UNIT-I

18CSC301T - FINITE AUTOMATA

*) **INTRODUCTION**:

TOC used for efficient computation. It concentrates on problem Solving using different models of computation.

The foundation of computer Science is divided into theoritical and non-theoretical parts, i.e., toc & top.

*) Theory of computation is used to understand the nature of computer.

*) Theory of programming is used to implement the computations.

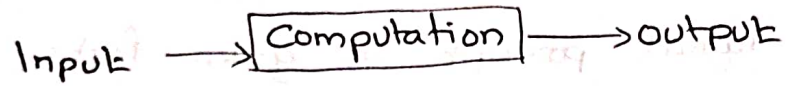


fig: computation process

AUTOMATON is a study of abstract computing devices as machines. It is used in designing and checking the behavior of digital circuits, lexical analyzers, Scanning large text, to Systems of all types that have a finite number of distinct States.

APPLICATIONS OF TOC:

*) Artificial Intelligence
*) compiler design
*) Robotics
*) Circuit Design
*) NLP
*) Quantum computing
*) Pattern recognition

1.2 BASIC MATHEMATICAL NOTATIONS & TECHNIQUES

FORMAL PROOF :

*) Formal methods offer techniques for the specification, development and verification of systems based on mathematical methods. Truth statement is solved by a detailed sequence of steps.

DEDUCTIVE PROOFS :

*) Sequence of statements whose truth leads from initial statement called the hypothesis or the given statem to a conclusion statement.

*) Each step in the proof must follow some accepted logical principle or given facts or some previous statements in deductive proof.

*) Hypothesis may be true or false.

"Hypothesis H to a conclusion C" => If H then C

Ex: if $x \geq 4$, conclusion $2^x \geq x^2$

C is true for $x = 4$, $2^4 \geq 4^2 \Rightarrow 16 \geq 16$

$x = 5$, $2^5 \geq 5^2 \Rightarrow 32 \geq 25$

IF-THEN STMTS

*) If H is true then C is true

-> H implies C

-> H only-if C

-> C if H

-> whenever H holds, C follows

IF and ONLY-IF STMTS

It has two if-then stmts.

"A if and only-if B"

i.e. If B then A => If part

If A then B => only-if part

Operators : $A \leftrightarrow B$, $A \equiv B$

EQUIVALENCE ABOUT SETS

Commutative law $R \cup S = S \cup R$

$R \cup S = E$ and $S \cup R = F$

It follows the form,

*) If x is in E, then x is in F.

*) If x is in F, then x is in E.

CONTRA POSITIVE

"If H then C" represents contrapositive of the statement,

"If not C then not H".

*) If x is not in E then x is not in F.

Proof by contradiction : To prove the statement, the proof should be shown by considering the contradiction statement to be false.

INDUCTIVE PROOFS

*) It deals with recursively defined objects.

Prove two things:

=> BASIS : Show that S(n) for a integer i. (0 or 1)

=> INDUCTION: Assume $n \geq i$, show that "if S(n) then S(n+1)."

PROBLEM ① $\sum_{n=1}^{n} n^2 = \frac{n(n+1)(2n+1)}{6}$

Sol: Let $P(n) = 1^2+2^2+\cdots+n^2 = \frac{n(n+1)(2n+1)}{6}$

Basis: $n=0, \quad 0 = 0$

$n=1, \quad P(1) = \frac{1(1+1)(2\times1+1)}{6} = 1.$

L.H.S = R.H.S

Induction Hypothesis:

Assume for any $k \le n$,

$$P(k) = 1^2+2^2+\cdots+k^2 = \frac{k(k+1)(2k+1)}{6}$$

Induction step:

$n = k+1$

$$P(k+1) = 1^2+2^2+\cdots+k^2+(k+1)^2 = \frac{(k+1)(k+2)(2k+3)}{6}$$ // To be Proved as above.

(Sub Induction Hypothesis in L.H.S.)

$$P(k+1) = \frac{k(k+1)(2k+1)}{6} + (k+1)^2 =$$

$$= \frac{k(k+1)(2k+1) + 6(k+1)^2}{6} = \frac{(k+1)(2k^2+3k+4k+6)}{6}$$

$$= \frac{(k+1)(2k^2+k+6k+6)}{6} =$$

L.HS = R.H.S

② $1^3+2^3+\cdots+n^3 = \frac{n^2(n+1)^2}{4}$

③ $n^3+(n+1)^3+(n+2)^3$ is divisible by 3.

3) BASIC DEFINITIONS

ALPHABET: Any finite set of Symbols (Letters / characters...)

$\Sigma = \{0, 1\}$ // binary alphabet.

STRING: Finite Series of Symbols drawn from alphabet.

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

Strings are 00, 0011, ...

PREFIX, SUFFIX, SUBSTRING, PROPER PREFIX, PROPER SUFFIX

LANGUAGE: Any set of Strings over fixed alphabet.

Ex: $L = \{0^n 1^n \mid \text{where } n \geq 1\}$

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

$L^+ = \{0, 1, 01, 00, \ldots$

*) Here L is the Language. $\{01, 00, \ldots\}$ are Strings over the alphabet $\{0, 1\}$.

CONCATENATION, REVERSE OF STRING, EMPTY STRING

x y xy w w^R ϵ

GRAMMAR: Gives the Structure of the language and defines rules.

Diff rules for diff grammar.

$G = (V, T, P, S)$ { Variables, Terminals, Production, Start Symbol }

GRAPH: → Directed graph → undirected graph.

TREE: Root node, Vertices, Interior vertices, Level, Height

POWER OF ALPHABET: $\Sigma = \Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$

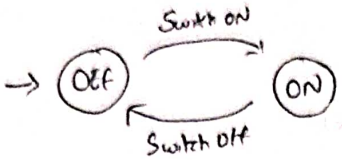
CLOSURE: KLEENE CLOSURE L^* => Zero or more occurrences

POSITIVE CLOSURE L^+ => one or more occurrences

(*) FINITE STATE SYSTEMS

*) FSM has finite no. of states.

*) Based on i/p, FSM generates an o/p form machine and move to new state or remains in same.



FSM for Light Bulb

2 types of FSM's

ACCEPTORS

*) Seq of i/p from Σ

*) Has accepting state

Ex: $\rightarrow \bigcirc \xrightarrow{i} \bigcirc \xrightarrow{f} \textcircled{\bigcirc}$

TRANSDUCERS

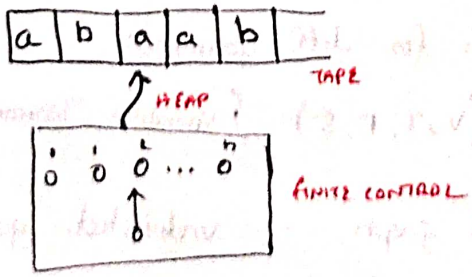
*) Behavior is defined by transitions.

Ex: Above Diagram

(*) FINITE AUTOMATON

*) Mathematical model with discrete i/p and o/p's and the system can be in any one of finite no. of states.

*) Computing devices that accepts / recognize regular languages are called finite automata.



Model consists of:

→ INPUT TAPE - No. of cells which holds one i/p symbol.

→ FINITE CONTROL - changes or remains in state based on current state & i/p Symbol.

→ TAPE HEAD - Reads one Symbol & moves ahead.

DEFINITION:

*) FA accepts only "Regular language".

*) If end of I/p reached to a final state then string is accepted otherwise rejected.

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

Q - Finite set of states

Σ - Finite set of I/p alphabets

δ - Transition function

$$\delta : Q \times F \rightarrow Q$$

q_0 - Initial State

F - Set of final/accepting States

*) FA can be represented in two ways:

TRANSITION DIAGRAM:

*) A directed graph

*) States are nodes connected by are

*) final state as double circle.

TRANSITION TABLE:

*) A table : States in rows, I/p in columns

*) Initial State : →

*) final state : *

States / Inputs	0	1
q_0	q_0	q_1
$*q_1$	q_1	q_1

TYPES: *) NDFA (Non deterministic Finite Automata)

- NFA with ε-moves
- NFA without ε-moves

*) DFA (Deterministic finite Automata)

Relational operator
Identifier
Integer

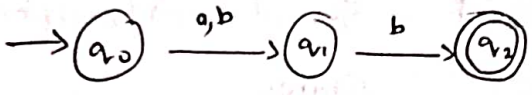
(*) DETERMINISTIC FINITE AUTOMATA

*) for a single i/p, the state goes to only one state.

*) State may be new or old state.

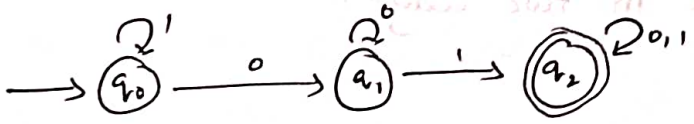
*) No state has ε - transition

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



	a	b
→ q_0	q_1	q_1
q_1	—	q_2
* q_2	—	—

Ex: check whether i/p string 010 is accepted.



$\delta(q_0, 0) = q_1$

$\delta(q_0, 01) = \delta(\delta(q_0, 0), 1) = \delta(q_1, 1) = q_2$

$\delta(q_0, 010) = \delta(\delta(q_0, 01), 0) = \delta(q_2, 0) = q_2$

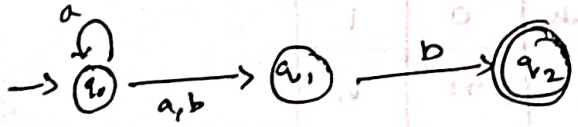
// q_2 is final state. ∴ 010 is accepted.

(*) NON DETERMINISTIC FINITE AUTOMATA

*) for single i/p it goes to more than one state.

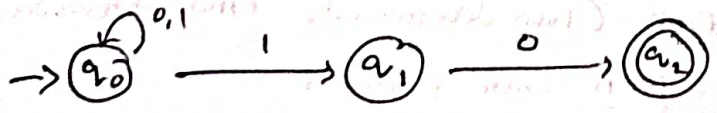
*) Multiple transitions for a single input.

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



	a	b
→ q_0	$\{q_0, q_1\}$	q_1
q_1	—	q_2
* q_2	—	—

Ex: check 010 is accepted.



$\delta(q_0, 010) = \delta(\delta(q_0, 01), 0) \Rightarrow \delta(\delta(\delta(q_0, 10), 1), 0)$

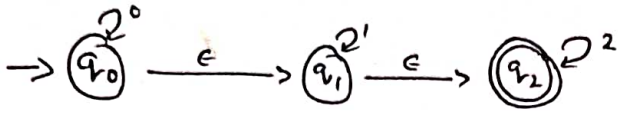
$\rightarrow \delta(\delta(q_0, 1), 0) \Rightarrow \delta(q_1, 0) = q_2$

*) FINITE AUTOMATON WITH ϵ-MOVES:

-> Allowed to make transition w/o receiving an i/p symbol.

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

ϵ-closure: vertices having path via ϵ



ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

*) REGULAR LANGUAGES:

-> Formal language written by "Regular expression."

-> It satisfies: - Represented by RE.
- Accepted by DFA and NDFA.
- can be generated from type-3 of regular grammar.

*) Empty i/p 'ϵ' is a regular lang.

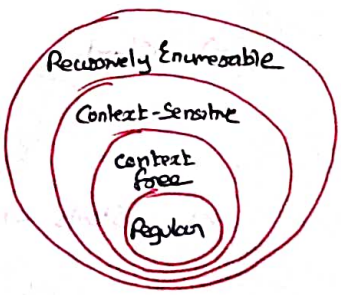
*) Single Symbol 'a' ,,

*) A and B are two regular languages.

-> Union : A+B or A|B or AUB
-> Concatenation : A.B or AB
-> closure : A^*

} is a regular lang.

Chomsky Hierarchy	Grammars	Languages	Automaton
Type 0	Unrestricted or phrase level	Recursively enumerable	Turing Machine
Type 1	context Sensitive	context Sensitive	Linear bound Automata
Type 2	Context-free	Context-free	PDA
Type 3	Regular grammar	Regular lang	FA



*) REGULAR EXPRESSION:

→ Reg. Lang accepted by FA is denoted by RE.

→ RE : combination of operands and operators.

→ Each RE 'α' denotes a Reg. lang L(α).

OPERATIONS IN RE:

UNION: LUM or L+M = Strings either in L or M.

$L = \{10, 11\}$ $M = \{10, 111\}$

$LUM = \{10, 11, 111\}$

CONCATENATION: Any String in L is concatenating with any String in M.

$L = \{10, 11\}$ $M = \{10, 111\}$

$L.M = \{1010, 10111, 1110, 11111\}$

CLOSURE: Set of Strings : taking any no. of Strings from L with replication, concatenation.

$L = \{0, 11\}$

$L^* = \{\epsilon, 0, 11, 011, 00, 1111, 110, \ldots\}$

$L^+ = \{0, 11, 011, 00, \ldots\}$

*) Construct RE ::

*) String consisting of zero or more occurences of 01 or alternating 0's and 1's (begin with 0 and end with 1).

$(01)^*$ $(01)^* + (10)^*$ (if beginning and ending not mentioned)

*) Starting and ending with Symbol zero. $\Sigma = \{0,1\}$

$0\,(0+1)^*\,0$

*) Starts with "011" $\{0,1\}$ *) End with "aba" $\{a,b\}$

$011\,(0+1)^*$ $(a+b)^*aba$

*) Single 0 followed by 1's

01^*

*) Single 0 followed by 1's or Single 1 followed by 0's

$01^* + 10^*$

*) Any no a's followed by b's

$a^* b^*$

*) ... followed by c's

$a^* b^* c^*$

*) contains atleast one a & one b

$(a+b)^* ab \qquad aba\,(a+b)^*$

*) atleast one pair of 11

$(0+1)^* 11 (0+1)^*$

*) Atmost one consecutive pair of 1's

$(00+01+10)^* 11 (00+01+10)^* + \underline{(00+01+10)^*}$ → option for no 11 pair

*) 3rd Symbol from left is one.

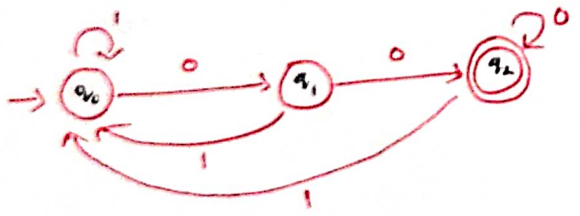
$(0+1)(0+1)1(0+1)^*$

*) No. of 0's is divisible by 5

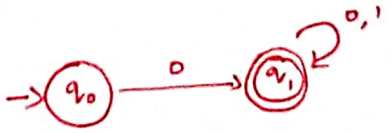
$(00000+1)^*$

*) Give DFA accepting following languages: $\Sigma = \{0,1\}$

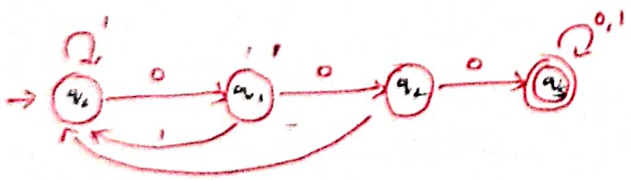
*) Ending with 00 / ~~Ends~~ ~~Starts~~ with 11 - DIY



*) Starts with 0 / Starts with 1 - DIY

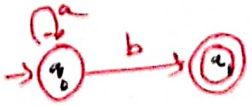


*) 3 consecutive 0's



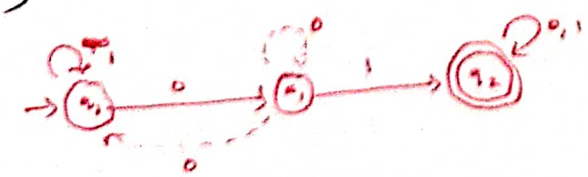
*) $L = \{ a^n b ; n \geq 0 \}$ / $a b^n, n \geq 0$ - DIY

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

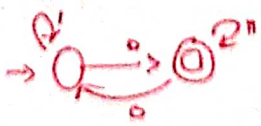


*) $a^* \cup b^*$

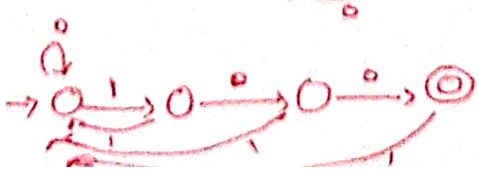
*) 01 as Substring:



*) odd no. of zeros



*) 3rd Symbol from right end is 1.



(7) CONVERT NDFA TO DFA:

Problem

Steps:

*) construct Transition table for NFA

*) Start from processing input symbol for first state

*) Start processing i/p symbols for new formed state

*) construct Transition table for DFA.

	0	1
q_0	$[q_0, q_1]$	q_0
q_1	–	q_2
q_2	–	–

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

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

	0	1
q_0	$[q_0, q_1]$	q_0

$\Rightarrow \delta([q_0, q_1], 0) \Rightarrow \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_1\} \cup \{\emptyset\} \Rightarrow \{q_0, q_1\}$

$\Rightarrow \delta([q_0, q_1], 1) \Rightarrow \delta(q_0, 1) \cup \delta(q_1, 1)$

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

	0	1
q_0	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$

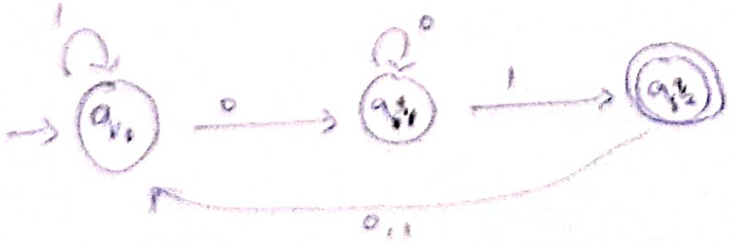
$\Rightarrow \delta(\{q_0, q_2\}, 0) \Rightarrow \delta(q_0, 0) \cup \delta(q_2, 0)$

$= q_0, q_1 \cup \emptyset \Rightarrow \{q_0, q_1\}$

$\delta(\{q_0, q_2\}, 1) \Rightarrow \delta(q_0, 1) \cup \delta(q_2, 1)$

$= q_0, q_1 \cup \emptyset \Rightarrow \{q_0, q_1\}$

	0	1
q_0	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_0, q_1]$	$[q_0, q_1]$

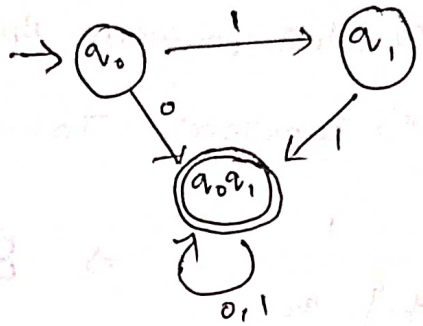


(*) CONVERT TO DFA :

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

	0	1
$\to q_0$	$\{q_0, q_1\}$	$\{q_1\}$
q_1	ϕ	$\{q_0, q_1\}$

	0	1
$\to q_0$	$\{q_0, q_1\}$	$\{q_1\}$
q_1	–	$\{q_0, q_1\}$
$*q_0, q_1$	$\{q_0, q_1\}$	$\{q_0, q_1\}$



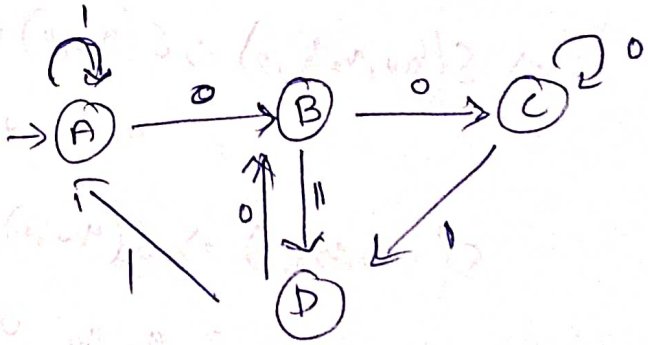
(*)

	0	1
$\to P$	$\{P, q\}$	$\{P\}$
q	$\{r, s\}$	$\{t\}$
r	$\{P, r\}$	$\{t\}$
$*s$	ϕ	ϕ
$*t$	ϕ	ϕ

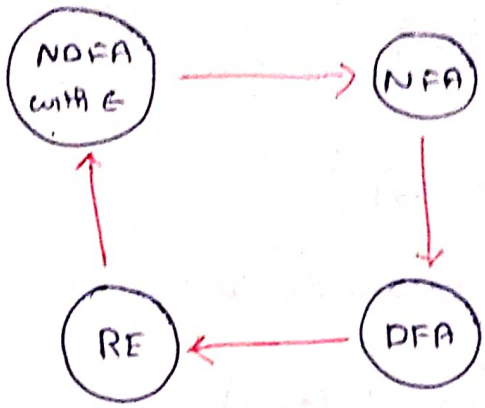
	0	1
$\to P$	$\{P, q\}$	$\{P\}$
$\{P, q\}$	$\{P, q, r, s\}$	$\{P, t\}$
$*\{P, q, r, s\}$	$\{P, q, r, s\}$	$\{P, t\}$
$*\{P, t\}$	$\{P, q\}$	$\{P\}$

$P \Rightarrow A$ $\quad P, q \Rightarrow B$ $\quad P, q, r, s \Rightarrow C$ $\quad P, t \Rightarrow D$

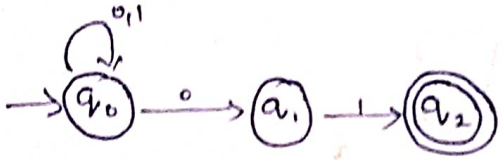
	0	1
$\to A$	B	A
B	C	D
$*C$	C	D
$*D$	B	A



EQUIVALENCE OF FOUR DIFFERENT NOTATIONS FOR REGULAR LANGUAGES



(*) Convert the following NDFA to DFA



Soln:

Step1: Consider initial state q_0 & process for input Symbol 0

$$\delta_N(q_0,0) = \{q_0,q_1\}$$

$$\therefore \delta_D(q_0,0) = [q_0,q_1]$$

for input Symbol 1

$$\delta_N(q_0,1) = \{q_0\}$$

$$\therefore \delta_D(q_0,1) = [q_0]$$

Step2: New state is $[q_0q_1]$. Process for input Symbol 0.

$$\delta_N([q_0,q_1],0) = \delta_N(q_0,0) \cup \delta_N(q_1,0)$$

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

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

$$\therefore \delta_D([q_0,q_1],0) = [q_0,q_1]$$

1) Define initial state
2) If new state is formed, Define the newly formed state
3) Repeat (2) until all states are defined.
4) Draw Table and DFA diagram.

*) No. of steps will differ according to the no of newly formed States.

Process for input Symbol 1

$$\delta_N([q_0,q_1],1) = \delta_N(q_0,1) \cup \delta_N(q_1,1)$$
$$= \{q_0\} \cup \{q_2\}$$
$$= \{q_0,q_2\}$$

$$\therefore \boxed{\delta_D([q_0,q_1],1) = [q_0,q_2]}$$

Step 3 : New state is $[q_0 q_2]$. Process for I/P 0

$$\delta_N([q_0,q_2],0) = \delta_N(q_0,0) \cup \delta_N(q_2,0)$$
$$= \{q_0,q_1\} \cup \phi$$
$$= \{q_0,q_1\}$$

$$\boxed{\delta_D(\{q_0,q_2\},0) = [q_0,q_1]}$$

Process for I/P 1

$$\delta_N(\{q_0,q_2\},1) = \delta_N(q_0,1) \cup \delta_N(q_2,1)$$
$$= \{q_0\} \cup \phi$$
$$= \{q_0\}$$

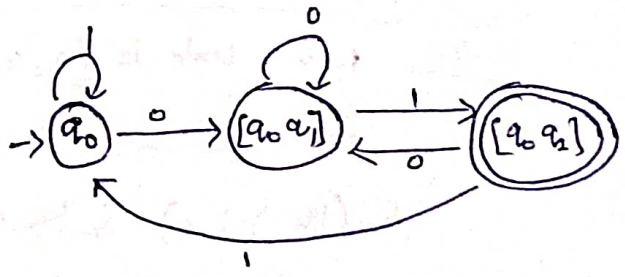
$$\boxed{\delta_D(\{q_0,q_2\},1) = [q_0]}$$

Step 4: Construct Transition Table

States	Inputs	
	0	1
$\rightarrow [q_0]$	$[q_0,q_1]$	$[q_0]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_2]$
$*[q_0,q_2]$	$[q_0,q_1]$	$[q_0]$

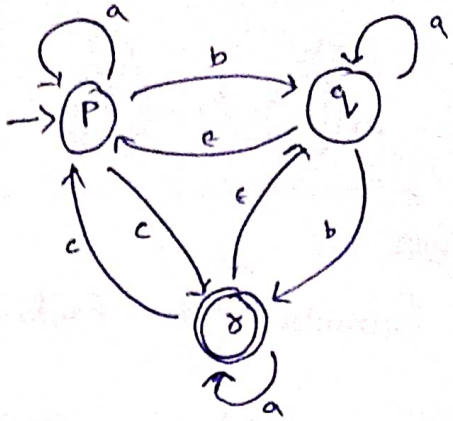
*) If a newly formed DFA state contains a final state from NFA, then it is also final state.

*) $[q_0 q_2]$ is final bcos, it contains 'q_2' (final state in NFA)



(*) Convert NDFA - ε to without ε

	ε	a	b	c
→p	ϕ	{p}	{q}	{r}
q	{p}	{q}	{r}	ϕ
*r	{q}	{r}	ϕ	{p}



Soln

Step1: Draw Transition diagram: ##

ε-closure (q_a): from q_a what are the states reachable by ε-transition

(*) (1) Find ε-closure of all states.
(2) Find Each state Process for all inputs.
(3) Construct table and draw diagram.

Step 2: Find the eclosure of each state.

ε - closure (p) = {p}
ε-closure (q) = {p, q}
ε - closure (r) = {p, q, r}

Step3: for each state process all i/p symbols: Here it is {a, b, c} are i/p's

States: {p, q, r}

So, find
δ(p,a) δ(p,b) δ(p,c)
δ(q,a) δ(q,b) δ(q,c)
δ(r,a) δ(r,b), δ(r,c)

$\delta(p,a) = \varepsilon\text{-closure}(\delta(\hat{\delta}(p,\varepsilon),a))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(p),a))$

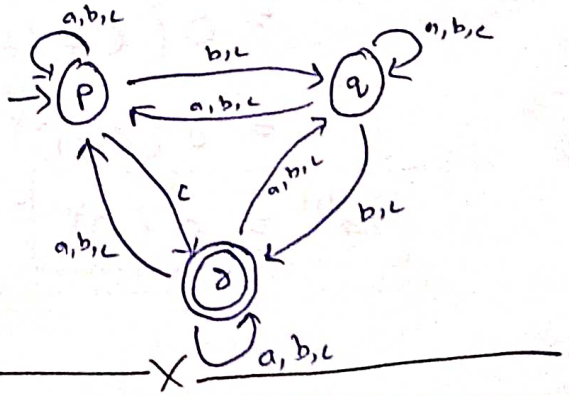
$= \varepsilon\text{-closure}(\delta(p),a)$

$= \varepsilon\text{-closure}(\delta(p,a))$.

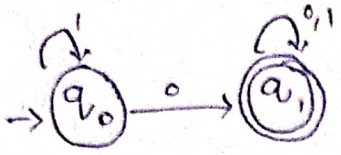
$= \varepsilon\text{-closure}(p)$

$\delta(p,a) = \{p\}$ // find for all combination, (9 values)

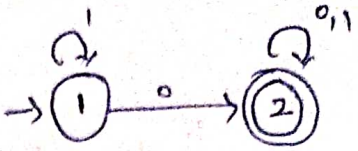
	a	b	c
→p	{p}	{p,q}	{p,q,r}
q	{p,q}	{p,q,r}	{p,q,r}
*r	{p,q,r}	{p,q,r}	{p,q,r}



(*) Convert given DFA to Regular Expression.



If given like this considers as =>



From this,
1 is initial state
2 is final state
No. of states is 2
i = 1
j = 2
k = 2
find $R_{12}^{(2)}$

Soln:

Formula for finding regular expression is,

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \cdot \left(R_{kk}^{(k-1)}\right)^{*} \cdot R_{kj}^{(k-1)}$$

i = 1, j = 2, k = 2 ; To find $R_{12}^{(2)}$

$$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)} \left(R_{22}^{(1)}\right)^{*} R_{22}^{(1)} \quad -(1)$$

To find $R_{12}^{(1)}$, i = 1, j = 2, k = 1, Apply formula, Sub ans in (1)

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)}\right)^{*} R_{12}^{(0)}$$

$R_{12}^{(0)} = ?$
If power is '0', we can get ans from given diag.
$R_{12}^{(0)}$ => Input alphabet in transition from "State 1" to "State 2"

Next to find value for

$$\boxed{R_{12}^{(0)} = 0}$$

// State 1 to State 2 i/p is "0".

and

$$\boxed{R_{11}^{(0)} = \epsilon + 1}$$

$$\therefore R_{12}^{(1)} = 0 + (\epsilon+1) \cdot (\epsilon+1)^{*} \cdot 0$$

$$= 0 + (\epsilon+1)^{*} \cdot 0 \quad \text{(By Rule)}$$

$$= 0\left(\epsilon + (\epsilon+1)^{*}\right)$$

$$= 0\,(\epsilon+1)^{*}$$

$$\boxed{R_{12}^{(1)} = 0\,(1)^{*}}$$

Here . is not Multiplication
It's "followed by". (concatenate)

$a + ab = a(1 + b)$
↳ In Maths

$a + ab = a(\epsilon + b)$
↳ In TOC

(E)

To find $R_{22}^{(1)}$, $i=2, j=2, k=1$

$$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}\left(R_{11}^{(0)}\right)^* R_{12}^{(0)}$$

To find $R_{22}^{(0)} = \epsilon + 0 + 1$

$$\boxed{R_{21}^{(0)} = \phi}$$

ϵ is added bcas ② ← ϵ will be there by default for self transition.

ϕ, bcas no transition

ϵ and ϕ were different | ϵ - empty string (length 0) | ϕ - NULL

$$R_{22}^{(1)} = (\epsilon + 0 + 1) + \phi\,(\epsilon+1)^*.0$$

$$= (\epsilon + 0 + 1) + \phi$$

$$\boxed{R_{22}^{(1)} = (\epsilon + 0 + 1)}$$

$$\therefore R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}\left(R_{22}^{(1)}\right)^* R_{22}^{(1)}$$

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

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

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

$$\boxed{R_{12}^{(2)} = 1^*0(0+1)^*}$$

RULES FOR SIMPLIFICATION: ← were used mostly

1. $(\epsilon+1)^* = 1^*$
2. $1^*(\epsilon+1) = 1^*$
3. $(\epsilon+1)^* = 1^*$
4. $(\epsilon+1)+1^* = 1^*$
5. $\epsilon + 00^* = 0^*$
6. $0 + 1^*0 = 1^*0$
7. $0 + 01^* = 01^*$
8. $(1+0)^* = (1^*0^*)^*$
9. $(01)^*0 = 0(10)^*$
10. $(0^*1)^*0^* = (0+1)^*$
11. $\phi 0 = \phi$
12. $\phi^* = \epsilon$
13. $\epsilon 0 = 0$
14. $11^* \neq 1^*$
15. $1^*1 \neq 1^*$
16. $\phi + 0 = 0$

ϕ. Anything $\Rightarrow \phi$

Anything. $\phi \Rightarrow \phi$

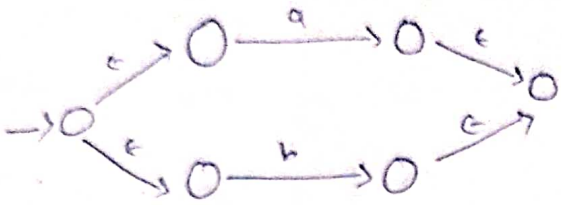
ϕ + Anything $\Rightarrow$ Anything

Anything + $\phi \Rightarrow$ Anything

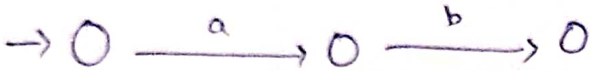
(*) Conversion of RE to DFA :

If three basic conversion is known, we can do for any RE.

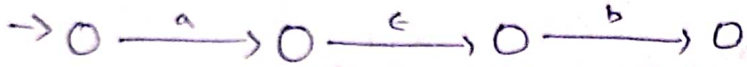
a + b (either a or b)



a.b (a followed by b i.e a and b)

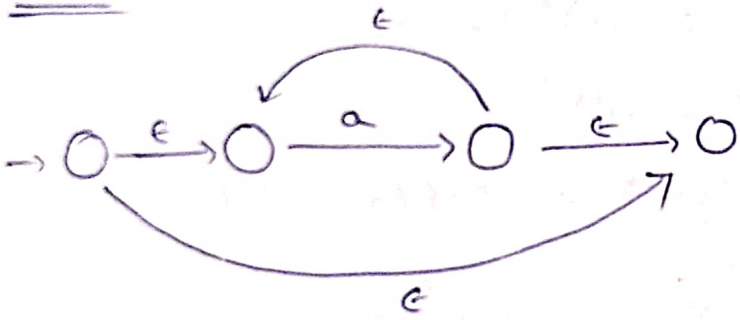


or



Use this diag mostly ⇒ (Recommended)

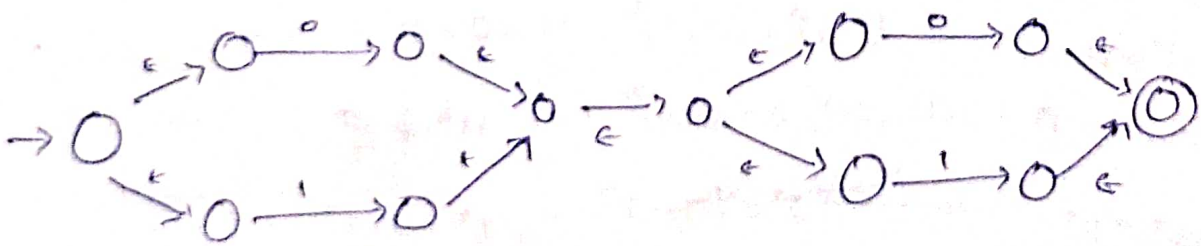
a*



(*) Construct NDFA for Regular expr (0|1) (0|1) ⇒ Similar to (0+1) (0+1)

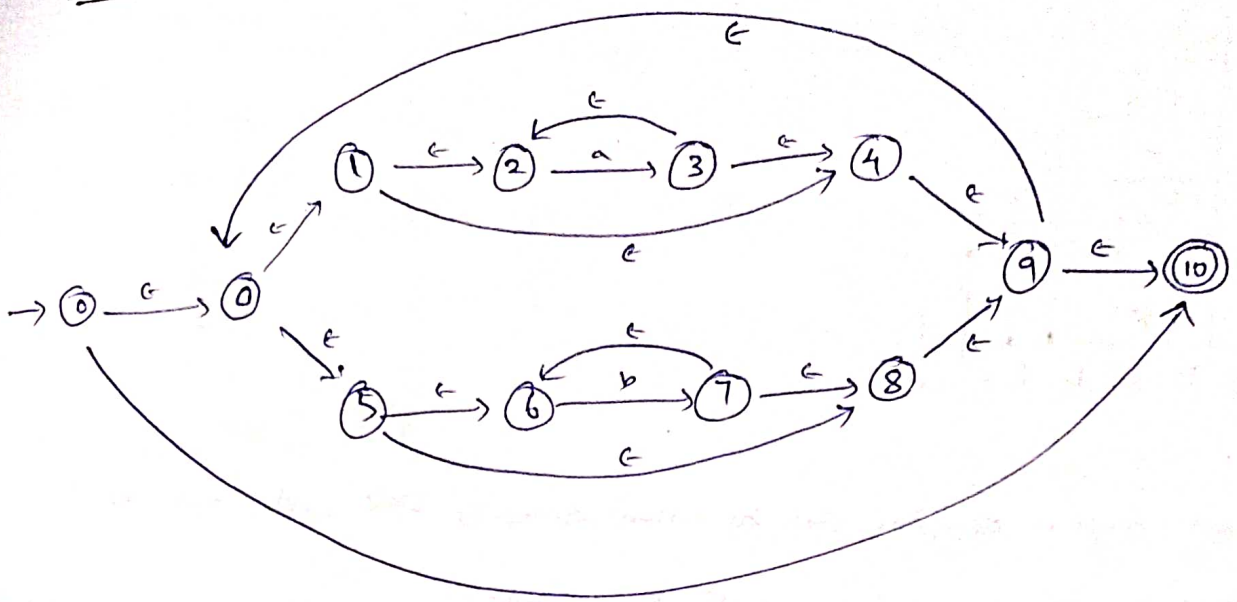
We know NDFA for (a+1).

∴



a b

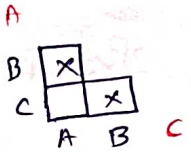
(*) Draw NDFA for RE $(a^*|b^*)^*$



X ε X X

(*) Minimization of DFA:

1) Table filling algorithm:

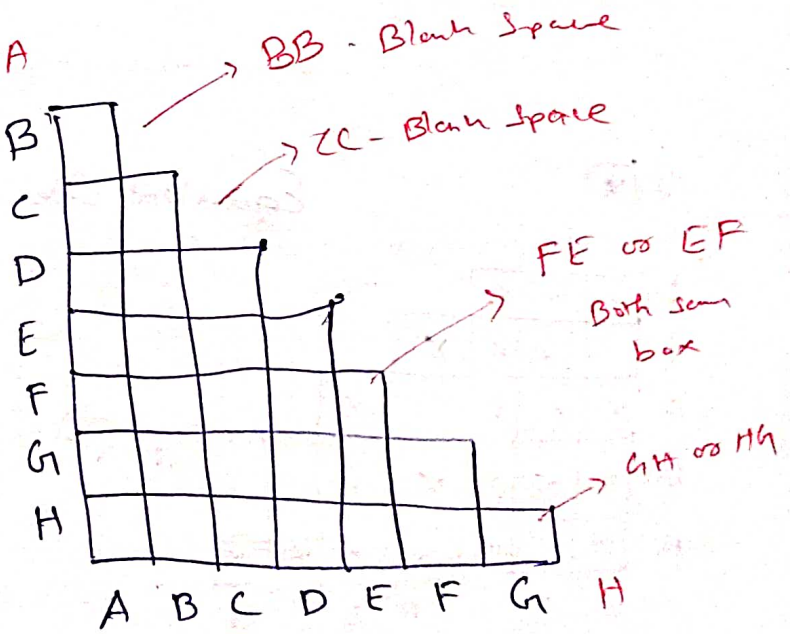


⇒ X Indicates non equivalent states.

⇒ Blank Space Indicates the equivalent states.

(*) Find equivalence and minimization of DFA

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



*) Put X,
If one state in the Box is final
*) Put X for all final state
If Both are final in the Box
don't put X

Soln:

Do not compare accepting stat to non accepting state and also do not compare same state with same state.

So (B,C), (D,C), (E,C), (F,C) (G,C) (H,C) (A,C) are not equivalent.

⇒) For each and every other pair (Empty boxes) find equivalence.

1) Take pair (A,B)

$\delta(A,0) = B$ $\delta(B,0) = G$

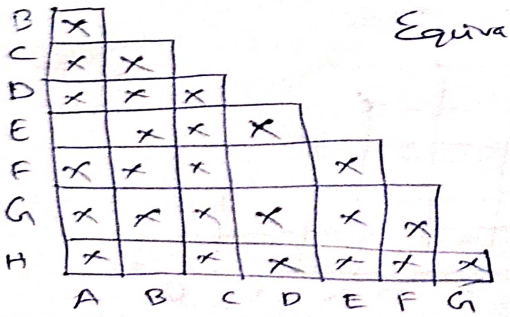
$\delta(A,1) = F$ $\delta(B,1 = C$

∴ (A,B) are not equivalent.

i) BF is Blank
ii) GC is X
⇒ So AB is also X

If Both are Blank, then result is Blank otherwise mark X.

// After finding all pairs. Some empty boxes will be there. Confirm it once again by finding equivalence for 2nd time.



Equivalent States are A≡E, B≡H, D≡F

*) A & E are Equivalent So ~~Any one of~~ can be Eliminated.

*) If E is Eliminated replace E with A wherever possible.

*) Construct table and diagram.

Find minimized DFA for following Automata.

	a	b
→A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

Soln

0 Equivalence: (A B C D) (E)

↓ Non Accepting State ↓ Accepting State

*) Find Equivalence 0, 1, 2 ... n until n and n-1 are same values.

Step1: Two partitions (Equivalence 0)
- i) Non Accepting States
- ii) Accepting States

Step2: Use set in "Equiv 0" for finding Equiv 1

i.e., Use set in "n-1 equiv" for finding "Equiv 1"

Repeat until two values are equal.

1 Equivalence:

Compare pairs in (A B C D)

(A B C) (D) (E)

i.e. AB, AC, AD
BC, BD
CD → find Equiv

AB: $\delta(A,a)=B$ $\delta(A,b)=C$
(AB ← $\delta(B,a)=B$ $\delta(B,b)=D$
Same Set | Same Set

AC: B | C (As Equiv 0)
(ABC ← B | C
Same Set | Same Set (As Equiv 0)

AD: B | C
(ABC) (D) ← B | E
Same Set | DIFF Set (As Equiv 0)

2 Equivalence: (AC) (B) (D) (E)

AB: B | C
B | D
Same Set | Diff Sets (As Equiv 1)
So AB are not equivalent
⇒ (A (B (D) (E)

BC: B | D
B | C
Same | DIFF → (A (B) (D) (E)

AC: B | C
B | C
Same | Same ⇒ (AC) (B) (D) (E)

3 Equivalence: (AC) (B) (D) (E)

AC: B | C
B | C (As in Equiv 2)
Same | Same

So stop Repeating Finding Equivalence

finally we got (AC) (B) (D) (E)

i.e. $A \equiv C$, we can eliminate any one state, and replace other states.

i.e. if C is eliminated replace C with A wherever it occurs.

	a	b
→ A	B	A
B	B	D
D	B	E
* E	B	A

this is the Minimized DFA.

Problems based on Pumping Lemma:

→ To prove given language is regular or not.

Take a String w, break into 3 Strings (i.e.) $w = xyz$

a) $|xy| \leq n$

b) $|y| \geq 1$ or $y \neq \epsilon$

c) For all $k \geq 0$, xy^kz should be in Language L.

(If not, then it is not regular language).

(*) **Show $L = \{a^n b^n \mid n \geq 1\}$ is not regular.**

(*) In all books they will split formula (condition) into three Strings and processed. Instead considering an example Step will be easy.

Possible strings = { ab, abbb, aaabbb, ... }

Considering $w = \underbrace{a}_{x}\underbrace{a}_{y}\underbrace{b}_{z}\underbrace{b}$

$x = a$

$y = ab$

$z = b$

(k)

Cond a) $|xy| \leq n$ — n is length of String

$|1+2| \leq 4$

$|3| \leq 4$

Cond b) $|y| \geq 1$

$|2| \geq 1$

Cond c) If $k=0$, $xy^0z \Rightarrow ab$ (belongs to L)

If $k=1$, $xy^1z \Rightarrow aabb$ (belongs to L)

If $k=2$, $xy^2z \Rightarrow aababb$ (not belongs to L)

$\therefore$ $L = \{a^n b^n / n \geq 1\}$ is not a regular language.

(*) Proof by Induction:

Inductive proofs deals with recursively defined objects.

To prove, we have three steps. (normally 2 steps, second step having 2 Process)

Basis: Prove for initial value. ($i=0$ or $i=1$ as per conditions).

Induction: Assume, (intermediate value k) it is true.

Induction Principle: Prove for value "k+1" using "k".

LHS => Split into two parts

i) upto "initial value to k"

ii) "k+1" as Separate.

Here "k+1" differs according to condition,

Like $(k+1)^2$, $(k+2)$, $3k$, ...

RHS => Substitute "k+1" as it is in formula.

(*) Prove by induction on 'n' that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

Soln

1) Basis of induction

Assume $n=1$ $\quad \sum_{i=0}^{1} i = \frac{1(1+1)}{2}$

$0+1 = \frac{1(2)}{2}$

$1 = 1$ $\quad$ L.HS = R.H.S

2) Induction Hypothesis

Assume $n=k$, Then $\boxed{1+2+3\ldots+k = \frac{k(k+1)}{2} \text{ is true.}}$

3) Induction step

Assume $n=k+1$,

L.H.S $\sum_{i=0}^{k+1} i$ $\quad$ R.H.S $= \frac{n(n+1)}{2}$ // Sub $n=k+1$

(Splitting into two parts)

$= \sum_{i=0}^{k} + (k+1)$ $\quad$ => upto k and $(k+1)^{th}$ term $= \frac{(k+1)((k+1)+1)}{2}$ => Applying k+1 for 'n'

$= \frac{k(k+1)}{2} + (k+1)$ $\quad$ *) Simplify both sides and show both are equal $= \frac{(k+1)(k+2)}{2}$

$= \frac{(k^2+k)+2(k+1)}{2}$ $\quad = \frac{k^2+2k+k+2}{2}$

$= \frac{k^2+k+2k+2}{2}$ $\quad = \frac{k^2+k+2k+2}{2}$

$\boxed{R.HS = L.HS}$

Hence, Given Hypothesis is proved.

$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ is proved by Induction.