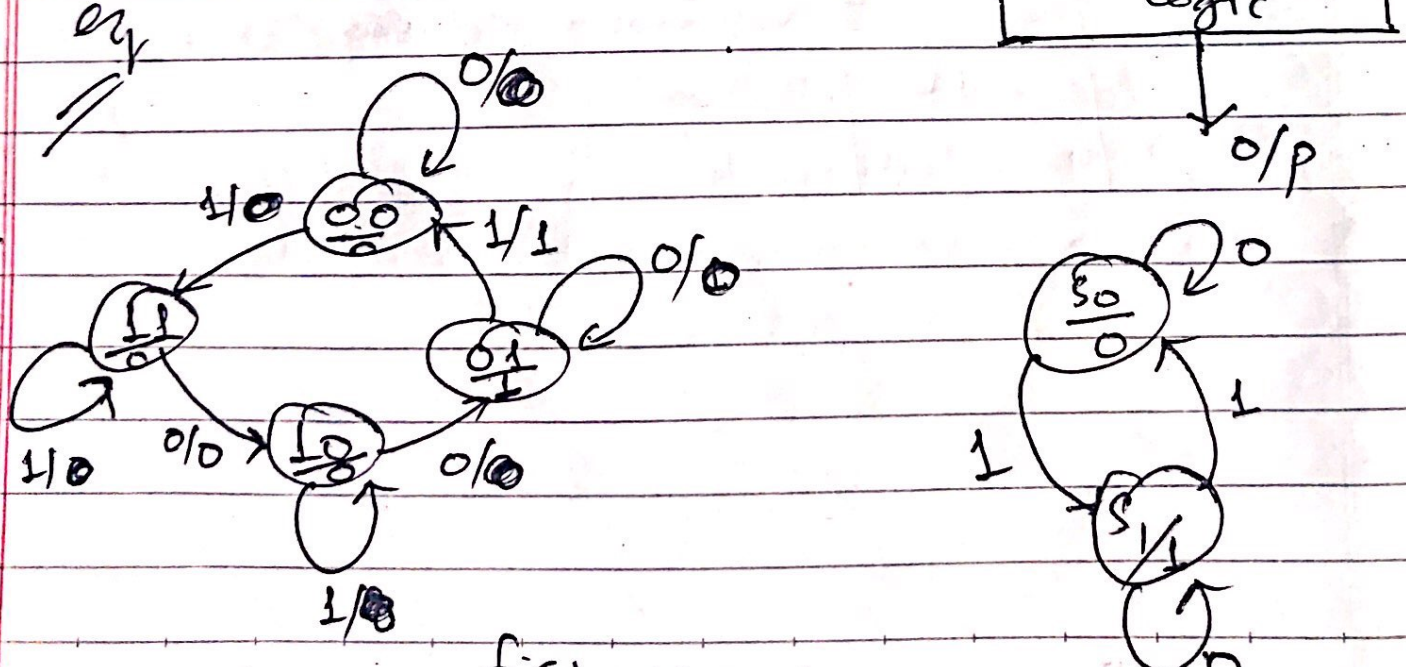


fig:- State transition diagram

First

Date: Page:

Mealy Machine:-

- It is a fsm whose o/p depends on present state as well as the current input.

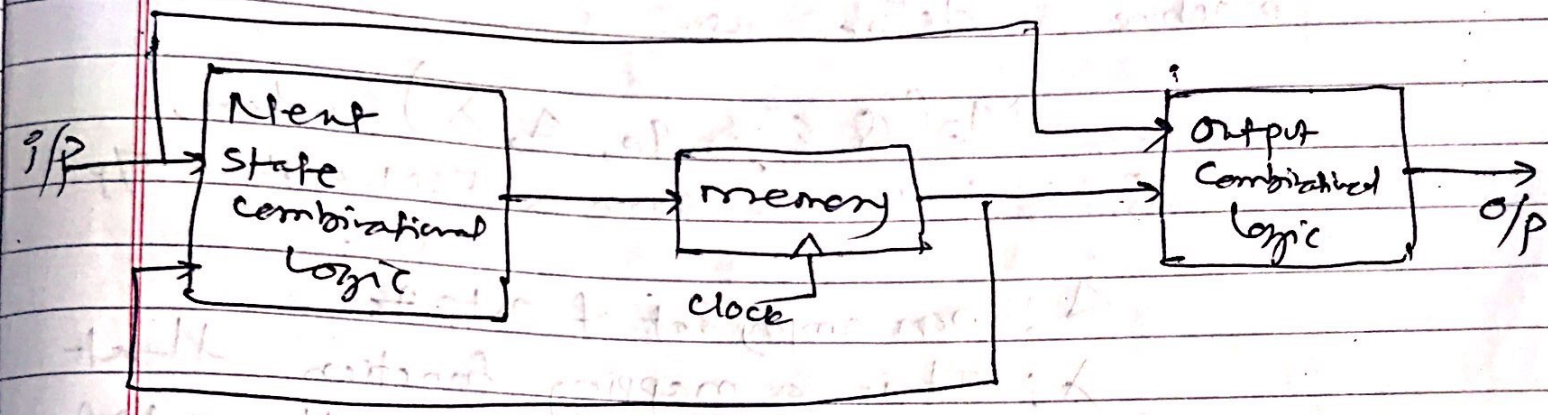
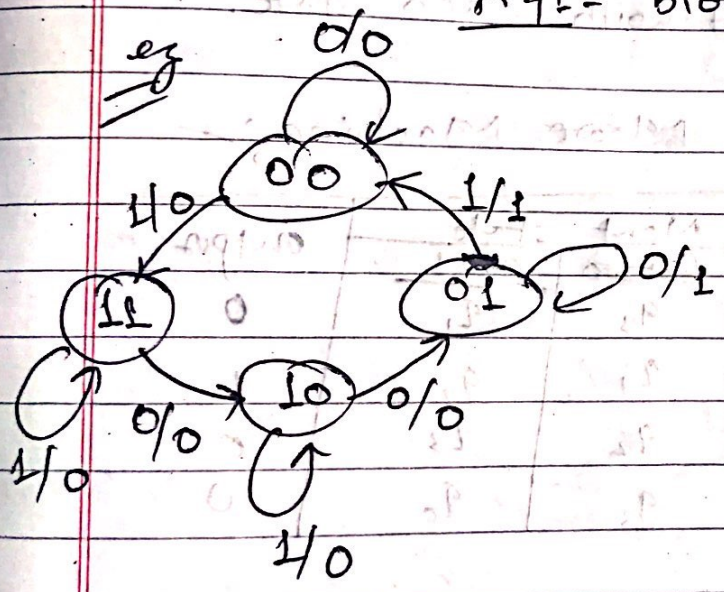


fig:- block diagram

eg



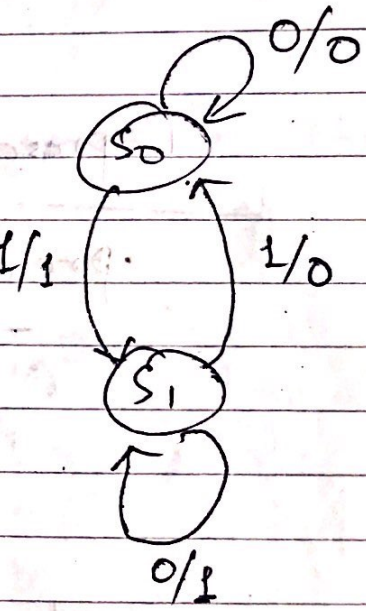


fig:- State transition diagram

Date: Page:

Moore Machines:-

- It is a FA in which o/p is associated with each state.

Mathematically, Moore machine is a Six tuple machine & defined as.

$M_0 = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$ where,

Q, Σ, δ, q_0 are same as FSM without o/p.

Δ : non empty set of outputs.

λ : It is a mapping function that maps Q to Δ, giving the output associated with each state.

Representation of Moore Machine:-

present state	Next state		output
	0	1	
$\rightarrow q_0$	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

where, $M_0 = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$ where.

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

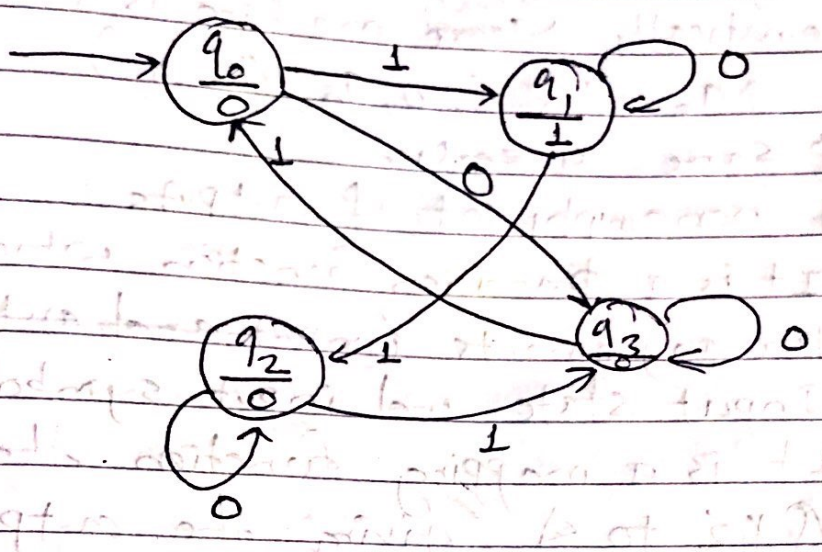
$\Sigma = \{0, 1\}$

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

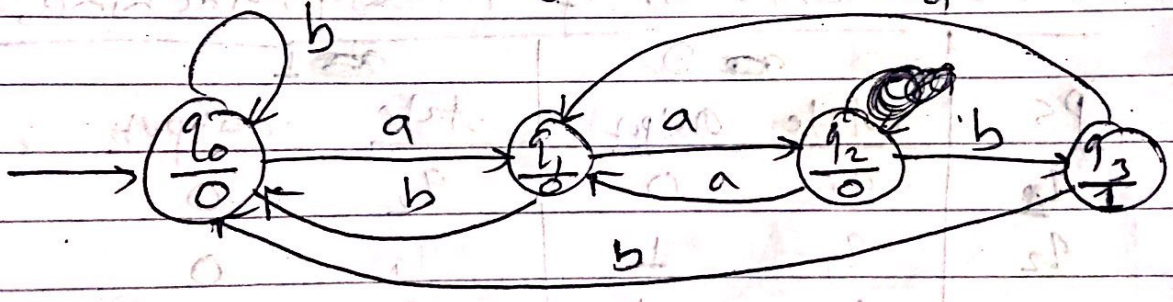
$\lambda(q_0) = 0, \quad \lambda(q_1) = 1, \quad \lambda(q_2) = 0, \quad \lambda(q_3) = 0$

- There is no concept of final state in moore machine since we are considering output for each corresponding state.

The transition diagram can be written as,



eg:- Design a Moore machine which counts the occurance of substring aab in input string.



w = a aab baabaa

Date: Page:

Mealy Machines:-

- It is a finite automata in which output is associated with each transition.
- In mealy machine every transition for a particular input symbol has a fix output.

Mathematically, ~~Moore~~ Mealy machine is a six tuple machine $M_e = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$

Q; Σ, q_0 same as earlier.

Δ: A nonempty set of outputs.

δ: It is a transition function which takes ~~two arguements (input and output~~ two arguements (Input state and input symbol).

λ: It is a mapping function that maps, $Q \times \Sigma$ to Δ, giving the output associated with each transition.

Representation of Mealy Machine:-

	0		1	
PS	State	output	state	output
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

$M_e = (Q, \Sigma, \delta, q_1, \Delta, \lambda)$

$Q = (q_1, q_2, q_3, q_4)$

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

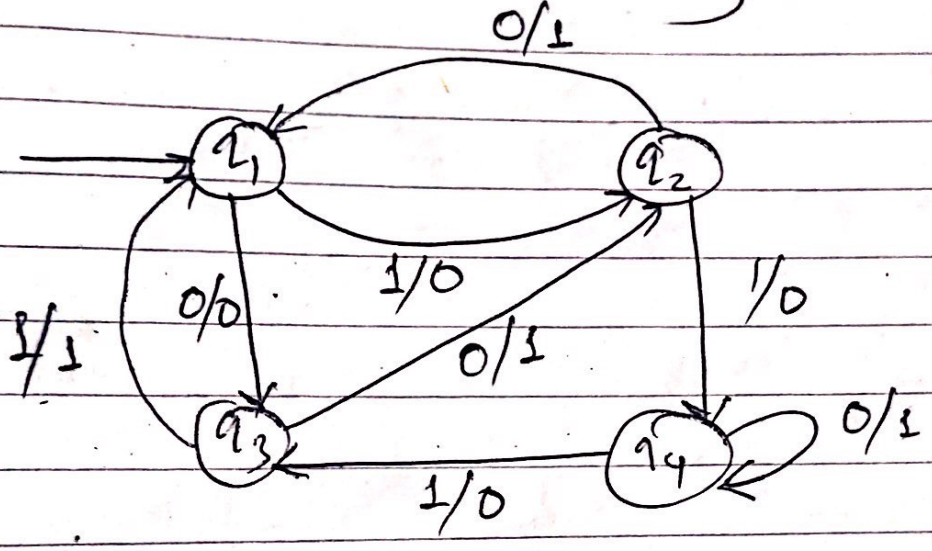
$\Delta = \{0, 1\}$

Date: Page:

q_0 is the initial state & δ is transition function.

$\lambda(q_0, 0) = 0$ $\quad$ $\lambda(q_1, 1) = 0$
$\lambda(q_2, 0) = 1$ $\quad$ $\lambda(q_2, 1) = 0$
$\lambda(q_3, 0) = 1$ $\quad$ $\lambda(q_3, 1) = 1$
$\lambda(q_4, 0) = 1$ $\quad$ $\lambda(q_4, 1) = 0$

The transition diagram can be written as,



Q:. Same as above ~~count~~ count occurrence of aab