Basic concepts (or) mathematical Notations in Automata Theory

1. Alphabet :

* A alphabet is set of symbols
* it is an finite set of symbols i.e (finite collection of symbols)
* An alphabet Represented by sigma

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

because contain 2 symbols

→ This is called binary alphabet

$\Sigma = \{a, b, \ldots z\}$ → lower case Alphabet

This contain lower case symbols

$\Sigma = \{A, B, \ldots Z\}$ → Upper case Alphabet

This contain upper case symbols

2. string :

* string means a finite collection of symbols over Alphabet Σ.
* a finite set of symbols which generated from the alphabet Σ.
* A string is always represented by w.
* string can also be called as word

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

(0 → 0 is a string, 1 → 1 is string, 01, 10, 11, 010, 101 . . .) 'n' no. of strings

likewise we have

The basic operations performed on the string.

1. Length of a string :- → total No. of letters Presented in string.

* length of a string is denoted by |w| . mod w

w = abc, } 3 symbols

|w| = 3

Empty string

* a string with No symbols •
* Empty string denoted by Epsilon (ϵ) symbol
* The length of Empty string is zero (0).

|w| = 0 → because Empty string have no symbols.

3. Prefix of string

* Prefix means any number of leading symbols

w = abc

* Then the prefixes are epsilon ϵ
* why? because we can write abc as ϵ abc X ϵ

→ string → Prefix leading Symbol

ϵ, a, ab, abc

4. proper prefix of string

* ϵ a, ab, abc X
* it means Except epsilon and the string. Remaining all are the proper prefixes

prefix Ex: ∈, a, ab, abc

proper prefix :- a, ab

suffix :- any number of trailing symbols

Ex: w = abc

∈, c, cb, cba or ∈, c, bc, abc

proper suffix

* it means Except ebsilon and the string. Remaining are the proper suffixes

Ex: w = abc

suffix = ∈, c, bc, abc

proper suffix = c, bc

7. substring

* substring means a part of a string

Ex: abc

∈, a, b, c, ab, ac, bc, abc

8. Language

* language mean a collection of a string. generated from embsilon ∈.

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

$L = \{0, 1, 00, 01, 10, 11, 000, 0111, 001, 100 \cdots\}$

likewise we have n number of strings.

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

Language L = any no. of 0's → we have to write zero only

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

Ex 3 L = any no. of 0's and 1's

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

* Now let us see about major two operations

1. Kleene closion
2. positive closion

1. Kleene closion

* Kleene closion also called as ~~positive~~ star closion.
* it is denoted by Σ^*. This can also called star closure

$\Sigma = \{0\}$

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

2. Positive closure

Positive closure is denoted by sigma plus Σ^+

* Σ^+ it means it contains minimum 1 occurance

that means ebsilon is not present here.

$\varepsilon^+ = \{0, 00, 000, 00 \cdots\}$

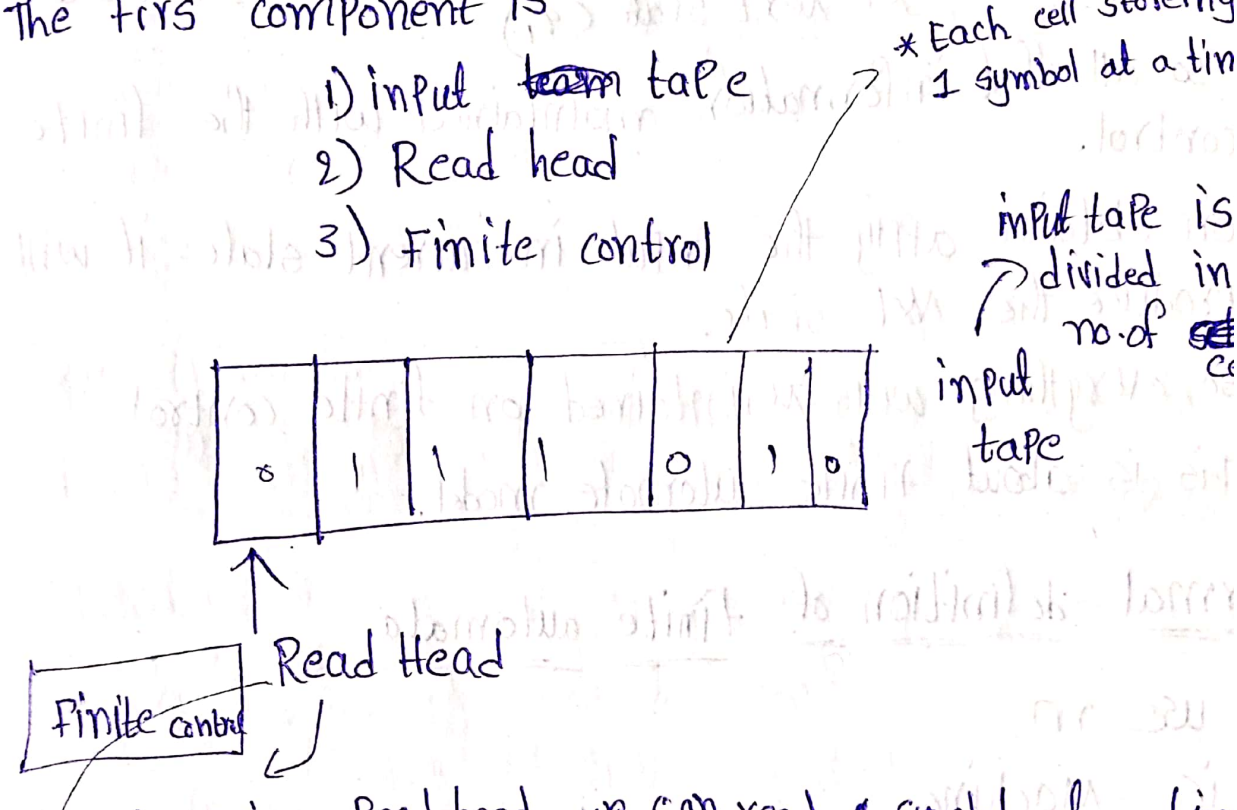
These are Various Basic concepts (or) mathematic Notations are used in finate Automata theory.

Finite Automata model & Formal definition of F.A

* first talk about Finite Automat model
* it mainly contains 3 models.

The Firs component is

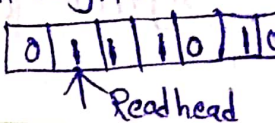
1) input ~~tearn~~ tape
2) Read head
3) Finite control



by using Read head. we can read 1 symbol at a tim

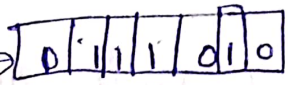
initially Read head points to the first symbol of input type.

* The First symbol '0' zero will be readed
* The Read head Moved to Right
* Read head Moved to Right only.



3. Finite control

* in Finite control all the states information will be maintained
* let our transition diagram contains 5 states.
* so, all the 5 states information maintained in Finite control.
* And we have current state
* some input symbol.
* current state is q_0 let input symbol = 0



* Then we will apply ~~the~~ 0 on the current state. Then we will get Next state (q_1)
* so all that information maintained with the finite control.
* on obtain apply the input in current state. it will produce the Nxt state.
* so, everything will Maintained on finite control
* This is about Finite automate. model.

Formal definition of Finite automata

- we use m
- M is Machine

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

inorder to define FA we have 5 types.

$Q \rightarrow$ represents set of states. (ex: let the transition diag contain 5 states)

$\Sigma \rightarrow$ sigma represent input alphabet (input alphabet contains

δ → delta is transition function (which maps like these $Q \times \Sigma \to Q$)
(constraint, input symbol, next state)

q_0 → q_0 not mean initial state
(any transition diagram contain 1 initial state)

F → F mean set of final states (we may have multi final sets)

* This about formal definition of FA.

Representation of Finite Automata

Representation of finite automata is 2 ways.

1. transition diagram
2. Transition table

Transition diagram

transition diagram means the pictorial representa of finite automata. is called as transition diagra

* in transition diagram mainly we use 2 states. They are
  1. initial state
  2. Final state

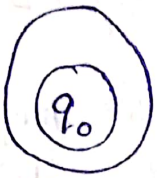
1. initial state

Is means we uses a single circle.

* arrow will be towards the circle * within the ci have write stat

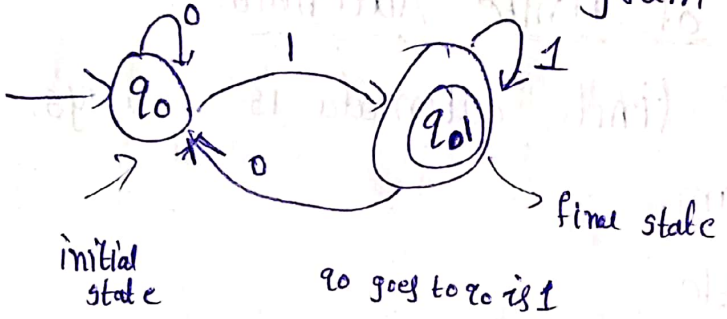
2. Final state

* we have to use double circles
* in the double circles we write the state name



* for the final state There is no need of arrow

let we draw the transition diagram



* This is transition diagram
* we can define FA with the help of 5 tuples.

They are $m = (Q, \Sigma, \delta, q_0, F)$

* m is equal what are the states present

$m = (\{q_0, q_1\}$ → Q means set of states collections of states so draw{}

$\{0, 1\}$ → sigma means input alphabet

δ → which maps

q_0 → Initial state

$\{q_1\})$ → Final state

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

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

if we apply 1 on q_0 it goes to q_1

$\delta(q_1, 0) = q_0$

$\delta(q_1, 1) = q_1$

Initial state

$\delta(q_0, 0) = q_0$ | $\delta(q_1, 0) = q_0$

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

Final state

Transition tape:

we have set of rows and columns

rows means states

column means input symbols

current state	Next state	
	0	1
→ q_0	q_0	q_1
q_1	q_0	q_1

q_0 is initial state so arrow →

in final state enclosed with circle()

in transition diagram double circles (◎) but transition table single o

* In this way we represent FA in transition diagram and transition table.

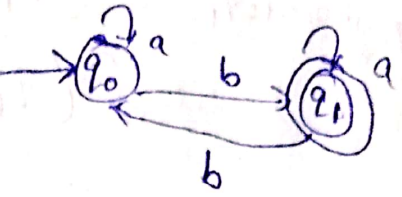
DFA (Deterministic Finite Automata)

* DFA defined with the help of 5 tuple.

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

$Q \rightarrow$ Finite set of state

$Q = \{q_0, q_1\}$



$\Sigma \rightarrow$ Input alphabet

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

$q_0 \rightarrow$ initial state (in DFA or NFA only 1 initial state)

q_0

$F \rightarrow$ set of Final states

$\{q_1\}$

$\delta \rightarrow$ Delta is transition function maps from

$Q \times \Sigma \rightarrow Q$

$\delta(q_0, a) = q_0$ | $\delta(q_1, a) = q_1$

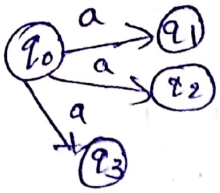
$\delta(q_0, b) = q_1$ | $\delta(q_1, b) = q_0$

* In this we define DFA
* why it is called Deterministic DFA ?

because we apply an input symbol on a state $(q_0 \xrightarrow{b} q_1)$ we can get only 1 state.

In DFA $q_0 \xrightarrow{a} q_1$

But in NFA we get mulliple states



different states, that why it is called NFA.

* Wheras DFA means
* if we apply an input symbol on particular state ~~state~~ x we get only 1 state
* That means on a particular symbol. for a particular symbol There will be only 1 edge (or) 1 transition
* it is called DFA
* one more property of DFA is.

$\Sigma = \{a, b\}$ → Input alphabet contain 2 alphabets.

* Then every state present in the diagram we must consume 2 symbols.
* q_0 must use These 2 ($\{a,b\}$) symbols
* q_1 must use These 2 symbols.
* Even though they are not need also we consume.
* because DFA means every state must consume Each and every symbol in the alphabet.
* That is the Property of DFA.

DFA Examples

1) construct DFA that accepts set of all strings over {a,b} of strings starting with 'a'

2) " Set of all string starting with "ab".

1) first derive the language

$\Sigma = \{a, b\}$ → input

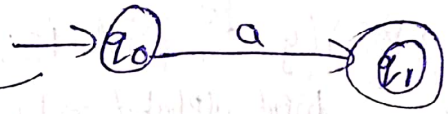
$L = \{a, aa, ab, aab \ldots\}$ → in question every str start with a.

The string length is n+1 we require n+1.

$a = 1, 2^{n+1}$ q_0, q_1

* The minimum length is 1 (a)
* we require 2 states (q_0, q_1)

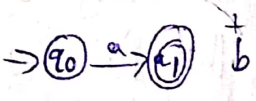
every string ever str everystring start with a



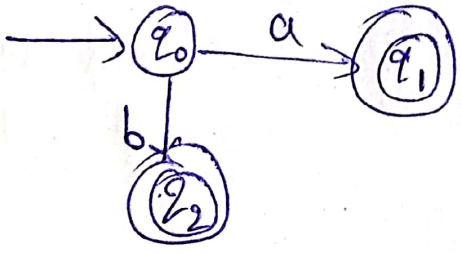
* let us check wheather it is DFA or not?

This is Not DFA. DFA means on every state we have to apply the symbol that are present in the set.

- Here the aphabet contain 2 symbols $\Sigma = \{a, b\}$
- we not applied b. but we apply b also
- but every starting is a only
- so if we apply b on q_0 Then what will happen. The correspond string start with b



- if string startwith b it is Not a valid string
- so we apply b on q_0. Then go to state q_2



→ This state called as dead state.

are we also called as Hang state.

→ what is dead state?

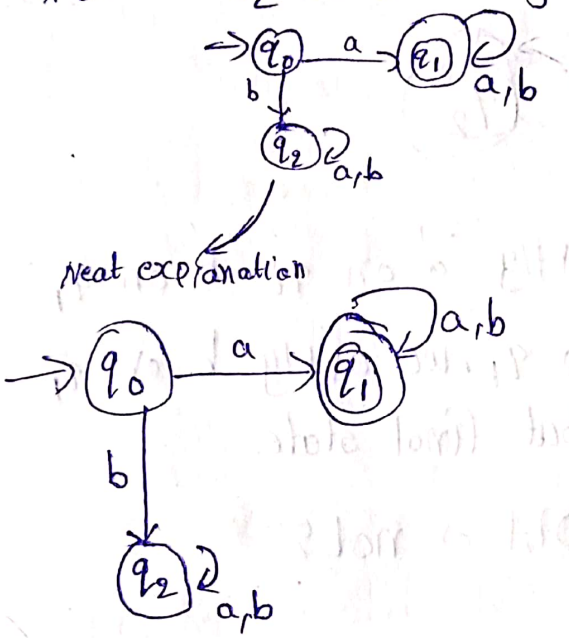
. from dead state we can not go to another state.

. so from q_2 we can not go to q_0 are q_1

ex: bab

* if the string is string with b it is not a valid string. that corresponding string not accepted by a finite automata

* so on q_2 we have any noof a's and b's



* so in q_0 'a' is applied 'b' is applied. This state is ok.

* on q_2 'a' is applied 'b' is applied. This state is also ok.

* on q_1 no symbol is applied we have to apply both 'a' and 'b' on q_1

* Here every string start with a after 'a' we have any no. of a's and b's. so There is no constraint on that.

* so draw some self Loop a and b

* so if we observe the first string i.e (a) we apply a on q_0 Then we go to the q_1. so This is Ntg but final state.

* so we can say that The string is accepted.

* if we conside second string (aa). if we apply 'a' on q_0 Then we go to the q_1. on q_1 we can apply 'a'

* so any string accepted

* with this the 1st problem is over.

2) Every string start with "ab".

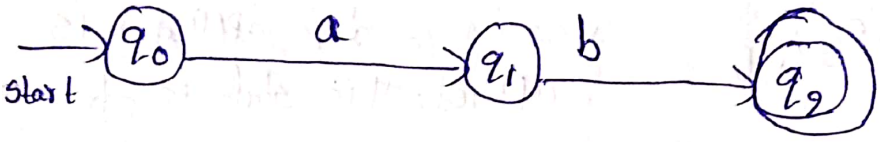
$\Sigma = \{a, b\}$

$L = \{ab, aba, abb, abaa, \ldots\}$

Here the minimum possible string is ab. it contains 2 characters

The DFA ~~requires~~ length 2 we require 3 states.

as we know DFA ~~require~~ length 'n' we require n+1 states.

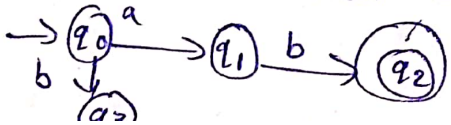


* minimum length ab we apply 'a' on q_0 'b' on q_1

* we apply 'a' on q_0 we go to q_1. we apply b on q_1 we go to q_2. q_2 is Ntg but final state.

* Now check whether it is DFA or not?

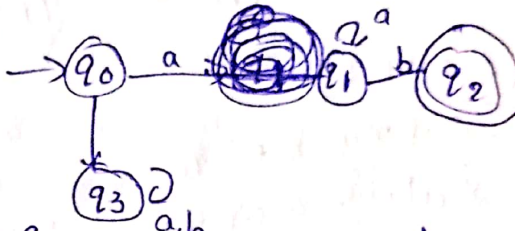
- it is Not a DFA.
- DFA means every state uses the symbols that are present in the alphabet $\in \{a, b\}$
- in q_0 b is not applied on q_1 'a' is not applied. on q_2 ab is not applied.
- Let us apply b on q_0. so we apply b on q_0 it is not a valid string.
- because every string start with ab only.
- we apply b first character is b. but Here string not starting with b.
- we apply b on q_0 it go to dead state. so the Next State q_3



• after b we have any No. of string we Not bother it.
• why because the string is starting with b. so we can say that it is not a valid state.

• on q_3 we apply a,b. we should not bother about it. why because it is a invalid string
• Here q_3 is called dead state. q_3 can not moved to any state

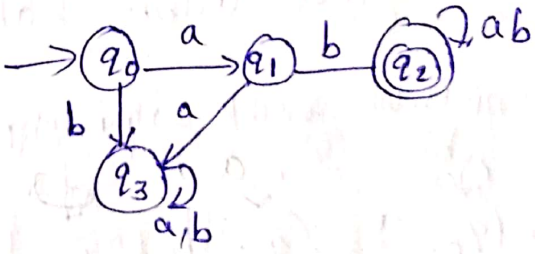
• Now let us focus on q_1. on q_1 b is already applied
• Now we apply 'a' on q_1
• if we apply the self loop



• if we apply 'a' on q_0 we got to q_1.
if we apply 'a' on q_1 we go to q_2 (final state)

aab

* Now aab is a valid string Not a valid string, why because every string starts with ab
* so we should not apply self Loop here.
* so if we apply a on q_1. Then we have to go to the dead state.



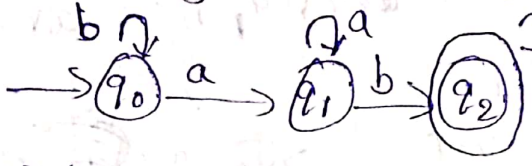
* on q_2 we can apply any a's and b's
* So starting with ab and Nx b'so are a's Not bother it

→ construct DFA that accepts set of all strings over {a, b} of strings containin "ab" as substring

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

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

length of string 'ab' so we require 3 states



* not DFA !

→ baaab

*apply b on q_0

* apply a on q_1 self loop

* on q_2 apply a and b.

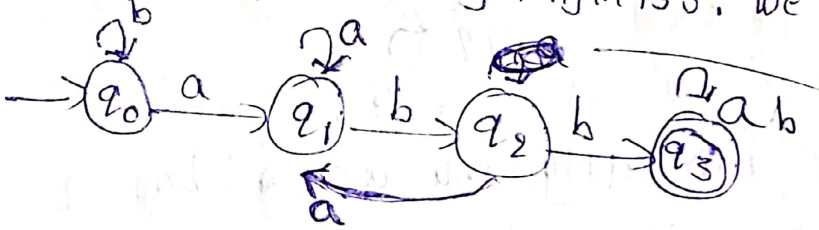
The

→ Having substring "abb"

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

$L = \{abb, aabb, babb \ldots\}$

Here the minimum possible string length is 3. we require 4 states



if we consid
self loop bab
so if it is to the
sub str

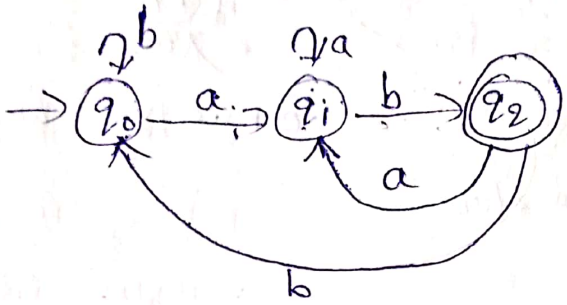
* so after a we
have abb as subst

→ construct DFA that accepts set of all strings over {a,b} of strings ends with "ab".

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

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

minimum is 2

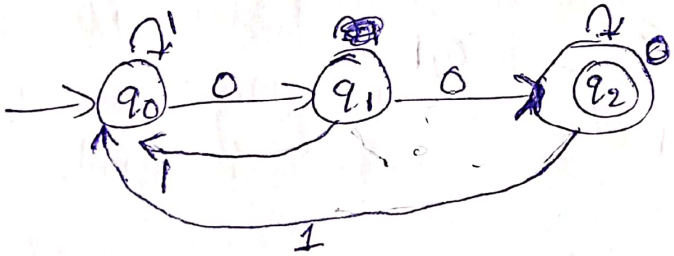


→ Every string ends with "00"

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

$L = \{00, 000, 100, 0100 \ldots\}$

minimum is 2

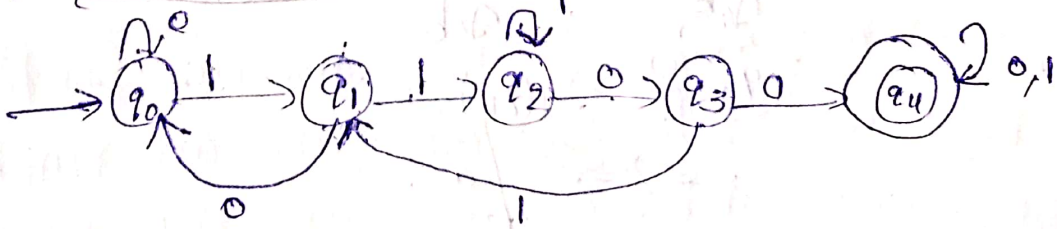


→ Design DFA which accepts set of all string containing 1100 as a substring.

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

$L = \{1100,$ 1110 1101

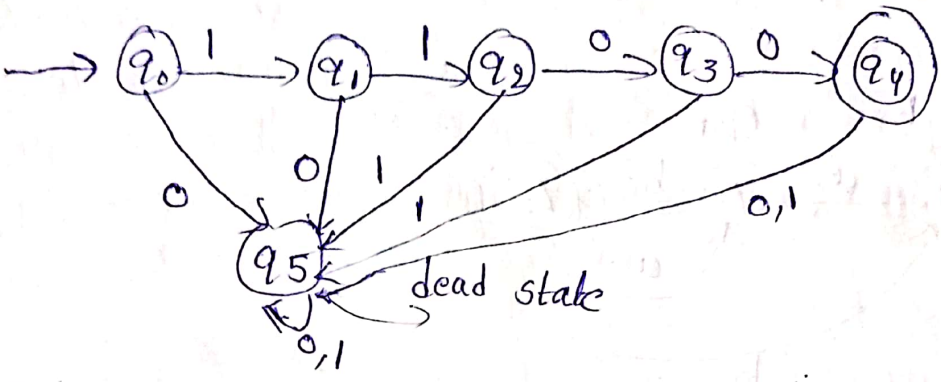
Min 4, require 5 states



→ Design DFA which accepts the string 1100 only?

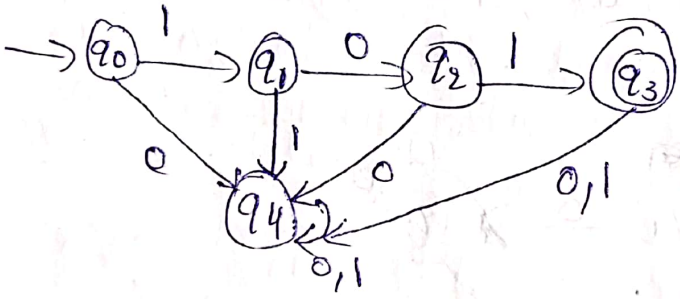
$\Sigma = \{0,1\}$

0 1100 X, not acc
1100 1 x It is
accepts only 11



→ Design DFA which Accepts the string 101 only?

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



0 1 0 1 X
101

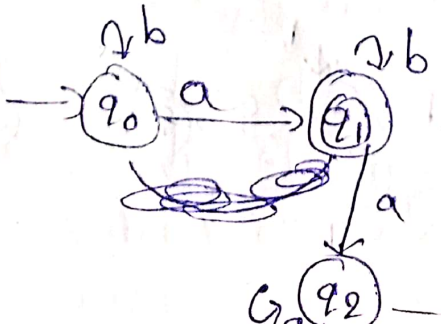
→ Design DFA which accepts set of all string over $\Sigma = \{a,b\}$ with Exactly one 'a'.

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

$L = \{a, ba, \cancel{aba}, bba, bbabb \ldots\}$

abbba X
only 1 'a'

Min: 1
2 states

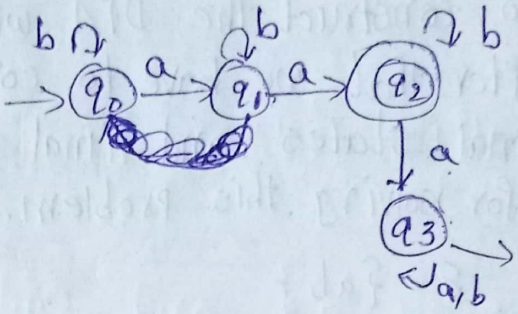


→ Design DFA which Accepts set of all strings over $\Sigma = \{a, b\}$ with Exactly two a's?

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

$L = \{aa, baba, bbbaa, \ldots\}$ abb ✗ 2 a's

min 2 . require 3 states

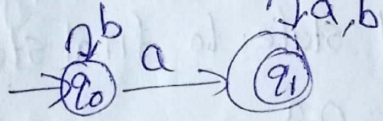


→ Design DFA which accepts set of all strings over $\Sigma = \{$ with Atleast one 'a' ~~Atleast state q3~~

$\Sigma = \{a\}$

$L = \{a, aa, aaab, aaabb \ldots\}$

min 1 require 2



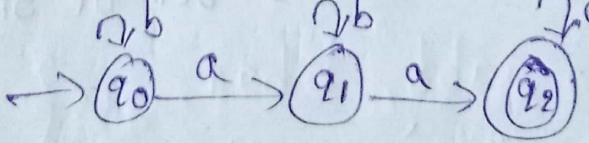
→ Design DFA which accepts set of all strings $\Sigma = \{a, b\}$ with Atleast two a's.

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

$L = \{aa, ababa, aaabb, aaaaba \ldots\}$

min-2

acceptable

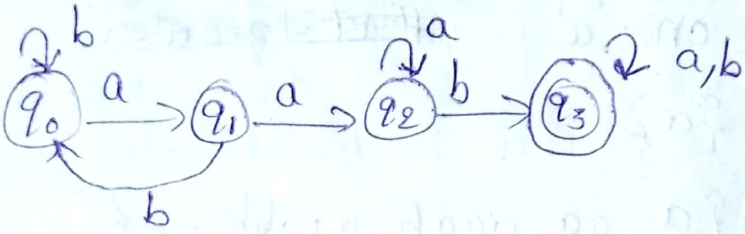


→ Construct DFA which accepts set of strings over {a,b} except the substring aab (or) not containing the substring aab.

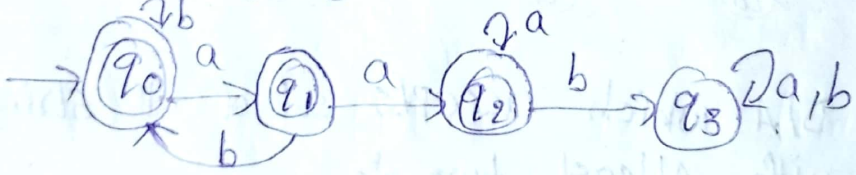
* inorder to solve these type of problems first we have to construct the DFA with the string aab and after that we have to convert a non final state into final states and final states into non-final states. for solving this problem.

$\Sigma = \{ab\}$

$L = \{aaab, baab \ldots\}$



* Now we construct the DFA not containing aab.
* Now convert Non-Final state to final state. Final state to Non-final state



* So here our DFA is not accepting the substring aab.

Suppose ex: aab
a to q_1 a to q_2 b to q_3
but q_3 not final state
so A is not accepte

→ construct DFA which accepts set of strings over {a,b} with

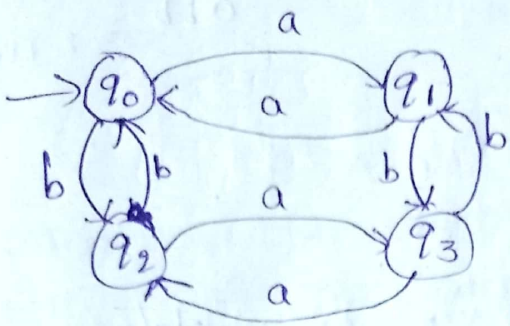
a) even number of a's and even number of b's
b) Even number of a's and odd number of b's
c) odd number of a's and Even number of b's
d) odd number of a's and odd number of b's

* inorder to solve These type of problems we have to take 4 states

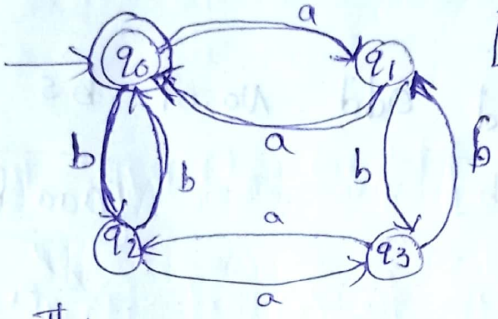
Even ↺ 2 2 ↻ even

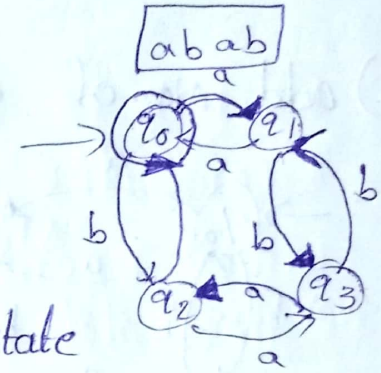
→ aabb

→ abab



a) for the first problem q_0 is final stat



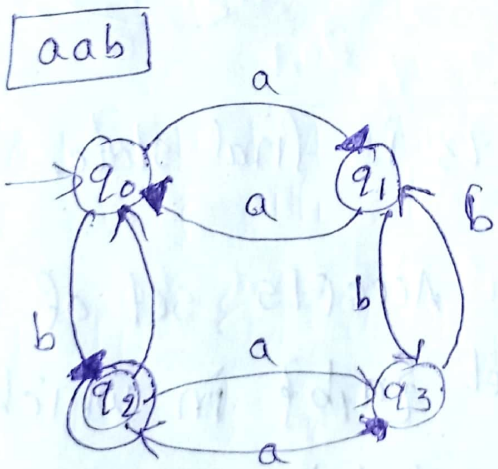


* so The q_0 is both the initial state and final state

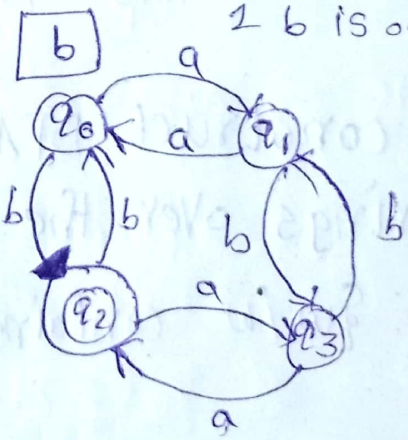
b) Even No. of a's and odd No. of b's

2 → even

aab → 1 odd



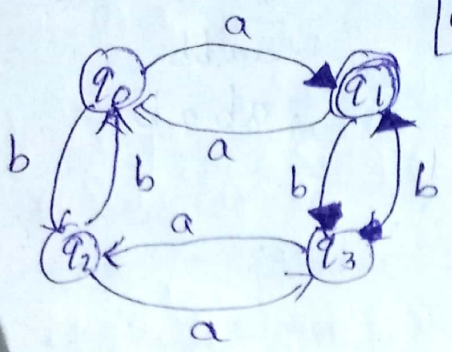
b 0 is also e
so No a'
1 b is odd.

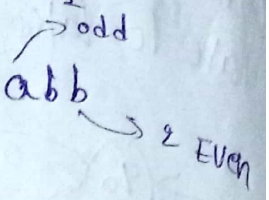


so make q_2 as final so ◎ circles

c) odd No. of a's and Even no. of b's

abb

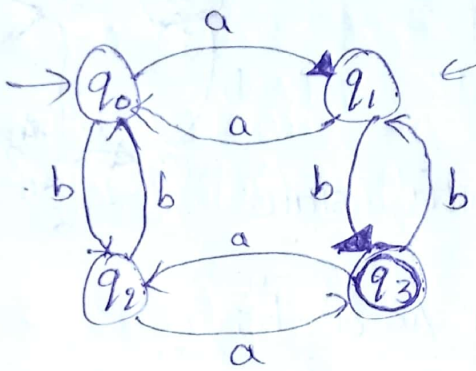




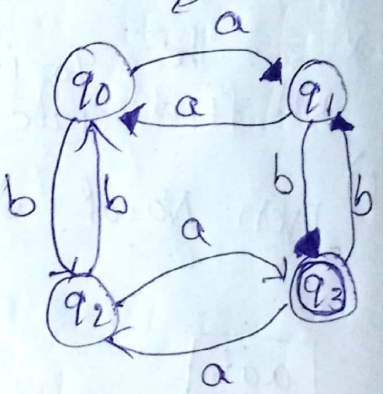
So for this problem q1 is final state.

d) odd No. of a's and odd No. of b's

ab



aaa bbb

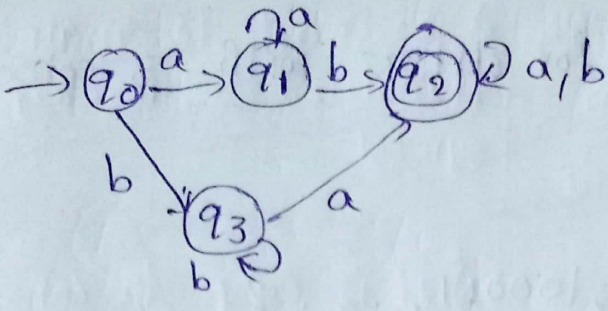


So for this problem q_3 is final state.

→ construct DFA which Accepts set of all strings over the alphabet {a,b} in which L = {w / w contains the substring ab or ba}

L = {ab, ba, aabb, bbaa, ...}

Min = 2, require 3 states

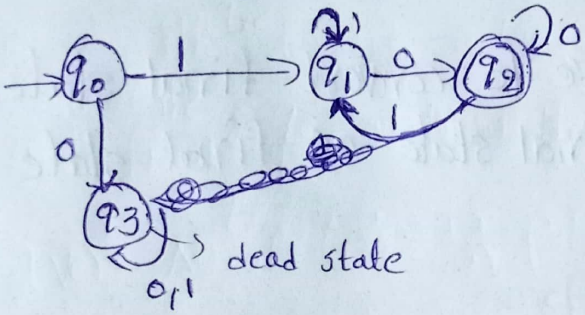


→ construct DFA which accepts set of all string over one alphabet $\{0,1\}$ in which

a) Every string starts with 1 & Ends with 0.

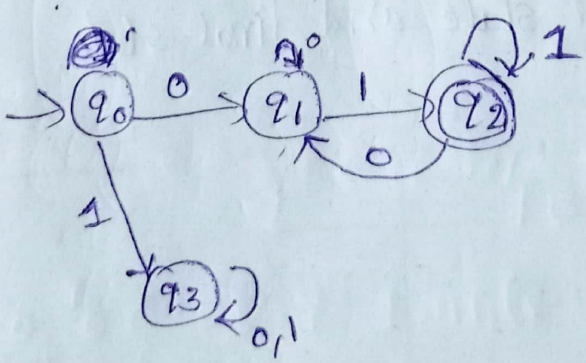
$\Sigma = \{0,1\}$

$L = \{10, 1010, 100 \ldots\}$



b) Every string starts with 0 & Ends with 1 in between 0 and 1 any strings are there.

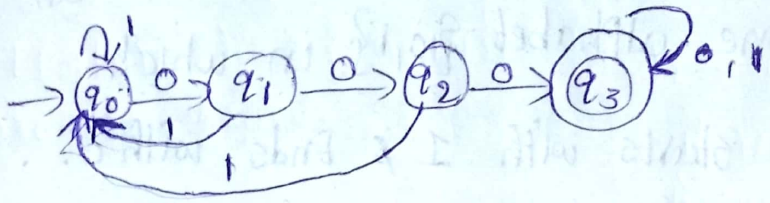
$L = \{01, 0011, 0111 \ldots\}$



→ Design a DFA for strings with 3 consecutive 0's

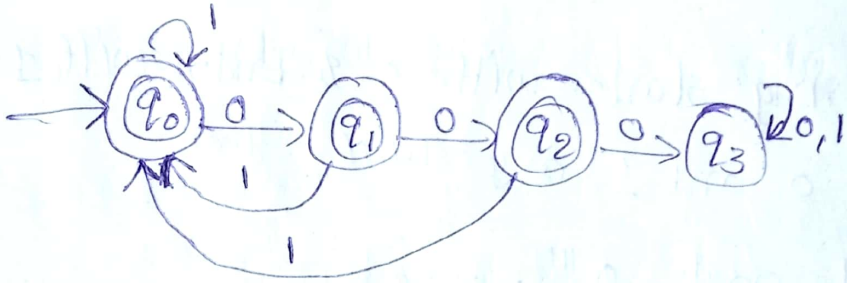
$\Sigma = \{0, 1\}$

$L = \{000, 100011, 100001, \ldots\}$



→ we have to Design a DFA for strings without 3 consecutive 0's?

without means, we have to convert final state Non-final state and Non final state as final state



Here q_0 acts a initial state and final state.

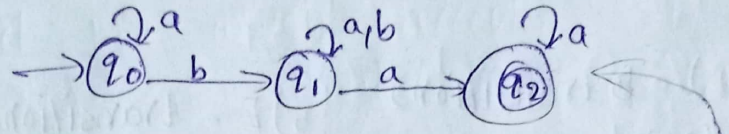
NFA

Non-Deterministic finite automata

NFA is defined by 5 tuples

in mean machine $m = (Q, \Sigma, \delta, q_0, F)$

ex:



$Q \rightarrow$ Finite set of state (countable states)

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

$\Sigma \rightarrow$ input alphabet.

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

$q_0 \rightarrow$ initial state (we have only 1 initial state in NFA or DFA)

$F \rightarrow$ set of Final states.

$F = \{q_2\}$

$\delta \rightarrow$ Transistion function which map from

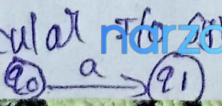
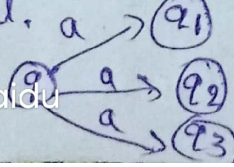
$\delta = Q \times \Sigma \rightarrow 2^Q$

$\delta(q_0, a) = q_0$

$\delta(q_0, b) = q_1$

$\delta(q_0, b) = q_0$

Diffirence of DFA and NFA

DFA	NFA
1. it is defined using 5tuples $m = (Q, \Sigma, \delta, q_0, F)$ $\delta : Q \times \Sigma \rightarrow Q$	1. $m = (Q, \Sigma, \delta, q_0, F)$ $\delta : Q \times \Sigma \rightarrow 2^Q$
). In case of DFA, every transition generates single state with a particular I/P symbol. 	2) In case of NFA, transition gene more than one outcome with a I/P symbol. a 

3. every state contain 1 transition with every I/o symbol in given alphabet.	3) No need to maintain transition with every I/p symbol
4) It is difficult to construct	4) It is easy to construct
5) Epsilon (E) transitions are not allowed.	5) E - transitions are allowed it means without applying input also we Moved.

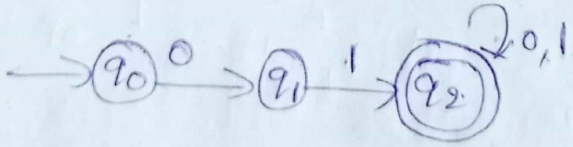
NFA examples

1. starts with '01'.

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

$L = \{01, 010, 011, 0100, 0101, 0110, 0111, \ldots\}$

Min = 2. requires 3 states

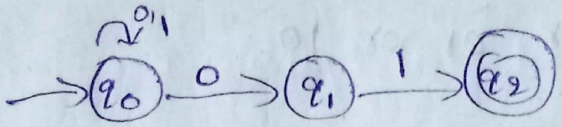


2. string ending with '01'

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

$L = \{01, 001, 101, 0001, 1101, \ldots\}$

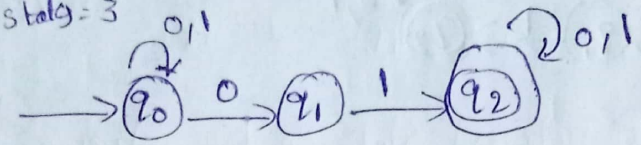
Min = 2 3 states



→ substring '01'

$L = \{01, 0011, 1010, \ldots\}$

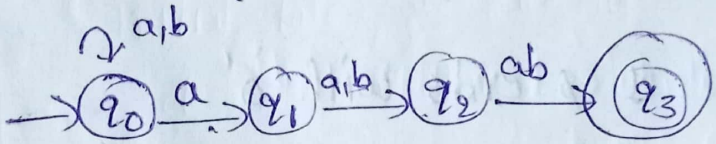
min = 2, states = 3



4) 3rd symbol from right end is 'a'

$\Sigma = \{a, b\}$ 3 symbol. so 3+ states.

$L = \{$



b a b a

↓

3rd right end.

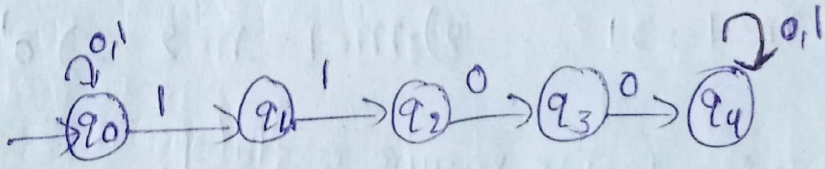
5) double '1' is followed by double '0'?

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

~~2 symbols requires 3 states~~

$L = \{1100$

4 min. 5 states

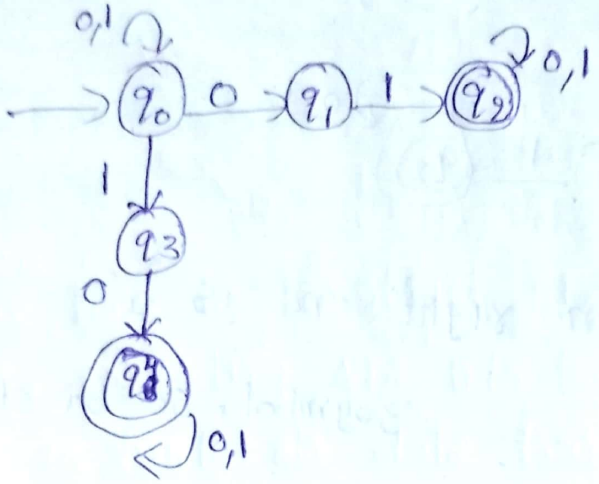


6. string contains either '01' or 10 ?

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

$L = \{01, 10$

Min 2.3 states

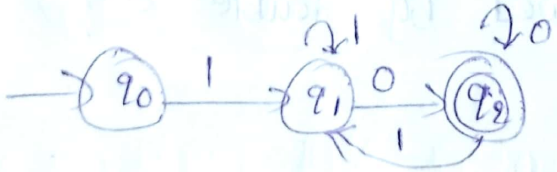


7. starts with 1 & ends with '0'

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

$L = \{10$

Min 2, require 3



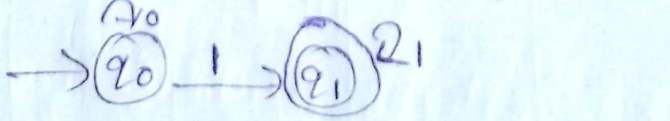
8. $L = \{0^m 1^n \mid m \geq 0$ and $n \geq 1\}$

$0^0 = 0$ ← $1^1 = 1$

1) $m = 0$, $n = 1 \Rightarrow 1$ 2) $m = 1$ $n = 2 \Rightarrow 0^1 = 0$, $1^2 = 11$

$= 1$ $= 011$

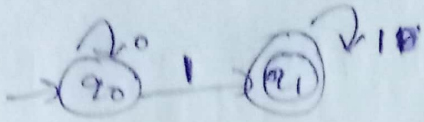
Min = 1. we require 2 states.



$m=2, n=3$

$0^2 = 00$ $1^3 = 111$

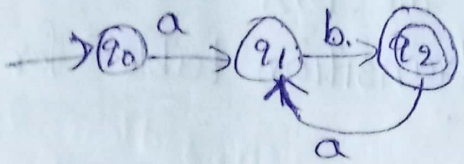
00111



9) $(ab)^n$ with $n \geq 1$

$n=1$

$n=1 \Rightarrow ab \rightarrow 2$ we require 3 states.



$n=1 \Rightarrow ab$

$n=2 \Rightarrow abab$

$n=3 \Rightarrow ababab$

10) $(ab)^n$ with $n>0 \rightarrow$

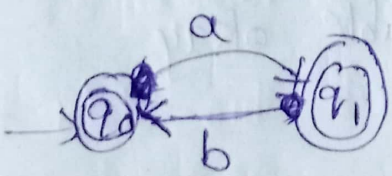
$n=0 \rightarrow n=0$

* if $n=0$ it is empty string

$n=0 \Rightarrow \phi$

$n=1 \Rightarrow ab$

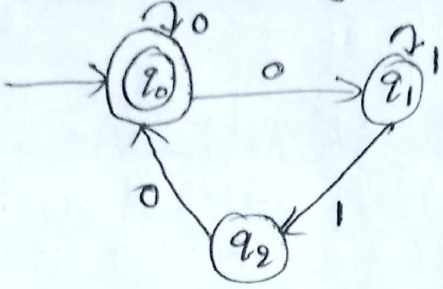
$n=2 \Rightarrow abab$



The both become final states.

Converting NFA to DFA (or) Equivalence of DFA & NFA

1) Design DFA from the given NFA



* So we have to construct DFA based on this NFA.
* before converting NFA into the DFA
* first construct state transition table for the NFA.
* so, for NFA calculate transition table

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

transition table (NFA)

* So construct DFA so for that let us ~~calcafate~~ let us construct the transition table okay

DFA transiton table

	0	1
→ q_0	$\{q_0, q_1\}$	ϕ
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	q_0	$\{q_1, q_2\}$

q_0, q_1 are new states

There is no need to process the ϕ because it is empty state.

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

$= \{q_0, q_1\}$

This is the new state

so no need q_1, q_2 is already explored

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

$= \phi \cup \{q_1, q_2\}$

$= \{q_1, q_2\}$

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

$= \phi \cup q_0$

$= q_0$

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

$= \{q_1, q_2\} \cup \phi$

$= \{q_1, q_2\}$

* According to the transition table we construc the DFA
* Here DFA mean if the input alphabet contains 2 sym Then Each state must consume those 2 symbols.
* but Here ϕ is there. ϕ means Empty symbol.
* for that purpose we assume the states as (q_d) q_d — dead state.
* Here what is the final state? q_0 is initial stat and final state only.
* The states (which contain) q_0 will become the final sta
* if we observe 1 state → q_0 make circle
* observe second row $\{q_0, q_1\}$. The second state contai q_0, so this state become the final state.
* Nxt observe third row $\{q_1, q_2\}$. There is No q_0 This is Not the final state.

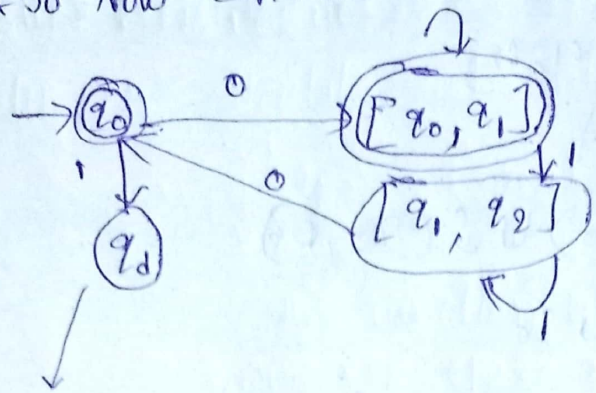
$\cup \delta(q_1, 0)$

ϕ

so all the explored, No New state. So The transition table is over

Now check any new states

* So Now construction transition table. constructing

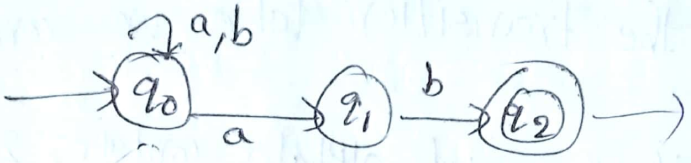


→ in this diagram instead of [] we use [] it's better.

q_0 on 1 means dead state based DFA table.

* So This is the DFA for this NFA. In this way we can solve the problem.

Ex: 2



→ in problem q_2 is final state which states ~~become~~ contain is become final state

* First construct transition table

	a	b
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
(q_2)	ϕ	ϕ

* Now let us calculate DFA transition table based on above Table

	a	b
→ q_0	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
($\{q_0, q_2\}$)	$\{q_0, q_1\}$	q_0

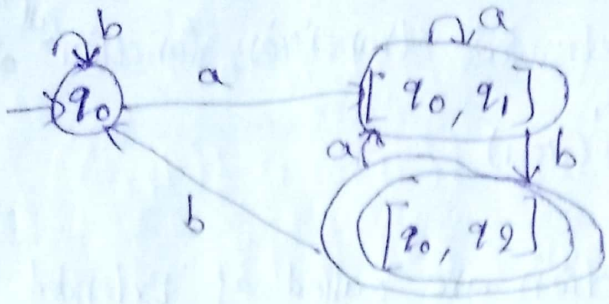
→ let check This states New or old

→ New state

→ New state

* So all are old states. Then the transition table is over.

* let us construct the DFA based on above dFA table.



* Here we no need to use q_b (dead state) because in DFA table there is No ϕ.

* So This is the DFA for the given NFA.

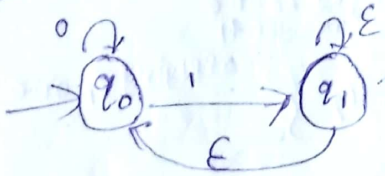
converting NFA with Epsilon Moves to NFA without ε-moves

* Here the Move also be called as transition.

* first we see what is the use of epsilon?

* epsilon transition is mainly useful inorder to move from one state to another state without applying any input symbol.

ex:



→ on aplying epsilon ε we are at q_1 onl

* Here we apply input symbols $\Sigma = \{0,1\}$

* whereas epsilon transision is ver useful inorder to move 1 transiti to another transition without applyi any input symbol

* Here our target is converting NFA with Epsilon Moves to NFA without epsion moves

* output is we have to produce NFA without ε-M

step 1 : Calculate ε-closure of all states.

step 2 : calculate extended transition function × all the states

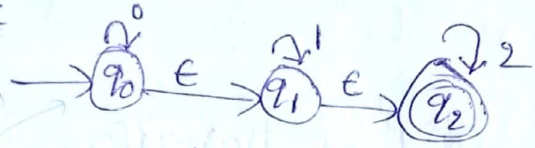
ex: $\delta'(q_0, 0)$ $\delta'(q_0, 1)$

↳ This function are called as Extended transition

* with NFA epsilon Moves we use δ.
* in NFA without epsilon moves δ' (delta dash)
* And This also called Extended

step 3 : based on the extend transition function construct NFA without ε-Moves

Ex



* Here This is the NFA with Epsilon Moves.
* first we find epsilon closure all 3 states. Know about ε definition.

ε-closure(q_0) :- set of states that are reacheable from state q with Epsilon-transition also be called Epsilon Moves.

self state

applying q_0 one ε we get q_1

applying q_1 one we get q_2

ε-closure(q_0) = $\{q_0, q_1, q_2\}$

ε-closure(q_1) = $\{q_1, q_2\}$

ε-closure(q_2) = $\{q_2\}$

- we Need to find the Expanded function.

Based on diagram

$\delta'(q_0, 0)$ = ε-closure (δ (ε-closure(q_0), 0)) → Formula

= ε-closure(δ ($\{q_0, q_1, q_2\}$, 0))

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

$\delta(q_0,1)$ $= \epsilon\text{-closure}(q_0 \cup \phi \cup \phi)$ → because q_1 are not applied not '0' zero.

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

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

delta bar $\hat{\delta}$

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

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

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

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

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

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

$= \{q_1, q_2\}$

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

$\delta'(q_0,2) = \epsilon\text{closure}(\delta(\epsilon\text{-closure}(q_0),2))$

$= \epsilon\text{closure}(\delta(\{q_0,q_1,q_2\}\,2))$

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

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

$= \epsilon\text{closure}(q_2)$

$= \{q_2\}$

$\rightarrow \delta'(q_0,2) = \{q_2\}$

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

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}, 0))$

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

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

$= \phi$

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

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}, 1))$

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

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

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

$= \{q_1, q_2\}$

$\delta'(q_1, 2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), 2))$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}, 2))$

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

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

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

$= \{q_2\}$

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

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

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

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

$= \phi$

$\delta'(q_2,1)$ = E-closure (δ (e-closure (q_2), 1))

= E-closure (δ ($\{q_2\}$,1))

= E-closure($\delta(q_2,1)$

= E-closure (ϕ)

= ϕ

$\delta'(q_2,2)$ = E-closure (δ(e-closure (q_2) 2))

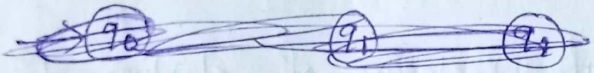
= E-closure (δ($\{q_2\}$ 2))

= E-closure($\delta(q_2,2)$)

= E-closure (q_2)

= $\{q_2\}$

* calculation of extended transition function is over
* based on transition function draw the NFA diagram.



* roughly i written answers

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

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

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

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

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

$\delta'(q_0,2) = \{q_2\}$

$\delta'(q_1,0) = \phi$

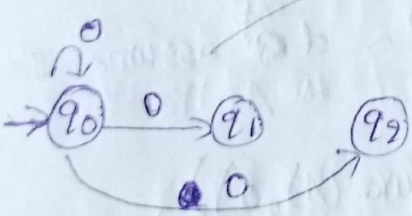
$\delta'(q_1,1) = \{q_1, q_2\}$

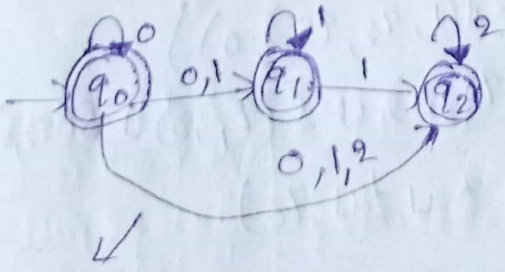
$\delta'(q_1,2) = \{q_2\}$

$\delta'(q_2,0) = \phi$

$\delta'(q_2,1) = \phi$

$\delta'(q_2,2) = q_2$





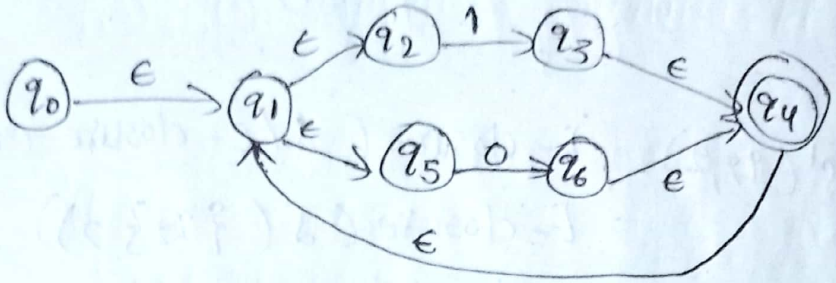
applying to ... it goes to ... applying ... it goes ... q_2. ... is final

* Now check which state to q_2. The final state

* So, this is the NFA without epsilon moves for this NFA
* Now, we have to find final state. in question NFA q_2 as final state

Converting NFA with ϵ-transitions to DFA

* input is NFA with ϵ-transition
* output is DFA



* There is no need to convert NFA with ϵ-transition to NFA without ϵ-transition. and then NFA will be converted to DFA
* So there is no need follow that approach
* Directly we convert NFA with epsilon transmission to DFA.

→ Here first we calculat ϵ-closure of starting state

* Epsilon is used to move to 1 state to another state without applying any input symbol

ϵ-closure (q0) = {q0, q1, q2, q5}

= A → let us assume the co as A.

δ(A, 0) = epsilon closure (δ(ϵ-closure (A), 0))

= ϵ-closure (δ(q0, q1, q2, q5), 0))

= ϵ-closure (δ(q0, 0) ∪ δ(q1, 0) ∪ δ(q2, 0) ∪ δ(

= ϵ-closure (ϕ ∪ ϕ ∪ ϕ ∪ q6)

= ϵ-closure (q6)

= {q6, q4, q1, q2, q5}

= B

$\delta(A,1)$ = E-closure (δ (E-closure {q_0, q_1, q_2, q_5}, 1))

= E-closure (~~E-closure~~ .

$\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1) \delta(q_5,1)$)

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

= Eclosure (q_3)

= {q_3, q_4, q_1, q_2, q_5}

= C

~~$\delta(q_1)$~~ =

* Now (A_0, 0) calculated (A,1) calculated. Next we calculate (b,0) (b,1) and then (c,0) (c,1)
* if we get any new state then we have to calculate them also.

$\delta(B,0)$ = ϵ-closure(δ (ex-closure(B), 0))

= ϵ-closure (δ ({q_6, q_4, q_1, q_2, q_5}, 0))

= ϵ-closure ($\delta(q_6,0) \cup \delta(q_4,0) \cup \delta(q_1,0) \cup \delta(q_2,0) \cup \delta(q_5,0)$)

= ϵ-closure ($\phi \cup \phi \cup \phi \cup \phi \cup q_6$)

= ϵ-closure (q_6)

= {q_6, q_4, q_1, q_2, q_5}

= B already existed

$\delta(B,1)$ = ϵ-closure (δ({q_6, q_4, q_1, q_2, q_5}, 1))

= ϵ-closure ($\delta(q_6,1) \cup \delta(q_4,1) \cup \delta(q_1,1) \cup \delta(q_2,1) \cup$

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

= C

$\delta(C,0)$ = ϵ-closure (δ(ex-closure(c), 0))

= ϵ-closure (δ({q_3, q_4, q_1, q_2, q_5}, 0)

= E-closure ($\phi \cup \phi \cup \phi \cup \phi \cup q_6$)

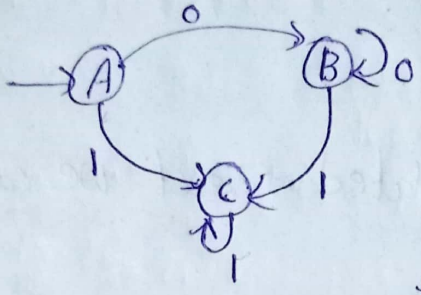
= B

$\delta(C,1) = \text{ex-closure}(d \cup \phi \cup \phi \; q_3 \cup \phi)$

$= \text{ex-closure}(q_3)$

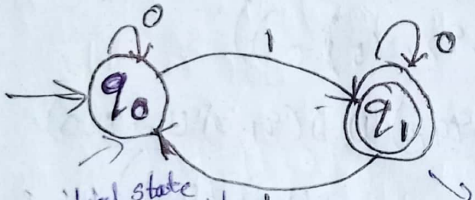
$= C$

* Now all the transition functions calculated let dr the DFA

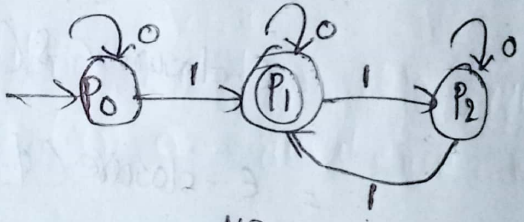


* So this is the DFA.
* All states uses both symbols (0,1).

Equivalence of 2 Finite state machines



M_1



M_2

* Let use see wheather 2 finite state machines ar equivalent or not.
* So, inorder to solve this problem first we have to the initial state of both the finite machines

	0	1
$\{q_0, P_0\}$	$\{q_0, P_0\}$	$\{q_1, P_1\}$
$\{q_1, P_1\}$	$\{q_0, P_1\}$	$\{q_0, P_2\}$
$\{q_0, P_2\}$	$\{q_0 P_0\}$	$\{q_1, P_1\}$

q_0 intermediate

P_0 interm

$\epsilon = (01) \rightarrow$ are

* 1 is inte other pair is Then we sa is not equi

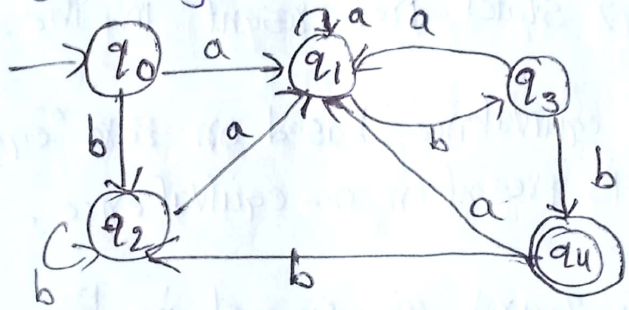
New state table over

New st Pair are final

* so all the states are explored. No new states here.
* Now we can say that these two atumoata are equivalent.
* suppose ~~any~~ The pair {q0, P1} are ther q0 is inter-mediat. P1 is final so we declare Not equivalent.
* but in above table ~~all are inte~~ it is not contains 1 pair as intermediate and 1 as final
* so, it is equivalent.

Minimization of DFA

* minimization of DFA means Reducing the no. of states in the DFA
* Here we have a DFS contain 5 states.
* so our target is to reduce the stats in DFA.



- we can minimize DFA by using 2 approches.
  1) Equivalence method (or) Partition method
  2) table Filling method (or) myhill-nerode Theorem

Equivalence method (or) partition method

Transition table

	a	b
→ q0	q1	q2
q1	q1	q3
q2	q1	q2
q3	q1	q4
(q4)	q1	q2

* first we have to find 0-equivalence.

0-equivalence :- 1st set represent set of non-final states
2nd set represents set of final states

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

1-equivalence → based on 0 equivalence
→ previous have 4 states we have to compare q_0 with q_1, q_1 with q_2, q_2 with q_3 whether they are equal or not.

let us tak the pair (q_0, q_1) and apply input a & b

$\delta(q_0, q_1)$

$\delta(q_0, a) = q_1$ | $\delta(q_0, b) = q_2$

$\delta(q_1, a) = q_1$ | $\delta(q_1, b) = q_3$

* After calculation these 2 states are present in the same state or not.
* Here we calculate one equivalence based on zero 0-equivalence
* we got q_1 checking it is present in 0-equivalence. Present.
* Next we got q_2 q_3. q_2 and q_3 present in the same set of 0-equivalence.
* so we can say that q_0 is equals to q_1. because q_1, q_1 means only one state so that is present in the same set
* we got the result as $q_2 q_3$, q_0 and q_3 present in the same set
* suppose we got q_4. q_0 is in one set q_4 is present in another set.
* then we can say that q_0 is not equal to q_1

$q_0 \neq q_1$

* but we have same states
  * so $q_0 = q_1$
* Now we have to compare q_0 with q_2 (or) can compare q_1 with q_2.
* let us compare (q_0, q_2)

$$\delta(q_0, a) = \begin{bmatrix} q_1 \\ q_1 \end{bmatrix} \quad \Big| \quad \delta(q_0, b) = \begin{bmatrix} q_2 \\ q_2 \end{bmatrix}$$
$$\delta(q_2, a) = \begin{bmatrix} q_1 \\ q_1 \end{bmatrix} \quad \Big| \quad \delta(q_2, b) = \begin{bmatrix} q_2 \\ q_2 \end{bmatrix}$$

* we got q_1 That is present in one set only.
* we got q_2 That is present in one set only.

→ so we can say that $q_0 = q_2$

* Now we compare q_0 with q_3 (q_0, q_3) (or) compare
* $(q_1$ with $q_3)$ or (q_2, q_3)

(q_0, q_3)

$$\delta(q_0, a) = q_1 \quad \Big| \quad \delta(q_0, b) = q_2$$
$$\delta(q_3, a) = q_1 \quad \Big| \quad \delta(q_3, b) = q_4$$

* q_1 is one state only There is No problem.
* you observe q_2 q_4. q_2 is present in one set q_4 is present in another set.
* so we can say q_0 and q_3 are Not equal

$$q_0 \neq q_3.$$

* Here $q_0 = q_1$ and $q_0 = q_2$. In 1-equivalence mean

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

(These all are equal)

* 0-equivalence calculated 1-equivalence calculated. Nex 2 eq 3-eq, 4-eq likewise we have to calculate until sub sequ equivalences are equal.

* suppose if 2-equivalence and 3-equivalence are same Then we quit the process merge the state.

* Now let us calculate 2-equivalence.

2-Equivalence → if 2 equivalence result also same Then we quit the process otherwise we calculate 3-equivalence.

* Here 2-Equivalence calculated based on 1 equivalence.

* The 1-Equivalence result $\{q_0, q_1, q_2\}\ \{q_3\}\ \{q_4\}$

* Now compare q_0 with q_1

(q_0, q_1)

$$\delta(q_0, a) = q_1 \quad | \quad \delta(q_0, b) = q_2$$
$$\delta(q_1, a) = q_1 \quad | \quad \delta(q_1, b) = q_3$$

* Here we got q_1 that is one state No problem.

* And also other pair we got q_2, q_3. The q_2 is present in one set, q_3 is present in another set.

So we can say that

$$q_0 \neq q_1$$

(q_0, q_2)

$$\delta(q_0, a) = q_1 \quad | \quad \delta(q_0, b) = q_2$$
$$\delta(q_2, a) = q_1 \quad | \quad \delta(q_2, b) = q_2$$

$$q_0 = q_2$$

Now the 2-equivalence is over

$\{q_0, q_2\}\ \{q_1\}\ \{q_3\}\ \{q_4\}$ → remaining set of symbols

* if 3 equivalence result also same as 2- equivalence then we can stop the process and we can merge q_0 and q_2.

* let us calculate 3-equivalence

3 - Equivalence

→ (q_0, q_2) 3 equivalence calculated based upon 2nd equivalence.

$\delta(q_0, a) = q_1$ | $\delta(q_0, b) = q_2$

$\delta(q_2, a) = q_1$ | $\delta(q_2, b) = q_2$

$q_0 = q_2$.

So 3rd Equivalence is similar second equivalen

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

* so now we can stop the process and here we can combine and merge q_0 & q_2

* we can merge q_0 & q_2 into 1 state $\{q_0, q_2\}$

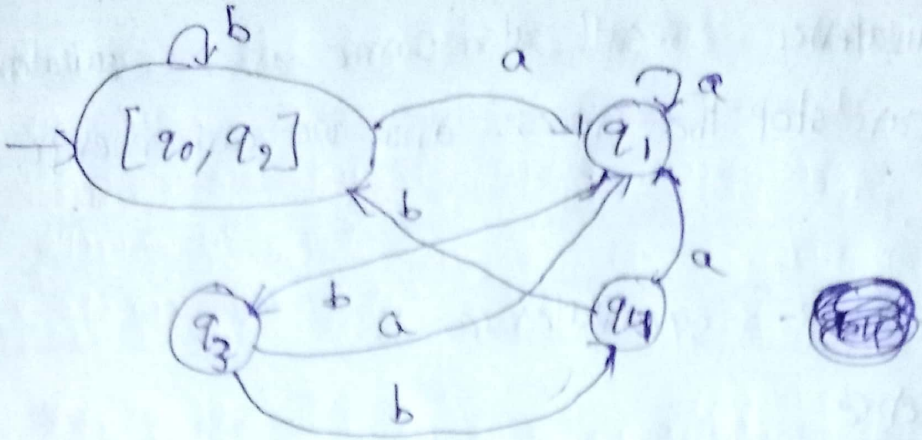
* initially in table we have 5 states q_0, q_1, q_2, q_3, q

* Now those 5 states are reduced to 4 states

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

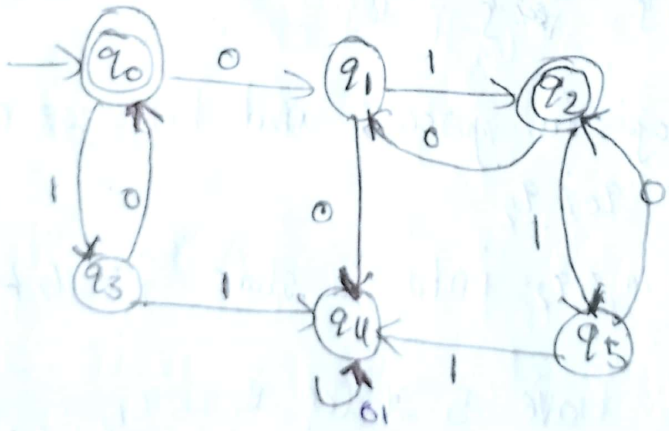
* so This is about minimization of DFA.

* NO we construct DFA



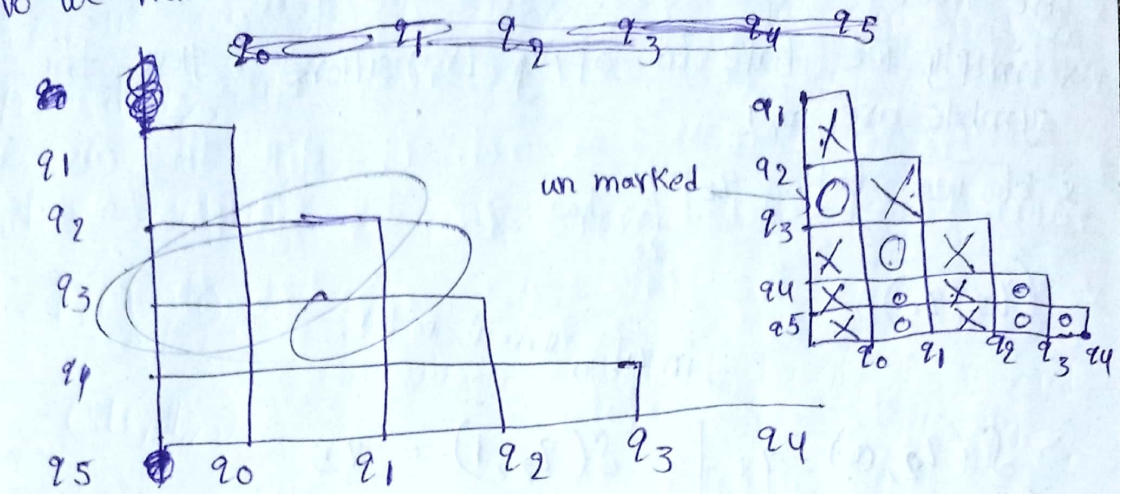
* In this way we construct the DFA
* ~~in~~ Minimization of DFA (or) NFA have same approach
* in NFA we have many final states. In DFA we have one final state
* Nxt we see second method

→ Minimization of finate automate using Table filling method (or) myhill



	0	1
→ (q0)	q1	q3
q1	q4	q2
(q2)	q1	q5
q3	q0	q4
q4	q4	q4
q5	q2	q4

* transition table is over
* No we have to construct other table.



* Let us eliminal last stage on horizontal first stage on vertical.
* So we have to draw a table like these now we have to mark the state like (✓ or x)
* we can mark a pair if the pair contain final state as well as non-final state (f.s & n.f.s)
* if the pair contain only final states Then marking is not possible (fs, fs)
* simply we have to leave that pair.
* like if the pair contains non final states (n.f.s) Then also leave the box.
* when we marking?
  -> one state is final state another state is Non-final st
* Here q_0 and q_2 are final states (q_1 q_3 q_4 q_5) are Non-f-nal states
* q_0 is final state q_1 is Not final state so mark (✓) X wrong mark.
* q_2 月 . so that box have q_2 and q_0 both are final states . simply we have to leave that box.

* now we have to check the Marking is Possible are not.

* simply we have to apply transitions. Here the symbols are 0, 1

* let we take q_2 $\square$ q_0 box.

(q_2, q_0)

$\delta(q_2, 0) = q_1$ → in $q_1 \square q_1$ there is no box.

$\delta(q_0, 0) = q_1$

$\delta(q_2, 1) = q_5$

$\delta(q_0, 1) = q_3$

Not Marked

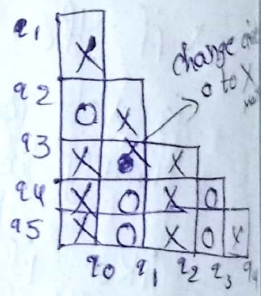
q_5 0 q_3

* So Marking that pair is not possible

* if one this pair $\begin{matrix} q_1 \\ q_1 \end{matrix}$ is Marked

* if one this pairs $\begin{matrix} q_1 \\ q_1 \end{matrix}$ or $\begin{matrix} q_3 \\ q_3 \end{matrix}$ are Marked. Then we can mark (q_2, q_0)

* so we have to mark (q_2, q_0) is Not Possible. So leave the box $\begin{matrix} q_2 \\ \square \\ q_0 \end{matrix}$ as it is.



→ (q_3, q_1) → we taking 0 boxes.

$\delta(q_3, 0) = q_0$

$\delta(q_1, 0) = q_4$

$\delta(q_3, 1) = q_4$

$\delta(q_1, 1) = q_2$ → There is no need apply 1 symbol because pair already

* Here (q_0, q_4) pair is already marked. so we can mark (q_3, q_1) pair

→ (q_4, q_1)

$\delta(q_4, 0) = q_4$ → No X pair. so we go to 1 symbol

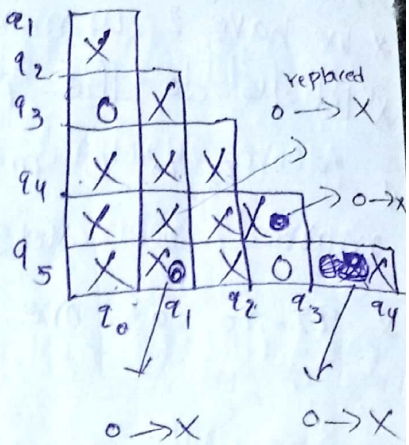
$\delta(q_1, 0) = q_4$

$\delta(q_4, 1) = q_4$

$\delta(q_1, 1) = q_2$

* (q_4, q_2) q_4 [X] q_2 is marked so we Mark the pair

→ (q_4, q_1)



→ (q_4, q_3)

$\delta(q_4, 0) = q_4$

$\delta(q_3, 0) = q_0$

* (q_4, q_2) is already marked. so we can mark the (q_4, q_3). we no need to go to 1 symbol.

→ (q_5, q_1)

$\delta(q_5, 0) = q_2$

$\delta(q_1, 0) = q_4$

* (q_4, q_2) is already marked. so we can mark $(q_5, q$

→ (q_5, q_3)

$\delta(q_5, 0) = q_2$ | $\delta(q_5, 1) = q_4$

$\delta(q_3, 0) = q_0$ | $\delta(q_3, 1) = q_4$

bd marked symbol.

There is no marked pair. so we can not m

(q_5, q_3) box.

→ (q_5, q_4)

$\delta(q_5, 0) = q_2$

$\delta(q_4, 0) = q_4$

* (q_4, q_2) is already marked. so we can mark $(q$

* Now we have to do one more iteration.
* We have 2 unmarked (○) pairs are there.
* let us do the process one more time
* if (q_2, q_0) is marked or not. if still they are not marked we Merge
* q_2-q_0 as one state and q_3 and q_5 to one state.

→ (q_0, q_2)

$\delta(q_0, 0) = q_1$	$\delta(q_0, 1) = q_3$
$\delta(q_2, 0) = q_1$	$\delta(q_2, 1) = q_5$

* (q_3, q_5) is not Marked. So marking (q_0, q_2) is not Possible

→ (q_3, q_5)

not marked

$\delta(q_3, 0) = q_0$	$\delta(q_3, 1) = q_4$
$\delta(q_5, 0) = q_2$	$\delta(q_5, 1) = q_4$

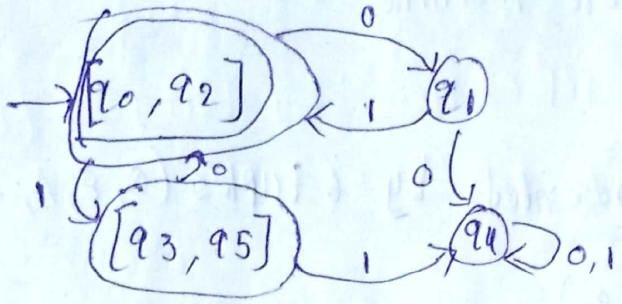
* (q_4, q_4) are no boxes at these.
* So we can say that q_2 ☐ q_0 and q_5 ☐ q_4 are not marked
* so now we can merge these 2 states into the one state.

(q_0, q_2) q_1 (q_3, q_5) q_4

* Initially the finate automata have q_0..q_5 6 state.
* It reduceses to 4 states.

(q_0, q_2) q_1 (q_3, q_5) q_4

* Now let us construct the finite automad. (or) ~~NFA~~



* In this way we can minimize DFA by using 2 approaches such as first approach is 1) Equivalence method (or) Partitioning method
2) Table filling (or) myhill Nerode theorem

Finite automata with ~~automata~~ outputs

* The best example of finite automata with output are
  1) Moore Machine
  2) Mealy Machine.

* Wheras as DFA and NFA are the examples of finite automata without outputs
* because DFA (or) NFA doesnt produce any results
* after reading the input string if we get the final state then the string is said to be accepted
* after reading the input string we don't get the final state then we can say that the string is not accepted. String is rejected

~~* String is~~

* that's why the DFA and NFA are the Examples for finite automata without outputs
* Wheras according to the input if we want to produce some output then we have to use finite automata with outp

* So the best examples are 1) moore Machine 2) Melay

* first let us see about Moore Machine.

Moore Machine

Moore Machine is represented by 6 tuples ($Q, \Sigma, \Delta, \delta,$

ϵ - tuple $(Q, \Sigma, \Delta, \delta, \tau, q_0)$ ← big delta

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ input alphabet

$\Delta \rightarrow$ output alphabet → in Moore machine within the circle we specifies output ($q_0/1$) → Here o/p is 1.

$\delta \rightarrow$ transition function → * in Moore Machine output is specified within the state only

$Q \times \Sigma \rightarrow Q$ → output

(current state, input)

(lambda) $\lambda \rightarrow$ output function → in Moore machine the output always depends up on the present state

$Q \rightarrow \Delta \rightarrow$ output

(present state)

$q_0 \rightarrow$ initial state

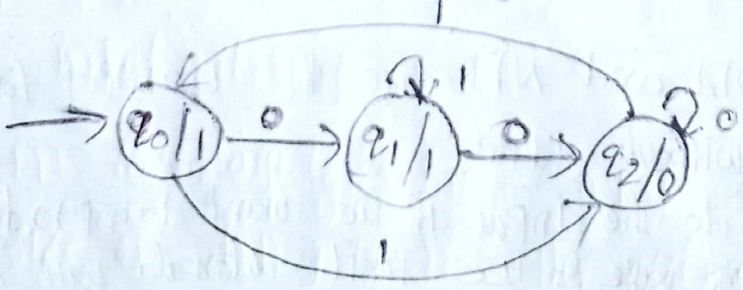
* in Moore Machine (or) Melay Machine doesn't contain any final state

* our target is to produce an output according to the string

* so, that why Moore Machine and melay Machine doesn't have any final state.

* so it is Explanation of Moore Machine.

* Now let us see the diagram.



* if we observe the diagram q_0 is initial state we don't have any final states here.

* The output (q_0) is 1 ($q_1/1$) q_1 output = 1 ($q_2/0$) q_2 output = 0

transition table

current state	Next state		output (λ)
	0	1	
q_0	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

→ doesnot depend only on prsnt state.

* in more machine output is specified in the state

I/P string = 0110

* let we have input string 0110
* Now we have to generate the output string
* in more machine o input string is n. it produces n+1.

0110

$\delta(q_0, 0110) = 1$

q_0 output = 1

$\delta(q_0, 0110) = 1$ → output

applied

$\delta(q_1, 110) = 1$

remaining

q_0 on 0 → q_1

$\delta(q_1, 10) = 1$

$\delta(q_1, 0) = 1$

q_1 on 1 → q_1

q_1 on 0 → q_2 ← q_2 → but the entire string is processed.

so the output string is

11111

So, This is about ~~More~~ Moore Machine.

* Now we go to Meelay Machine.

Meelay Machine

Meelay Machine also represented by using 6 tuples

6-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$Q \rightarrow$ finite set of states

$\Sigma \rightarrow$ input alphabet

$\delta \rightarrow$ transits function

$Q \times \Sigma \rightarrow Q$

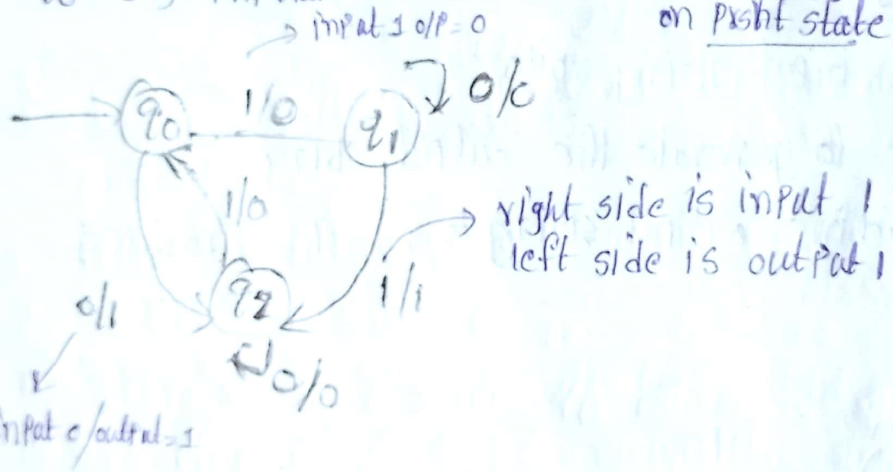
$\lambda \rightarrow$ output function

$Q \times \Sigma \rightarrow \Delta$ →

* in More machine the output depends on present state $Q \leftarrow \lambda$

* in meelay machine output depends on present state and present input

$q_0 \rightarrow$ Initial state



current state	input symbol 0		1	
	next state	output	next state	output
q_0	q_2	1	q_1	0
q_1	q_1	0	q_2	1
q_2	q_2	0	q_0	0

input string = 100

* in meelay machine input string contain. The size of string is n The output also n.

* The input string size is 4. The output size string also 4.

$\delta(q_0, 1001) \rightarrow 0$

$\delta(q_1, 001) = 0$ — q_1 on 0 goes to q_1

$\delta(q_1, 01) = 0$ — q_1 on 0 goes to q_1, q_1 output is 0

$\delta(q_1, 1) = 1$ — q_1 on 1 goes to q_2, q_1 output is 1

$\delta(q_2$ → it ends at q_2 state

The output string = 0001

* so input string length is 4 same as out string also 4

input string = 1001

output string = 0001

* so This is about finite Automata with output using Moore Machine and meeley machine.

Design a moore machine to find the 1's complement of a given binary number

* binary number means 0's and 1's.

$\Sigma = \{0, 1\}$ → binary number → The input is converted into 1's complement

* 1's compliment 1's will be converted to zeros. Zeroes will be converted to ones.

ex: 1 0 1 1

↓ ↓ ↓ ↓

0 1 0 0 → output alphabets also contain 0 and 1's

* output alphabet is represented by Δ

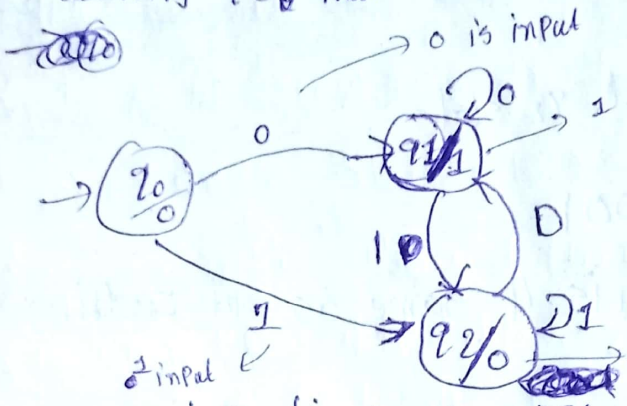
$\Delta = \{0, 1\}$

* Now let us design the Moore machine problem
* Here we can solve this ~~method~~ using 2 ~~states~~ Methods

1) first method is by taking 3 states.
2) by taking 2 states

method 1 (3 states)

* let initial state q_0. we can take output of q_0 is 0 or 1 according to that we change the transition.



* More and Machines are DFA s . so we use both input symbols.
* Tnital ~~q0~~ taking q_0 and output as 0.
* Method 1 3 states so q_0 q_1 q_2
* The q_0 output is '0' so q_0 ~~output~~ is 0
* q_1 ~~output~~ input is '0'. The q_1 output is 1.

* This Moore machine sore solve this problem.
* let us check wheather the machine is correct or not.

take

* let the input string 1011
* The output is 0100 is output
* initially we are at q_0 state

1 0 1 1

q_0 q_2 q_1 q_2 q_2

0 0 1 0 0 ⟶ outputs of q_0, q_2, q_1, q_2, q_2.

q_0 on 1 go to q_2 on '0'

~~if the size~~

* in more machine the size of string is n, The output string is n+1.

* Here input string size is 4. out string size is 5

input string 1 0 1 1

output string 0 0 1 0 0

There is no need of zero we have to neglect it.

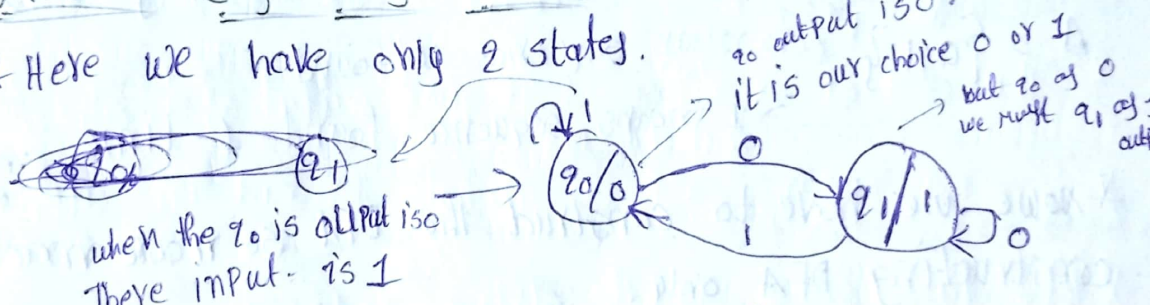
* in more machine means output is same irrespective of input ex: $(q_0/0)$ The q_0 is 0. wheather we apply zero(0) or 1.

* like wise The output of q_1 is 1. we apply 0 or 1.

* like wise The output of q_2 is 0. we apply 0 or 1

Methodz (by taking 2 states)

* Here we have only 2 states.



* Now we check machine is correct or not

* let take input string 1011

1 0 1 1

q_0 ~~q_0 q_1~~ q_1 q_0 q_0 q_1

0 0 1 0 0

* in more machine input string n output string size n

* in more machine we neglect first 0. (or) left most bit.

1 0 1 1

0 1 0 0

* So in this we design a more machine to find the 1's complement of a given binary number

Design a moore machine that counts the occurrence of the sequence 'abb' in any input string over {a,b}

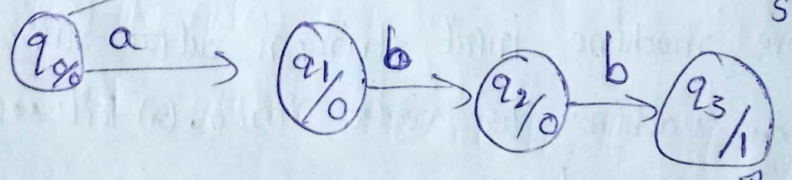
$\Sigma = \{a, b\}$

* Here we have to find out how many types abb is present in the given string.
* The No. of occurences in the given string
* for example aba it is not as abb so the output is 0
* for example abb. 1 time abb present
* for example abbbabb. 2 times abb present.
* Here the output alphabet contains mainly 2 symb.

$\Delta = \{0, 1\}$ → zero mean sequence not found → 1 means sequence found 1 time.

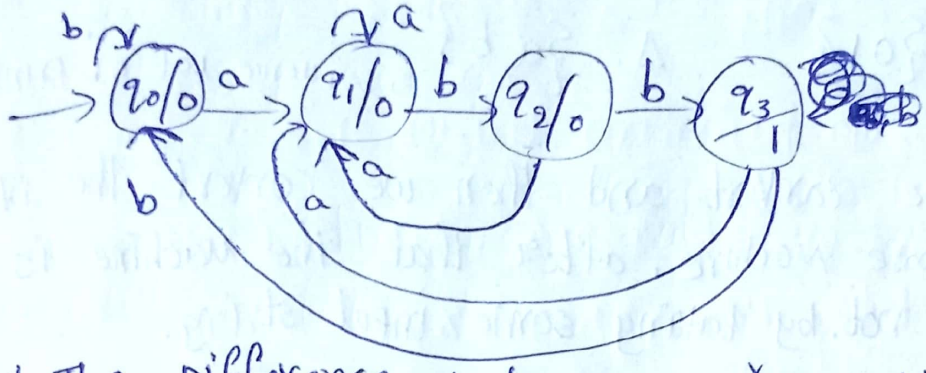
* Nowe we have to construct the DFA so moore machine (or) mealy machine constructing DFA only.
* Here the sequence contains 3 symbols abb. size of abb is 3.
* so Here the DFA contains 3+1 4 states.
* so states of DFA are. 4

→ we know that more machine have starting state only No Final state



* so Here the sequence is abb so output as 1. otherwise 0
* '0' Means the sequence Not found
* '1' Means that sequence is found.

* Now we have to make this as DFA.
* DFA means states we apply the input symbols
* Here input alphabet $\Sigma = \{a, b\}$ 2 symbols. so we apply both a & b on each state.



* The Difference between moore machine and melay machine are
* in Moore Machine each state has an output irrespective of any input
* DFA doesn't produce output. but Moore machine (or) Mealy produce output.
* let's take an input string abb

abb → abb means 1 time. Now check whether it is crct or not.

q_0 q_1 q_2 q_3
0 0 0 1 → it means abb occared one time. we are cr

* let us take an input string abbbbabb
* the out put becomes 2 because 2 abb's present. Now ch it is crct or not.

abbbbabb

q_0 q_1 q_2 q_3 q_0 q_0 q_1 q_2 q_3
0 0 0 1 0 0 0 0 1

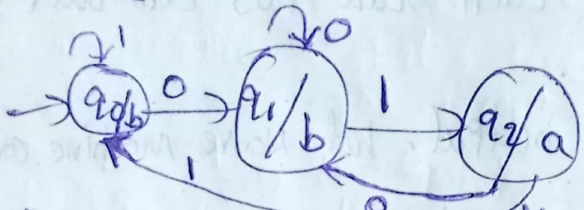
* it means abb occured twice in the input string.

Design a more machine that ~~counts~~ prints a when ever the sequence '01' is encountered in any input string.

input → $\Sigma = \{0, 1\}$; output → $\Delta = \{a, b\}$; if sequence '01' prints a ; sequence not '01' Prints b

* first we convert and then we convert the DFA into Moore Machine. after that the Machine is valid or not. by taking some input string.

* The ~~input alp~~ sequence contain 01. so we require 3 states



* The ~~of~~ 01 are present it gives output of 'a'. otherwise

each state uses

* in DFA means ~~each~~ all symbols present in alphabet

* lets take some input string 1010001

1 0 1 0 0 0 1

q_0 q_0 q_1 q_2 q_1 q_1 q_2

~~b b b a b b b a~~

0 0 0 1 0 0 0 1

* The output of q_2 is 1. remaining states '0' only.

* with this we got 1+1

* 2 times 01 sequence present in the string.

→ Design a moore machine that takes set of all strings over {0,1} and produces 'A' as output if input ends with '10' or produces 'B' as o/p is I/p ends with '11' otherwise produces 'C' as o/p.

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

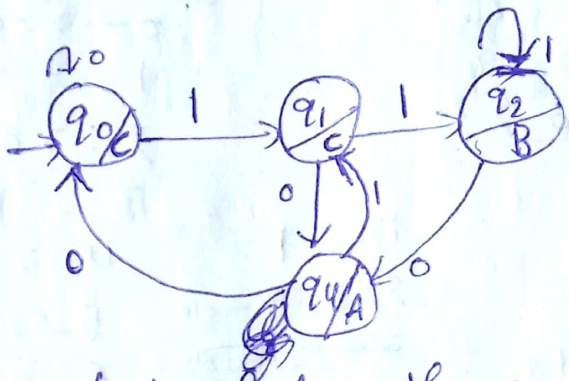
~~Ans~~

~~if input~~ * produces 'A' as output input ends with 10

$A \rightarrow 10$
$B \rightarrow 11$
$C \rightarrow$ otherwise

Δ → output aphabet $= \{A, B, C\}$

* in moore construct DFA only but each state has output
* The ~~in~~ each ~~output~~ string contains 2 letters. so we require 3 states.



* Now check wheather the moore Machine correct or no

→ input string 0010

0010
$q_0\ q_0\ q_0\ q_1\ q_4$
C C C C A → so we get A of output

let we take another string ends with 11

010011

$q_0\ q_0\ q_1\ q_4\ q_1\ q_4\ q_1\ q$
C C C A C A C

~~Ans~~ let the output doesn't contain '10' or '11' it Produces C

→ input string 001

0 0 1
$q_0\ q_0\ q_0\ q_1$
C → 'c' of output.

B of outp
because which is 'a'
pattern 'B' that is out

Design a moore machine to determine residue modulo 3 for binary number.

→ what is modulo after performing Division with 3

* binary number is a combination of 0, and 1.

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

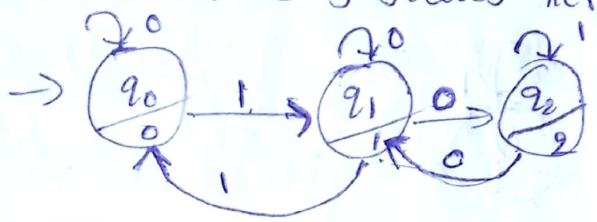
* The modulo performing Division on 3 we may get zero or 1 or 2
* Maximum we get these 3 values only. so $\Delta = \{0,1,2\}$
* if we take 1%3, 2%3, 3%3

3)1(0 → 1 → Mod

3)2(0 → 2

3)3(1 → 0 → remainder

* for example 72%3. 3*24 = 72. so the result is 0.
* so we require 3 states here



suppose

0 → 000
1 → 001
2 → 010
3 → 011
4 → 100
5 → 101
6 → 110
7 → 111

1%3 = 1 → 001

↓

we change at output 1

2%3 = 2 → 001

3%3 = 0 → 011

↓

we make changes output 0. it means q_0.

1%3 = 1

2%3 = 2 → 010

3%3 = 0 → 011

4%3 = 1 → 100

5%3 = 2 → 101

↓ so we will be at q_2 only.

reach → apply 011 we get q_0 state

4%3 = 1 = 100

↓

we Make changes as output 1. it mean q_1

5%3 = 2 = 101

6%3 = 0 = 110

* Now this is the Moore Machine
* Each state applied both symbols.
* Now check it is correct or not.

* let us take 15 as input string

15 → 1111 15%3 = 0 → q_0

q_0 q_1 q_0 q_1 q_0
0 1 0 1 0 → This 0 is output. so crct.

→ let us take 10 as input string

10 → 1010 10%3 = 1 → q_1

q_0 q_1 q_1 q_0 q_1 0

1 → output. so it is correct.

* so we can conclude that our moore machine is crct one

Design a moore machine to determine residue module 4 for binary number.

→ binary means 0 & 1 → max output 3 only. 8 4 2 1

$\Sigma = \{0,1\}$ $\Delta = \{0,1,2,3\}$ 0

* let us consider 0 1 2 3

0%4 4)0(0 0

1%4 4)1(0 0 1

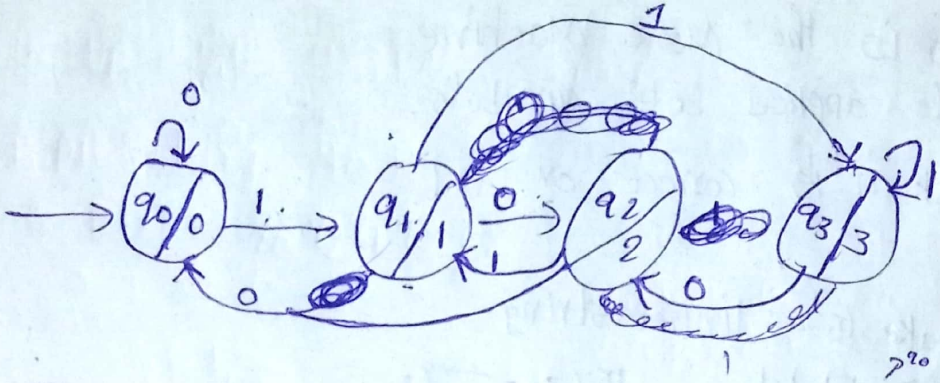
2%4 4)2(0 2 0

3%4 4)3(0 0 3

* let us take 72%4

4)72(18
4
32
32
0

* Now we design the moore machine
* The output alphabet contain 4 symbols. we require 4 states



4 %. 4 = 0 → 0100

↓

for 0100 we reach q_0 state

5 %. 4 = 1 → 0101

6 %. 4 = 2 → 0110

↓ q_2

7 %. 4 = 3 = 0111

↓

q_3 apply 0111 to reach q_3.

8 %. 4 = 0 = 1000

↓

q_0

9 %. 4 = 1 → 1001

↓

q_1

10 %. 4 = 2 → 1010

↓

q_2

0 %. 4 = 0 → 0000 → q_0

1 %. 4 = 1 → q_1 → 0001

2 %. 4 = 2

↓

we perform 0010 we get q_2 state.

3 %. 4 = 3

↓

0011 we reach to q_3.

8 4 2 1

0 → 0000

1 → 0001

2 → 0010

3 → 0011

4 → 0100

5 → 0101

6 → 0110

7 → 0111

8 → 1000

9 → 1001

10 → 1010

* So Each state applied both symbols. so The Mode Machine is over.

* Now check whether it is crct (or) not. by taking input str

15 → 1111

q_0 q_1 q_3 q_3 q_3 15 %. 4 = 3

3 → so output same. correct ✓

4) 15 (3

12

(3)

10 → 1010

1 0 1 0

q_0 q_1 q_2 q_1 q_2 ← 10/4 = 2

2 - so it is corred.

So In this we design a moore machine in reciduc module of 4 for binary number.

Design a moore machine to find the 1's complemen of a given binary number

*binary means 0 & 1's

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

* Here we find 1's complement. it means 1's converte to zero. Zero converted to 1.

1 0 1 1

↓ ↓ ↓ ↓

0 1 0 0 → so output alphabet also contain 0 & 1

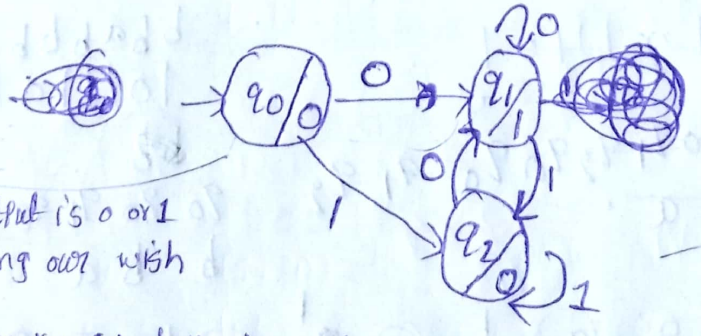
* output alphabet represented as Δ.

* Now let us design moore machine. Here we can solve this problem by using 2 methods.

1) by taking 3 states.
2) by taking 2 states.

* Now let us solve the problem in 2 methods.

* by considering method 1 (3 states)



The output is 0 or 1 according our wish

whe q_2 got outp as 0. because of input is 1. so se

The more & relay are DFA's so must use both symbols

* let us take input string it is correct or not.

input string 1011

$q_0\ q_2\ q_1\ q_2\ q_2$

0 0 1 0 0 → w.k.T in Moore Machine input string size n, output size n+1.

No need of this zero

→ input string 1011

0100

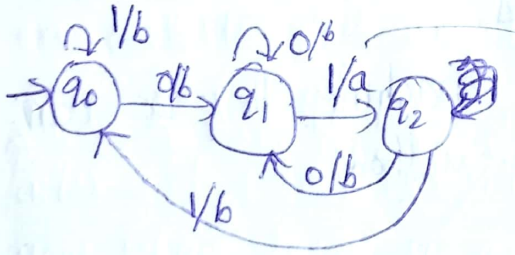
Meelay Machine example

→ construct a melay machine that prints 'a' whenever the sequence '01' is encountered in any input string.

$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

* Melay Machine constructs DFA only. but in DFA have final states in melay we don't have final states

* The sequence '01' it means we require 3 states



* So this is the melay machine for this problem

* Now we have to check wheather our melay machine is correct or not. by taking an input string

input string 1011101

a a

$q_0\ q_0\ q_1\ q_2\ q_0\ q_0\ q_1\ q_2$

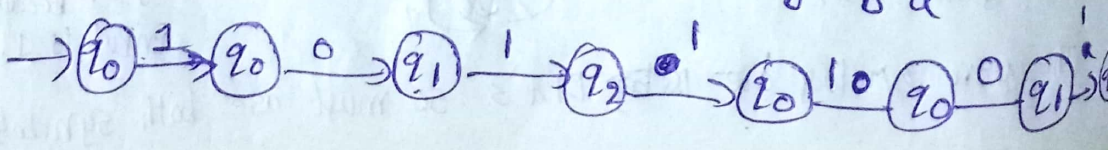
b b a

b b a b b b a

1 0 1 1 1 0 1

$q_0\ q_0\ q_1\ q_2\ q_0\ q_1\ q_2$

b b a



* this way 2 a's are named.

* in this way we solve the problem.

→ Design a melay machine inorder to find 1's complement

* 1's complement means converting 0 to 1's. 1's to zeros.

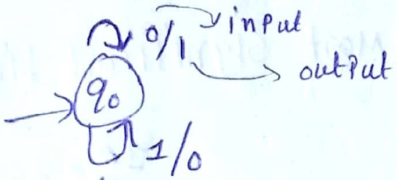
Ex: Input string 1001
↓↓↓↓
0110 → output string

* because in binary input string and output string are 0's & 1's

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

* input is zero output 1 . input 1 output 0

* so only 1 state is enough here.

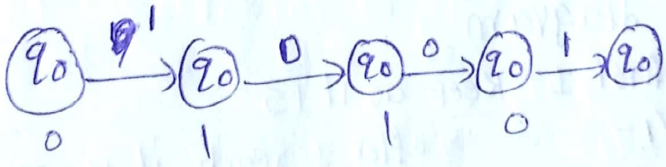


* so this is the melay machine for this problem.

* Now check wheather the diagram is crct or not.

input string → 1001.

1 1 0
0



so this way our melay machine is corred.

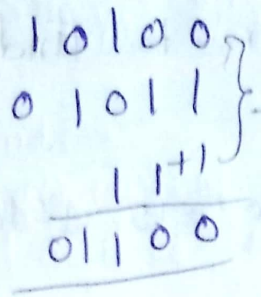
→ Design a melay machine inorder to find 2's comp

2's complement means 1's compliment +1

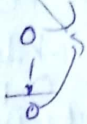
* suppose 0100100
1011011
+1
1011100

* let us take another example

we have a string like this 10100



* if you observe the logic input 0 → output is 0. if the input is 1 → output is 1.

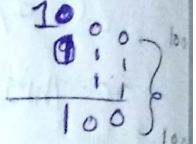


* so that means here instead of Reading data more MSB to the LSB (least significant bit)

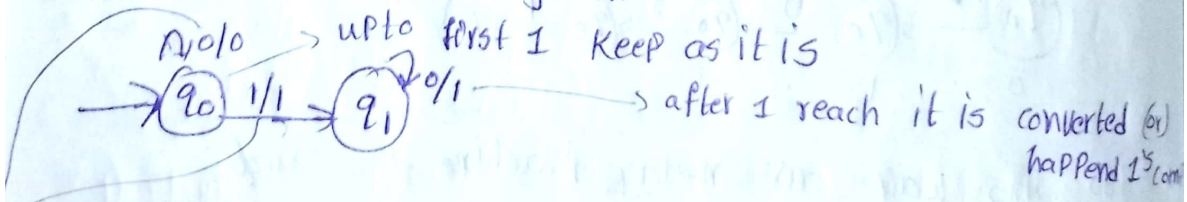
* we have to read LSB to MSB (Most significant bit)

MSB ← 0 1 00 1 00 → LSB

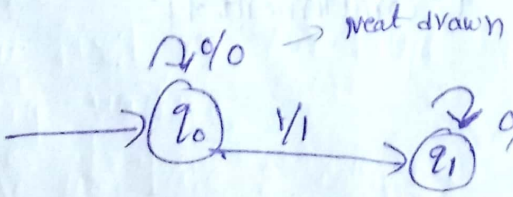
* if you observe is least significant Bit (100) leaf wise reach 1 No change. after the 1 They was converted This was Logic here.



* up to 1 Keep the bits as it is
* after reach 1 complement the bits.
* Now let us draw diagram



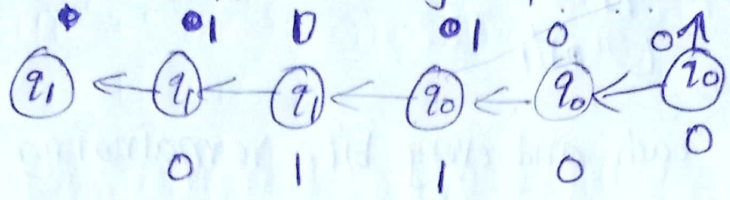
upto first 1 keep as it is
mean '0' as input '0' as output.



* this is the melay machine for this problem. Let us check wheather it is crct or not.

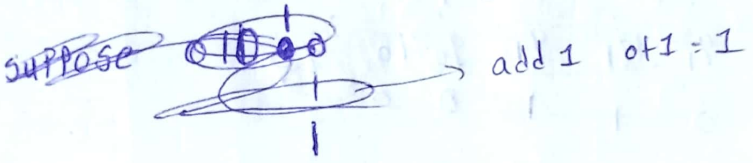
here data Read from LSB to MSB.

Take an input string 1 0 1 0 0



* so in this way we find the 2's complement of given binary number.

→ Design a mealy Machine inorder to find Increment given binary number by 1.



suppose $0\,1\,0\,\overline{1\,1}$

$1\,1\,1$

$0\,1\,1\,0\,0$

* Here 1 is changed to 0. 1 → 0. 0 → 1
* so Here also Read the data from LSB to MSB.
* upto to first 0 Keep the bits as it is
* Another example

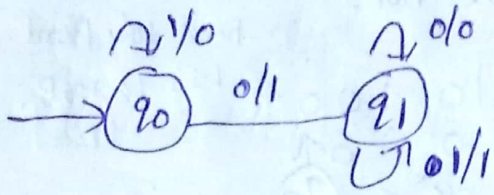
$0\,1\,0\,0\,1\,0\,0$

$\quad\quad\quad\quad\quad\quad\;\; 1$

$0\,1\,0\,0\,1\,0\,1$

* Here what is the logic upto first '0'. ~~Black~~ invert each and every bit.
* first the 0 become 1. remaining bits are placed as same. (as it is)

Now draw the Melay machine for this.



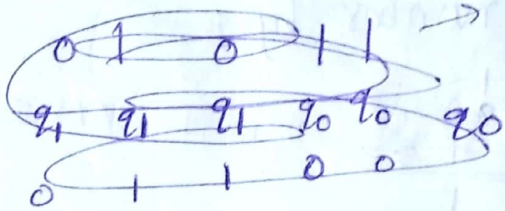
First zero

upto compliment each and every bit. remaining are placed as it is

* After the first zero '0'. we remain the bits as it is.

* Now check whether it is crct or not.

* input string → Read LSB to MSB



0 1 0 1 1

$q_1 \leftarrow q_1 \leftarrow q_1 \leftarrow q_0 \leftarrow q_0 \leftarrow q_0$

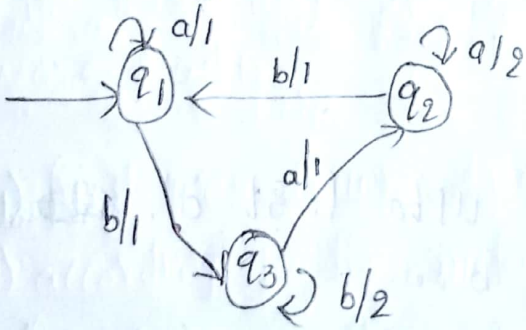
0 1 1 0 0

so it is crct.

* In this way we solve the problem.

conversion of melay machine to more machine

* The melay machine we have to convert melay to more



Now first we construct transmission table to this melay machine.

* Here we have 2 columns
  1) current state
  2) Next state

current state	a		b	
	next state	o/p	next state	o/p
→ q_1	q_1	1	q_3	1
q_2	q_2	2	q_1	1
q_3	q_2	1	q_3	2

→ Melay transition table

* The output of q_1 in a is 1 . q_1 is b is 1
* so we have to construct the transition table for more machine
* in more machine the output depends only on prsnt state
* so the output of each value is one value
* if it produces multiple value we split that state into 2 states depends upon the output
* The q_1 is one o/p so $q_1 \rightarrow 1$
* q_1 produces 2 o/ps $q_2 \begin{cases} q_{2,1} \\ q_{2,2} \end{cases}$ → q_2 'a' output 1, $q_{2\,2}$ 'a' output 2
* q_3 also splitted into 2 states

$q_3 \begin{cases} q_{3\,1} \rightarrow q_3$ 'b' output is 1 |
$q_{3\,2} \rightarrow q_3$ 'b' output is 2 |

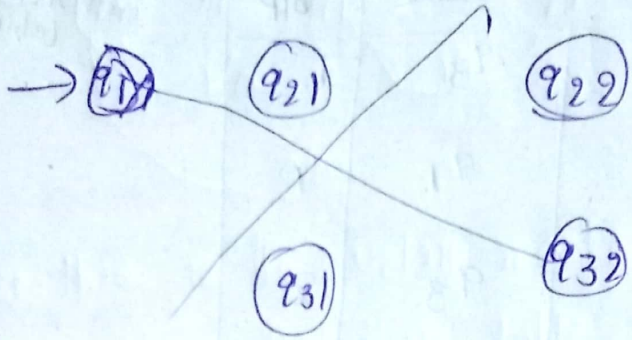
* let us construct More machine transition table

More transition table → based on Melay we construct it

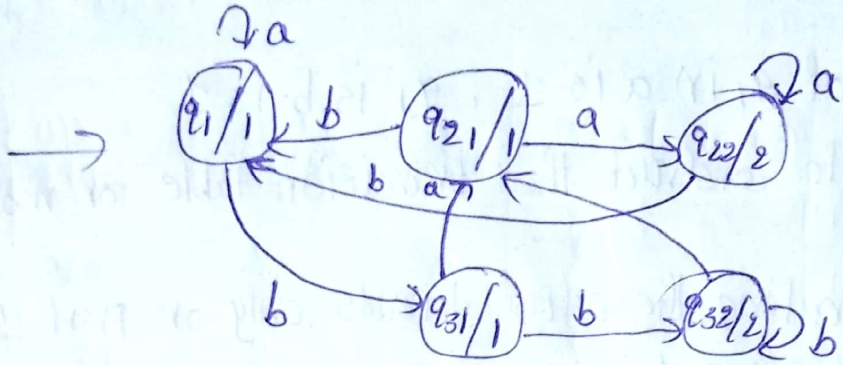
current state	nextstate		output
	a	b	
q_1	q_1	q_3	1
q_{21}	q_{22}	q_{18}	1
$q_{2\,2}$	q_{22}	q_1	2
$q_{3\,1}$	q_{21}	q_{32}	1
$q_{3\,2}$	q_{21}	q_{32}	2

left value output

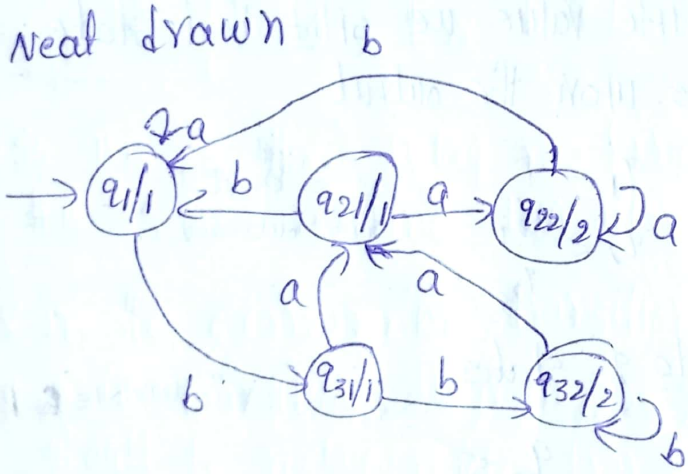
* let us draw transition diagram based on table



because More machine o/p written on state



Neal drawn



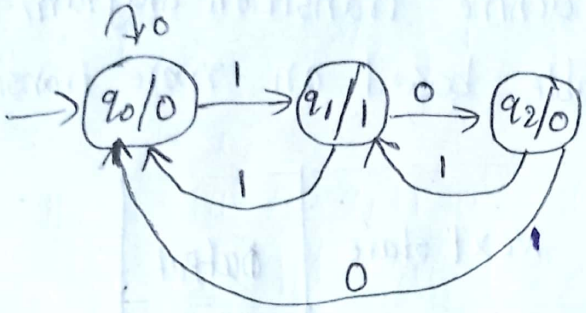
* in this way we convert Melay machine to Move machine.

1) first we drawn transition table for melay machine
2) according to the melay machine transition table draw the Move machine transition table.
3) according to Move machine transition table draw the transition diagram.

conversion of moore machine to melay machine

* in the examination transition table may be given or transition diagramme may be given

* so this is transition diagrame.



* Now let us construct transition table for this diagram

* because in the examination a transition table may be given so directly we need to convert that more machine transition table to melay machine transition table

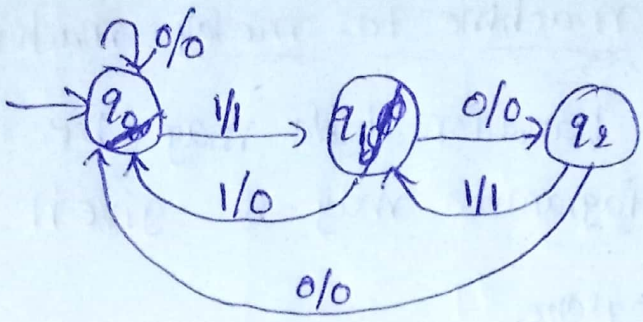
output depends only on present state
all input same output
more machine transition table

* let se the transition table

current state	Next state		output
	0	1	
q_0	q_0	q_1	0
q_1	q_0	q_2	1
q_2	q_0	q_1	0

* Now let us construct the melay machine based on transition diagram

* after that we need to construct transition table of melay machine based on more machine table.



* So this is the Melay machine transition diagram
* Now obtain transition table based on more transi

Current state	Next state	o/p	Next state	Output
q0	q0	0	q1	1
q1	q2	0	q0	0
q2	q0	0	q1	1

* So this is the Melay machine transition table for this more machine transition table
* in the examination transition diagram may be given
* so if transition diagram is given Then the output should be melay miche transition diagram.
* wheras transition table is given. The output should be melay machine transition table.
* In this way we can solve This joroblem.