Theory Of Computation

UNIT-1

Traditional Area of theory of computation: ① Automata ② computability ③ complexity

→ What is TOC?

In theoretical computer science, the theory of computation is the branch that deals with whether and how efficiently problems can be solved on model of computation using an algorithm. eg. of Automata: automatic machine / automatic machine tool / packaging machine

* It's study of (abstract machine) & the computation problem that can be solved using these machine.

Theory of computation (Area of TOC)

machine that is automated

Automata theory (mathematical "represent")
It deals with the defination & properties of various mathematical model of computer

Computability (capability) theory
It deals with what can and cannot be computed by model

Complexity (hardness) theory
It groups the computable problem based on their hardness.

* The motivation of developing the automata theory was to develop methods to describe and analysis the dynamic behaviour of discrete systems

It enables to understand how machines compute the functions and solve problems.

Eg- Set of binary strings that ends with 0.
Set of binary strings that represent a legal java code - Yes.

TOC tells -
- What we can compute
- What we cannot compute
- What is model of computation
- How we can compute.

TOC is also called as "Automata Theory".

Symbol are entity or individual object which can be any letter, alphabet or any picture

Automata theory :-

(1) Symbols - smallest building block, which can be any alphabet, letter or any picture. a, b, c, 0, 1 ....

(2) Alphabets (Σ) - It is set of symbols, which are always finite. (finite set of symbols) denote by Σ

Σ $\Sigma = \{1\}$ → Unary
$\Sigma = \{0,1\}$ → Binary no's
$\Sigma = \{0-9\}$ → decimal digit $\Sigma = \{0,1,\ldots 5\}$
$\Sigma = \{A,B,C,D\}$

(3) String - Finite sequence of symbols taken from a particular alphabet. Denoted by w and length of string w is denoted as $|w|$. No. of symbols in a string

$|w|$ Eg. $w = 010$, $|w| = 3$, $\Sigma = \{0,1\}$ 01101 $\Sigma = \{a,b\}$ abba

• A string with zero occurance of symbol is known as an empty string. represented as ε

Eg - If $|w| = 2$ then no. of strings = 4

For alphabet $\{a,b\}$ with length n, number of strings can be generated $= 2^n$.

(4) Language - Set of string that are always defined on alphabet.
A language is a subset of Σ^*
A language formed over Σ^n can be finite or infinite.

Eg - L_4 - neither eng. nor hindi

L_2 = {Set of all strings of length 2}
= {aa, ab, ba, bb} - finite language.

A language is a collection of appropriate string. A language which is formed over Σ can be finite or infinite.

L_2 = {Set of all strings which starts with 'a'} (2)
= {a, aa, ab, aba, aba, aaa, abb, ---} → Infinite language.

(5) Powers of alphabet (Σ) –
Say Σ = {a, b} then
Σ^0 = Set of all strings over Σ of length 0.
{ε}
Σ^1 = Set of all strings over Σ of length 1.
{a, b}
Σ^2 = Set of all strings over Σ of length 2.
{aa, ab, ba, bb}

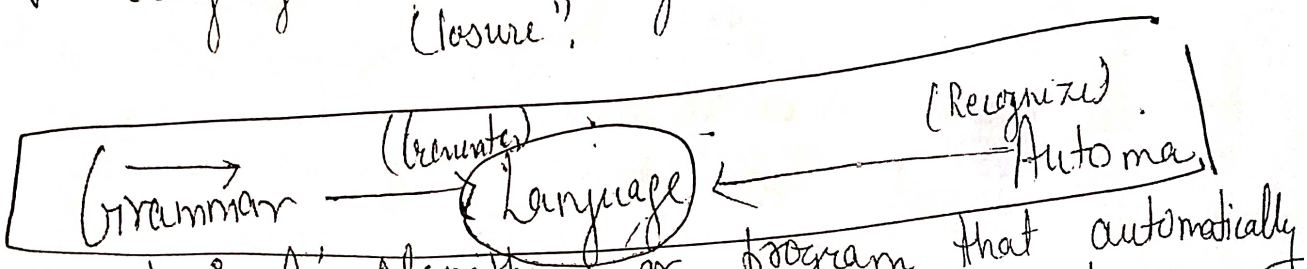
(6) Length of string – Counting the frequency / occurance of symbol in a string.

(7) Kleene Closure (Σ^*) – It contains all possible strings of any length defined over an alphabet including null string.

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$$

Eg – {00, 01} → Subset of {0, 1}.

"A language is always a subset of Kleen Closure".



Automata : An algorithm or program that automatically recognize if particular string belongs to the language or not, by checking the grammar of the string.

The automata consist of
1. States (represented by circles)
2. Transitions (represented by arrows) (jump)

Uses –
Compiler design and parsing

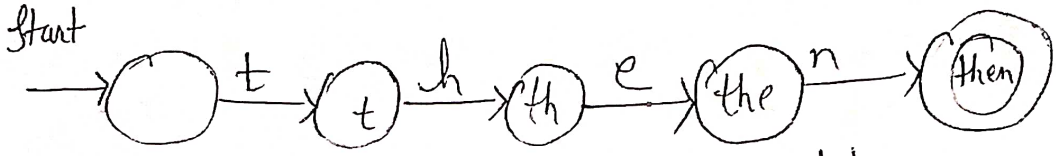


Fig. Finite automata model

As the automata sees a symbol of input, it makes a transition or jumps to another state, according to its transition function (which takes the current state & recent symbol as its inputs).

States – is an instantaneous description of that system which gives all relevant information.

Transition – are changes of states that can occur spontaneously or in response to inputs to the states.

FINITE AUTOMATA :-

Finite automata is the simplest machine to recognize patterns.
It is a machine that will answer Yes/No.
A system containing only a finite number of states and transitions among them is called as finite-state transition system or finite system.

A finite automata consist of the following:

$\{Q, \Sigma, q, F, \delta\}$

where —

- Q = Finite set of states
- Σ = Set of Input symbols.
- q = Initial state
- F = Set of final states
- δ = transition function.

* FA is collection of 5 tuple
* FA are used to recognize pattern
* takes string of symbol as input and changes its state accordingly. when desire symbol is found then transition occurs
* At the time of transition the automata can either move to next state
* FA have two states, Accept state or Reject state when the i/p string is proc-d successfully and the automa reached it final St then will accept

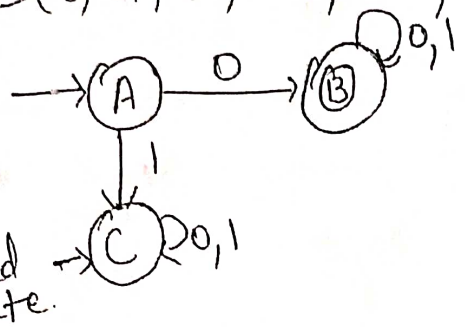
Finite Automata

- Deterministic FA without O/P capability
  - DFA — Deterministic finite automata
  - NFA — Non-deterministic finite automata
  - ε-NFA — Epsilon-NFA
- FA with O/P capability
  - Moore machine
  - Mealey machine

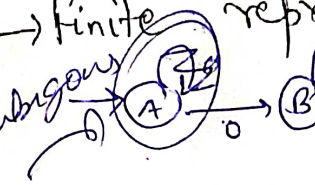
Examples of finite automata :—

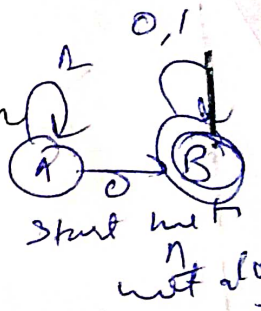
(1) Set of all binary strings starting with zero.

$\Sigma = \{0, 01, 00, 001, 010, 011 \ldots\}$ — Infinite representation



→ Finite representation





2. * Any system as a m/c answered in a finite rep. as Yes/No but not in infinite representation

"001"

$A \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{1} B$

So, when from initial we reach final state assuming all states then m/c says Yes otherwise No.

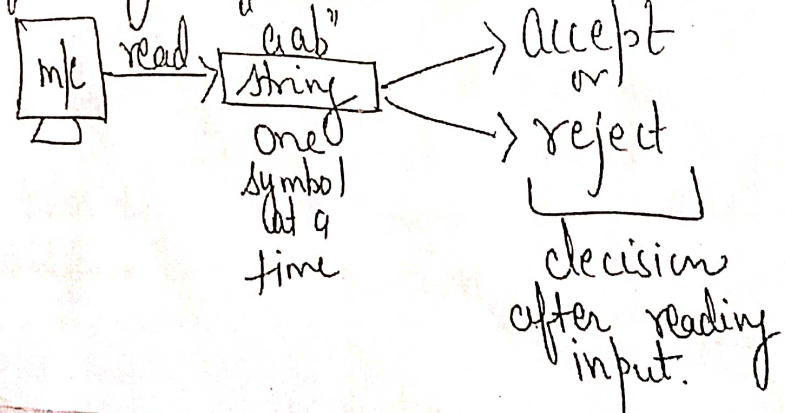
"101"

Reject as we didn't reach the final state

Finite automata without O/P capability

(1) Deterministic finite automata-

Informally,



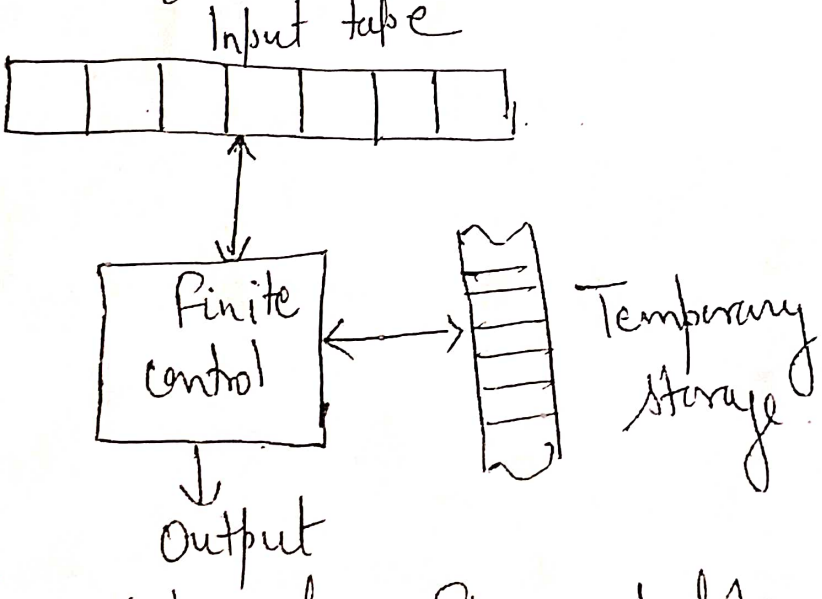
So, A m/c for which deterministic code can be formulated & if there is only one unique way to formulate the code then m/c is called as DFA.

DFA consist of 3 parts:—

1) Tape —to hold input string.
   Divided into finite no of cells.
   Each cell holds a symbol from Σ

2) Tape head — for reading symbol from tape

3) Control —
   - Finite no. of states that m/c allows
   - A current state, initially a start state
   - State transition function for changing current state.



DFA consist of five tuples as—

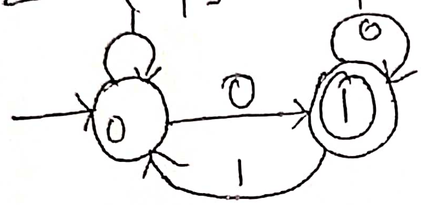
$$\{Q, \Sigma, q, F, \delta\}$$

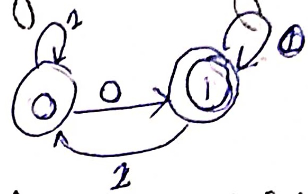
where

Q = set of all states
Σ = Input Symbols
q = Initial states
F = final states
δ = transition functi

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

Eg.- $\Sigma = \{0,1\}$ accepts all strings ending with 0.

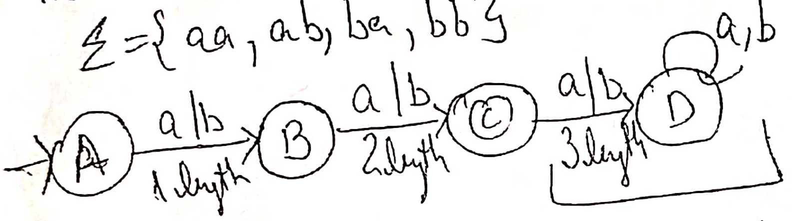




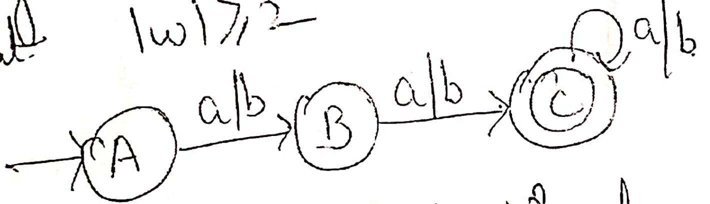
*Note— There can be many possible DFAs for a pattern but minimum no. of states is preferred.

Q- Design a DFA over the alphabet $\Sigma = \{a, b\}$ that will accept all strings w such that $|w| = 2$

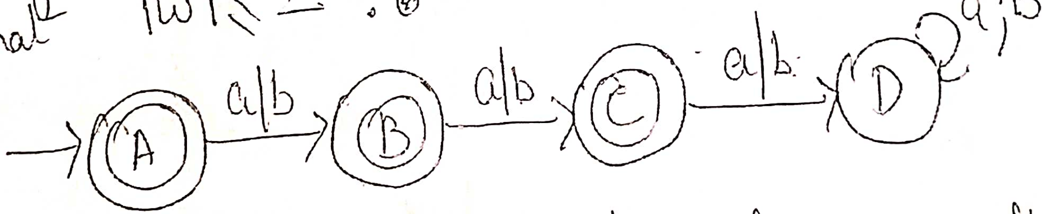
$\Sigma = \{aa, ab, ba, bb\}$



Q. Design a DFA $\Sigma = \{a,b\}$ for all strings such that $|w| \geq 2$



Q. Design a DFA $\Sigma = \{a,b\}$ for all strings such that $|w| \leq 2$.



* If initial state is treated as the final state then it means ϵ move is acceptable.

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

automata model that represents acceptance X rejection on language

* If A is set of all strings that machine M accepts we say that A is the language of machine M,

$L(M) = A$ or M recognizes A or M accepts A

Language for finite automata M_1,

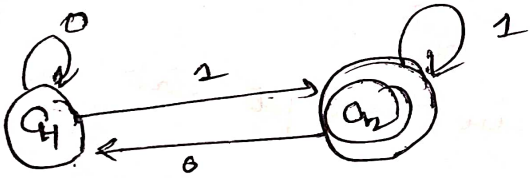
$L(M_1) = A = \{ w \mid w$ contains at least one 1 $\}$

eg $M_2 = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$

$\delta =$

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_2) = A = \{ w \mid w$ ends in a 1 $\}$

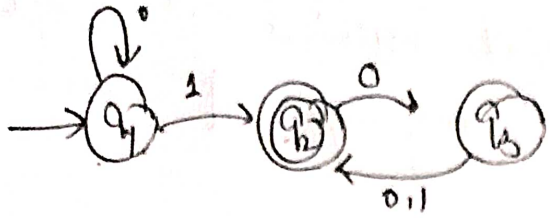


~~1, 11, 01, 11~~

1, 01, 11, 011

001, 111, 101

- Language?
- Language is called Regular Language if some finite automata recognize it



(finite automata with 3 states)

q_1 : Start state

q_2 : accept state / final state

q_3 : transition state / intermediate state

Finite automata

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

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

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

(1101)

$q_0 = q_1$

$F = q_2$

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Q

for input string 1101 into machine M_1

① Start in state q_1

② Read 1, follow transition from q_1 to q_2

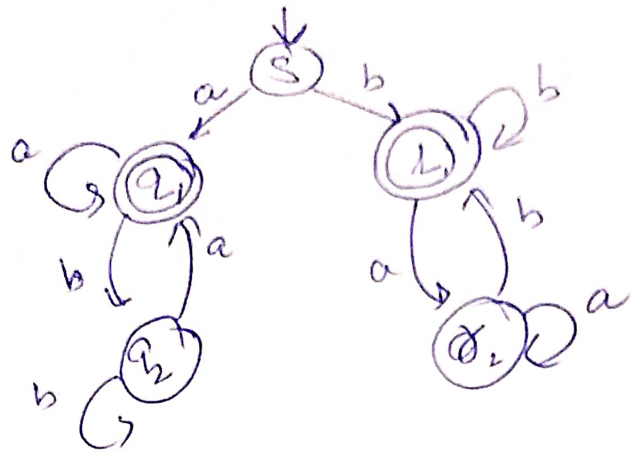
③ Read 1, follow transition from q_2 to q_2

④ Read 0, follow transition q_2 to q_3

⑤ Read 1, follow transition q_3 to q_2

⑥ Accepted, because M_1 is in accept state q_2 at the end of input.

9. Five state machine M_2



$Q = \{ S, q_1, q_2, f_1, f_2 \}$

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

$q = S$

$F = \{ q_1, f_1 \}$

$\delta =$

	a	b
S	q_1	f_1
q_1	q_1	q_2
q_2	q_1	q_2
f_1	f_2	f_1
f_2	f_2	f_1

It accept the string like

{a, aa, aaa, aaba ... }

b, bb, bbb, bbab ... }

$L = \{ w | w$ have same start & end symbol $\}$

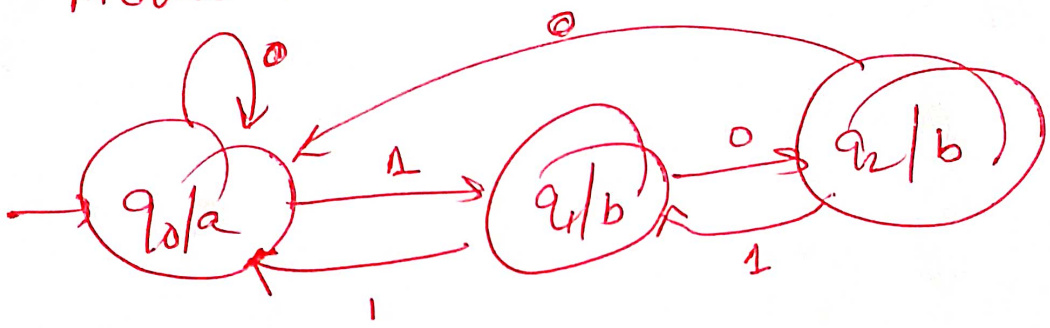
Rejected string = { ab, ba, bba, aab.

Moore Machine

$M_1 = \{ Q, \Sigma, q_0, \delta, \Delta, \lambda \}$

- Q → finite set of state
- Σ → input symbol
- q_0 → start state
- δ → transition function
- Δ → output symbol
- λ → output function

Moore machine — FA with output



1: $Q \rightarrow \Delta$

$q_0 \rightarrow a$

$q_1 \rightarrow b$

$q_2 \rightarrow b$

δ: $Q \times \Sigma \rightarrow Q$.

$q_0 \quad \times (0,1)$

q_1

q_2

$q_0,0 \rightarrow q_0$

$q_0 1 \rightarrow q_1$

$q_1 0 \rightarrow q_2$

$q_1 1 \rightarrow q_0$

$q_2 0 \rightarrow q_0$

$q_2 1 \rightarrow q_1$

If the length of input is n then output will be $n+1$

eg: 00110 — O/P

(5)

initially at q_0 then, output

q_0 q_0 q_1 q_0 q_0

0 0 0 1 1 0

a a a b a a

(6)

5 → 6

Current	Next State		Output
	0	1	
q_0	q_0	q_1	a
q_1	q_2	q_0	b
q_2	q_0	q_1	b

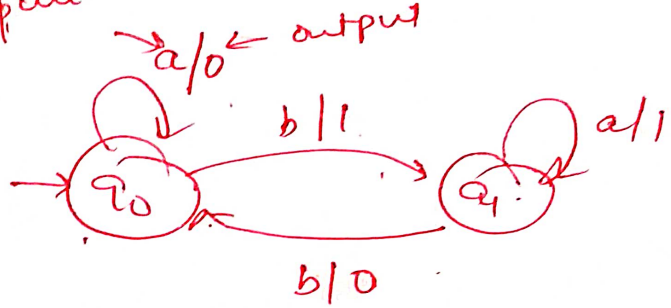
Melay Machine

$M_1 \{ Q, \Sigma, \Delta, \delta, \lambda, q_0 \}$ → $Q \times \Sigma \to \Delta$

(0,1)

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

input



if n size input then n size output

$n \to n$

Initial state	Next state			
	Input a		Input b	
	Input	output	Input	output
q_0	q_0	0	q_1	1
q_1	q_1	1	q_0	0

λ : $Q \times \Sigma$

$q_0 \times (a,b) \to$ $q_0, a \to q_0$; $q_0, b \to q_1$; $q_1, a \to q_1$; $q_1, b \to q_0$

q_1

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

$q_0 \times (a,b) =$ $q_0 a \to 0$; $q_0 b \to 1$; $q_1 a \to 1$; $q_1 b \to 0$

q_1

	a	b
→ q_0	$\{q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_3\}$
q_2	$\{q_3\}$	$\{q_0\}$
q_3		
q_4	q_2	q_4

Minimization of DFA

Step 1 : Remove all state that are unreachable from the initial state via any set of transition of DFA

Step 2 : Draw the transition table for all pair of states.

Step 3 : Now split the transition-table into two tables T1 & T2. T1 contains all final states & T2 contains non-final states.

Step 4 Find similar row from T1 such that

$\delta(q,a) = p$

$\delta(r,a) = p$

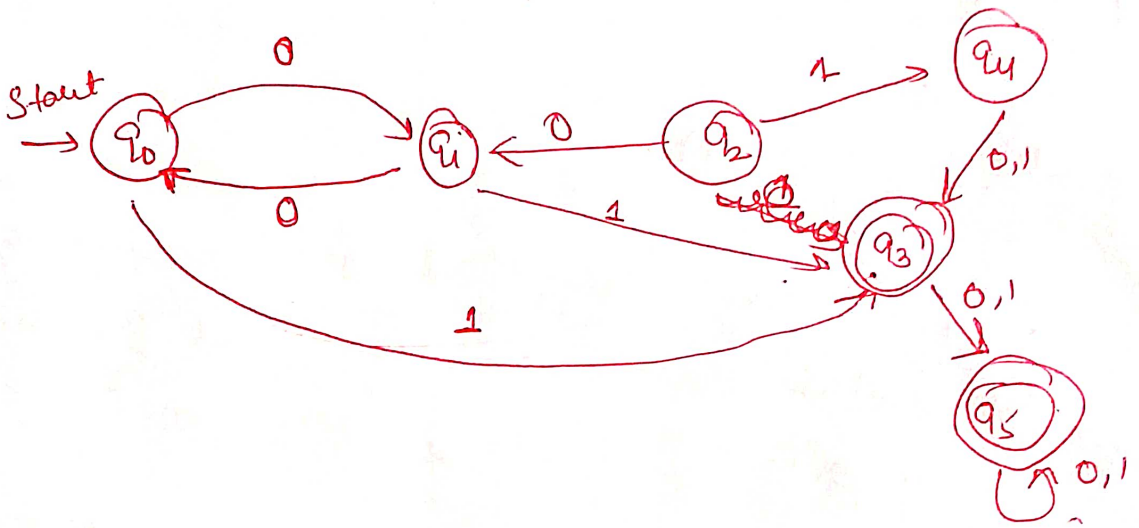
that means find two states which have same value of a & b & remove one.

Step 5 : Repeat step 3 until we find no similar row available in transition table T1

Step 6 Repeat step 3 & step 4 for table T2 also

Step 7 Now combine the reduced T1 & T2 tables. The combined transition table is the transition table of minimized DFA.

eg



Sol (1) In the given DFA q_2 & q_4 are the unreachable state so remove them.

(2) Draw transition table for the rest of the state.

State	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
(93)	q_5	q_5
(95)	q_5	q_5

(3) Now divide row of transition table into two set as

3.1) Contain non-final states

State	0	1
q_0	q_1	q_3
q_1	q_0	q_3

} set 1

3.2) Contain final states

State	0	1
q_3	q_5	q_5
q_5	q_5	q_5

} set 2

(4) set 1 has no similar rows so set 1 will be same

(5) set 2, row 1 & row 2 are similar since q_3 & q_5 transit to same state on 0 & 1. So skip q_5 & then replace q_5 by q_3 in the rest.

State	0	1
q_3	q_3	q_3

(6) Now combine set 1 & set 2 as:

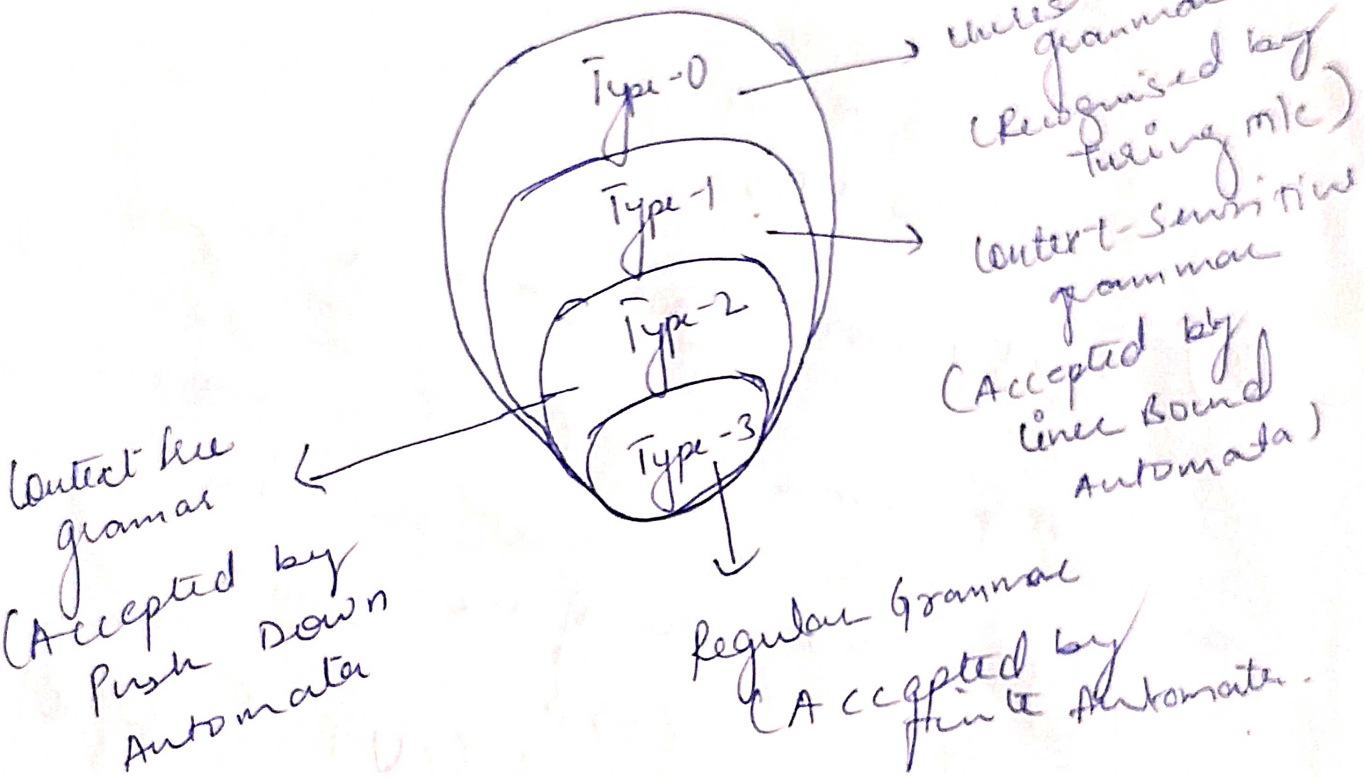
minized DFA.

State	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
(f 3)	q_3	q_3

Chomsky Classification

Grammar — A grammar consist of finite set of rules of production that specifies syntax of the language.

- A grammar imposes restriction on the sentence of a language.



Type 0 : Unrestricted Grammar

A grammar $G = (V, T, P, S)$ is called unrestricted if all the productions are of the form

$$\alpha \rightarrow \beta$$

where α is in $(V \cup T)^+$ i.e it can be a

string of terminal or non-terminals with language
& at least one non-terminal
β is $(V \cup T)^+$ - a string of terminal or
+ non-terminals).

eg: $S \to AaB$
$A \to S$
$Bc \to acB$

eg: $S \to ACaB$
$Bc \to acB$
$CB \to DB$
$aD \to Db$

(2) Type 1 grammar - Context sensitive Grammar.

A grammar $G = (V, T, P, S)$ is said to be context sensitive if all production are of the form

$\alpha \to \beta$

where $\alpha, \beta \in (V \cup T)^+$ & $|\alpha| \leq |$

Note: except for $S \to \epsilon$ & start symb
does not appear on the right hand
side of any production

for eg $S \to AB$
$AB \to abc$
$B \to b$.

free grammar.

A grammar $G = (V, T, P, S)$ is said to be context free ~~language~~ if all production are of form

$$\alpha \to \beta$$

where $\alpha \in V$ & $\beta \in (V \cup T)^*$

This type of grammar generate context free language that are recognized by Non Deterministic PDA.

Eg. $S \to AB$

$A \Rightarrow \lambda$

$B \to y$

Type 3 – Regular Grammar

A grammar $G = (V, T, P, S)$ is said to be context free if all production are of the form

$$\alpha \to \beta$$

where $\alpha \in V$ & $\beta \in VT^* + T^*$

or

$\beta \in T^* V + T^*$

(i.e grammar must have a single terminal on left hand side and on right hand side must have a single non terminal followed by a terminal.

eg. $S \to aS \mid bS \mid \epsilon$

or

$S \to \to ACaB$

$Bc \to aCB$

$CB \to DB$

$aD \to Db$

Regular Expressions

In arithmetic operation + & × are used to build expression such as (5+3)*4

By Regular Expression can be build using regular operation (0∪1)0*

Value of expression 32

In this case language consist of strings starting with 0 or 1 followed by no. of 0s.

(0∪1) means ({0} ∪ {1}) → the value of this part is the language {0,1}

& {0}* language consisting of all strings containing any no. of 0s

Formal Defination

Say that R is a regular expression if R is

1) a for some a in alphabet Σ
2) ε
3) ϕ (→ empty)
4) $(R_1 \cup R_2)$ where R_1 & R_2 are reg

5) $(R_1 \circ R_2)$ where R_1 & R_2 are regular expression.

6) (R_1^*) where R_1 is regular expression.

ε ⟶ represents language containing a single string

ϕ → empty string — represents the language that doesn't contains any string.

Equivalance with finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

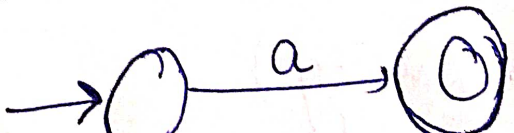
Lemma

If a language is described by a regular expression then it is regular.

Proof idea Say R is regular expression some language A.

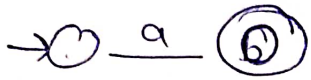
1) $R = a$ for some $a \in \Sigma$.

Then $L(R) = \{a\}$

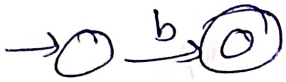


$(a \cup b)^* aba$

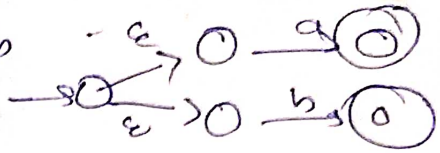
① a :



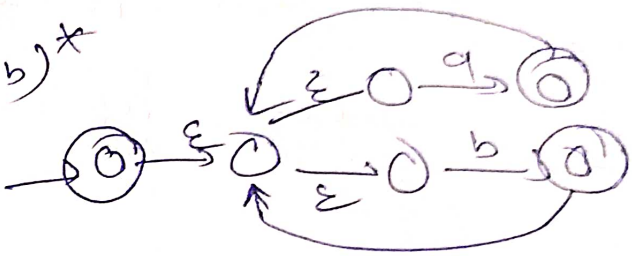
b : ②



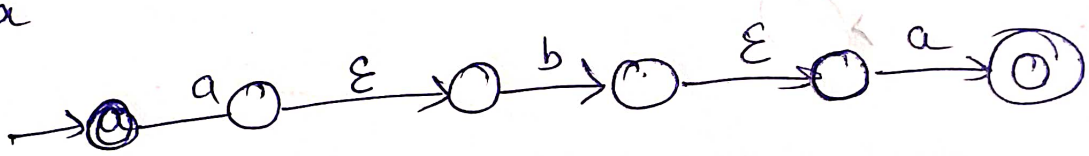
② $a \cup b$



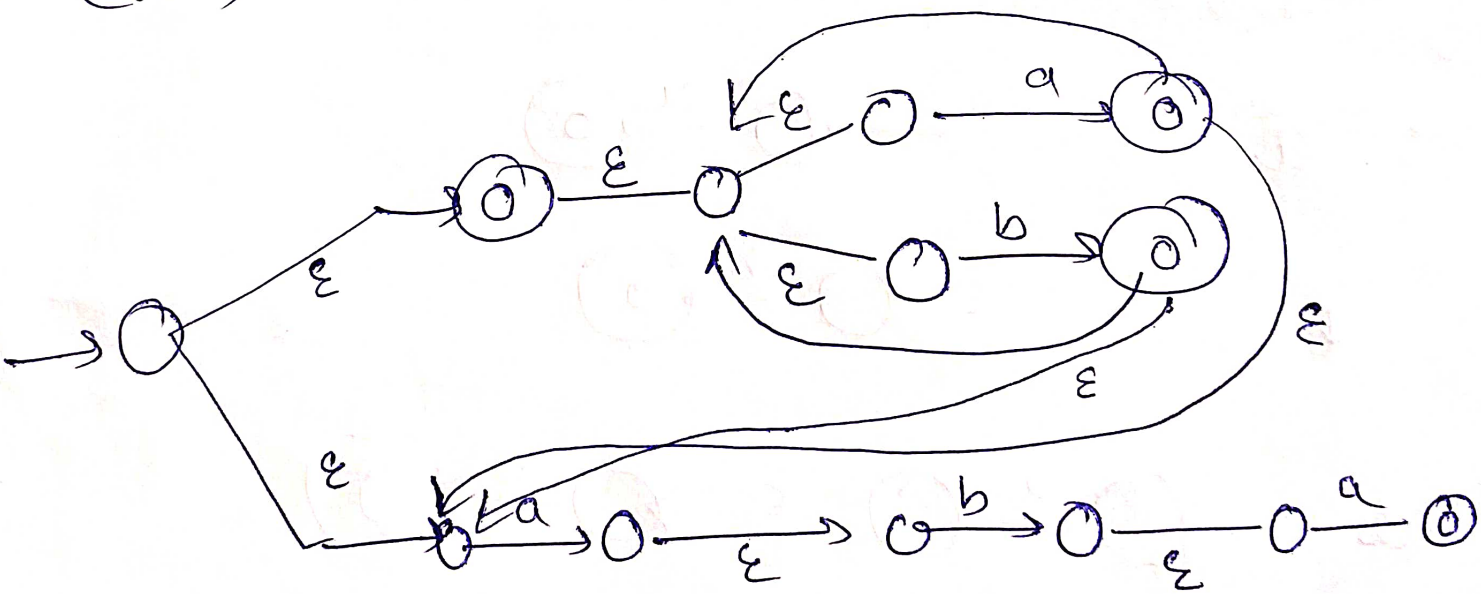
④ $(a \cup b)^*$



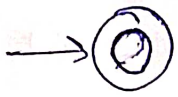
⑤ a b a



⑥ $(a \cup b)^* aba$



(2) $R = \varepsilon$. $L(R) = \{\varepsilon\}$



$N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$

where $\delta(q, b) = \phi$ for any q & b

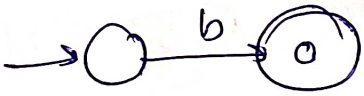
(3) $R = \phi$. Then $L(R) = \phi$

eg Convert the regular expression $(ab \cup a)^*$ to NFA.

Sol

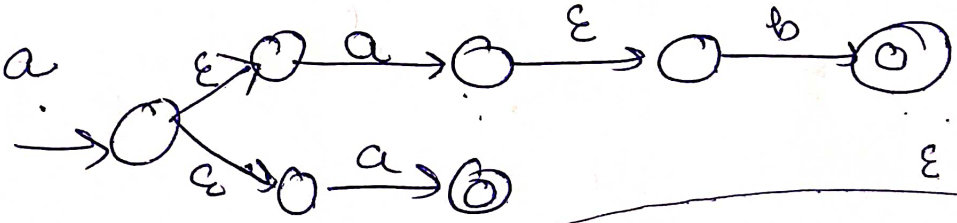
a

b

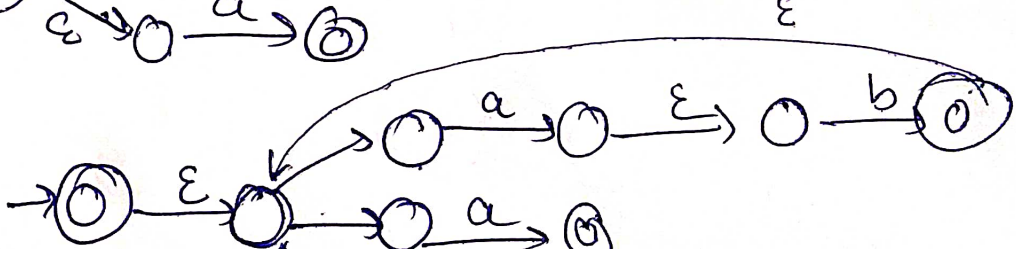


ab

$ab \cup a$



$(ab \cup a)^*$



DFA for even a & even b

Q. let $L = \{w \mid w$ has even no. of a's & even no of b's$\}$

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

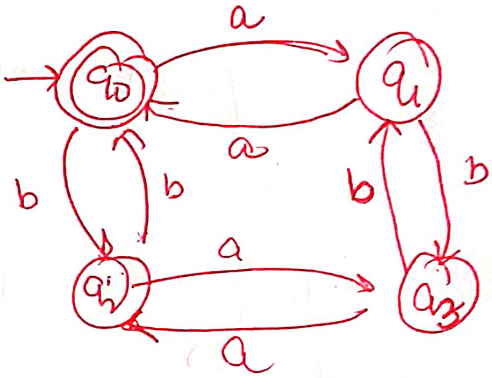
$L = \{\varepsilon, aa, bb, abab, aabb, bbaa, baba\}$

Zero no of a & b

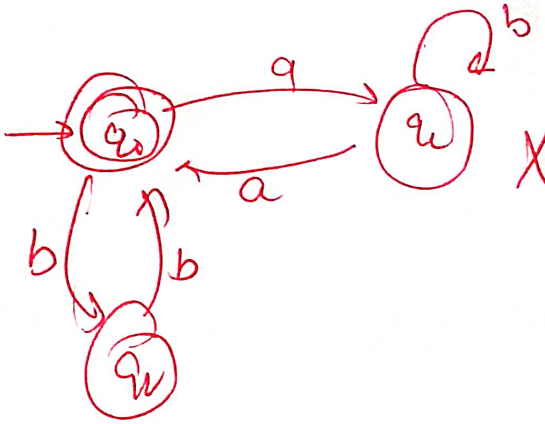
neither a is accepted nor b is also

Remain at 1st state

both me pahuch gaye first & last state are same.

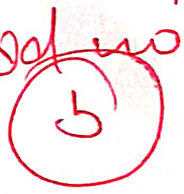


✗ abba
baba
baab



✗ Not acceptable as single b is also accepted

aba = odd no of b ✓



q_0 = final for both even

q_1 = final · odd a & even b

q_2 = even a & odd b

q_3 = odd a & odd b.

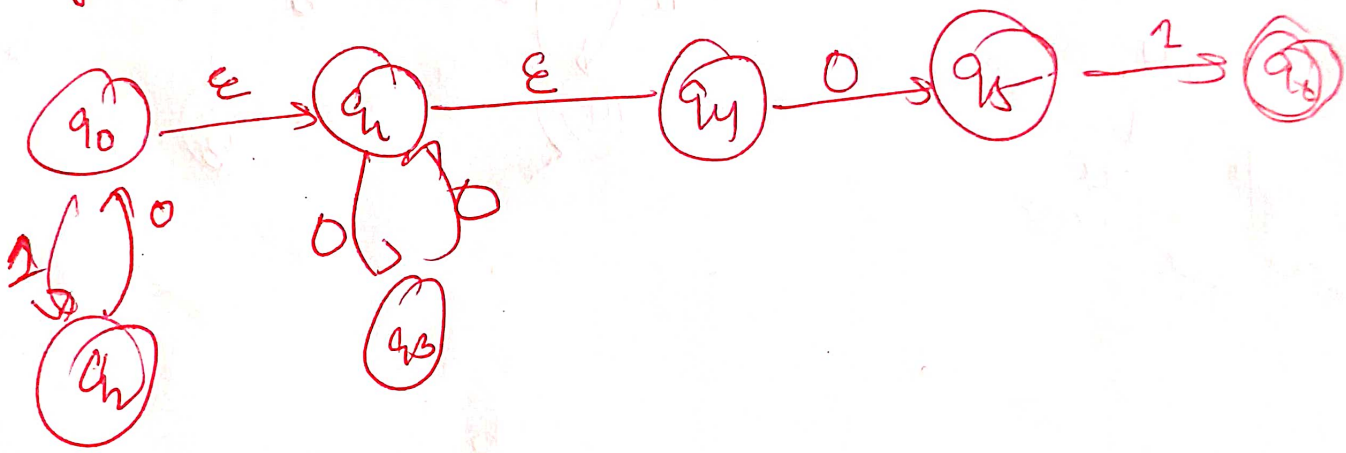
ε- NFA → eliminating

1) find all edges starting from S_2

2) Duplicate all edges to S_1 without changing edge labels

3) if S_1 is initial state, make S_2 initial

4) if S_2 is final state make S_1 final



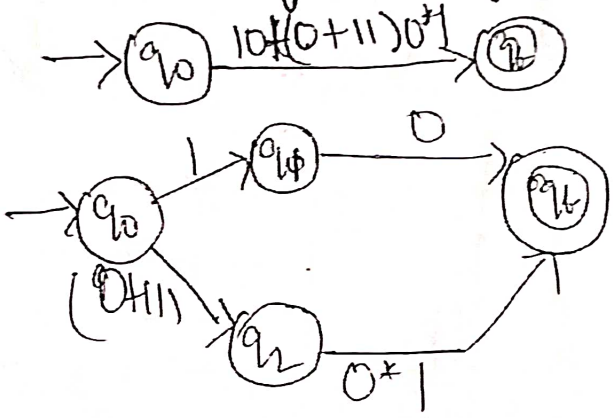
→ DIRECT METHOD TO CONVERT RE into FA:-

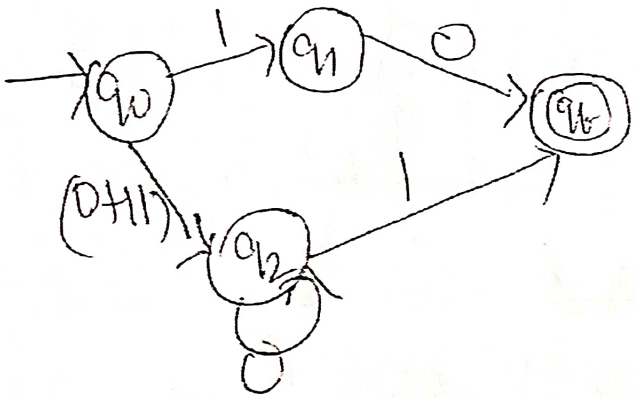
Step 1:- Design a transition diagram for give RE, using NFA with ε moves.

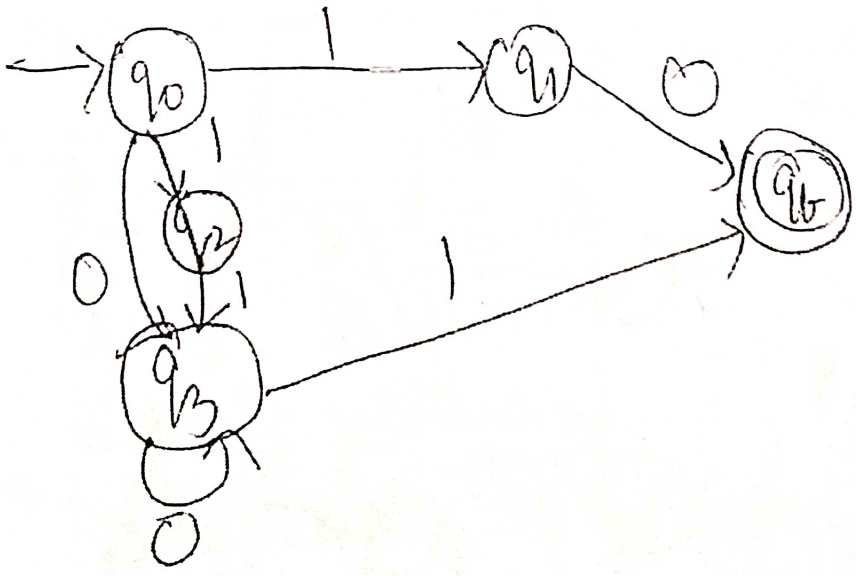
Step 2:- Convert this NFA with ε to NFA without ε

Step 3:- Convert the obtained NFA to equivalent DFA.

Q. Design a FA from given RE $10+(0+11)0^*1$







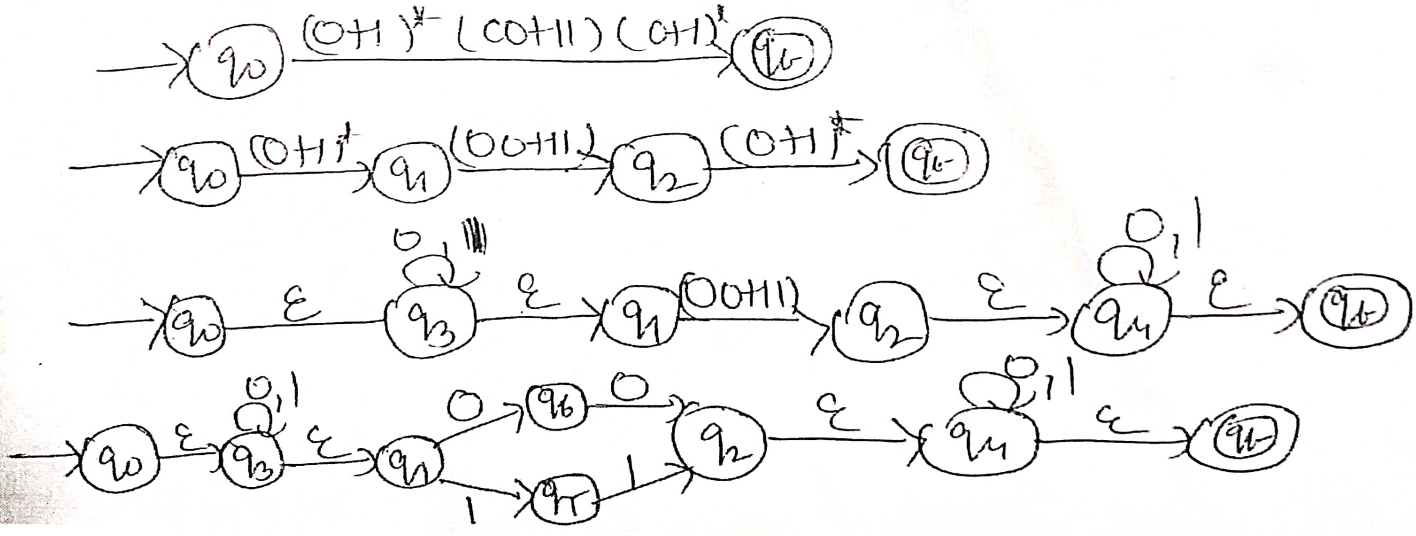
Transition Table for NFA

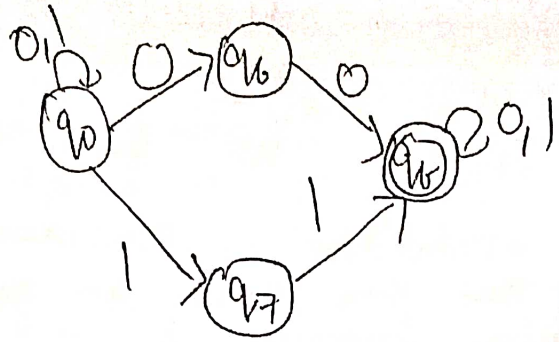
State \ Input	0	1
q_0	q_3	$\{q_1, q_2\}$
q_1	q_6	ϕ
q_2	ϕ	q_3
q_3	q_3	q_6
q_6	ϕ	ϕ

The equivalent DFA will be

State \ Input	0	1
$[q_0]$	$[q_3]$	$[q_1 q_2]$
$[q_1]$	$[q_6]$	ϕ
$[q_2]$	$[q_3]$	$[q_3]$
$[q_3]$	$[q_3]$	$[q_6]$
$[q_1 q_2]$	$[q_6]$	$[q_3]$
* $[q_6]$	ϕ	ϕ

- Construct FA equivalent to RE $(0+1)^*(00+11)(0+1)^*$





q \ Σ	0	1
→q_0	$q_6 q_7$	$q_0 q_7$
q_6	q_6	—
q_7	—	q_6
*q_6	q_6	q_6

For DFA

q \ Σ	0	1
→$[q_0]$	$[q_0 q_6]$	$[q_0 q_7]$
$[q_0 q_6]$	$[q_0 q_6 q_6]$	$[q_0 q_7]$
$[q_0 q_7]$	$[q_0 q_6]$	$[q_0 q_7 q_6]$
$[q_0 q_6 q_6]$	$[q_0 q_6 q_6]$	$[q_0 q_7 q_6]$
$[q_0 q_7 q_6]$	$[q_0 q_6 q_6]$	$[q_0 q_7 q_6]$

we can reduce this to one state as providing same state on input 00 & 1.

So final Transition table.

q \ Σ	0	1
→$[q_0]$	$[q_0 q_6]$	$[q_0 q_7]$
$[q_0 q_6]$	$[q_0 q_6 q_6]$	$[q_0 q_7]$
$[q_0 q_7]$	$[q_0 q_6]$	$[q_0 q_6 q_6]$
$[q_0 q_6 q_6]$	$[q_0 q_6 q_6]$	$[q_0 q_7 q_6]$

KLEEN'S THEOREM (WITH PROOF):-

PART-2 (To obtain RE from FA)

Theorem:- Let $M=(Q,\Sigma,\delta,q_1,F)$ be an FA recognizing the language L. Then there exists an equivalent RE R for regular language L such that $L=L(R)$.

OR

If language L is accepted by a DFA then prove there is a RE R for the language L so that $L=L(R)$

OR

Prove that if $L=L(A)$ for some DFA A, then there is a RE R such that $L=L(R)$.

Proof: Let $Q=\{q_1,q_2\ldots q_n\}$ are states of m/c M where $n=$ no. of states.

The path from state i to state j through an intermediate state (whose no. is not greater than K) is given by RE R_{ij}^K as—

$R_{ij}^K = \{w \in \Sigma^* \mid w$ is label (path) from state i to state $j\}$

where $i>K$ & $j>K$. So

$w=xy$

where $|x|>0$ & $|y|>0$ & $\delta(i,x)=K$ & $\delta(K,y)=$

Basis - If $K=0$

This shows that there is no intermediate state & path from i to j given by following two conditions:-

(1) Direct edge from i to j
Possible when $i \neq j$
So, a DFA H is needed with all input symbol such that there is a transition with following cases:-

a) No input symbol s corresponding RE -
$$R_{ij}^{(0)} = \phi$$

b) Exactly one input symbol s RE is -
$$R_{ij}^{(0)} = a$$

c) Multiple inputs $a_1, a_2, a_3 - - a_k$ s RE is -
$$R_{ij}^{(0)} = a_1 + a_2 + a_3 - \text{---} + a_n.$$

2) There is only one state such that $i=j$ s making a self loop or path of length s is denoted with ε.

a) $R_{ij}^{(0)} = \phi + \varepsilon$

b) $R_{ij}^{(0)} = a + \varepsilon$

c) $R_{ij}^{(0)} = a_1 + a_2 + a_3 + - - - a_k + \varepsilon$

CLOSURE PROPERTIES OF REGULAR LANGUAGES:

If certain languages are regular & language L is formed from them by certain operations (such as ∪ or concatenation) then L is also regular. These properties are called closure properties of regular languages.

These properties express that when one or many languages are regular then certain related languages are also regular.

Properties:-

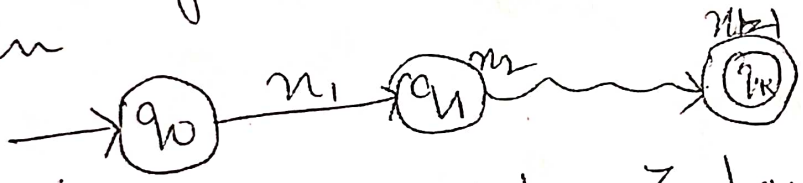
Union of two regular languages is regular.
Intersection of two RL is regular
Complement of regular language is regular.
Difference of two RL is regular.
Reversal of RL is regular.
Closure operation on RL is regular.
Concatenation of RL is regular.
Homomorphism of RL is regular
nverse homomorphism of RL is regular.

* PUMPING LEMMA OF REGULAR LANGUAGES

1) It is a tool to prove that a language is not regular
2) It establishes a necessary (but not sufficient) condition for a language to be regular
3) It prove that a language is not regular by showing that the language doesnot obey the lemma.

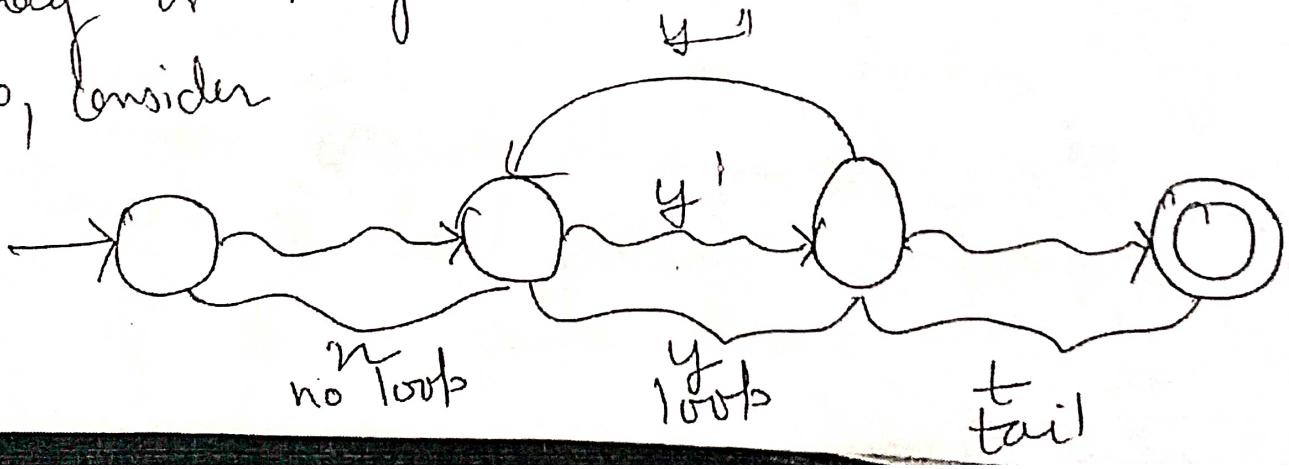
Consider FA with n states,
Let z = string of length atleast n
In order to accept z transition is made for each input & moves through atleast n+1 states.
Therefore automaton will go through atleast one loop in accepting the string.

Let string be $z = x_1 x_2 x_3 \ldots x_{n-1}$
then



The path divides into 3 parts as -
a) Dosnot contain loop
b) First encountered loop
c) May or maynot contain additional loops.

Now, consider



false by contradiction.

Steps of PL-

1) Assume that language is regular.
2) Let some parameter n be constant of PL.
3) Pick a 'suitable' string z with $|z| \geq n$.
4) Show that for every legal decomposition of z into uvw, there exists $i \geq 0$ such that uv^iw doesnot belong to L.
5) Conclude assumption (1) was false.

Q Prove $\{a^n b^n \mid n \geq 0\}$ to be non-regular through pumping lemma.

Sol $RE = \{ab, aabb, aaabbb, ---\}$

(1) $\underset{u}{aa}\underset{v}{ab}\underset{w}{b}$

$|uv| \not\leq n$

$3 \leq 4$ True

(2) $|v| > 0$

$2 > 0$ True

(3) uv^iw

for $i=2$ $aa(ab)^2w$.

False.

So, this language is not regular