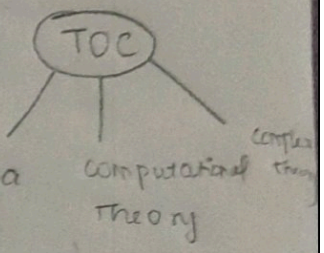
UNIT - I

Introduction to Automata

Need for automata theory - Introduction to formal proof - determinate finite automata (DFA), (NFA) Non deterministic finite automata - Finite automata with epsilon transitions, Equivalence of DFA and NFA - Applications of Finite Automata Equilance of NFA with and without transitions (epsilon) - conversion of NFA into DFA - Minimization of DFA

Need for Automata Theory



Automata Theory is an exciting theoritical branch of computer science.

The Automata computer Scientists are able to understand how machines are computing the functions and how to solve the problems.

In this Automata theory we are dealing with the definitions and Properties of various Mathematical Models of computation.

Alphabets:

An alphabet is a finite collection of symbols

Example: $\Sigma = \{0, 1\}$ $\Sigma = \{a, b\}$

Here 0,1 or a,b these are the alphabetical elements.

String:

String is a finite collection of symbols

from an alphabet

For Example If an alphabet $\Sigma = \{a, b\}$ then various strings that can be formed from Σ are $W = \{a, b, aa, bb, ab, ba\}$ aaa, aab ... }

Null String is used to Move from one state to another State by accepting empty input

$W = \{\epsilon \in \Sigma$ → value either 0 or nothing

Operations on Strings:

1. Concatenation → $W_1 = ab$, $W_2 = bb$ → $W_1 \cdot W_2$ abbb
2. Transpose → $W_1^T \rightarrow ba$
3. Palindrome → $W_1 = aba$

Languages:

Language is a collection of Strings.

The operations on languages:

1. Concatenation → $L_1 \cdot L_2$
2. Intersection → $L_1 \cap L_2$
3. union → $L_1 \cup L_2$
4. Difference → $L_1 - L_2$

Applications of Automata theory:

* Automata theory Plays an important role in compiler design.
* Automata theory deals with design of finite state Machines or finite Automata.

* The design and analysis of digital circuits uses automata theory.

* TO Prove the correctness of a Program automata theory is used.

Finite Automata:

The finite Automata (or) finite state transition System represents a mathematical model of a System with certain inputs.

Finite automata is a collection of five tuples

$(Q, \Sigma, \delta, q_0, F)$ — δ: transition function; F: Final state; q_0: initial state

Q: No. of states; Σ: No. of inputs

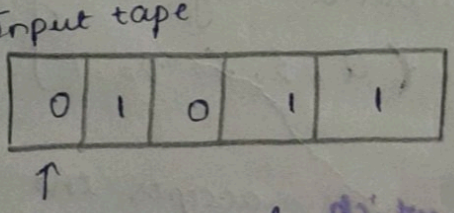
Q is the finite Set of states.

Σ is the input (alphabets or numbers)

q_0 is an initial state

F is a final state

δ is the transition function.



Input tape: It is a linear tape having some number of cells. Each input Symbol is placed in each cell.

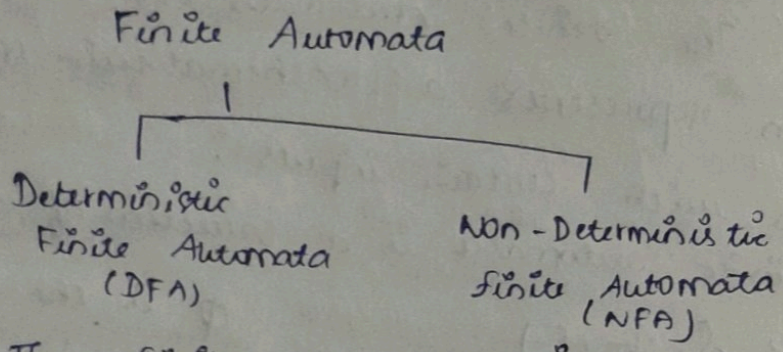
Finite control: It decides the next state on receiving the particular input from the input tape.

$$(q, w) \vdash (q', w')$$

If W is a String and M is the finite automata then W is accepted by Finite Automata Then a Machine accepts a language.

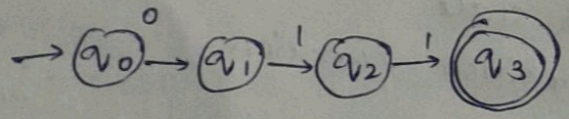
$L = L(M)$

Types of Automata:



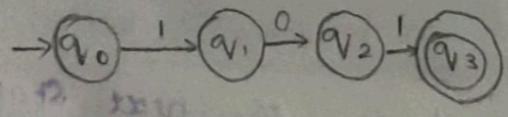
DFA: The finite Automata is called Deterministic Finite Automata if there is only one path for a specific input from current state to next state.

For Example



The DFA can be represented by using the same five tuples $\{Q, \Sigma, \delta, q_0, F\}$

1. Design a DFA which accepts the only input 101 over the input set $\Sigma = \{0, 1\}$



Transition function:

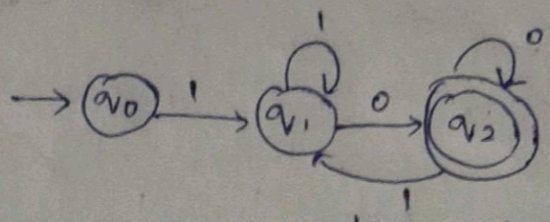
$\vdash \delta(q_0, 101)$

$\vdash \delta(q_1, 01)$

$\vdash \delta(q_2, 1)$

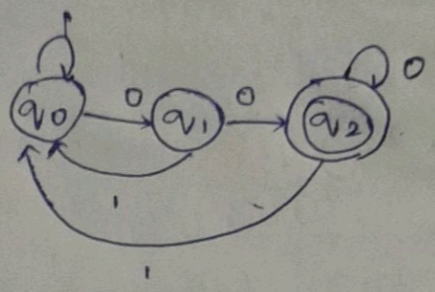
$\vdash \delta(q_3)$

2. Design a DFA which accepts those strings which starts with 1 and ends with 0.



3. Design a DFA that always ends with 00

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



4. Design a DFA should accept even number of 0 and even number of 1

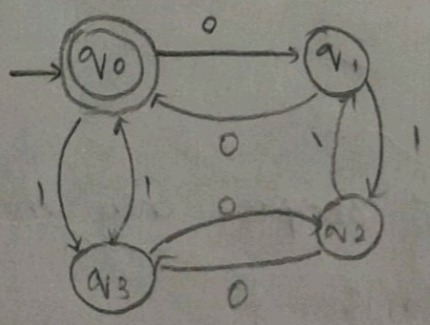
5. Design a DFA that checks whether the given unary number is divisible by 3.

6. Design a DFA to accept strings of a's and b's ending with abb over a,b

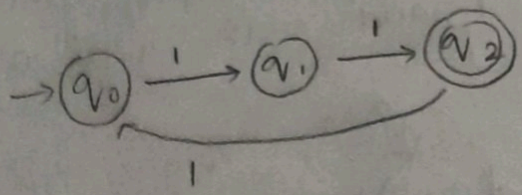
7. Design a DFA $L(M) = \{w \mid w \in \{0,1\}^*\}$ and w is a string that does not contain consequtive 1's

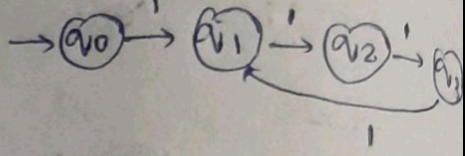
8. Which Recognize the language $L = \{b^m a b^n , m,n>0\}$

4.

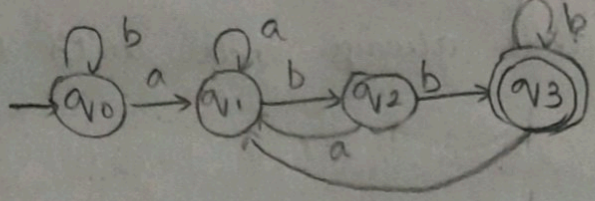


5.

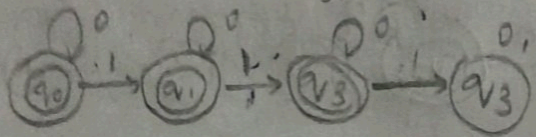




6.

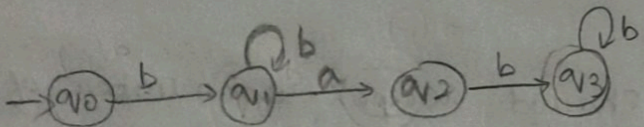


7.

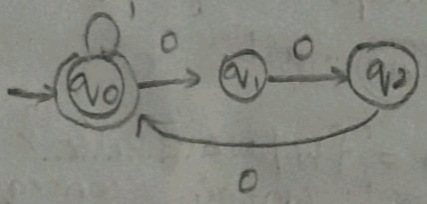


→ dead state

8.

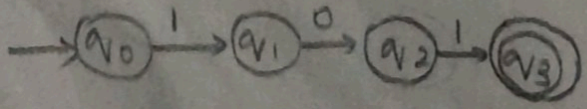


1. Some the DFA to accept the string $\Sigma = \{0,1\}$ containing 3 consecutive zeroes.



2. Construct a DFA over a,b which produces not more than 3 a's.

3. Create a FA which accepts only the input 101, $\Sigma = \{0,1\}$



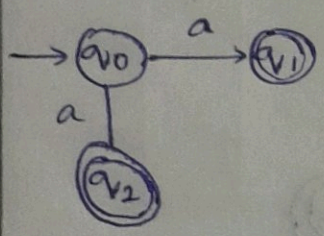
Difference between DFA and NFA

NFA

NFA is the non-deterministic finite Automata.

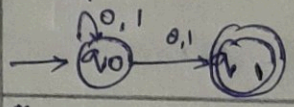
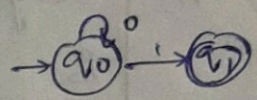
When there exises many paths for a specific input from current state to next state

For example:

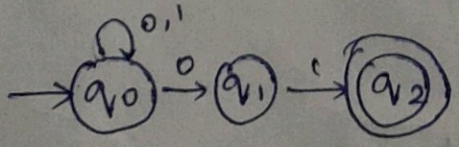


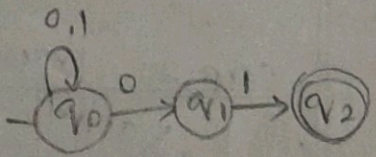
The NFA consists of five tuple $(Q, \Sigma, \delta, q_0, F)$

DFA

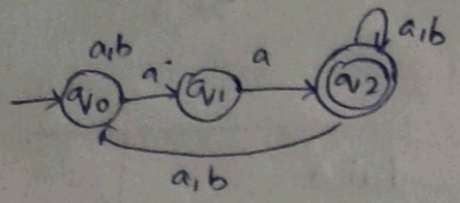
NFA	DFA
* Non-deterministic Finite Automata	* Deterministic Finite Automata
* There are more than one transition for a given input from current state to next state	* There is only one fixed transition for a given input from current state to next state
Example: 	Example: 
* Transition function: $\delta(q_0, a) = \{q_0, q_1\}$	* Transition function: $\delta(q_0, a) = q_1$
* In NFA epsilon ϵ-transition are not allowed	* In NFA ϵ-transitions are not allowed
* All NFA's are not DFA	* All DFA's are NFA

1. Design a NFA accepting all strings ending with 0, 1



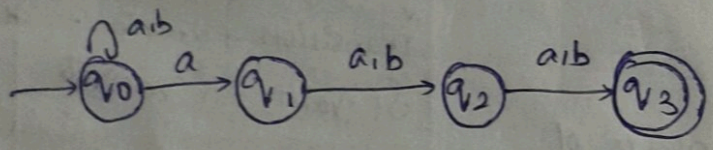


2. Accepts the string with aa



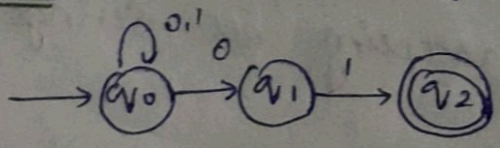
3. Design a NFA to accept the string that should have 11 is followed by 00

4. Design a NFA third symbol from right end is always a.



5. Solve the NFA that accepts all strings ends in 01 give the transition table and the extended transition function for the input string 0101

Step 1 Transition diagram



Step 2: Transition

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

Step 3:

String Acceptance by using transition function

0101

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

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

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

$\delta(q_1, 1) = q_2$

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

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

0101 → string Acceptance

$\vdash \delta(q_0, 0101)$

$\vdash \delta[(q_0, q_1), 101)$

$\vdash \delta((q_0, 1) \cup (q_1, 1), 01)$

$\vdash \delta((q_0, q_2), 01)$

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

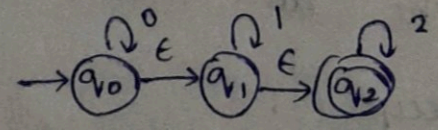
$\vdash (q_0 \Rightarrow q_2)$

$\therefore q_2$ is the final state so the given string 0101 is accepted.

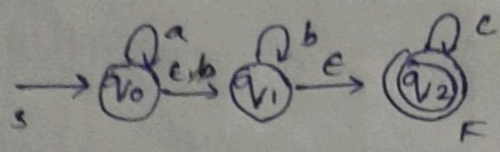
NFA - ϵ (NFA with Epsilon transitions)

The Epsilon transitions in NFA are given inorder to move from one state to another state without having any input symbol from the input set Σ.

Example:



1. Construct a NFA with epsilon which accepts a language consisting the strings of any number of a's followed by any number of b's followed by any number of c's.



2. Step:2 State Transition table

State / Input	a	b	c	e
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

Step:3 State Transition Function

$\vdash \delta(q_0,$

$\vdash \delta$

Step:3 String acceptance aabbcc

$\vdash \delta(q_0, aabbcc)$

$\vdash \delta(q_0, abbcc)$

$\vdash \delta(q_0, \epsilon bbcc)$
$\vdash \delta(q_1, bbcc)$
$\vdash \delta(q_1, bcc)$

$\vdash \delta(q_1, \epsilon cc)$
$\vdash \delta(q_2, cc)$
$\vdash \delta(q_2, c)$

$\vdash \delta(q_2)$

$\therefore q_2$ is the final state so the given string is accepted.

closure - ϵ

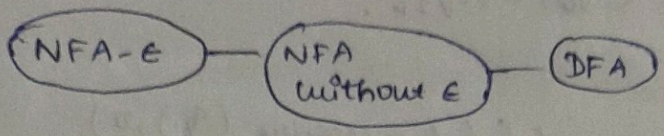
ϵ - closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ - closure $(q_1) = \{q_1, q_2\}$

ϵ - closure $(q_2) = \{q_2\}$

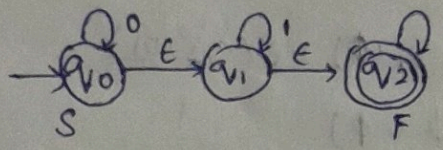
Equivalence of NFA & DFA

NFA conversion from with epsilon to without Epsilon



Convert the following NFA Epsilon to NFA without Epsilon

1.



Step1: find ϵ-closure of all states

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

ϵ - closure $(q_1) = \{q_1, q_2\}$

ϵ - closure $(q_2) = \{q_2\}$

Step2:

$\delta'(q_0, 0) = \epsilon$ - closure $(\hat{\delta}(q_0, 0))$

$= \epsilon$ - closure $(\delta(\epsilon$-closure $(q_0), 0)$

$= \epsilon$ - closure $(\delta(q_0, q_1, q_2), 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$

$= (q_0 \cup \phi \cup \phi)$

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

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

$= \epsilon$ - closure $(\delta(q_0, q_1, q_2), 1)$

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

$= \epsilon$ - closure $(\phi \cup q_1 \cup \phi)$

$= \epsilon$ - closure (q_1)

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

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

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

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

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

ϵ closure $(q_0) = \{q_2\}$.

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

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

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

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

ϵ closure $(q_1) = \{\phi\}$

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

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

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

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

ϵ-closure $(q_1) = \{q_1, q_2\}$

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

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

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

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

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

ϵ-closure $(q_1) = \{q_2\}$

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

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

$= \epsilon$ closure $(q_2, 0)$

ϵ-closure $(q_2) = \phi$

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

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

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

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

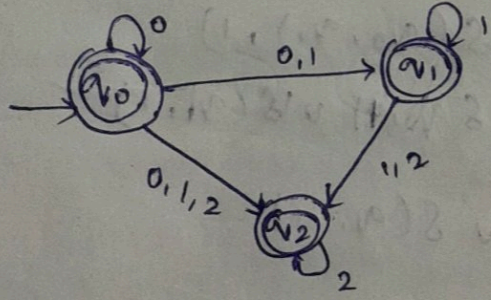
$= \epsilon\text{-closure}(q_2, 2)$

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

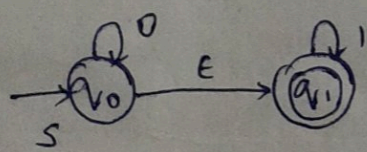
Step 3: State Transition Table for NFA

States \ Inputs	0	1	2
→ * q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
* q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
* q_2	ϕ	ϕ	$\{q_2\}$

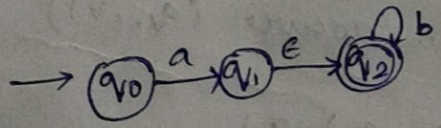
Step 4: Transition diagram or state diagram of NFA.

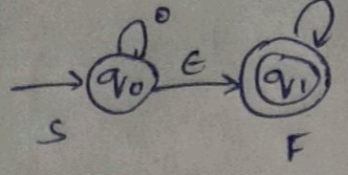


2.



3.



2. 

Step 1: Find ε-closure of all states

ε-closure $(q_0) = \{q_0, q_1\}$

ε-closure $(q_1) = \{q_1\}$

Step 2:

$\delta'(q_0, 0) = \epsilon$-closure $(\hat{\delta}(q_0, 0))$

$= \epsilon$-closure $(\delta(\epsilon\text{-closure}(q_0), 0)$

$= \epsilon$-closure $(\delta(q_0, q_1, 0)$

ε-closure $\delta(q_0, 0) \cup \delta(q_1, 0)$

& ε-closure $\{q_0 \cup \phi\}$

ε-closure $q_0 = \{q_0, \phi\}$

ε-closure $(q_0) = \{q_0, q_1\}$

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

$= \epsilon$-closure $(\delta(q_0, q_1), 1)$

$= \epsilon$-closure $(\delta q_0, q_1) \cup \delta(q_1, 1)$

ε-closure $\delta(q_0, 1) \cup \delta(q_1, 1)$

$= \phi \cup q_1$

ε-closure $(q_0) = \{\phi \cup q_1\}$

$= \{q_1\}$

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

$= \epsilon$-closure $(\delta(q_0, q_1), 0)$

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

$= q_0 \cup \phi$

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

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

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

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

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

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

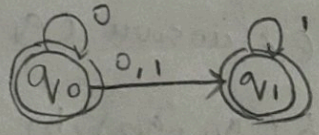
$\epsilon\text{-closure}\,\delta(q_1, 1)$

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

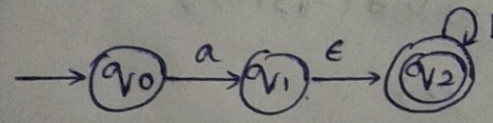
Step 3: State Transition table.

States \ Inputs	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	ϕ	$\{q_1\}$

Step 4: Transition diagram or state diagram



3.



Step 1: Find ϵ-closure of all states

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

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

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

Step 2:

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

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

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

ϵ-closure $(\phi \cup q_1 \cup \phi)$

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

$\delta'(q_0, b) = \epsilon$-closure $(\delta(\epsilon$-closure $(q_0, q_1, q_2), b)$

$= \epsilon$-closure $\delta((q_0, q_1, q_2), b)$

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

$= \epsilon$-closure $(\phi \cup \phi \cup q_2)$

ϵ-closure $(q_0) = \{q_2\}$

$\delta'(q_1, a) = \epsilon$-closure $(\delta(\epsilon$-closure $(q_1, q_2), a)$

$= \epsilon$-closure $(\delta(q_1, q_2), a)$

$= \epsilon$-closure $(\delta(q_1, a) \cup \delta(q_2, a))$

$= \epsilon$-closure $(q_1 \cup \phi)$

ϵ-closure $(q_1) = \{q_1, q_2\}$

$\delta'(q_1, b) = \epsilon$-closure $(\delta(\epsilon$-closure $(q_1, q_2), b)$

$= \epsilon$-closure $(\delta(q_1, q_2), b)$

$= \epsilon$-closure $((\delta(q_1, b) \cup \delta(q_2, b))$

$= \epsilon$-closure $(\phi \cup q_2)$

ϵ-closure $(q_1) = \{q_2\}$

$\delta'(q_2, a) = \epsilon$-closure $(\delta(\epsilon$-closure $(q_2), a)$

$= \epsilon$-closure $(\delta(q_2, a)$

$= \epsilon$-closure (ϕ)

ϵ-closure $(q_2) = \phi$

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

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

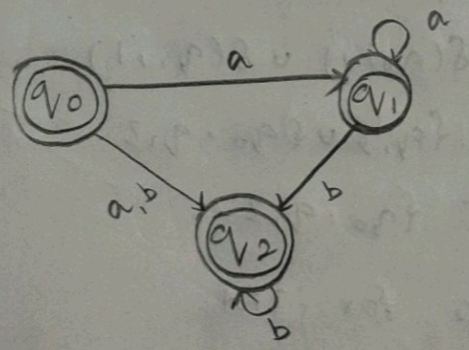
$= \epsilon\text{-closure}(q_2)$

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

Step 3: Transition table

States | Inputs	a	b
q_0	$\{q_1, q_2\}$	$\{q_2\}$
q_1	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	$\{q_2\}$

Step 4: Transition diagram or state diagram



1. Let $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$

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

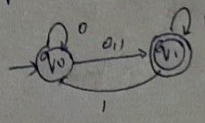
$\delta(q_0, 1) = \{q_1\}$

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

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

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

Step 1:



Step 2: States are

From the transition table we can compute that $[q_0], [q_1], [q_0, q_1]$

So we need to compute the transition from q_0, q_1.

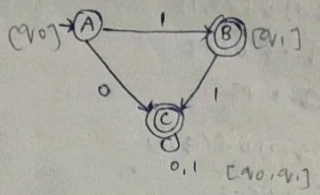
$$\delta([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

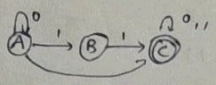
$$\delta([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$$
$$= \{q_1\} \cup \{q_0, q_1\}$$
$$= \{q_0, q_1\}$$

Step 3: Transition table for DFA

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

Step 4: DFA State transition diagram





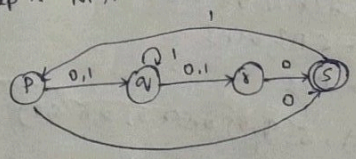
∴ Thus q_1 is Present in B, C so there are the final states.

2. NFA to DFA

	S	0	1
→	P	$\{q, s\}$	$\{q\}$
	q	$\{r\}$	$\{q, s\}$
	r	$\{s\}$	$\{p\}$
	s	-	$\{p\}$

Solution:

Step 1: NFA State Transition diagram



Step 2: From the transition table we can compute that there are $[q_0]$, $[q_1]$, $[q_2]$, $[q_3]$

So we need to compute the transition $[q, s]$, $[q, r]$

States are $[P]$, $[q]$, $[r]$, $[s]$, $[q, s]$, $[q, r]$

$$\delta([q, s], 0) = \delta(q, 0) \cup \delta(s, 0)$$
$$= \{r\} \cup \phi$$
$$= \{r\}$$

$\delta([q,s],1) = \delta(q,1) \cup \delta(s,1)$
$= \{q,r\} \cup \{p\}$
$= \{q,r,p\}$

$\delta([q,r],0) = \delta(q,0) \cup \delta(q,0)$
$= \{r\} \cup \{s\}$
$= \{r,s\}$

$\delta([q,r],1) = \delta(q,1) \cup \delta(r,1)$
$= \{q,r\} \cup \{p\}$
$= \{q,r,p\}$

$\delta([p,q,r],0) = \delta(p,0) \cup \delta(q,0) \cup \delta(r,0)$
$= \{q,s\} \cup \{r\} \cup \{s\}$
$= \{q,r,s\}$

$\delta([p,q,r],1) = \delta(p,1) \cup \delta(q,1) \cup (r,1)$
$= \{q\} \cup \{q,r\} \cup \{p\}$
$= \{p,q,r\}$

$\delta([r,s],0) = \delta(r,0) \cup \delta(s,0)$
$= \{s\} \cup \phi$
$= \{s\}$

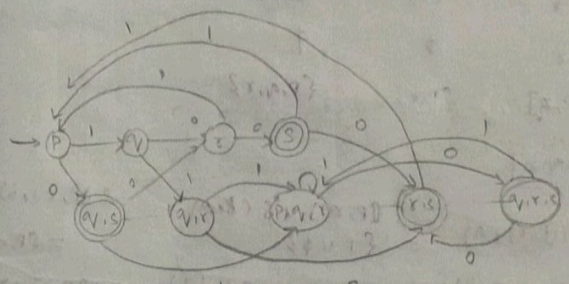
$\delta([r,s],1) = \delta(r,1) \cup \delta(s,1)$
$= \{p\} \cup \{p\}$
$= \{p\}$

$\delta([q,r,s],0) = \delta(q,0) \cup \delta(r,0) \cup \delta(s,0)$
$= \{r\} \cup \{s\} \cup \phi$
$= \{r,s\}$

$\delta([q,r,s],1) = \delta(q,1) \cup \delta(r,1) \cup \delta(s,1)$
$= \{q,r\} \cup \{p\} \cup \{p\}$
$= \{p,q,r\}$

Step 3: Transition table for DFA

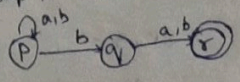
Input state	0	1
p	{q,s}	{q}
q	{r}	{q,r}
r	{s}	{p}
s	ϕ	{p}
[q,s]	{r}	{p,q,r}
[q,r]	{r,s}	{p,q,r}
[p,q,r]	{q,r,s}	{p,q,r}
[r,s]	{s}	{p}
[q,r,s]	{r,s}	{p,q,r}.



3.

	a	b
→ p	{p}	{p,q}
q	{r}	{r}
r	ϕ	ϕ

Solun: Step 1: NFA state transition table.



Step 2: [P] [q] [r] [P,q]

$\delta([P,q],a) = \delta(P,a) \cup \delta(q,a)$

$= [P] \cup [P,r,q]$

$= \{P,q\}$

$\delta([P,q],b) = \delta(P,b) \cup \delta(q,b)$

$= \{P,q\} \cup \{r\}$

$= \{P,q,r\}$

	a	b
P	{P}	{P,q}
q	{r}	{r}
r	ϕ	ϕ
[P,q]	{P,r}	{P,q,r}

[P,q,r]

$\delta([P,r],a) = \delta[P,a) \cup \delta(r,a)$

$= \{P \cup \phi\}$

$\delta([P,q],b) = \delta([P,b] \cup [q,b])$

$= \delta\{P,q\} \cup \{r\}$

$= \{P,q,r\}$

$\delta([P,r),b) = \delta(P,b) \cup \delta(r,b)$

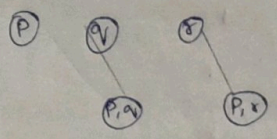
$= \{P,q\} \cup \{\phi\}$

$\delta([P,q],a) = \delta(P,a) \cup \delta(q,r)$

$\{P\} \cup \{r\}$

$\{P,r\}$

	a	b
P	{P}	{P,q}
q	{r}	{r}
r	ϕ	ϕ
[P,q]	{P,r}	{P,q,r}
[P,r]	{P}	{P,q}
[P,q,r]	{P,r}	{P,q,r}



$\delta([P,q,r],a)$

$= \delta[(P,a) \cup (q,a) \cup (r,a)]$

$= \{P\} \cup \{r\} \cup \{\phi\} = \{P,r\}$

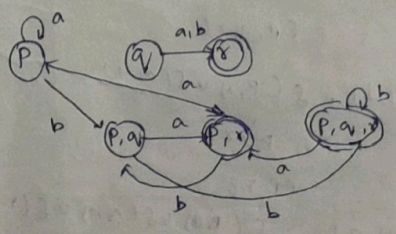
$\delta([P,q,r],b)$

$= \delta[(P,b) \cup (q,b) \cup (r,b)]$

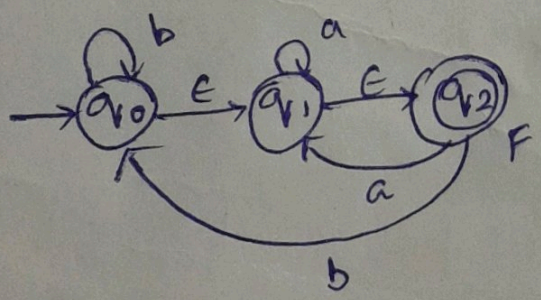
$= \{P,q\} \cup \{r\} \cup \{\phi\} = \{P,q,r\}$

Step 3: Transition table for DFA

Input state	a	b
P	{P}	{P,q}
q	{r}	{r}
r	ϕ	ϕ
[P,q]	{P,r}	{P,q,r}
[P,r]	{P}	{P,q}
[P,q,r]	{P,r}	{P,q,r}



1) Convert the following NFA-ϵ to DFA



Step1: Finding ϵ-closure for all states

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

ϵ- closure $(q_1) = \{q_1, q_2\}$

ϵ- closure $(q_2) = \{q_2\}$

Step 2: We have

States as (A)

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

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

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

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

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

$= \{q_1, q_2\} \rightarrow (B)$

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

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

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

$= \epsilon\text{-closure}(q_0) = \{q_0,q_1,q_2\} \rightarrow (A)$

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

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

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

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

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

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

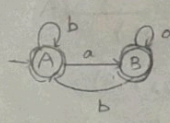
$= \epsilon\text{-closure}(\phi \cup q_0)$

$\epsilon\text{-closure}\{q_0\}$

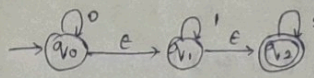
$= \{q_0, q_0, q_2\}$

Input	a	b
* A	B	A
* B	B	A

Transition diagram for DFA.



2.



Step 1: Finding ϵ-closure for all states

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

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

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

Step 2: We have to consider ϵ-closure of initial state as (A)

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

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

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

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

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

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

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

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

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

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

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

$= \{q_1, q_2\} \rightarrow (B)$

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

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

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

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

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

$= \{q_2\}$

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

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

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

$= \phi$

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

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

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

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

$= \{q_1,q_2\}$

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

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

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

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

$= \{q_2\}$

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

$= \{\phi\}$

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

$= \{\phi\}$

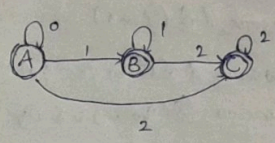
$\delta'(C,2) = \epsilon\text{-closure}\,\delta(q_2,2)$

$= \{q_2\}$

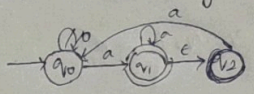
Transition table

State \ Input	0	1	2
A	A	B	C
B	ϕ	B	C
C	ϕ	ϕ	C

Transition diagram for DFA



1) Convert the following NFA-ϵ to DFA



Step 1: Finding ϵ-closure for all states

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

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

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

Step 1: Finding ϵ-closure for all states

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

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

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

Step 2: We have to consider ϵ-closure of initial state as (A)

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

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

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

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

$= \{q_1,q_2\} \rightarrow (B)$

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

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

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

$= \epsilon\text{-closure}\{q_1 \cup q_2\}$

$= \{q_1,q_0,q_2\} \rightarrow (C)$

$\{q_0$

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

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

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

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

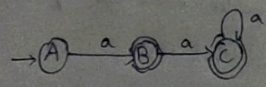
$= \epsilon\text{-closure}\{q_1 \cup q_0\}$

$= \{q_0, q_1, q_2\} \rightarrow (C)$

Transition table

State	Input a	B
→ A		
* B	C	
* C	C	

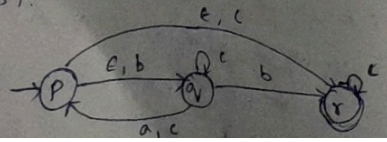
Transition diagram



22

States	I/P ϵ	a	b	c
P	$\{q,r\}$	ϕ	$\{q\}$	$\{r\}$
q	ϕ	$\{p\}$	$\{r\}$	$\{p,q\}$
r	ϕ	ϕ	ϕ	$\{r\}$

Step 1:



$\epsilon\text{-closure}(P) = \{P,q,r\}$

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

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

Step 2: We have to consider ϵ-closure of intial states as (A)

$\epsilon\text{-closure}(P) = \{P,q,r\} \rightarrow (A)$

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

$= \epsilon\text{-closure}(\delta(P,a) \cup \delta(q,a) \cup \delta(r,a))$

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

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

$= \{P,q,r\} \rightarrow (B)$

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

$= \epsilon\text{-closure}(\delta(P,b) \cup \delta(q,b) \cup \delta(r,b))$

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

$= \epsilon\text{-closure}\{q,r\}$

$= \{q,r\} \rightarrow (B)$

$\delta'(A,c) = \epsilon\text{-closure}(\delta(A,c))$

$= \epsilon\text{-closure}(\delta(P,c) \cup \delta(q,c) \cup \delta(r,c))$

$= \epsilon\text{-closure}(r \cup (P,q) \cup r)$

$= \epsilon\text{-closure}\{P,q,r\}$

$= \{P,q,r\}$

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

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

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

$= \epsilon\text{-closure}\{p \cup \phi\}$

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

$= \{P,q,r\}$

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

$= \epsilon\text{-closure}(\delta[q,r),b)$

$= \epsilon\text{-closure}(\delta(q,b) \cup \delta(r,b))$

$= \epsilon\text{-closure}\{r \cup \phi\}$

$= \{r\} \rightarrow ©$

$\delta'(B,C) = \epsilon\text{-closure}(\delta(B,C))$

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

$= \epsilon\text{-closure}(\delta(q,C) \cup \delta(r,C))$

$= \epsilon\text{-closure}\{p,q,r\}$

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

$= \phi$

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

$= \phi$

$\delta'(r,c) = \epsilon\text{-closure}(\delta(r,c))$

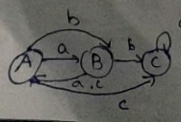
$= \epsilon\text{-closure}\{r\}$

$= \{r\}$

Transition table

State \ Input	a	b	c
A	A	B	A
B	A	C	A
C	ϕ	ϕ	C

Transition diagram:



Theorem

Introduction to formal Proof:

The formal Proof can be classified into two types.

1. Deductive Proof
2. Inductive proof

The deductive Proof consists of sequence of statements given with logical Reasoning in order to Proove the first or initial statement. The initial statement is called as hypothesis. Th

The inductive Proof is a Recursive kind of proof which consists of sequence of statements

Deductive

Various forms of Proofs:

1. Proofs about Sets
2. Proof by contradiction
3. Proof by counter example

1. Proofs about Sets:

The Set is a collection of elements or items

to proue the statements

$P \cup Q = Q \cup P$.

Proving L.H.S:

x is in $P \cup Q$

x is in P or x is in Q

x is in Q or x is in P

x is in $P \cup Q$.

R.H.S

x is in $Q \cup P$.

x in Q or x is in P

x is in P or x is in Q

x is in $Q \cup P$

∴ It is Proved.

2. Proofs by contradiction.

In this type of Proof the statement of the form

if A then B

We start with statement A is not true, thus By assigning or assuming false to A we try to get the conclusion of statement B.

When it becomes impossible to reach statement B we contradict ourself and accept that A is true.

For Example, PUQ = QUP

Proof:

PUQ = QUP

Initially we assume that

PUQ = QUP is not true, that is PUQ ≠ QUP.

Now consider that x is in Q or x is in P hence we say that x is in PUQ.

But also implies that x is in QUP according to definition of union. Hence the assumption we made initally is false

Thus PUQ is equal to QUP is proved.

Example: 1

Prove that $\sqrt{2}$ is rational number.

Proof:

We will assume that $\sqrt{2}$ is a rational number

$$\sqrt{2} = \frac{a}{b} \rightarrow (1)$$

where a and b are integers and $\frac{a}{b}$ is irreducible

Squaring on both sides

$$2 = \frac{a^2}{b^2}$$

$$2b^2 = a^2$$

If we substitute $a = 2k$

$$2b^2 = (2k)^2$$

$$\frac{2}{2}b^2 = \frac{4k^2}{2}$$

$$b^2 = 2k^2$$

This means b is an even number. This is contradiction to our assumption that $\frac{a}{b}$ is simplified to lowest term because a and b are even. So Square root of 2 $\sqrt{2}$ is not rational number.

3. proofs by counter Example:

Inorder to prove certain statements we need to see all possible conditions in which the statements remain true. There are some situations in which the statement cannot be true.

For Example consider

a = 2, b = 3 then a mod b ≠ b mod a

2 mod 3 ≠ 3 mod 2.

Thus the given pair is true for some integers

if a=b

a mod b = b mod a

Such Proof is true only at some Specific condition.

Inductive Proof:

Inductive Proofs are special Proofs based on some observations it is used to Prove Recursively defined objects. This type of proof is also called as Proof by Mathematical induction.

There are three steps:

Step 1: Basis
Step 2: Induction hypothesis
Step 3: Inductive step.

Basis:

In this step we assume the lowest possible value This is an initial step in the Proof of Mathematical induction for example we can Prove that the result is true for n=0 and n=1

Induction hypothesis:

In this step we assign value of n to some other value k. We will check whether the result is true for n=k. or not.

Inductive Step:

If n=k is true then we check the result is true for n=k+1 or not If we get the same result at n=k+1 then we can state that given Proof is true by Principle of Mathematical induction.

Example:

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

1) Basis of Induction

Assume n=1 then

$L.H.S = \sum_{i=0}^{n} i = 1$

$L.H.S = 1$

2) Induction hypothesis:

n=k

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

L.H.S

$0+1+2+3+\ldots+k = \frac{k(k+1)}{2} \rightarrow (1)$

3) Inductive step:

$$\sum_{i=0}^{k+1} = \frac{(k+1)(k+2)}{2} \qquad 0+1+2+3+\ldots+k+1 = \frac{(k+1)(k+2)}{2}$$

n = k+1

L.H.S

$= 0+1+2+3+\ldots+k+k+1$

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

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

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

∴ Hence Proved.

Kleen closure (or) Star closure (Σ^*) and Positive closure (Σ^+)

Kleen closure
(or)
Star closure (Σ^*)

* This is a set of strings of any length including the null string ϵ.

* $\Sigma = \{0,1\}$
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
$\Sigma^* = \{\epsilon, 0, 1, 00, 11, 01, 10, 000, 111, 010, \ldots\}$

Positive closure (Σ^+)

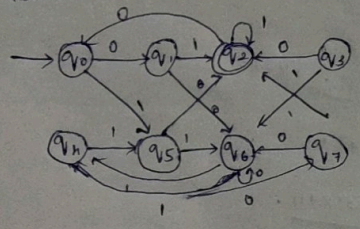
* It consists of all the string of any length except a null string ϵ.

$\Sigma = \{0,1\}$
$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$
$\Sigma^+ = \{0, 1, 00, 11, 01, 10, 111, 000 \ldots\}$

The Minimization of FA or FSM means reducing the number of states from the given FA. We get the FSM with redundant states after Minimizing the FA.

First we have to find out which states are equivalent then we can represent those states by one representative state.

1. Minimize the given DFA by using equivalence method



Step 1:

State transition table for DFA.

States \ Inputs	0	1
→ q_0	q_4	q_5
q_1	q_6	q_2
* q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

Step 2: Finding 0-equivalence

$\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$ $\{q_2\}$
final non-final

Finding 1-equivalence

$\{q_0\} = \{q_1, q_3, q_4, q_5, q_6, q_7\}$

$\{q_1, q_7\} = \{q_0, q_3, q_4, q_5, q_6\}$

$\{q_0, q_4, q_6\}$
$\{q_1, q_7\}$
$\{q_3, q_5\}$
$\{q_2\}$

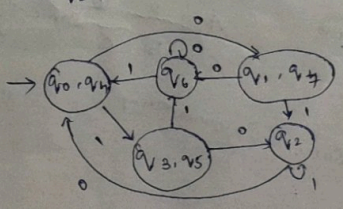
$\{q_0, q_4, q_6\}$
$\{q_1, q_7\}$
$\{q_3, q_5\}$
$\{q_2\}$

Finding 2-equivalance.

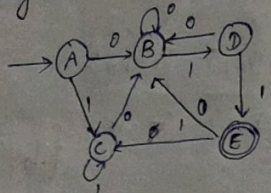
$\{q_0, q_4\}$
$\{q_6\}$
$\{q_1, q_7\}$
$\{q_3, q_5\}$
$\{q_2\}$

$\{q_0, q_4\}$,
$\{q_6\}$
$\{q_1, q_7\}$
$\{q_3, q_5\}$
$\{q_2\}$ //

Ip / states	0	1
$\{q_0, q_4\}$	$\{q_1, q_7\}$	$\{q_5, q_3\}$
$\{q_6\}$	$\{q_6\}$	$\{q_4, q_6\}$
$\{q_1, q_7\}$	$\{q_6\}$	$\{q_2\}$
$\{q_3, q_5\}$	$\{q_2\}$	$\{q_6\}$
$\{q_2\}$	$\{q_0, q_4\}$	$\{q_2\}$



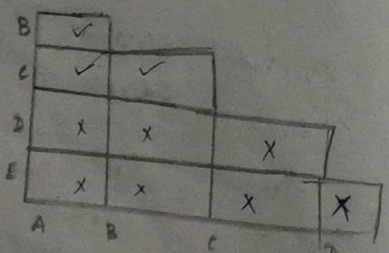
1. Convert the following DFA by using Table filling Method.



Step 1:
State Transition Table for DFA

States	Inputs 0	1
→ A	B	C
B	B	D
C	B	E
D	B	C
E	B	C

Step 2:
Table for mapping final and non-final state



Step 3:

(A, B)

	0	1
A	B	C
B	B	D

Same

(A, C)

	0	1
A	B	C
C	B	C

Same

(A, D)

	0	1
A	B	C
D	B	E

Same, different group

(B, D)

Same, different group

	0	1
B	B	D
D	B	E

(B, C)

	0	1
B	B	D
C	B	C

Same

(C, D)

	0	1
C	B	C
D	B	E

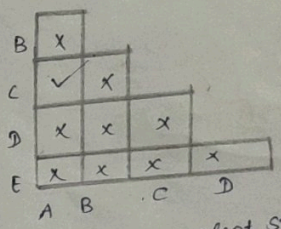
Same, different group.

Step 4:

(A, B)	0	1
→ A	B	C
B	B	D

	0	1
A	B	C
C	B	C

	0	1
B	B	D
C	B	C

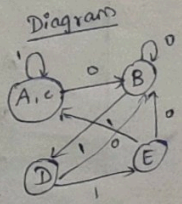


So, [A, C] are equivalent states. The new states are [A, C], B, D, E.

The Minimized DFA Transition table

State	IP 0	1
[A, C]	B	[A, C]
B	B	D
D	B	E
E	B	[A, C]

Diagram



By Equivalence Method.

Finding 0- equivalence

{A, B, C, D} {E}

Finding 1- equivalence

{A, B, C, D} {A, B, C}

{B, C, D}

{C, D}

{A, B, C}

{A}

{E}

Finding 2- equivalence

{A, C}

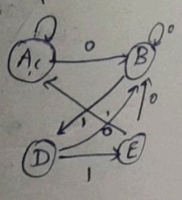
{B}

{D}

{E}

The Minimization of DFA Transition table

State / I/P	0	1
[A,C]	B	[A,C]
B	B	D
D	B	E
E	B	[A,C]



Unit- II

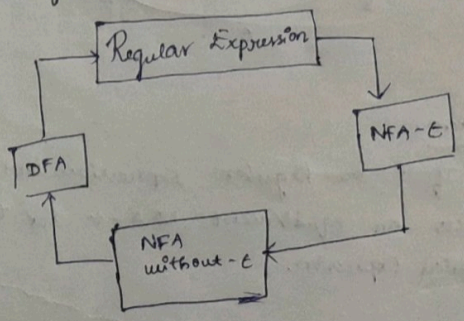
Regular Expressions and languages.

Regular languages are the languages that can be generated from one element languages by applying certain standard operations as a finite number of times. The language which is accepted or recognized by the finite Automata is called as the regular language.

The language accepted by finite Automata can be easily described by the simple expression is called as Regular expression.

1. union (+, |)
2. concatenation (.)
3. closure (*)

Equivalence of Finite Automata and Regular Expression.



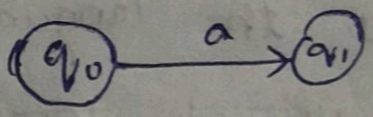
union(+, |):

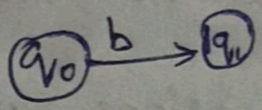
$r_1 = a$ — a - is a regular Exp|

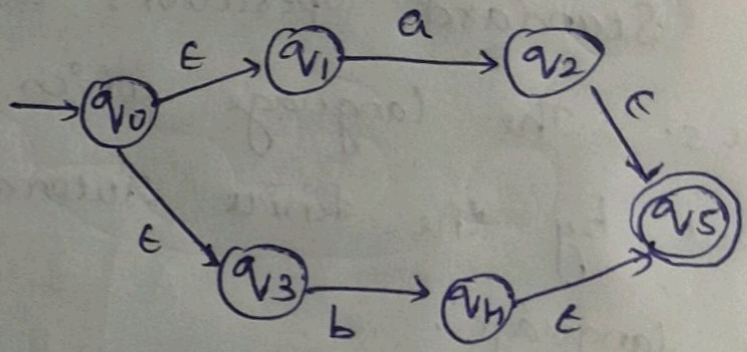
$r_2 = b$ — b - is a " " 2

Ex:

$r_1 + r_2 = (a+b)$ | $r_1 | r_2 = a/b$

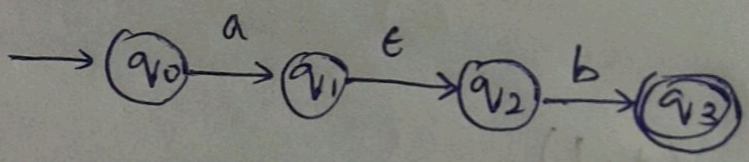
$r_1 \Rightarrow$ 

$r_2 \Rightarrow$ 

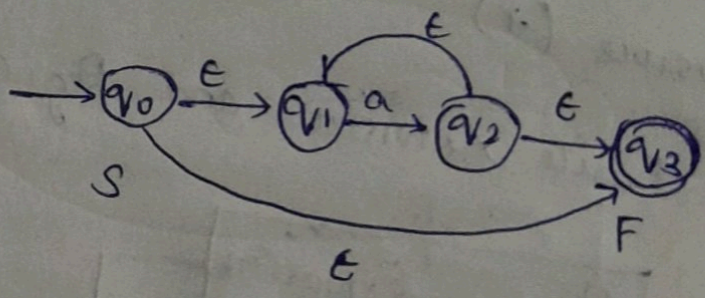
$r_1 + r_2 \Rightarrow$ 

Concatenation (·):

a·b



Closure (*): a*



Theorem:

If r be Regular Expression then there exists an equivalent NFA-ϵ for the given Regular Expression.