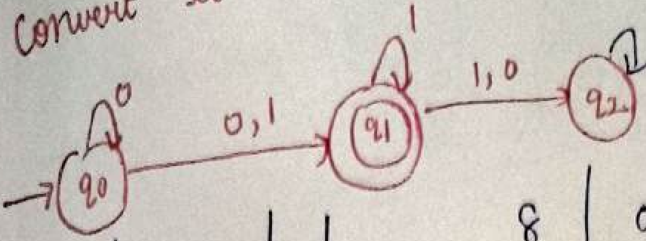
UNIT No-2

Finite Automata.

Topics :-

→ NDFA to DFA
→ Elimination of Λ- moves
→ Minimization of Automata

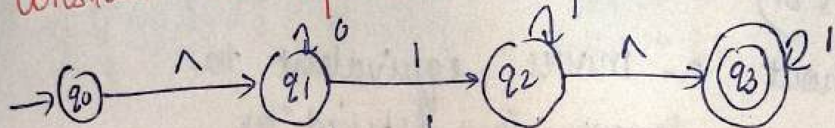
Convert it into deterministic machine



δ	0	1
→q0	{q0,q1}	{q1}
q1	q2	{q1, q2}
q2	φ	{q2}

δ	0	1
→[q0]	[q0 q1]	[q1]
[q1]	[q2]	[q1 q2]
[q0 q1]	[q0 q1 q2]	[q1 q2]
[q2]	φ	[q2]
[q1 q2]	[q2]	[q1, q2]
[q0 q1 q2]	[q0, q1, q2]	[q1, q2].

Q. Consider the following NDFA with ∧ moves. Construct an equivalent NDFA without ∧ moves



	0	1
→q0	{q1}	{q2, q3}
q1	{q1}	{q2, q3}.
q2	φ	{q2, q3}.
q3	φ	{q3}.

$q_2 \xrightarrow{\varepsilon^*} q_2 \xrightarrow{0} \phi \quad \varepsilon^*$

$q_3 \to \phi$

$q_2 \xrightarrow{\varepsilon^*} q_2 \xrightarrow{1} q_2 \xrightarrow{\varepsilon^*} q_2, q_3$

$q_3 \to q_3 \quad q_3$

$q_0 \xrightarrow{\varepsilon^*} q_0 \xrightarrow{0} \phi. \quad \varepsilon^*$

$q_1 \to q_1 \to \underline{q_1}$

$q_0 \xrightarrow{\varepsilon^*} q_0 \xrightarrow{1} \phi \quad \varepsilon^*$

$q_1 \to q_2 \to q_2, q_3$

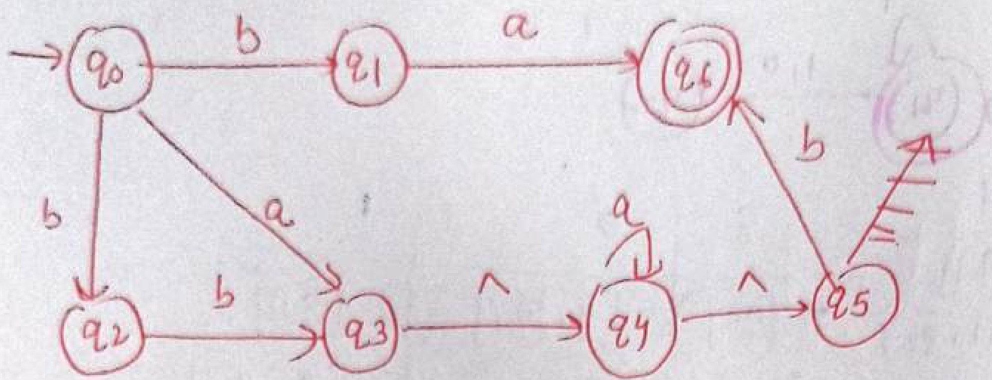
$q_1 \xrightarrow{\varepsilon^*} q_1 \xrightarrow{0} q_1 \xrightarrow{\varepsilon^*} q_1$

$q_1 \xrightarrow{\varepsilon^*} q_1 \xrightarrow{1} q_2 \xrightarrow{\varepsilon^*} q_2, q_3$

$q_3 \xrightarrow{\varepsilon^*} q_3 \xrightarrow{0} \phi. \quad \varepsilon^*$

$q_3 \xrightarrow{\varepsilon^*} q_3 \xrightarrow{1} q_3. \quad \varepsilon^*$

Q construct an NFA without Λ. moves



Sol $M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b\}, \delta, q_0, \{q_6\})$

Λ-closure $(q_0) = \{q_0\}$

Λ- closure $(q_1) = \{q_1\}$

Λ -closure $(q_2) = \{q_2\}$

Λ- closure $(q_3) = \{q_3, q_4, q_5\}$

$(q_4) = \{q_4, q_5\}$

$(q_5) = \{q_5\}$

$(q_6) = \{q_6\}$

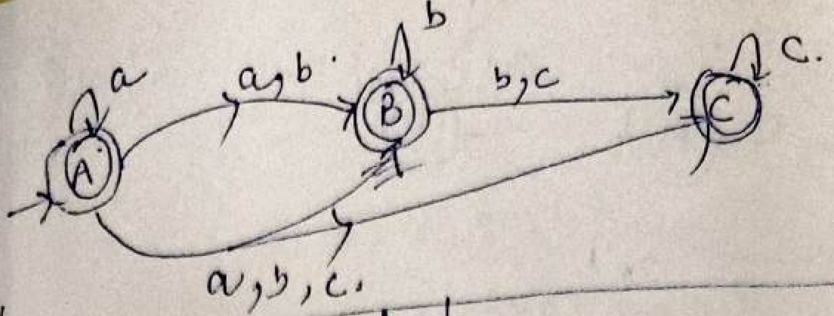
NDFA without Λ- moves equivalent to NDFA having Λ- moves is defined as

$M' = (\{\{q_0\}, \{q_1\}, \{q_2\}, \{q_3, q_4, q_5\}, \{q_4, q_5\}, \{q_5\}, \{q_6\}\}, \{a, b\}, \delta', q_0', F')$

where δ' is defined as

δ'

ε-closure of a state q is set of states reachable from q through ε- moves including itself



x

	a	b
$\rightarrow q_0$	$\{q_3, q_4, q_5\}$	$\{q_1\}, \{q_2\}$
q_1	$\{q_6\}$.	$\{\phi\}$
q_2	ϕ	$\{q_3, q_4, q_5\}$
$\{q_3, q_4, q_5\}$	$\{q_4, q_5\}$	$\{q_6\}$
$\{q_4, q_5\}$		
$\{q_5\}$		
q_6		

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

$$q_0 \quad \overset{\varepsilon^*}{q_0} \quad \overset{a}{q_3} - \overset{\varepsilon^*}{q_3, q_4, q_5}$$

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

$$\overset{\varepsilon^*}{q_0} \quad q_0 - \overset{b}{q_1} - \overset{\varepsilon^*}{q_1}$$
$$\quad \backslash\, q_2 - q_2$$

3) $\delta(\{q_1\}, a) = \quad q_1 \quad \overset{\varepsilon^*}{q_1} - \overset{a}{q_6} - \overset{\varepsilon^*}{q_6}$

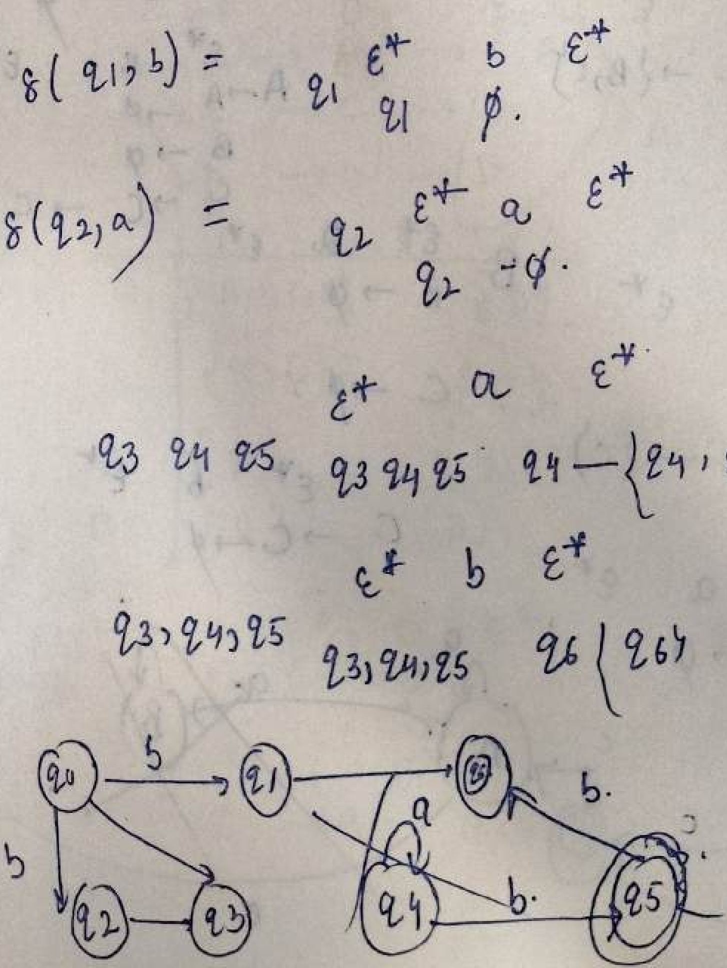
(4) $\delta(q_1, b) = \quad q_1 \quad \overset{\varepsilon^*}{q_1} \quad \overset{b}{\phi}.$

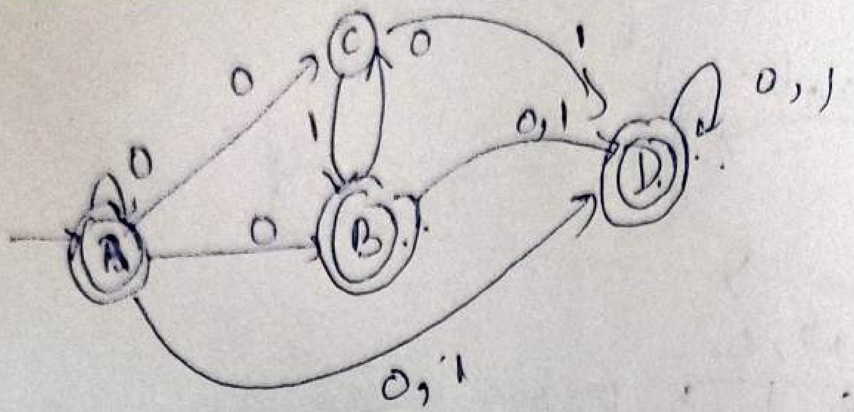
(5) $\delta(q_2, a) = \quad q_2 \quad \overset{\varepsilon^*}{q_2} \quad \overset{a}{-\phi}. \quad \overset{\varepsilon^*}{}$

$q_2 \quad \overset{\varepsilon^*}{q_2} \quad \overset{b}{q_3} \quad \overset{\varepsilon^*}{q_3,}$

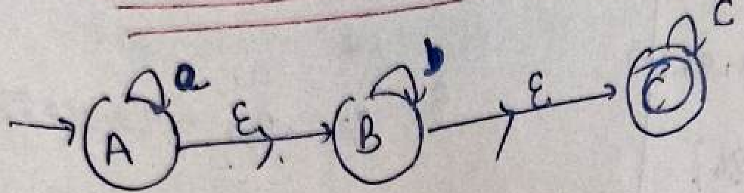
$$q_3\ q_4\ q_5 \quad \overset{\varepsilon^*}{q_3\, q_4\, q_5} \quad \overset{a}{q_4} - \overset{\varepsilon^*}{\{q_4, q_5\}}$$

$$q_3, q_4, q_5 \quad \overset{\varepsilon^*}{q_3, q_4, q_5} \quad \overset{b}{q_6} \quad \overset{\varepsilon^*}{\{q_6\}}$$





Convert ε-NFA to NFA



	a	b	c
A	{A,B,C}	{B,C}	{C}
B	ϕ	{B,C}	{C}
C	ϕ	{ }	{C}

$A \xrightarrow{\varepsilon^*} A \xrightarrow{a}$

$B \rightarrow \phi$

$C \rightarrow \phi$

$A \xrightarrow{\varepsilon^*} A \xrightarrow{b} \phi$ ε^*

$B \rightarrow B$.

$C \rightarrow \phi$

$A \xrightarrow{\varepsilon^*} A \xrightarrow{c} \phi$ ε^*

$B \rightarrow \phi$

$C \rightarrow C \rightarrow C$.

$B \xrightarrow{\varepsilon^*} B \xrightarrow{b} B \xrightarrow{\varepsilon^*} \{B, C\}$

$C \rightarrow \phi$

B ε^* a ε^*

$B \rightarrow \phi$

$C \rightarrow \phi$.

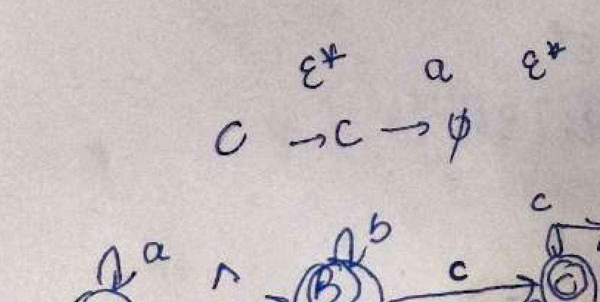
B ε^* c ε^*

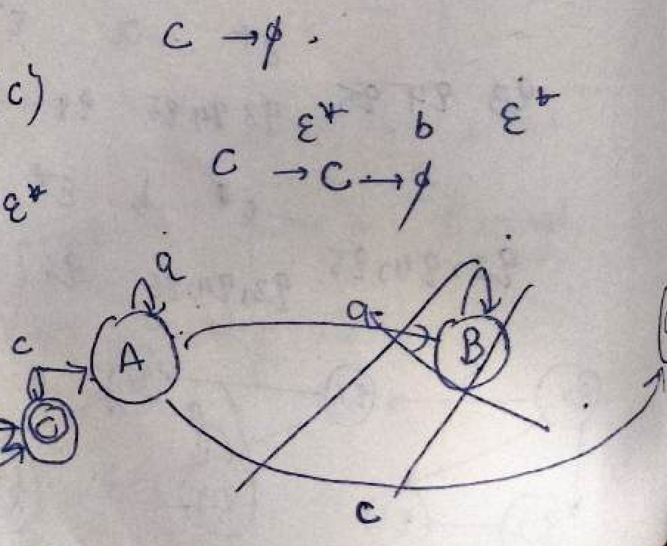
$B \rightarrow \phi$

$C \rightarrow C \rightarrow \{C\}$

$C \xrightarrow{\varepsilon^*} C \xrightarrow{b} \phi$ ε^*

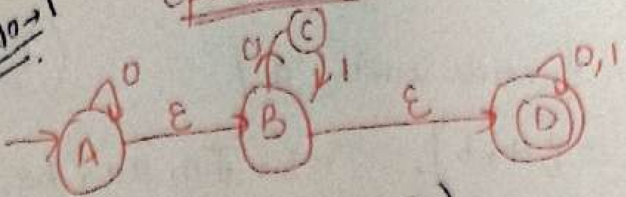
$C \xrightarrow{\varepsilon^*} C \xrightarrow{a} \phi$ ε^*





Example No→1

Epsilon NFA

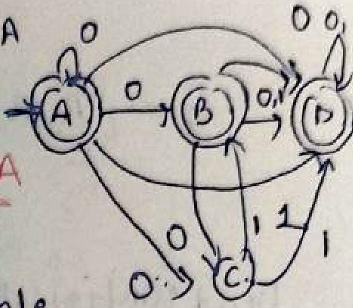


$(Q, \Sigma, \delta, q_0, F)$

$\delta : Q \times \Sigma \cup \{\epsilon\} \to 2^Q$

ε-closure (A): states from capital A on seeing A

$= \{A, B, D\}$



Convert Epsilon NFA to NFA

ε-NFA to NFA

go with state transition Table

$\delta(A, 0)$ A ϵ^* 0 ϵ^*

ε-closure δ(ε-closure of (A), 0))

$$
\begin{array}{lllll}
 & \epsilon^* & 0 & \epsilon^* \\
A \to & A \to & A & \to & A, B, D \\
 & B & \to C & \to & C \\
 & D & \to D & \to & D
\end{array}
$$

	0	1
A	{A, B, C, D}	{D}
B	{C, D}	{D}
C	{∅}	{B, D}
D	{D}	{D}

$$
\begin{array}{lllll}
 & \epsilon^* & 1 & \epsilon^* \\
A & A \to & \phi & \\
 & B \to & \phi & \\
 & D & \to D & \to D
\end{array}
$$

$$
\begin{array}{lllll}
 & \epsilon^* & 0 & \epsilon^* \\
B \to & B & \to C & \to C \\
 & D & \to D & \to D
\end{array}
$$

$$
\begin{array}{lllll}
 & \epsilon^* & 1 & \epsilon^* \\
B & \to B & \to \phi & \\
 & D & \to 1 & \to D
\end{array}
$$

$$
\begin{array}{llll}
 & \epsilon^* & 1 & \epsilon^* \\
D & \to D & \to D
\end{array}
$$

$$
\begin{array}{llll}
 & \epsilon^* & 0 & \epsilon^* \\
C & C & \to \phi &
\end{array}
$$

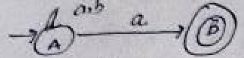
$$
\begin{array}{llll}
 & \epsilon^* & 1 & \epsilon^* \\
C & C. & \to B & \to B, D
\end{array}
$$

$$
\begin{array}{llll}
 & \epsilon^* & 0 & \epsilon^* \\
D & D & \to D & \to D
\end{array}
$$

(1)

NFA

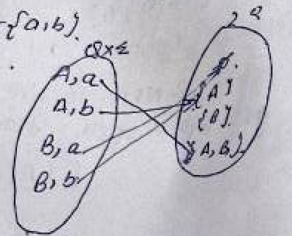
L = { ends with 'a' }
$\Sigma = \{a, b\}$.   L = {a, aa, ba, ...}.



$A \cdot \xrightarrow{a} A$
$\to B$.

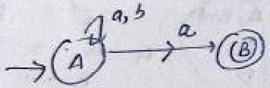
$\delta: Q \times \Sigma \to 2^Q$
any subset of states poss...

$Q = \{A, B\}$. $\Sigma = \{a, b\}$.
$Q \times \Sigma$.



Dead configuration
$A \xrightarrow{b} x$

L_1 = { ends with 'a' }

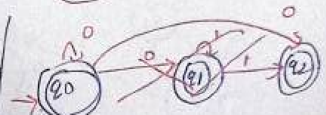


	a	b
→ A	[A,B]	{A}
(B)	φ	φ

	a	b
→ [A]	[A,B]	[A]
([A,B])	[A,B]	[A].

Q Imp

	0	1	2
→ q0	{q0, q1, q2}	{q1, q2}	{q2}
q1	φ	{q1, q2}	{q2}
(q2)	φ	φ	{q2}



	ε*	0	ε*
q0	q0	q0	→ q0, q1, q2
	q1	φ	
	q2	φ	

	ε*	1	ε*
q0	q0	φ	
	q1	q1	→ q1 q2
	q2	φ	

	ε*	2	ε*
q0	q0	→ φ	
	q1	→ φ	
	q2	→ q2	q2

	ε*	0	ε*
q1	q1	φ	
	q2	φ.	

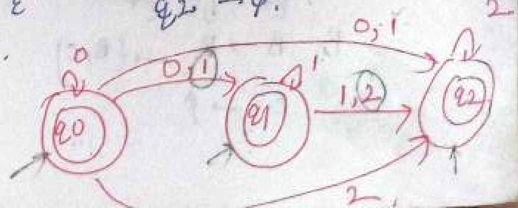
	ε*	1	ε*
q1	q1	q1	→ q1 q2
	q2	φ	

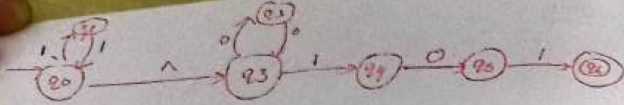
	ε*	2	ε*
q1	q1	→ φ	
	q2	q2	q2

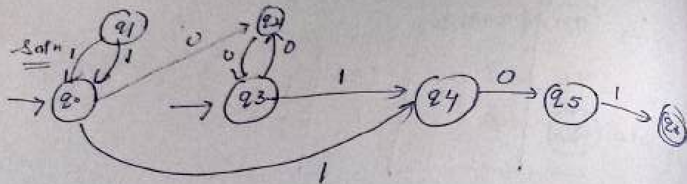
	ε*	0	ε*
q2	q2	→ φ	

	ε*	1	ε*
q2	q2	→ φ.	

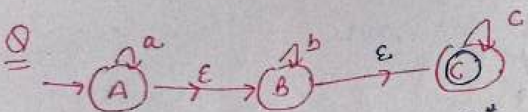
	ε*	2	ε*
q2	q2	—	q2.







Q



	a	b	c
A	{A,B,C}	{B,C}	{C}
B	ϕ	{B,C}	{C}
C	ϕ	ϕ	{C}

ε^* a ε^*
A: A → A → {A, B, C}
B φ
C φ

ε^* b ε^*
A: A → φ
B → B. → {B, C}
C → φ

ε^* c ε^*
A: A — A — φ
B — φ
C → C

ε^* a ε^*
B: B → φ
C → φ.

ε^* b ε^*
B: B → B → {B, C}
C → φ

ε^* c ε^*
B: B → φ
C → C → C

ε^* a ε^*
C: C → φ

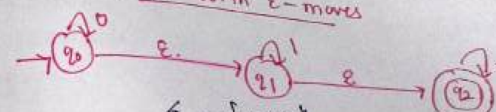
ε^* b ε^*
C: C → φ.

ε^* c ε^*
C: C → C → {C}

DFA ⇐ ε- NFA → NFA. → are equal in power

Q ~~FSA with ε-moves~~

FSA with ε-moves

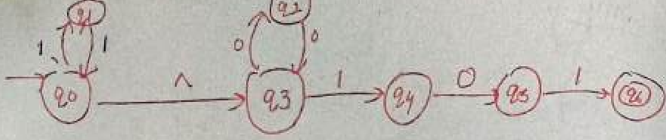


$\Sigma = \{0, 1, 2\}$

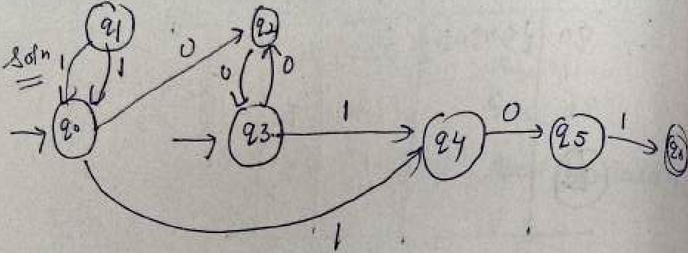
ε-closure of $q_0 = \{q_0, q_1, q_2\}$
ε-closure of $q_1 = \{q_1, q_2\}$
ε-closure of $q_2 = \{q_2\}$

ε-closure of a state q is set of states reachable from q through ε-moves including itself

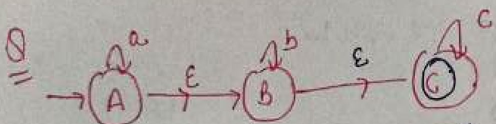
extend δ to $K \times \Sigma^*$.
$\hat{\delta}(q, \varepsilon)$ = ε-closure of q



Soln



Q



	a	b	c
A	{A,B,C}	{B,C}	{C}
B	ϕ	{B,C}	{C}
C	ϕ	ϕ	{C}

A $\quad \varepsilon^* \quad a \quad \varepsilon^*$
A → A = {A, B, C}
B ϕ
C ϕ

A $\quad \varepsilon^* \quad b \quad \varepsilon^*$
A → ϕ
B → B. → {B, C}
C → ϕ

A $\quad \varepsilon^* \quad c \quad \varepsilon^*$
A – A – ϕ
B – ϕ
C → C

B $\quad \varepsilon^* \quad a \quad \varepsilon^*$
B → ϕ
C → ϕ.

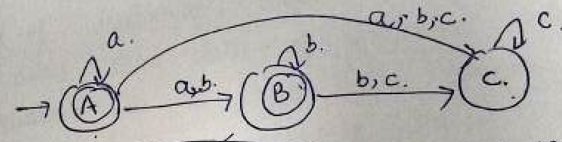
B $\quad \varepsilon^* \quad b \quad \varepsilon^*$
B → B → {B, C}
C → ϕ

B $\quad \varepsilon^* \quad c \quad \varepsilon^*$
B – B → ϕ
C → C → C

C $\quad \varepsilon^* \quad a \quad \varepsilon^*$
C → ϕ

C $\quad \varepsilon^* \quad b \quad \varepsilon^*$
C – C → ϕ.

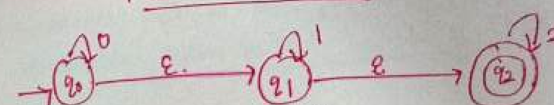
C $\quad \varepsilon^* \quad c \quad \varepsilon^*$
C → C → {C}



DFA ← ε-NFA → NFA. → are equal in power

Q ~~FSA with ε-moves~~

FSA with ε-moves



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

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

ε-closure of a state q is set of states reachable from q through ε-moves including itself.

extend δ to $K \times \Sigma^*$.

$\hat{\delta}(q, \varepsilon) =$ ε-closure of q

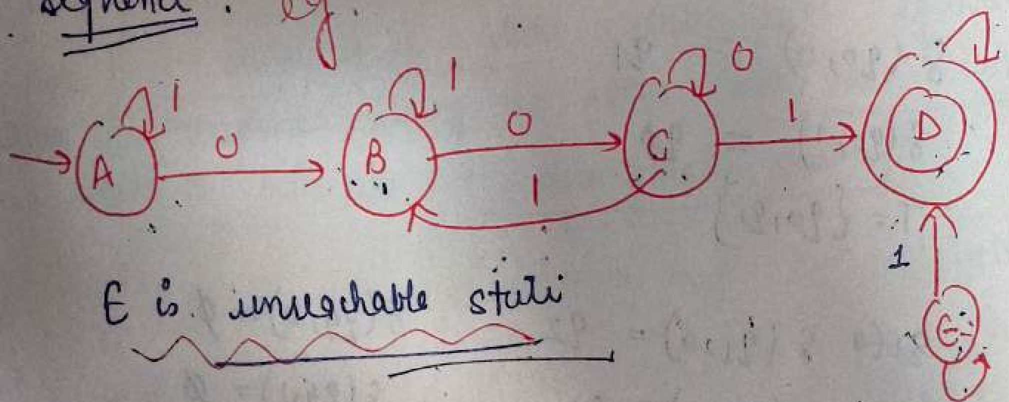
Minimization of ~~DFA~~ Finite Automata

Q. Minimization refers to construction of finite m/c with minimum no. of states, which is equivalent to the given finite machine. The no. of states of an automata has direct affect to the size of automata. When we minimize the automata we delete those states of automata whose presence or absence does not affect the language accepted by e.g. automata without affecting the language accepted by that automata.

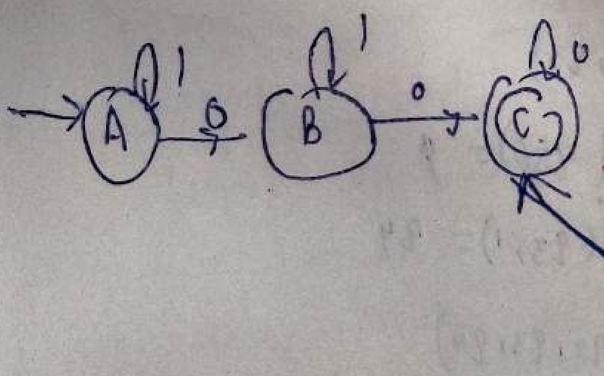
[1011] Not required 6 marks

Before this, we should know.

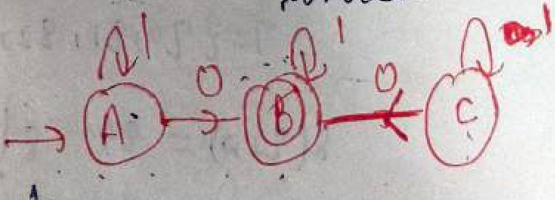
Unreachable states 2 marks means a state where the machine never reaches. It is a state of automata which are not reachable from initial state of DFA on any input sequence. eg.



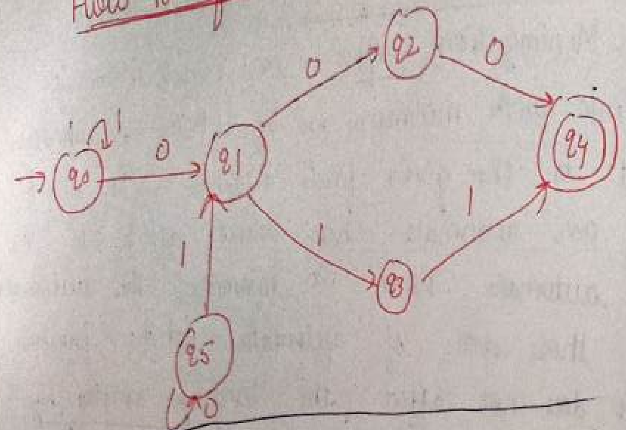
E is unreachable state



unreachable



How to find unreachable



Initial state $= q_0$

Final state $= q_4$

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

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

U = is unreachable state

Let T be a temporary set

Step1 :- Start from Initial state. Add q_0 to T

$T = \{q_0\}$

Step2 $\delta(q_0,0) = q_1$

$\delta(q_0,1) = q_0$

$T = \{q_0, q_1\}$

Step3 $\delta(q_1,0) = q_2$ $\quad \delta(q_4,0) = \phi$

$\delta(q_1,1) = q_3$ $\quad \delta(q_4,1) = \phi$

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

$\delta(q_2,0) = q_4$ $\quad \delta(q_3,0) = \phi$

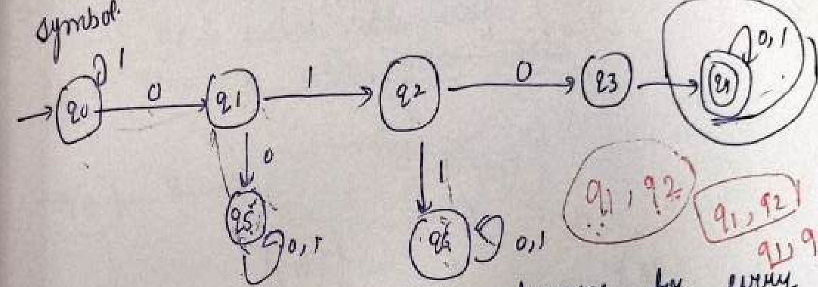
$\delta(q_2,1) = \phi$ $\quad \delta(q_3,1) = q_4$

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

$U = Q - T = \{q_5\}$

Dead states

A dead state is non accepting state (NF) which goes to itself on every possible input symbol.



q_5 & q_6 are dead states because for every input symbol transition ends on them.

Equivalent states

[q & q' are equivalent if final or both non final states]

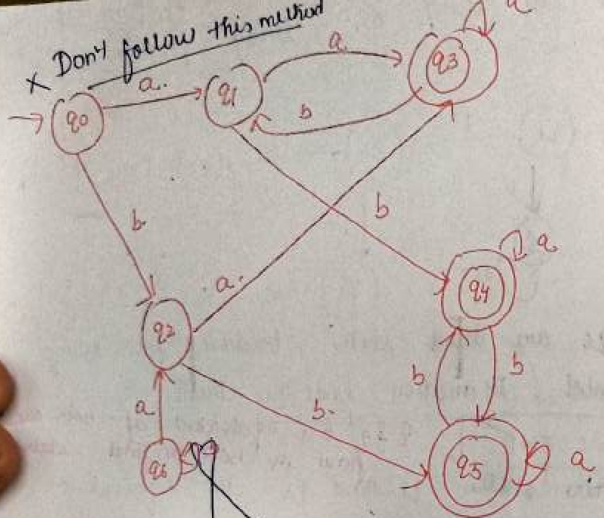
Two states q_1 and q_2 be equivalent if & only if for every input string w if $\delta(q_1,w)$ is in final state, then $\delta(q_2,w)$ must also be in final state & if $\delta(q_1,w)$ is in non-final state then $\delta(q_2,w)$ must also be in non-final state. ie if two states are equivalent then on accepting string w either both are in final state or both are in non final state.

$\delta(q_2,a) \rightarrow q_1$

$\delta(q_1,a)$

Minimization refers to detecting those states of given DFA whose presence or absence does not affect the language accepted by automaton. Hence, these states can be eliminated from automaton to save the memory and to ensure faster execution.

X Don't follow this method



(1) Detect unreachable states

(i) $T = \{q_0\}$

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

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

$\delta(q_1, a) = q_3$ $\quad \delta(q_1, b) = q_4$

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

$\delta(q_2, a) = q_3$ $\quad \delta(q_2, b) = q_5$

$T = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$\delta(q_3, a) = q_3$ $\quad \delta(q_4, a) = q_4$

$\delta(q_3, b) = q_1$ $\quad \delta(q_4, b) = q_5$

$\delta(q_5, a) = q_5$

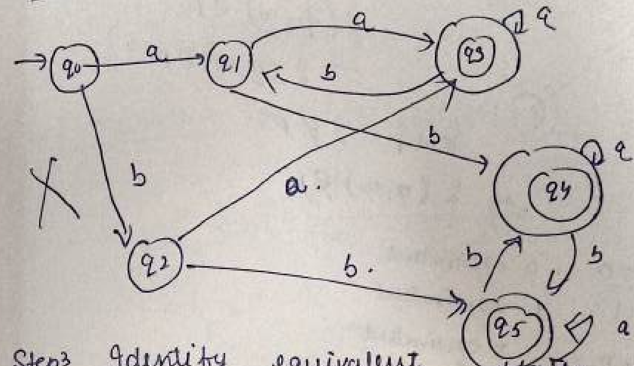
$\delta(q_5, b) = q_4$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\} - \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$= \{q_6\}$

q_6 is unreachable state

Step2 Eliminate unreachable states



Step3 Identify equivalent states & merge them

Group A – set of all final states

Group B set of all nonfinal states

$A = \{q_3, q_4, q_5\}$

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

For group A for input a

$\delta(q_3, a) = q_3$

$\delta(q_4, a) = q_4$

$\delta(q_5, a) = q_5.$

For group A for input b

$\delta(q_3, b) = q_1$

$\delta(q_4, b) = q_5$

$\delta(q_5, b) = q_4.$

Partition A group: $\{q_4, q_5\}$ $\quad \{q_3\}$ $\quad$ group B: $\{q_0, q_1, q_2\}$

For group B

$\delta(q_0, a) = q_1$ B.

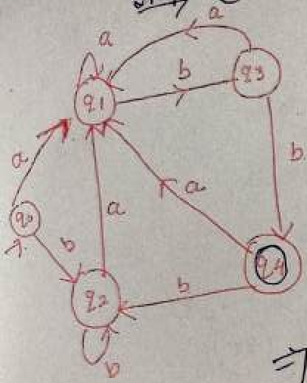
$\left.\begin{array}{l}\delta(q_1, a) = q_3 \\ \delta(q_2, a) = q_3\end{array}\right\} A_1$

$\delta(q_0, b) = q_2 \rightarrow$ belongs to B

$\left.\begin{array}{l}\delta(q_1, b) = q_4 \\ \delta(q_2, b) = q_5\end{array}\right\} A_2$

$\underbrace{\{q_0\}}_{B_1}$ $\quad \underbrace{\{q_1, q_2\}}_{B_2}$

Imp? (1) Example. Minimization of D...



(p, q) are equivalent if p on seeing any input string

$\delta(p, w) \in F \Rightarrow \delta(q, w) \in F$

$\delta(p, w) \notin F$

$\Rightarrow \delta(q, w) \notin F$

$|w| = 0$, 0 equivalent

$|w| = 1$, 1 equivalent

$|w| = n$ — n equivalent

(1) Identify unreachable states (all states reachable from initial state)

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
* q_4	q_1	q_2

0-equivalent state:-

i) Separate non-final states from final states

$[q_0, q_1, q_2, q_3]$ $[q_4]$

1 equivalent.

$[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$.

$q_0\ q_1$

$q_0\ q_2$

q_1, q_2

2-equivalent →

$[q_2\ q_3]$

$[q_3]$ $[q_4]$.

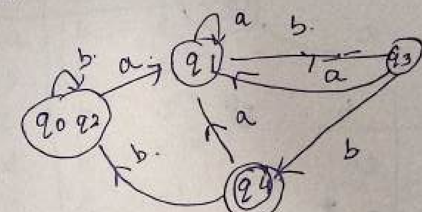
$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

$[q_0\ q_1]$

3-equivalent $[q_0\ q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

$q_0\ q_2$

$q_1\ q_2$.



(1) $[q_4]$ $[q_0, q_1, q_2, q_3]$

(2) 1equivalent $[q_4]$ $[q_3]$ $[q_0, q_1, q_2]$

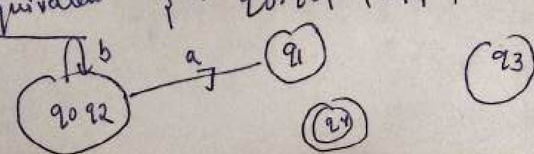
(3) 2equivalent $[q_4]$ $[q_3]$ $[q_0, q_2]$ $[q_1]$

(1) 0 eq. $[q_0, q_1, q_2, q_3]$ $[q_4]$.

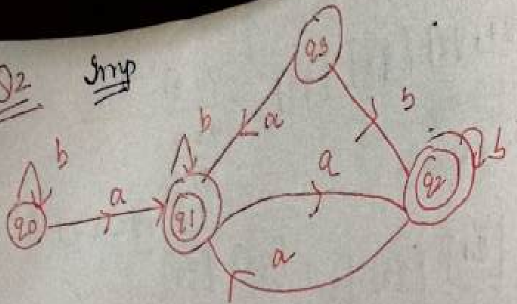
(2) 1-equivalent $\{q_0, q_1, q_2\}$ $\{q_3\}$ $\{q_4\}$

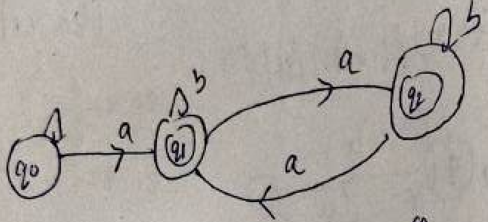
(3) 2 equ $\{q_0, q_2\}$ $\{q_1\}$ $\{q_3\}$ $\{q_4\}$.

(4) 3 equivalent $\{q_0, q_2\}$ $\{q_1\}$ $\{q_3\}$ $\{q_4\}$



Q2 Imp

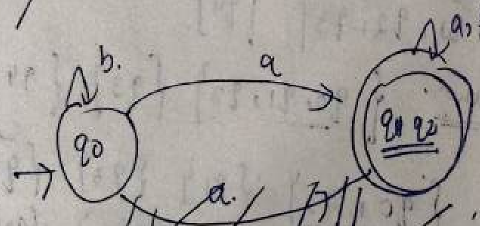




	a	b
→ q0	*q1	q0
* q1	*q2	q1 *
* q2	*q1	q2 *

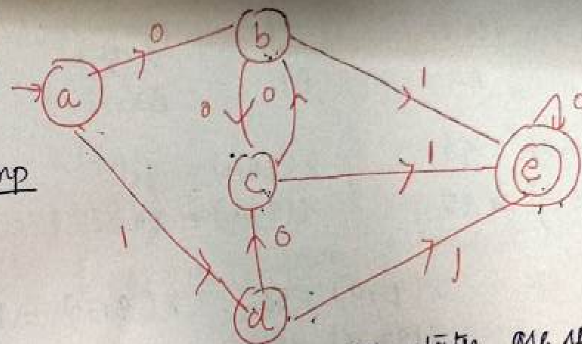
0-equivalent {q0} {q1, q2}

1equivalent {q0} {q1, q2}



Q3

Imp



All states are reachable.

	0	1
→ a	b	d.
b	c	e.*
c	b	e*
d	c..	e *
* e	*e	e*

	a	b
q0	q1*	q0
q1	q2*	q1*
q2	q1*	q2*

0q {q1, q2} {q0}

1equi {q0}

0-equivalence

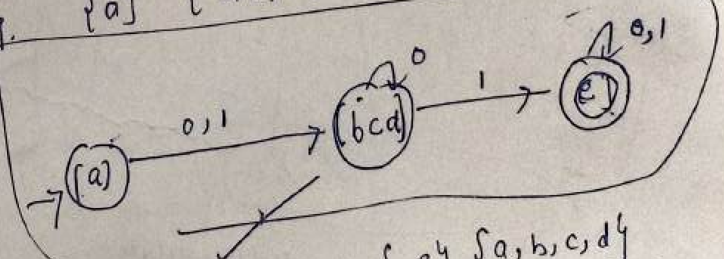
a, b

0-eq. {a, b, c, d} {e}.

b c

1-equivalence {a}, {b, c, d} {e}

2-eq. {a} {b, c, d} {e}



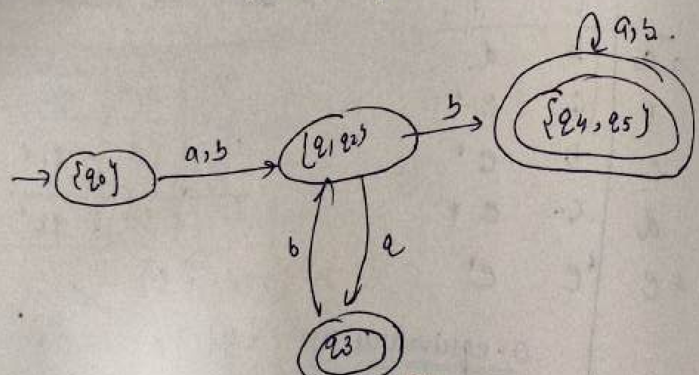
0equivalent {e} {a, b, c, d}

1equivalent → {e} {a} {b, c, d}

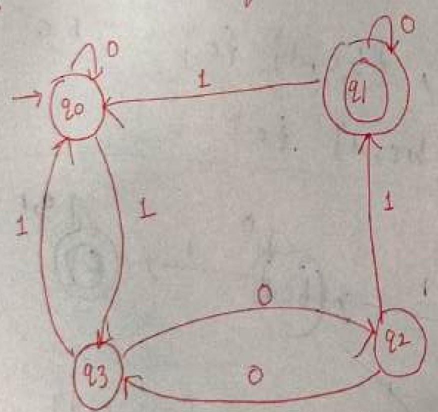
* $\boxed{q_3}$ A1 ✓ $\boxed{q_4, q_5}$ A2 $\boxed{q_0}$ B1 $\boxed{q_1, q_2}$ B2

A2. $\delta(q_4, 0) = q_4$ $\delta(q_4, b) = q_5$
$\delta(q_5, 0) = q_5$ $\delta(q_5, b) = q_4$.

For B2 $\delta(q_1, a) = q_3$ $\delta(q_1, b) = q_4$
$\delta(q_2, a) = q_3$ $\delta(q_2, b) = q_5$



Q: Minimize the following DFA



Step1: There is no unreachable state

Step2: Find equivalent states & merge them

$\boxed{q_1}$ Group A $\boxed{q_0, q_2, q_3}$ Group B

Check group B for both inputs

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

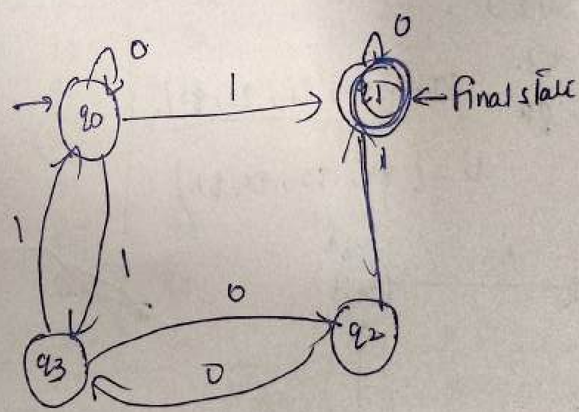
q0, q3 & q2 belong to group B.

q1 belongs to group A for input 1.

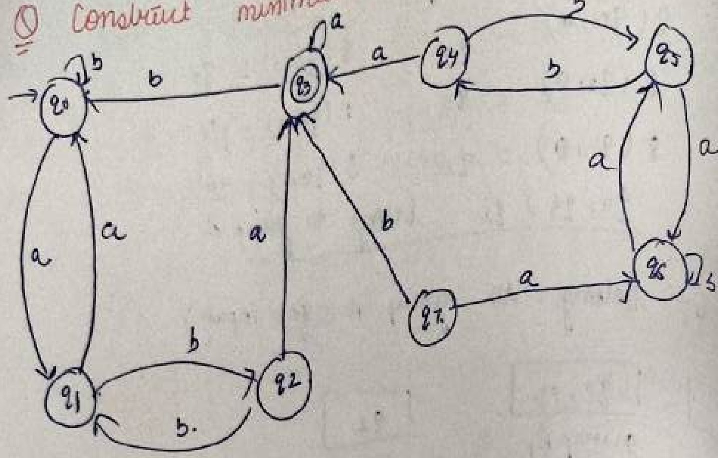
$\boxed{q_1}$ group A $\boxed{q_0, q_3}$ group B1 $\boxed{q_2}$ group B2.

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

$\boxed{q_1}$ $\boxed{q_0}$ $\boxed{q_3}$ $\boxed{q_2}$



Q Construct minimum state automaton



$\delta(q_0, a) = q_1$

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

$T = \{q_0, q_1\}$ $\delta(q_1, b) = q_2$

$T = \{q_0, q_1, q_2\}$ $\delta(q_2, a) = q_3$

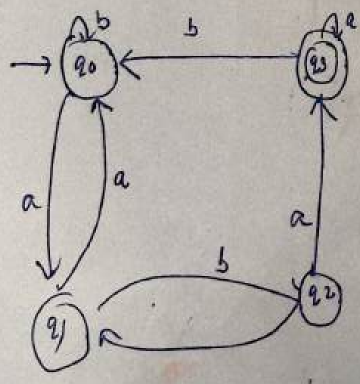
$\delta(q_2, b) = q_1$

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

$\delta(q_3, a) = q_3$

$\delta(q_3, b) = q_0$ $\pi = \{q_0, q_1, q_2, q_3\}$

$U = \{q_4, q_5, q_6, q_7\}$



(2) Find the accepting & non-accepting states

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

B A

(3) $\delta(q_0, a) = q_1$ ~~$\delta(q_3)$~~ $\delta(q_0, b) = q_0$

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

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

~~$\boxed{q_0, q_1}$~~ $\boxed{q_2}$ $\boxed{q_3}$ $\boxed{q_1}$ $\boxed{q_0}$

$\delta(q_0, 0)$ $\delta(q_0, b) = q_0$

$\delta(q_1, 0)$ $\delta(q_1, b) = q_2$

Second Method Construct transition Table

State/Σ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_2
q_2	$*q_3$	q_1
q_3	$*q_3$	q_0
q_4	$*q_3$	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	$*q_3$

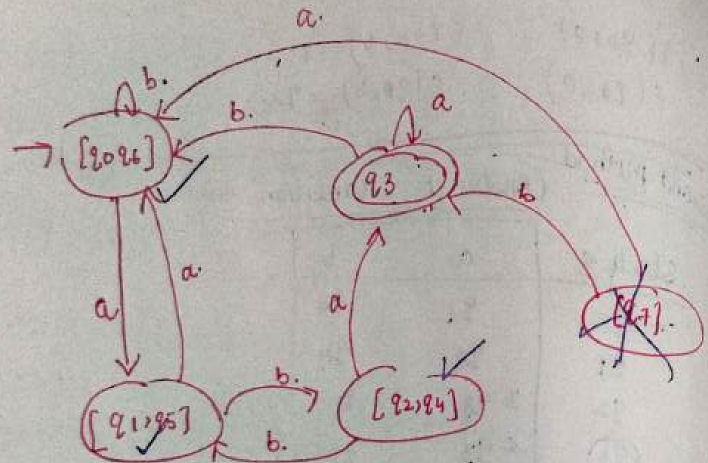
0-equivalent $\{q_3\}$ $\{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}$

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

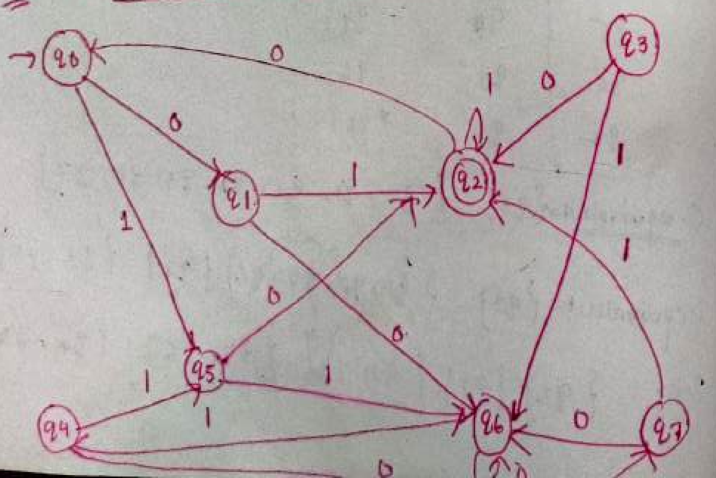
2-eq. $\{q_3\}$ $\{q_7\}$ $\{q_2, q_4\}$ $\{q_0, q_6\}$ $\{q_1, q_5\}$

$q_0 = \{q_0, q_6\}$ $F' = \{q_3\}$

State/ε	a	b
$[q_0, q_6]$	$[q_1, q_5]$	$[q_0, q_6]$
$[q_1, q_5]$	$[q_0, q_6]$	$[q_2, q_4]$
$[q_2, q_4]$	$[q_3]$	$[q_1, q_5]$
$[q_3]$	$[q_3]$	$[q_0, q_6]$
$[q_7]$	$[q_0, q_6]$	$[q_3]$.



Q2 Construct a minum automaton



State/ε	0	1
$\rightarrow q_0$	q_1.	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.

(1) 0- equivalent (Π_0)

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

(2) 1- equivalent (Π_1)

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

(3) 2- equivalent

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

(4) 3- equivalent $\{q_2\}$. $\{q_0, q_4\}$, $\{q_6\}$, $\{q_1, q_7\}$, $\{q_3, q_5\}$.

	0	1
$[q_0, q_4]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
$[q_2]$	$[q_0, q_4]$	$[q_2]$
$[q_3, q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

Q Construct minimum state

	a	b
→ q_0	q_1	q_2
q_1	q_4	q_3
q_2	q_4	q_3
q_3	q_5	q_6
q_4	q_7	q_6
q_5	q_3	q_6
q_6	q_6	q_6
q_7	q_4	q_6

$\Pi_0 = \{q_0, q_1, q_2, q_5, q_6, q_7\}\ \{q_3, q_4\}$

$\Pi_1 = \{q_0, q_6\}, \{q_1, q_2\}, \{q_5, q_7\}, \{q_3, q_4\}$

$\Pi_2 = \{q_0\}, \{q_6\}, \{q_1, q_2\}, \{q_5, q_7\}, \{q_3, q_4\}.$

State	a	b
$[q_0]$	$[q_1]$	q_2
$[q_1, q_2]$	$[q_3, q_4]$	$[q_3, q_4]$
$[q_3, q_4]$	$[q_5, q_7]$	$[q_6]$
$[q_5, q_7]$	$[q_3, q_4]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_6]$

Q

	a	b
→ q_1	q_2	q_6 ✓
q_2	q_1	q_3
q_3	q_4	q_2 ✓
→ q_4	q_4	q_1
q_5	q_4	q_6 ✓
q_6	q_7	q_5
q_7	q_6	q_7
q_8	q_7	q_4

$\Pi_0 = \{q_1, q_2, q_3, q_5, q_6, q_7, q_8\}\ \{q_4\}$

$\Pi_1 = \{q_1, q_2, q_6, q_7\}\ \{\cancel{q_3, q_4}\ \{q_3, q_5\}, \{q_4\}$

$\{q_8\}$

$\Pi_2 = \{q_8\}, \{q_4\}, \{q_3, q_5\}, \{q_1, q_7\}, \{q_2, q_6\},$

$\Pi_3 = \{q_8\}, \{q_4\}\ \{q_3, q_5\}\ \{q_1, q_7\}, \{q_2, q_6\}$

$q_0' = [q_1, q_7]$

$F = [q_4]$

	a	b
$[q_1, q_7]$	$[q_2, q_6]$	$[q_1, q_7]$
$[q_2, q_6]$	$[q_1, q_7]$	$[q_3, q_5]$
$[q_3, q_5]$	$[q_4]$	$[q_2, q_6]$
$[q_4]$	$[q_4]$	$[q_1, q_7]$
$[q_8]$	$[q_1, q_7]$	$[q_4]$

Finite Automata with O/P

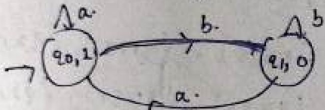
Moore Machine (Deterministic) — Mealy M/C

$(Q, \Sigma, \delta, q_0, \Delta, \lambda)$ — Δ: delta, λ: Lemda.

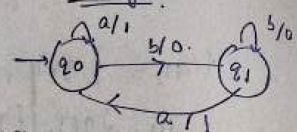
5

$Q \rightarrow$ set of finite states
$\Sigma \rightarrow$ I/P alphabet
$\delta \rightarrow$ Transition function
$Q \times \Sigma \rightarrow Q$.
$q_0 \rightarrow$ Initial state
$\Delta \rightarrow$ Output alphabet
$\lambda \rightarrow$ O/P function

$(Q, \Sigma, \delta, q_0, \Delta, \lambda)$
$\lambda: Q \rightarrow \Delta$
$\lambda: Q \times \Sigma \rightarrow \Delta$



Meally.



$\lambda: Q \rightarrow \Delta$. (for every state O/P is associated)

Moore M/C
$q_0 \rightarrow 1$
$q_1 \rightarrow 0$

$\lambda: Q \times \Sigma \rightarrow \Delta$
$(q_0, a) \rightarrow 1$
$(q_0, b) \rightarrow 0$
$(q_1, b) \rightarrow 0$
$(q_1, a) \rightarrow 1$

Moore
One Output

a b
$q_0 - q_0 - q_1$
1 1 0

a b
$q_0 \xrightarrow{} q_0 — q_1$
1 0

$(n+1)$ output are formed for string of length

n bit input
n bit output

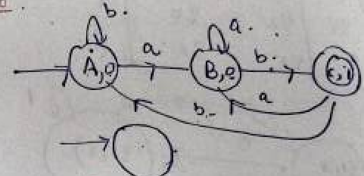
Construct a moore m/c that takes set of all strings over {a,b} as I/P and prints '1' as O/P for every occurrence of 'ab' as substring.

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

ab
1.

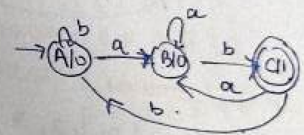
ababab.
1 1

abbbab
1 1



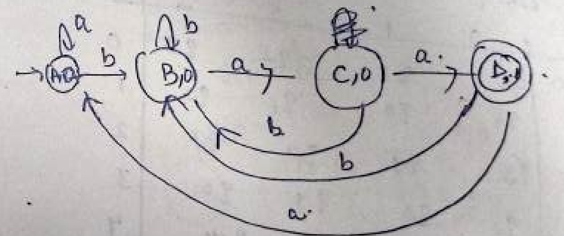
a b.
A → B. → C.
0 0 1

ab ab
A → B → C → B → C.
0 0 1 0 1



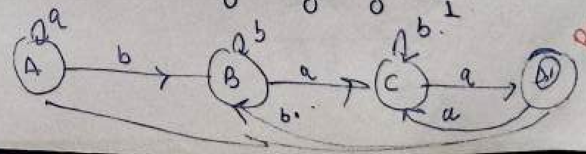
Q 'baa' as string.

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



b a a
A → B → C → D
0 0 0 1

b a a b a a
A → B → C → D → B → C → D
0 0 0 1 0 0 1

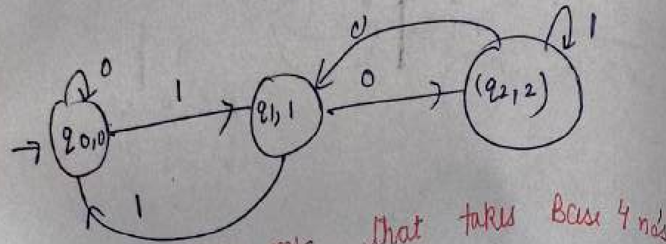


Rapinder

Q Construct a moore m/c that takes binary no. as Input & produce residue modulo '3' as O/P

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

	0	1	Δ
$\to q_0$	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2.



Q. Construct a moore m/c that takes Base 4 no's as I/P / produces residue module 5 as O/P →.

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

$\Delta = \{0,1,2,3,4\}$

	Inputs 0	1	2	3	Δ
$\to q_0$	q_0	q_1	q_2	q_3	0
q_1	q_4	q_0	q_1	q_2	1
q_2	q_3	q_4	q_0	q_1	2
q_3	q_2	q_3	q_4	q_0	3
q_4	q_1	q_2	q_3	q_4	4

IT/D/08

Q Construct a mealy m/c that takes binary number as I/P and produces 2's complement of that number as O/P. Assume the string is read LSB to MSB and end carry is discarded.

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

1011

0100

11100

1's 00011

2's + 1

00100

1011

$q_0 \xrightarrow[0]{1} q_0 \xrightarrow[1]{0} q_0 \xrightarrow[0]{1} q_0 \xrightarrow[0]{1} q_0$

MSB. LSB

1100

0011

+1

0100.

11101100

00010011

+1

00010100

1011

1's 0100

2's 0100

+1

0101

1011

0100

+1

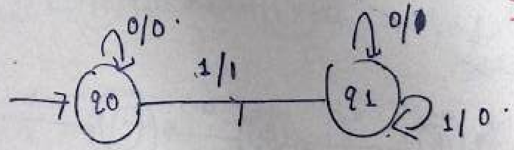
0101

1100

0011

+1

0100



eg 1011

0100

+1

01.01

1011

1100 1100

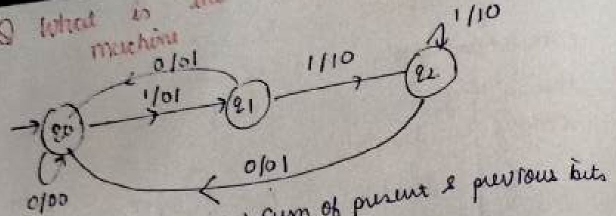
1's 0011 $q_0 \to q_1$

2's +1 1 0011

0100

$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1}$

Q What is the output produced by this machine



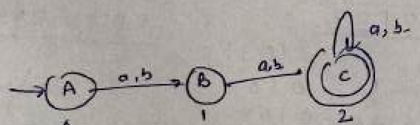
(c) Sum of present & previous bits
(d) NOT.

11 → 01.
10 → 00.

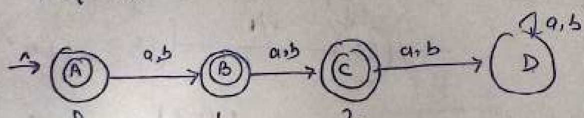
0
01

1 1 0 1.
$q_0 \to q_1 \to q_2 \to q_0 \to q_1$
01 10 . 01 . 01

1 1 0 1.
$q_0 \to q_1$ –



|w| ≤ 2
L = { ∧, a, b, aa, bb, ab, ba }

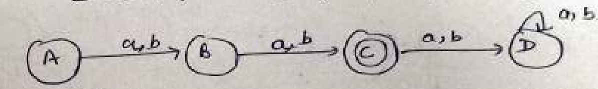


Tutorials

(1)

Q1. Construct a DFA that accepts set of all strings over {a, b} of length 2

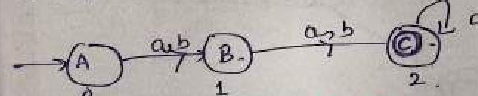
Σ = {a, b}   L = {aa, ab, ba, bb}



A finite automata must accept all strings which are in language & reject all which are not in language.

DFA w ∈ {a, b} |w| ≥ 2   (at least 2)

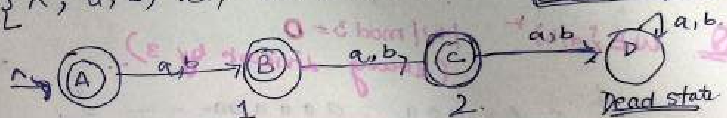
L = { aa, ab, ba, bb, aaa – bbb– }



2 or more

Q3. DFA w ∈ {a, b} |w| ≤ 2   (at most 2).

L = { ∧, a, b, aa, bb, ab, ba } → Finite Language



Q4

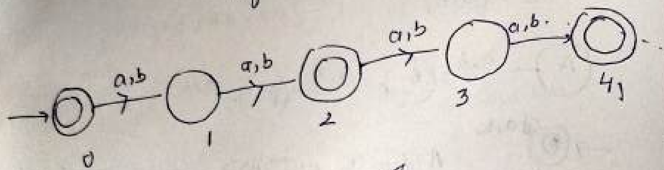
|w| = n → (n+2)	|w| ≤ n (n+2)
|w| ≥ n (n+1)	

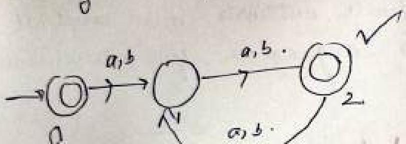
gate.

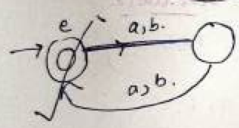
Q. $w \in \{a,b\}$, $|w| \bmod 2 = 0$

$L = \{ \Lambda, aa, bb, ab, ba, aaaa, bbbb ---- \}$

infinite

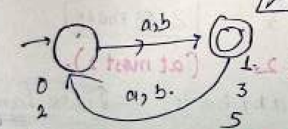






odd strings

$|w| \bmod 2 = 1$



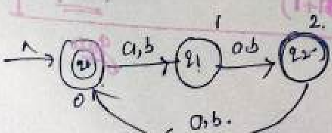
Q. $w \in \{a,b\}^*$ $|w| \bmod 3 = 0$

(string divisible by 3).

3)14(4
12
2

$L = \{ \Lambda, aaa, aaaaaa ---- \}$
aab
bbb

0,1,2 possible remainders when a no. divided by 3.

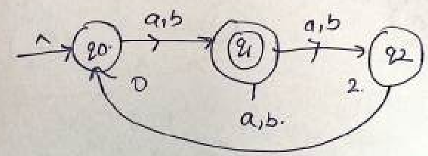


$|w| \equiv 1 \bmod 3$.

$\boxed{|w| \bmod 3 = 1}$

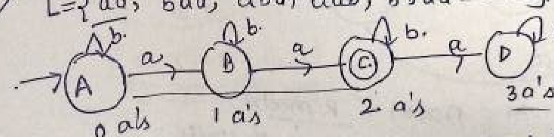
$|w| \bmod n = 0$

n no. of states



Q. Minimal DFA accepting $w \in \{a,b\}^*$ where $n_a(w) = 2$.

$L = \{aa, baa, aba, aab, bbaa ---\}$.

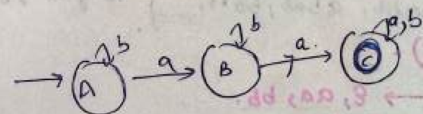


(No path to final state)

Deadstate: There is no path to final state

$n_a(w) \geq 2$

$L = \{ aa, baa, aba, aaba, ----$



$n_a(w) \leq 2$.

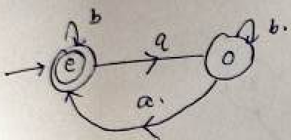
Q strings of a's is even.

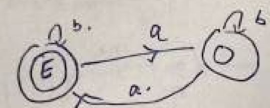
$w = \{a, b\}^*$

$n_a(w) \bmod 2 = 0.$

$n_a(w) \cong 0 \bmod 2.$

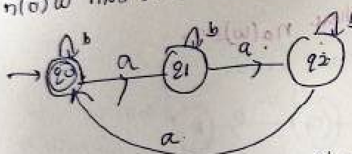
$L = \{\Lambda$





no of a's is odd

$n(a)w \bmod 3 = 0$ multiple of 3.



no. of a's in w $\quad n_a(w) \cong \dfrac{k \bmod n}{n \cdot \text{no. of states}}$

Minimal DFA $w \in \{a, b\}^*$

$n_a(w) \cong 0 \bmod 2.$

& b

$n_b(w) \cong 0 \bmod 2.$

every even lengths string does not contain even no of a's & b's.

$L = \{\Lambda, aa, bb, aabb, abab, baab, \ldots\}.$

$n_a(w)$	$n_b(w)$	
even	even	→ ε, aa, bb.
e	o	→ aab.
o	e	— aaabb
o	o.	— ab.

Even even ε, aabb

even odd

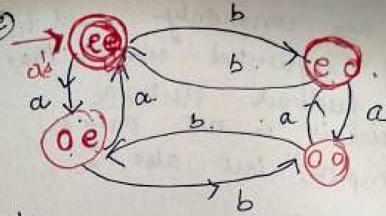
odd even

odd odd

ae

a.b. be

a's



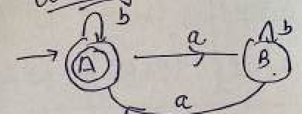
ab

a, b.

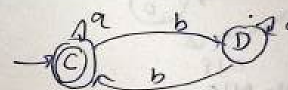
	a	b.
AC.	[BC]	[A D]
BC	[AC]	[B.D].
A D.	[B D].	[AC].
B D	[AD].	[B.C].

Q $w \in (a, b)^*$ $\quad n_a(w)$ = even $\quad n_b(w)$ = even

Counting A's

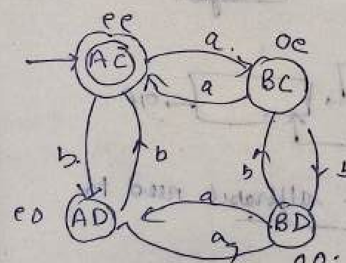


counting B's



$\{A, B\} \times \{C, D\}. = \{AC, AD, BC, BD\}$

Even even every even



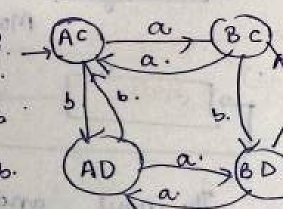
A →a B. / C. C.

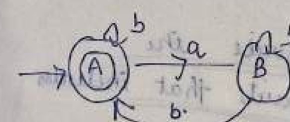
AC → b

A b. / C b

ab. b's / e or e

{ee}{eo, oe}

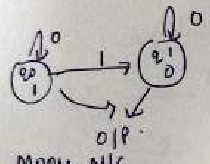




Mealy and Moore Machines are finite automata with output.

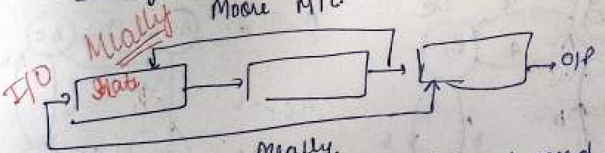
These m/c are commonly used to describe the action of sequential circuits that involve flip-flops and feedback electronic devices for which circuit is not only a fx of instantaneous inputs, but also a fx of previous state of system

Transducer I/O



Moore M/C

Moore M/C



The input and output alphabet need to be same in DFA but in moore & mealy they can be different

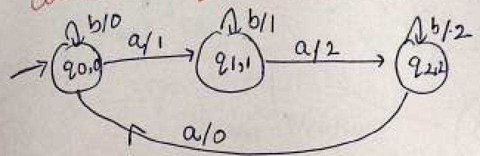
Uses They are used to describe the behavior of sequential circuits that includes flip-flops

①

Conversions

Moore & Mealy

count no. of a's % 3

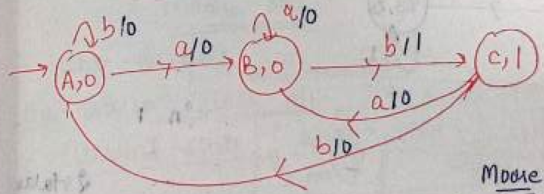


Moore M/C

	a	b	Δ
→ q0	(q1,1)	(q0,0)	0
q1	(q2,2)	(q1,1)	1
q2	(q0,0)	(q2,2)	2

Mealy M/C.

Moore to Mealy



Moore

	a	b.	Δ.
A	B	A	0
B	B	C	0
C	B	A	1.

Mealy

	a	b.
A	(B,0)	(A,0)
B	(B,0)	(C,1)
C	(B,0)	(A,0)

a b.

A → B → C

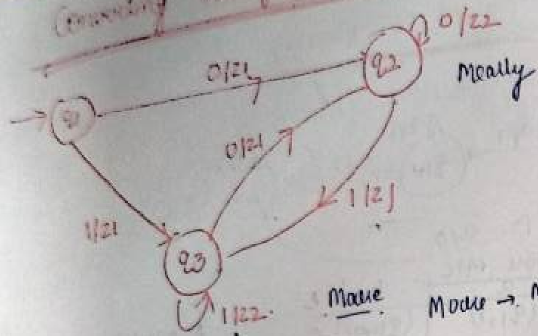
↓ 0, ↓ 1.

ab ab.

A → B → C → B → C

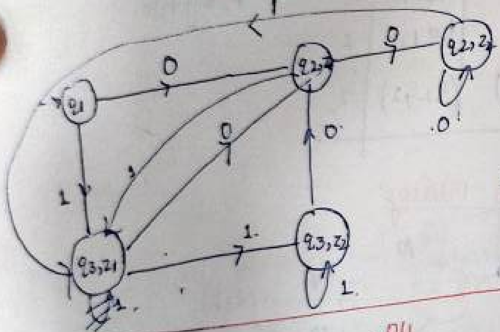
↓ 0, ↓ 1., ↓ 0, ↓ 1.

Converting Meally to Moore M/C



Meally

Moore



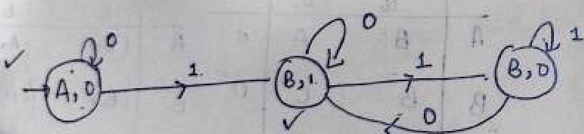
Moore → Meally (No states same

Meally → Moore

N×M. → M×N states

in 1

2 states
2 outputs



In Meally M/C

n Inputs 0 0 1 1

n outputs 1 1 1 0

4 bits input
4 bit output

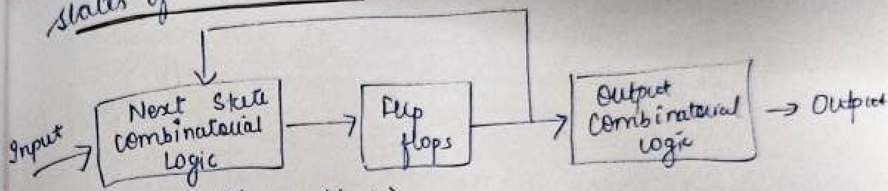
Moore — Meally

no. of states ...

Meally — Moore

NXM. — NXM. states

Moore M/C → output depends only on the states of the machine



Moore Machine

Moore Machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where Q → finite set of states

Σ → input alphabet

Δ → output alphabet

δ is transition function mapping $Q \times \Sigma \to Q$

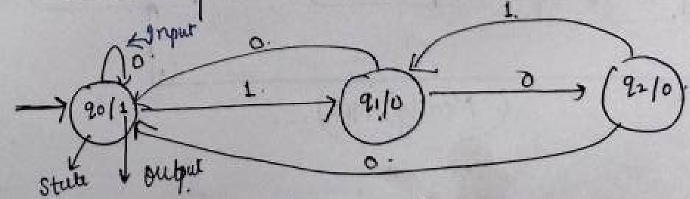
λ is output function mapping Q into Δ. $\lambda: Q \to \Delta$.

q_0 is initial state

Moore M/C $\Delta(t) = \lambda(q(t))$

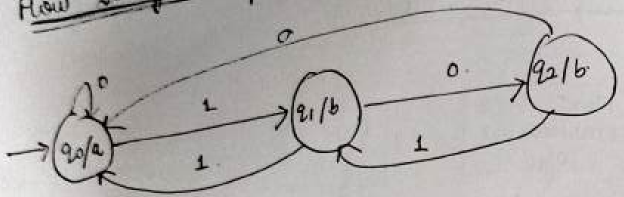
The output in moore m/c depends only on the current state of the machine. ~~independent of current input~~

Current state	Next state (δ) Input 0	Input 1	Output λ
→ q_0	q_0	q_1	1
q_1	q_2	q_0	0
q_2	q_0	q_1	0



Transition Diagram.

How string is processed by



Input string 00110.

$\rightarrow q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_0$

a a a b a a.

output- aaabaa.

~~00110~~ is 00110.

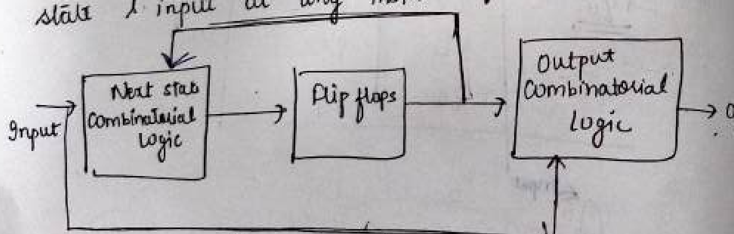
The input to the moore m/c. The length of input string is five & it produces output aaabaa, the length of output is six.

So, in moore machine if the input string is of length n, then output string is of length n+1 Imp

Moore n. output → n+1

Meally Machine

In meally m/c the output depends on the state & input at any instant of time.



(Meally M/c)

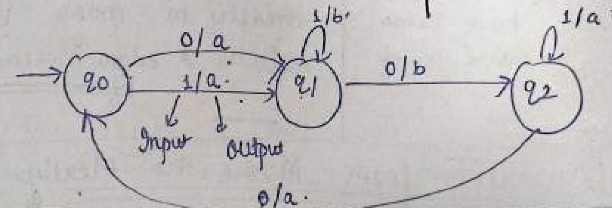
A meally m/c is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where λ → output fx. mapping $\Sigma \times Q \rightarrow \Delta$.

Transition Table for Meally M/c

$\Delta = \quad \lambda \rightarrow \Sigma \times Q$

Present state	Next State: Input = 0 State	Input = 0 Output	Input = 1 State	Input = 1 Output
→ q0	q1	a	q1	a
q1	q2	b	q1	b
q2	q0	a	q2	a

$\Delta(t) = \lambda(q(t), x(t))$

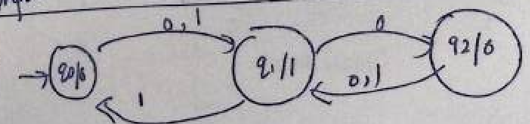


$\lambda: \boxed{Q \times \Sigma \rightarrow \Delta}$

Imp

Moore. $\lambda: \boxed{Q \rightarrow \Delta}$

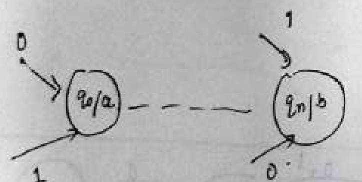
meally $Q \times \Sigma \rightarrow \Delta$.
state input



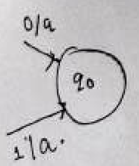
Differentiate Moore & Mealy

Moore	Mealy → 5 marks
1. Output depends only on present state & is independent of current input	1. In mealy m/c, the output depends on the present state & present input
2. If input string is of length n, the output string is of length n+1.	2. If input string is of length n, then output string will be of same length n. Δ λ
3. Output fn λ is defined as:- mapping Q into Δ giving output associated with each state. $\lambda: Q \to \Delta$	3. $\lambda \to$ $\Sigma \times Q \to \Delta$
4. When we convert moore to mealy we have same no. of states & same no. of edges.	4. When we convert mealy to moore both states & edges increase

Conversion from Moore to Mealy



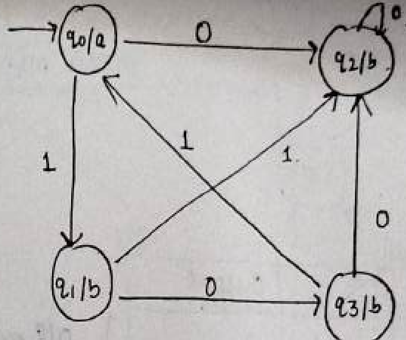
Moore M/c



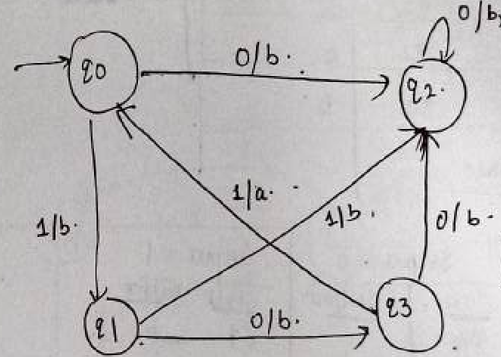
Q Convert the following Moore M/c to Mealy M/c.

Q1

(i)

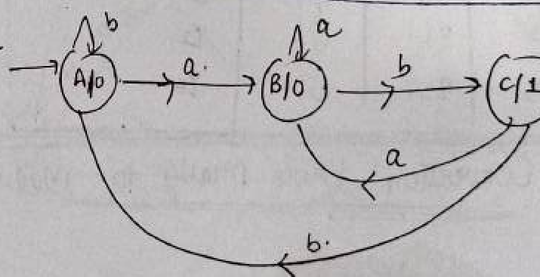


Moore



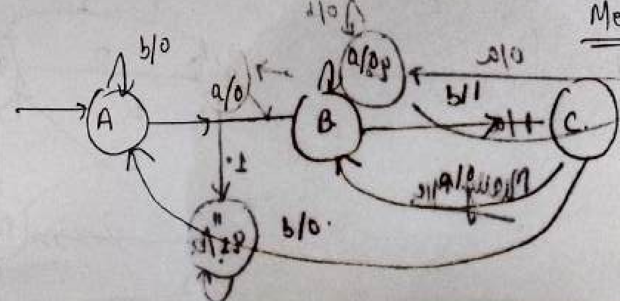
Mealy

Q2

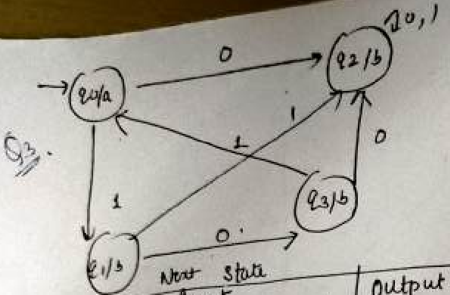


Moore

Mealy



Q3.



Moore M/c

Current state	Next state Input 0	1	Output
→ q0	q2	q1	a
q1	q3	q2	b
q2	q2	q2	b
q3	q2	q0	b

O/P on state

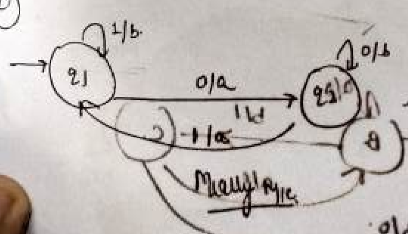
Moore

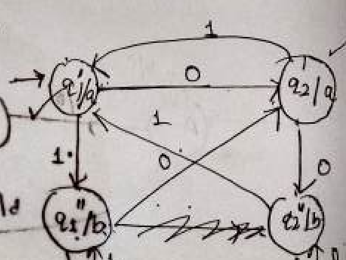
Mealy

Current state	Input = 0 State	output	Input = 1 State	output
q0	q2	b	q1	b
q1	q3	b	q2	b
q2	q2	b	q2	b
q3	q2	b	q0	a

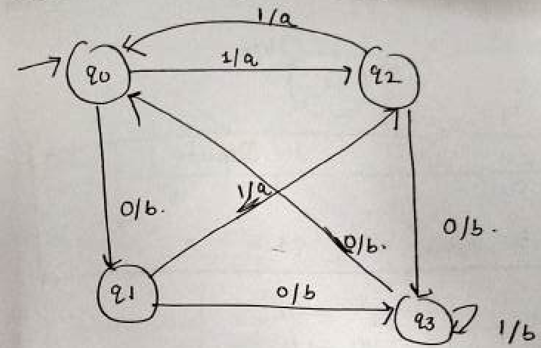
Conversion from Mealy to Moore

(1)

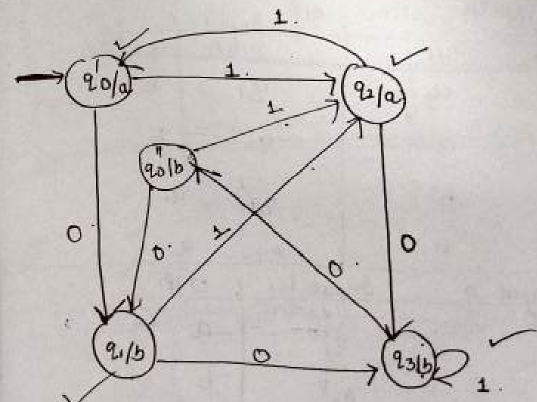




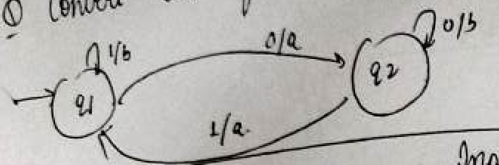
Convert Mealy to Moore



Sol^n



Q Convert the following mealy into moore m/c



	Input = 0		Input = 1	
→ q_1	q_2	a	q_1	b
q_2	q_2	b	q_1	a

$q_1' \to a$ $\quad q_2' \to a$

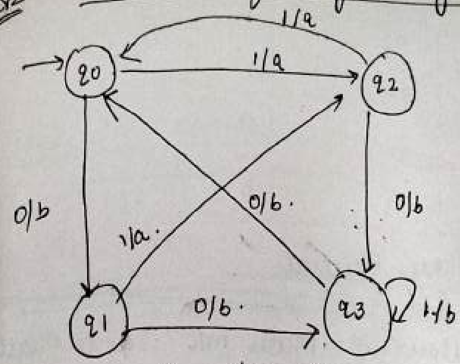
$q_1'' \to b$ $\quad q_2'' \to b$.

We split these state, because they are not associated with same o/p

	Input = 0 State	O/P	Input = 1 State	O/P
→ q_1'	q_2'	a	q_1''	b
q_1''	q_2'	a	q_1''	b
q_2'	q_2''	b	q_1'	a
q_2''	q_2''	b.	q_1'	a

	Input 0 State	Input 1 ~~State~~	O/P.
→ q_1'	q_2'	q_1''	a
q_1''	q_2'	q_1''	b
q_2'	q_2''	q_1'	a
q_2''	q_2''	q_1'	b

Q2 Convert the following mealy m/c to moore m/c



Mealy to Moore

Soln

Present state	Next state Input = 0 State	O/P	Input = 1 State	O/P
→ q_0	q_1	b.	q_2	a
q_1	q_3	b	q_2	a
q_2	q_3	b	q_0	a.
q_3.	q_0	b	q_3	b.

$q_0' \to a$

$q_0' \to a$

$q_0'' \to b$.

$q_0 \to a$ $\quad$ ~~$q_1 \to b$~~

$q_0 \to b$ - $\quad q_2 \to a$

$q_3 \to b$.

$q_0' \to a$

$q_0'' \to b$

	Next state Input = 0 State	O/P	Input = 1 State	O/P
→ q_0'	q_1	b.	q_2	a
q_0''	q_1	b	q_2	a
q_1	q_3	b	q_0'	a
q_2	q_3	b		
q_3	q_0''	b	q_3	b

	Input 0	Input 1	O/P
→ q_0'	q_1	q_2	a
q_0''	q_1	q_2	b
q_1	q_3	q_2	b
q_2	q_3	q_0'	a
q_3	q_0''	q_3	b

Moore Machine

Exercise practice

Q2 Convert the following Moore M/C into Mealy Machine

Current state	Next state 0	Next state 1	Output
→ q_0	q_2	q_3	a
q_1	q_3	q_2	b
q_2	q_1	q_2	a
q_3	q_2	q_1	b

Ans

State	Next state Input 0	output	Input 1	output
→ q_0	q_2	a	q_3	b
q_1	q_3	b	q_2	a
q_2	q_1	b	q_2	a
q_3	q_2	a	q_1	b

$q_0 \to a$

$q_1 \to b$

$q_2 \to a$

$q_3 \to b$

Q

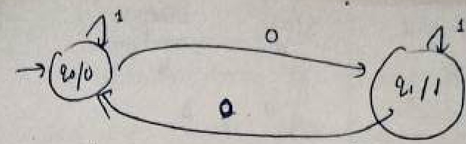
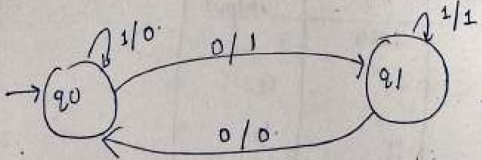


Table :-

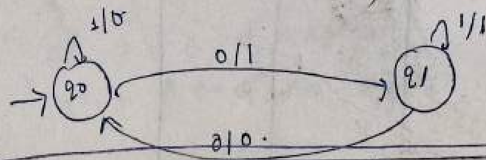
	Next state 0	Next state 1	Output
q_0	q_1	q_0	0
q_1	q_0	q_1	1

	0	Output	1	O/P
q_0	q_1	1	q_0	0
q_1	q_0	0	q_1	1

Mealy M/C



→ Same



same.

Mealy Moore

state	Input a state	Output	Input b state	Output
→ q_0	q_1	a	q_3	b
q_1	q_2	b	q_0	a
q_2	q_3	a	q_1	b
q_3	q_0	b	q_2	a

$q_0' \to a$

$q_0'' \to b$

State	Input a: State	Output	Input b: State	Output
q_0'	q_1'	a	q_3''	b
q_0''	q_1'	a	q_3''	b
q_1'	q_2''	b	q_0'	a
q_1''	q_2''	b.	q_0'	a
q_2'	q_3'	a	q_1''	b
q_2''	q_3'	a	q_1''	b
q_3'	q_0''	b	q_2'	a
q_3''	q_0''	b.	q_2'	a

$q_1' \to a$ $\quad q_2' \to a$ $\quad q_3' \to a$

$q_0' \to a$ $\quad q_1'' \to b$ $\quad q_2'' \to b.$ $\quad q_3'' \to b.$

$q_0'' \to b.$

State	~~Input a~~ a State	b State	Output
q_0'	q_1'	q_3''	a
q_0''	q_1'	q_3''	b
q_1'	q_2''	q_0'	a
q_1''	q_2''	q_0'	b
q_2'	q_3'	q_1''	a
q_2''	q_3'	q_1''	b
q_3'	q_0''	q_2'	a
q_3''	q_0''	q_2'	b.

Regular Expressions

representations of languages which are accepted by finite automata. Operations

(i) + (UNION)

(ii) · (concatenation)

(iii) * (Kleene closure)

(a) ϕ, ϵ, $a \in \Sigma$ } regular expression (Primitive regular expression)

$\phi \to \{\ \}$ $\quad \{\epsilon\} \longrightarrow$ language $\quad \{a\} \to$ language

(b) $r_1 + r_2$, $r_1 \cdot r_2$, ~~$r_1 * r_2$~~ r_1^*

$r_1 + r_2 \to$ union of two regular expression

$\phi = \{\ \}$ — languge.

$\epsilon = \{\epsilon\}$

$a = \{a\}$

$a^* = \{\wedge, a, aa, aaa, \ldots\}$.

$a^+ = a \cdot a^*$ or $a^* \cdot a$

$= \{a, aa, \ldots\}$

$(a+b)^* =$ (set of all strings of any length)

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

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

Length of all strings whose length is exactly 2

$L_1 = \{aa, ab, ba, bb\}$

R.ex. $aa + ab + ba + bb.$

$a(a+b) + b(a+b)$

$= (a+b)(a+b)$ $\quad \underline{(a+b)^2}$.

L_1 = length is at least 2

$L_1 = \{aa, ab, ba, bb, aaa \ldots\}$

$= (a+b)(a+b)(a+b)^*$

$= (a+b)(a+b)(a+b)^* \ldots\}$

at most 2
0, 1, 2

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

$\epsilon + a + b + aa + ab + ba + bb$.

$= (a+b+\epsilon)(a+b+\epsilon)$.

$a\,\cancel{(a+b+b)} + bb$.

$\cancel{(a+bb)}$

of even length

$L = \{\epsilon, aa, ab, ba, bb \ldots\}$

so => $\left(\underline{(a+b)(a+b)}\right)^* = \left((a+b)^2\right)^* = (a+b)^{2*}$

$= (a+b)^{2n} \mid n \geq 0$

$(a+b)^{2n+1} \mid n \geq 0$

$(a+b)^{2n}(a+b) = \left((a+b)(a+b)\right)^*(a+b)$.

length of string should be divisible by 3

① $\left((a+b)(a+b)(a+b)\right)^*$.

$\equiv 2 \bmod 3 \rightarrow$ take a no. divide by 3 & ans = 2

$(a+b)^{3n+2} \mid n \geq 0$

$\left((a+b)(a+b)(a+b)\right)^*(a+b)(a+b)$.

$3\overline{)n}$ 2 mod 3 = 2

2 $\quad$ 3n+2

Regular expression in which $\equiv$ 2 mod 3.

① $\boxed{n(a) = 2}$ $\boxed{b^* a b^* a b^*}$. $\qquad b^* a b^* a (a+b)^*$

② a's at least 2. $\qquad$ ③ a's at most 2

$b^* a b^* a (a+b)^*$ $\qquad b^*(\epsilon + a) b^* (\epsilon + a) b^*$.

$b^*(\epsilon + a) b^* (\epsilon + a) b^*$

a's are even ④ $(b^* a b^* a b^*)^* + b^*$.

$b^* a b^* a b^* = (b^* a b^* a)^* \cdot b^*$. $\qquad b^*(\epsilon + a) b^*(\epsilon + a) b$

⑤ Set of all strings which start with a

$a(a+b)^*$.

⑥ set of all strings which ends with a $(a+b)^* a$.

⑦ containing a $\quad (a+b)^* a (a+b)^*$.

starting & ending with different symbols.

$a(a+b)^* b \;+\; b(a+b)^* a$

starting & ending with same symbol.

$a(a+b)^* a$.

$L = \{\epsilon, a, b, aa, bb, aba\}$

$a(a+b)^* a + b(a+b)^* b + a + b + \epsilon$.

2 a's should not come together.

$L = \{\epsilon, b, bb, bbb \ldots, a, ab, aba, abab, ababa, ba, bab, babab \ldots\}$

$(b+ab)^* = \epsilon, b, bb, bbb \ldots ab, abab \ldots$

$+ (b+ab)^* a = (b+ab)^*(\epsilon + a)$

$= (b+ba)^* \Rightarrow a(b+ab)^* + (\epsilon + ba)^*$

no 2 a's & no 2 b's come together

$L = \{ \Lambda, a, b, ab, ba, aba, bab, \ldots \}$

	starts with	ends with	
$(ab)^*a$ or $a(ba)^*$ $\{a, aba, ababa\}$	a	a	
$\{ab, abab, ababab \ldots\}$	a	b	= or $a(b\ldots)$ $(ab)^*$
$\{ba, baba, \ldots\}$	b	a	$= (ba)^*$
$\{b, bab, babab \ldots\}$	b	b	

$b(ab)^*$ or $(ba)^*b$

$$\left[(ab)^*a + (ab)^* + \frac{b(ab)^*a}{(\cancel{ba})^*} + b(ab)^*\right]$$

$$(ab)^* \boxed{a+\varepsilon} + \quad = (\varepsilon + b)(ab)^*(a+\varepsilon)$$

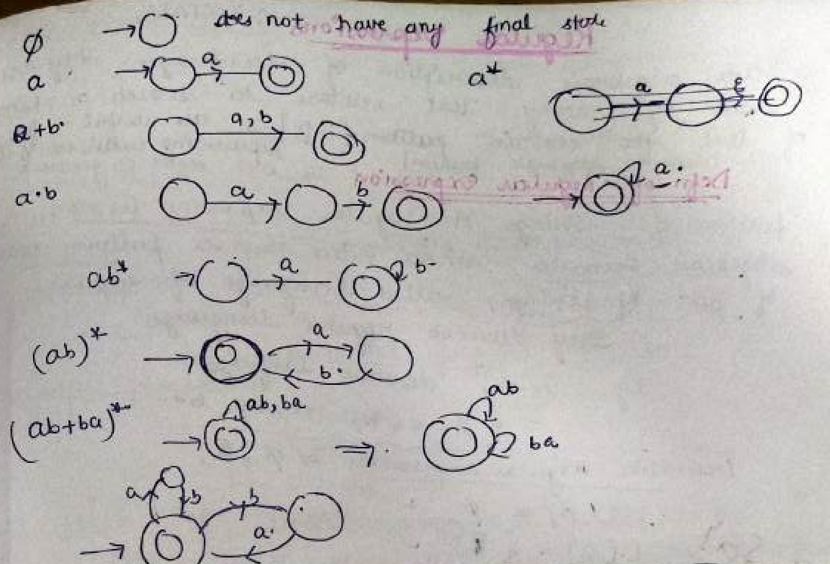
$$= (ab)^*(a+\varepsilon) + b(ab)^*(a+\varepsilon)$$

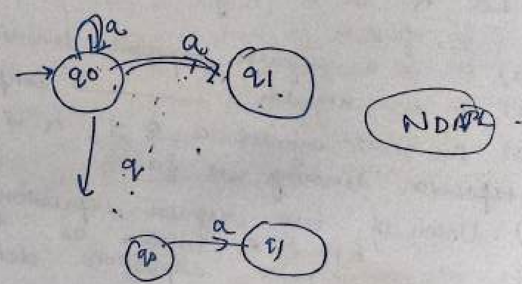
Identities of Regular Expressions

1. $\phi + R = R + \phi = R$
2. $\phi \cdot R = R \cdot \phi = \phi$
3. $\varepsilon \cdot R = R\varepsilon = R$
4. $\varepsilon^* = \varepsilon$
5. $\boxed{\phi^* = \varepsilon}$
6. $\varepsilon + RR^* = R^*R + \varepsilon = R^*$
7. $(a+b)^* = (a^* + b^*)^*$
   $= (a^* b^*)^*$
   $= (a^* + b)^*$
   $= (a + b^*)^* =$

1. $\phi + R = R + \phi \rightarrow R$
2. $\phi R = R\phi = \phi$
3. $\Lambda \cdot R = R\Lambda = R$
4. $\Lambda^* = \Lambda$
5. $\phi^* = \Lambda$
6. $\Lambda + RR^* = R^*R + \Lambda = R^*$
7. $(a+b)^* = (a^* + b^*)^*$
   $= (a^* b^*)^*$
8. $(PQ)^*P = P(QP)^*$

$a^*(ba^*)^* = b^*(ab^*)^*$





Regular expressions grammer

are algebraic description of languages. They are used by many text editors to search a block of text for certain pattern. [Defn → are useful for representing certain set in algebraic fashion] → are used to generate

Defn. of Regular Expression

patterns of strings. A regular expression is an algebraic formula whose value is a pattern consisting of set of strings, called language of expression.

They describe regular language

eg $(a+b)^*$ describe language
$= \{\lambda, a, b, ab, bb, ba, \ldots\}$

Primitive regular expressions :⇒ ϕ, ε,

$L(\phi) = \phi$

a {a}

$L(\varepsilon) = \{\varepsilon\}$

$L(a) = \{a\}$.

Let R be a regular expression over alphabet Σ, if R is

1) ε is a regular expression denoting set $\{\varepsilon\}$

(2) ϕ is regular ______ empty set ϕ

(3) For each symbol $a \in \Sigma$, a is a regular expression denoting set $\{a\}$

) Union of two regular expressions R_1 & R_2 written as $R_1 + R_2$ is also a regular expression denoting set $\{R_1 \cup R_2\}$

5. Concatenation of two regular expressions R_1 & R_2 written as R_1R_2 is also a regular expression denoting set (R_1R_2)

(6) Iteration (or closure or star) of a regular expression R written as R^* is also a regular expression denoting the set (R^*)

(7) If R is a regular expression then (R) is also a regular expression

Operators used in Regular Expression

Three permitted operators for building regular expressions.

(1) UNION

(2) concatenation (dot)

(3) closure (star)

18/10

(1) Union of two languages P & Q denoted by $P \cup Q$ is set of string that are in either P or Q or both

eg $P = \{a, b\}$ $Q = \{a, b, c, d\}$

$P \cup Q = \{a, b, c, d\}$

(2) Concatenation - The concatenation of two languages P and Q denoted by $P \cdot Q$ or is set of strings that can be formed by taking any string in P & concatenating it with any string in Q

if $P = \{a, b\}$ & $Q = \{\varepsilon, d\}$ then

$PQ = \{a, b, ad, bd\}$.

(3) Closure of a language P denoted by P^*. defines set of all string that can be formed by taking any no. of strings from P

$P = \{a, b\}^*$

$P = \{\varepsilon, a, \ldots\}$

b^* – mean $\{\lambda, b, bb \ldots\}$

$(ac)^* = \{\lambda, ac, acac, \ldots$

Any string contains exactly one c

$(a+b)^* c (a+b)^*$

Any string not containing $c \Rightarrow (a+b)^*$

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

Q

$\{a\} \longrightarrow a$

$\{ab, ba\} \longrightarrow ab+ba$

$\{abbb\} \longrightarrow abbb$

$\{\cancel{b+b+ab}\} = \{b, b, ab\} \rightarrow b+b+ab$

$\{\varepsilon, b, bb, \ldots\} \Rightarrow b^*$

$\{\varepsilon, ab\} \Rightarrow \varepsilon + ab$

Q $R = a(a+b)^* ab (a+b)^*$ over $\Sigma = \{a, b\}$

a — any string of a & b — ab — any string of a & b

Begg with a and containing substring ab

$\{aab, aaaba, aabb \ldots\}$.

Q3. Consider the alphabet $\Sigma = \{a, b\}$ and describe the regular expression for following statements:—

(i) All strings having a single b.

$a^* b a^*$

(ii) All strings having at least one b

$(a+b)^* b (a+b)^*$

(iii) All strings having bbbb as substring

Sol: $(a+b)^* bbbb (a+b)^*$

(iv) All strings end with ab.

$(a+b)^* ab$.

Sdn

(v) All strings start with ba

$ba(a+b)^*$

(vi) All strings in which a single a is followed by any number of b's or a single b followed by any no. of as

$ab^* + ba^*$.

(vii) Set of all strings over alphabet $\Sigma = \{0, 1\}$ begg and ending with 0

$0(0+1)^* 0$

(viii) $b^2, b^5, b^8 \ldots$ $bb(bbb)^*$

$bb(bbb)^*$.

(ix) $\{a^{2n+1} \mid n \geq 0\} \Rightarrow a(aa)^*$ $\{a, a^3, a^5, a^7$ $(aa)^* a$. $a(aa)^*$

Example $L = \{0^{2n} 1^{2m+1} \mid n \geq 0, m \geq 0$

$\{\cancel{0, 0, 1}$ $1^{2\times 0+1}$

$\{\cancel{\lambda}\ 1, 00111, 111$ $\{0^{2\times 0} 1, \underline{00}\,\underline{111}, \underline{111}, 000011111$

$(00)^*(11)^* 1$

(ii) $L = \{a, bb, \underline{aa}, \underline{bb}b, ba, bbb \ldots$

$(a+b)^+$

(iii) $(a+b)^* bb$ (string of a's & b's ending in bb)

$0^{2\times 0}$ 0

$0^{2n} 1^{2m+1} \mid n \geq 0, m \geq 0$

$1^{2\cdot 0+1}$

$\{\cancel{01, 001},$

$\{1, 001, 111 \ldots$

Q. ~~All strings containing~~ no $(0+10^*)$

Regular set :-

(1) $L=\{0,1,10,100,1000,1000\ldots\}$ ~~$(0+10^*)$~~

$(0+10^*)$

(2) $L=\{1,01,10,010,0010,\ldots\}$ 0^*1^*

$\Rightarrow 0^*10^*$

(3) $L=\{\varepsilon,0,1,01\}$ $\Rightarrow$ ~~$0+\varepsilon$~~

$0+\varepsilon+1+01$

$0(\varepsilon+1)+(\varepsilon+1)$

$(\varepsilon+0)(\varepsilon+1)$

(4) set of a's & b's of any length including the null string.

$(a+b)^*$

(5) set of string of a's & b's ending with abb.

$(a+b)^*abb$.

(6) set consisting of even no. of 1's including empty string $L=\{\wedge,11,1111\ldots\}$

$(11)^*$

(7) set of strings consisting of even no. of a's followed by odd no. of b's.

$(b), aab, aabbb$ — $(aa)^*(bb)^*$

$\{b, aab, aabbb, aabbbbb,\ldots$

$(aa)^*(bb)^*b$. $(aa)^*(bb)^*b$.

(8) $L=\{$ ~~aa, ab, ba, bb, ...~~ $\}$

$\boxed{(aa+ab+ba+bb)^*}$

strings of a's & b's of even length can be obtained by concatenating any combination of strings aa, ab, ba & bb

Regular sets Any set that represents the value of regular expression is called a set

Properties of Regular sets

Property 1 :- Union of two regular set is regular

$RE_1 = a(aa)^* \Rightarrow 1$

$RE_2 = (aa)^*$

$L_1 = \{a, aaa, aaaaa\ldots\}$ (strings of odd length excluding Null)

$L_2 = \{\lambda, aa, aaaa, aaaaaa,\ldots$ (strings of even length including Null)

$L_1 \cup L_2 = \{\lambda, a, aa, aaa, aaaa,\ldots$

$RE(L_1 \cup L_2) = a^*$ (which is a regular expression itself)

Property 2 :- Intersection of two regular sets is regular

$RE_1 = a(a^*)$ $RE_2 = (aa)^*$.

$L_1 = \{a, aa, aaa, aaaa\ldots\}$ (strings of all possible lengths excluding NULL)

$L_2 = \{\varepsilon, aa, aaaa,\ldots\}$ (strings of even length including NULL)

$L_1 \cap L_2 = \{aa, aaaa, aaaaaa\ldots\}$ (strings of even length excluding NULL)

$aa(aa)^*$.

$aa(aa)^*$ which is a regular expression itself.

Typecasting

int x = 10;
byte y = (byte)x.

int float = 4.;
int 4;
float int = (int) float ...;

float f = 4.5 Not included
int f = (int) f ;

Property 3 The complement of a regular set is regular

$RE = (aa)^*$

$L = \{\epsilon, aa, aaaa, aaaaaa\ldots\}$ (strings of even length including NULL)

$L' = \{a, aaa, aaaaa, \ldots\}$.

$a(aa)^*$ which is a regular expression itself

Property 4. The difference of two ~~regular~~ set is regular

$RE_1 = a(a^*)$

$RE_2 = (aa)^*$

$L_1 = \{a, aa, aaa \ldots\}$ (strings of all possible lengths excluding NULL)

$L_2 = \{\wedge, aa, aaaa, aaaaaa\ldots\}$ strings of even length including NULL.

$L_1 - L_2 = \{a, aaa, aaaaa\ldots\}$

$\Rightarrow a(aa)^*$ is regular expression

Q5 Reversal of regular set is regular

$L = \{01, 10, 11, 10\}$

$RE(L) = 01 + 10 + 11 + 10$

$L^R = \{10, 01, 11, 01\}$

$RE(L^R) = 10 + 01 + 11 + 10$

6. Sol The closure of regular set is regular

If $L = \{a, aaa, aaaaa\ldots\}$ (set of strings of odd length excluding $\wedge$)

$RE(L) = (a(aa)^*)^*$

$L^* = \cancel{\{a, aa, aaa}$ $L = \cancel{\{a, aaa, aaa}$..

$\cancel{L^* = (a(aa)^*)^* = \{\wedge, a, aa}$

$L^* = \{\wedge, a, aa, aaa, aaaa \ldots\}$

(7) Concatenation of two regular sets is regular

Let $RE_1 = (0+1)^*0$

$RE_2 = 01(0+1)^*$

$L_1 = \{0, 00, 10, \ldots\}$ set of strings ending in 0

$L_2 = \{01, 010, 011, \ldots\}$ (set of strings beginning with 01)

$L_1L_2 = \{001, 0010, 0011, \ldots\}$

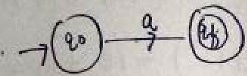
$(0+01)^* = \{$

000, 0101, 0001

ϕ is a regular expression denoting empty set

ϵ

a $\{0\} \longrightarrow$



If r_1 & r_2 are regular expression =

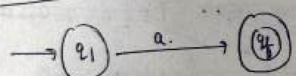
$r_1 + r_2$ — $L_1 \cup L_2$

$r_1 r_2$ — $L_1 L_2$.

r_1^* — L_1^*.

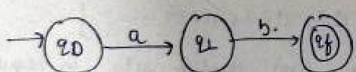
$(0+01)^*$

(1)



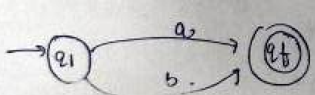
RE = a

(2)



RE = ab.

(3)



RE = a+b.

(4)

RE = $(a+b)^*$

Describe the following sets by regular expressions.

(a) $\{101\} \longrightarrow 101$

(b) $\{abba\} \longrightarrow abba$

(c) $\{01, 10\} \longrightarrow 01+10$

(d) $\{\Lambda, ab\} \longrightarrow \Lambda + ab$

(e) $\{abb, a, b, bba\} = abb + a + b + bba$

$= a(\cancel{bb+\Lambda}) \cancel{+} b(\Lambda + ba)$

(f) $\{\Lambda, 0, 00, 000, \ldots\} \rightarrow 0^*$

(g) $\{1, 11, 111, \ldots\} \Rightarrow 1^+$ or 11^*

Q Describe the following sets by regular expressions

(a) L_1 = the set of all strings of 0's & 1's ending in 00

Soln $(0+1)^* 00$

(b) L_2 = the set of all strings of 0's & 1's beg. with 0 & ending with 1

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

(c) $L_3 = \{\Lambda, 11, 1111, 111111, \ldots\}$

Soln $(11)^*$

Find regular exp. to

(a) set of all strings containing exactly 2a's

$b^* a b^* a b^*$

(b) at least 2a's

$(a+b)^* a (a+b)^* a (a+b)^*$

(c) at most 2a's.

$b^* a b^* a b^* + b^* a b^*$

$b^*(a+\Lambda) b^* (a+\Lambda) b^*$

$b^* [\Lambda + ab^*]$ $b^* + b^* a b^*$ $+ b^* a b^* a b^*$

$b^* [\cancel{\Lambda + ab^* + aa}]$

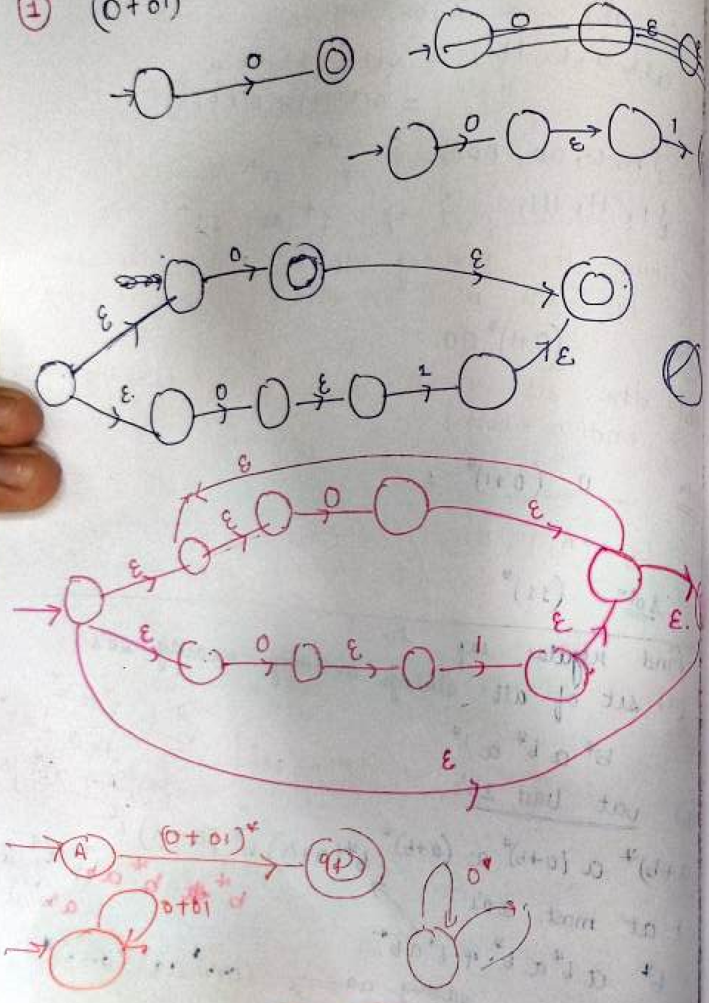
$b^* + b^* a b^* + b^* a a b^*$

$a b^* a$

(d) containing substring aa.

Construction of finite automata from a regular expression

(1) $(0+01)^*$



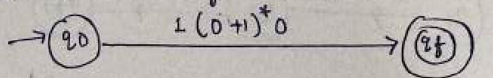
Step 1. Convert an NFA with NULL moves from regular expression.

2.) Remove Null transition from NFA & convert it into its equivalent DFA.

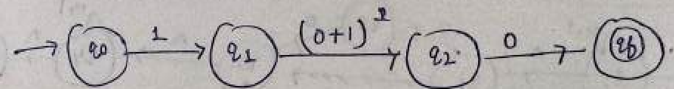
Problem Convert the following ~~DFA~~ Regular expression into its equivalent DFA

$1(0+1)^*0$

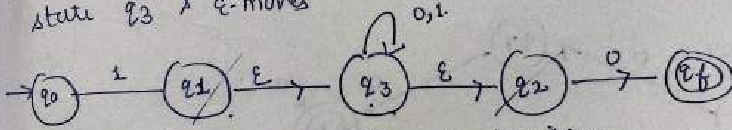
1) First we introduce initial state & final state for whole regular expression.



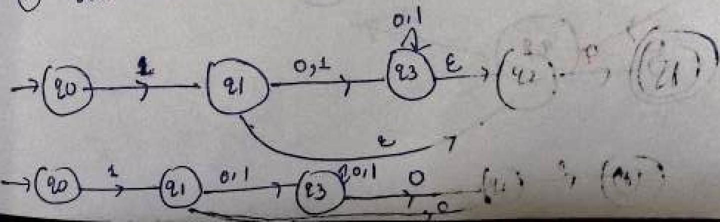
2) We eliminate concatenation by introducing new states q_1 and q_2

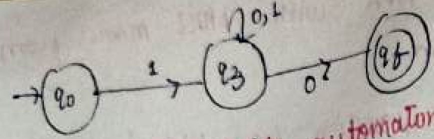


(3) We eliminate star from regular expression between states q_1 and q_2 by introducing state q_3 & ε-moves



(4) We will remove the ε-transitions.

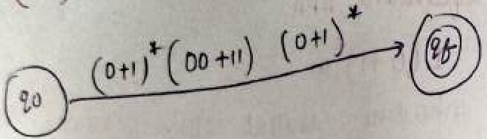


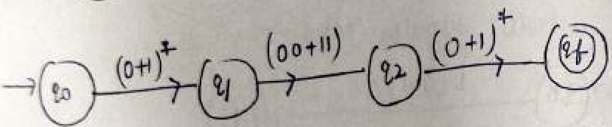


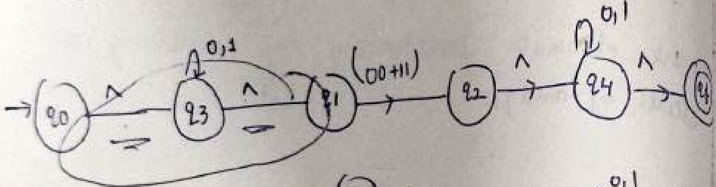
Q2 Construct the finite automaton equivalent to regular expression.

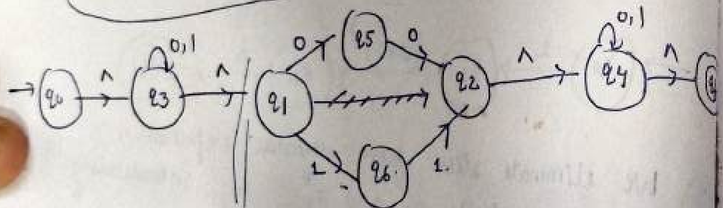
$(0+1)^*(00+11)(0+1)^*$

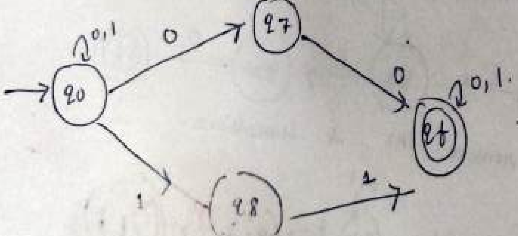
Sol





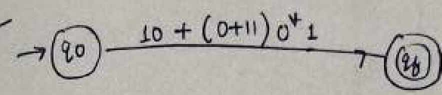




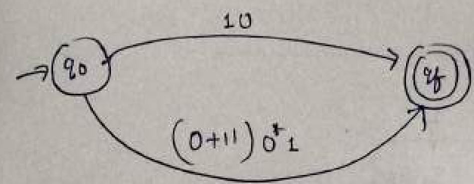


Q Construct a DFA with reduced states eq. to $10+(0+11)0^*1$

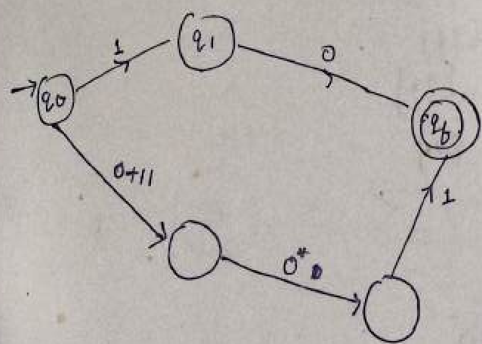
Soln



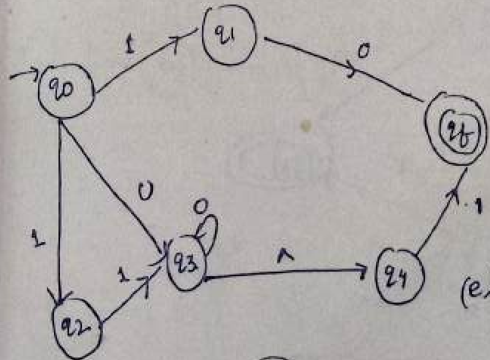
(a)



(b) Elimination of +.

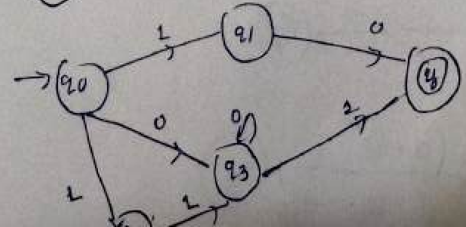


(c) Elimination of Concatenation & *



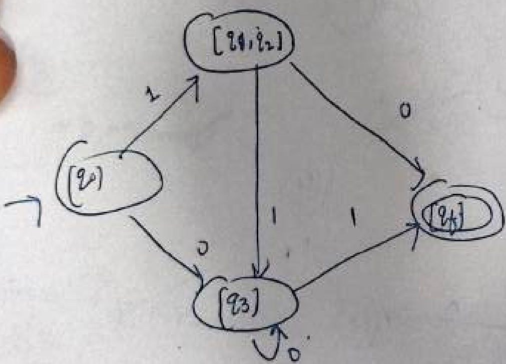
(d) Elimination of +.

(e) Elimination of Λ-moves

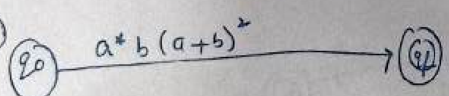


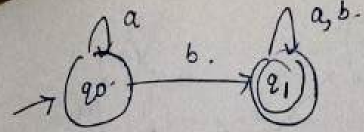
State	0	1
$\rightarrow q_0$	q_3	q_1, q_2
q_1	q_f	q_3
q_2		
q_3	q_3	q_f
q_f		

Q	0	1
$\rightarrow [q_0]$	$[q_3]$	$[q_1, q_2]$
$[q_3]$	$[q_3]$	$[q_f]$
$[q_1, q_2]$	$[q_f]$	$[q_3]$
$[q_f]$	$[\phi]$	$[\phi]$

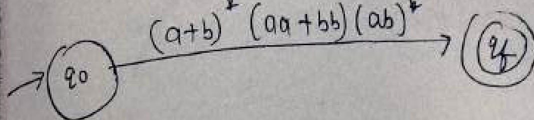


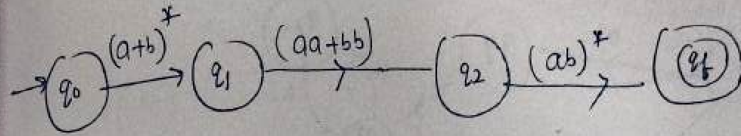
3

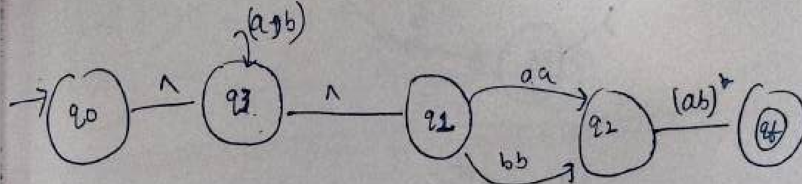


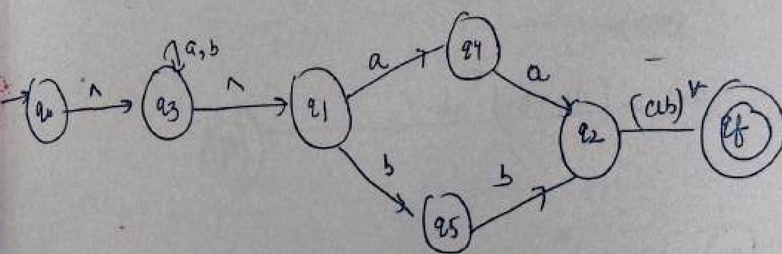


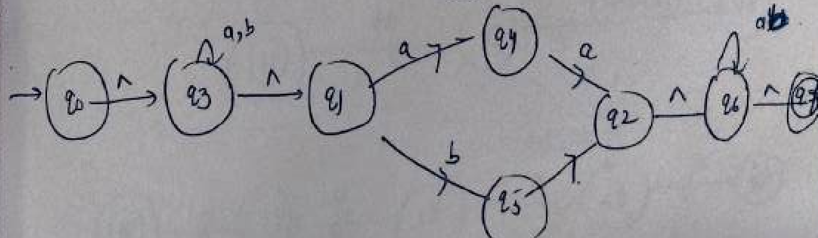
Q $(a+b)^* (aa+bb) (ab)^*$

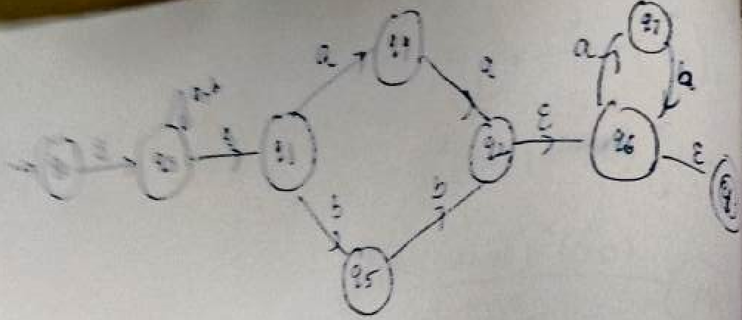


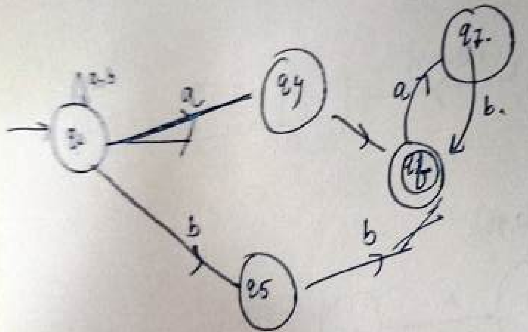






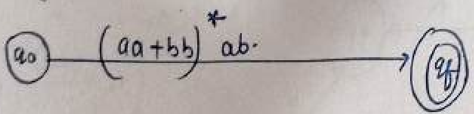


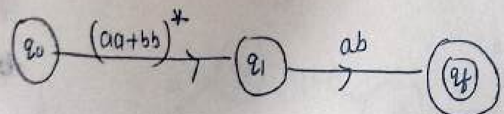


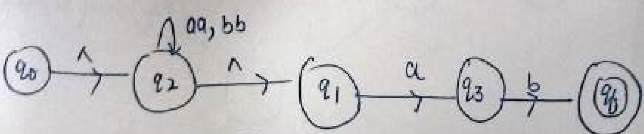


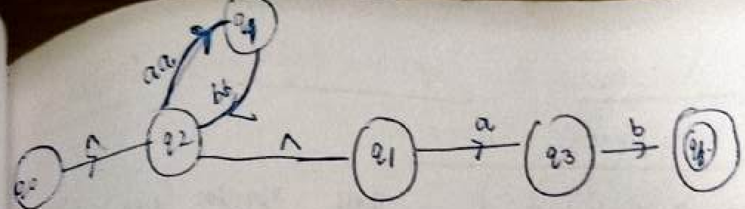
$(aa+bb)^* ab.$

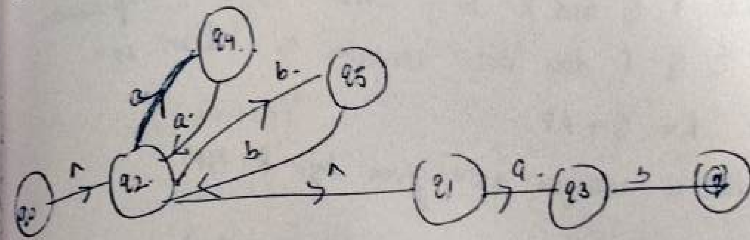
~~(aa)~~

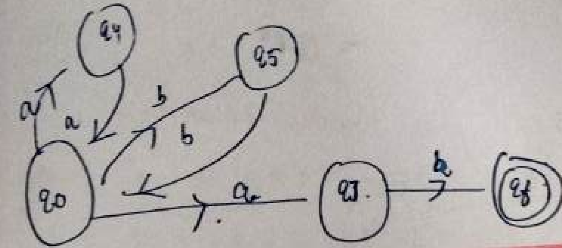




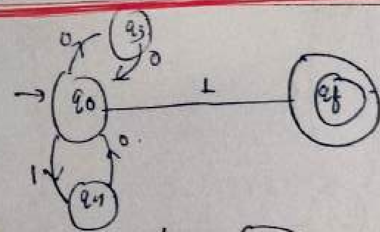


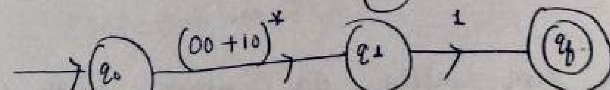


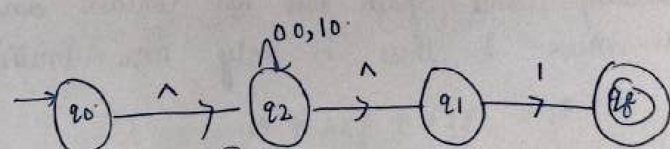


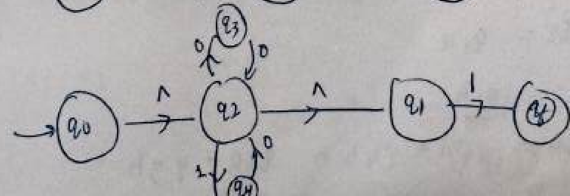


$(00+10)^* 1$









Finite automata to Regular Expression Conversion using ARDEN'S THEOREM

Let P, Q and R be three regular expressions over Σ. If P does not contains Λ, then eqⁿ

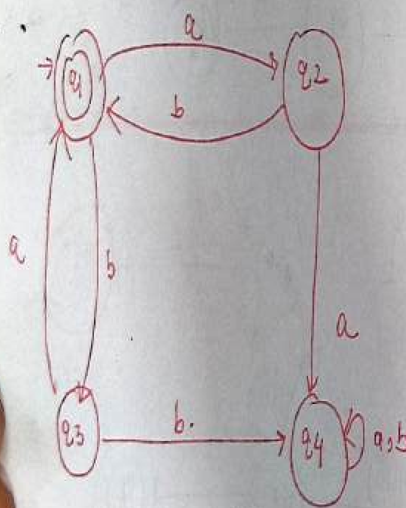
$R = Q + RP$

has a unique solⁿ given by $R = QP^*$.

$Q + RP = Q + (QP^*)P = Q(\Lambda + P^*P)$

$= QP^*$.

Q Convert this into regular expression.



We can apply the above method directly since graph does not contain any null move & there is only one initial state

$q_1 = q_2 b + q_3 a + \Lambda$

$q_2 = q_1 a$

$q_3 = q_1 b$

$q_4 = q_4 a + q_4 b + q_2 a + q_3 b$

$q_1 = q_1 ab + q_1 ba + \Lambda$

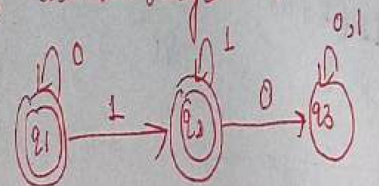
$q_1 = \underbrace{q_1}_{R}\underbrace{(ab+ba)}_{P} + \underbrace{\Lambda}_{Q}$. $R = Q + RP$

$= QP^*$.

$= \Lambda(ab+ba)^*$

$= (ab+ba)^*$

Q2 Describe in english the set accepted by FA whose transition diagram is



Solⁿ $q_1 = q_1 0 + \Lambda \Rightarrow \Lambda 0^* = 0^*$.

$q_2 = q_1 1 + q_2 1$

$q_3 = q_2 0 + q_3 0 + q_3 1$.

$q_2 = 0^* 1 + q_2 1$

$R = Q + RP$.

~~$q_2 = q_2(1 + 0^* 1)$~~ ~~$R = P + QR$~~.

$q_2 = q_2 1 + \underbrace{0^* 1}$ $= QP^*$.

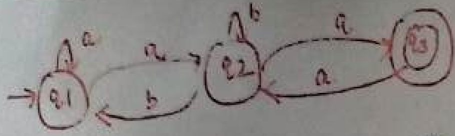
$q_2 \ (0^* 1) 1^*$

$\Rightarrow q_1 \ \& \ q_2$

$q_1 + q_2 = 0^* + 0^* 1 1^*$

$= 0^*(\Lambda + 11^*) = \cancel{0^*(1^*)}$

$= 0^* 1^*$



$q_1 = q_1 a + q_2 b + \Lambda$ — (1)

$q_2 = q_2 b + q_3 a + q_1 a$ — (2)

$q_3 = q_2 a$ — (3)

$q_2 = q_2 b + q_2 aa + q_1 a$

$\underset{R}{q_2} = \underset{Q}{q_1 a} + \underset{R}{q_2}\underset{P}{(b+aa)}$ $R = Q + RP$, QP^*

$$\boxed{q_2 = q_1 a (b+aa)^*}$$

Substituting q_2 in q_1 $R = Q + RP$, $\boxed{QP^*}$

$q_1 = q_1 a + q_1 a (b+aa)^* b + \Lambda$

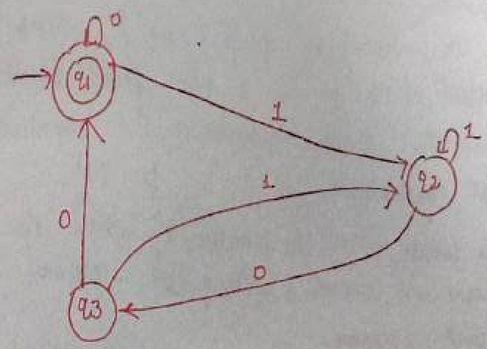
$q_1 = q_1 (a + a(b+aa)^* b) + \Lambda$

$= \Lambda (a + a(b+aa)^* b)^*$

$q_2 = (a + a(b+aa)^* b)^* a (b+aa)^*$

$q_3 = (a + a(b+aa)^* b)^* a (b+aa)^* a$

Construct a regular expression corresponding to state diagram



$q_1 = q_1 0 + q_3 0 + \Lambda$ — (1)

$q_2 = q_2 1 + q_3 1 + q_1 1$ — (2)

$q_3 = q_2 0$ — (3)

$q_2 = q_2 1 + q_1 1 + (q_2 0) 1$

$q_2 = q_1 1 + q_2 (1 + 01)$

$q_2 = q_2 (1+01) + q_1 1$

$$\boxed{q_2 = q_1 1 (1+01)^*}$$

$$\boxed{q_3 = q_1 1 (1+01)^* 0}$$

$q_1 = q_1 0 + q_1 1 (1+01)^* 0 0 + \Lambda$

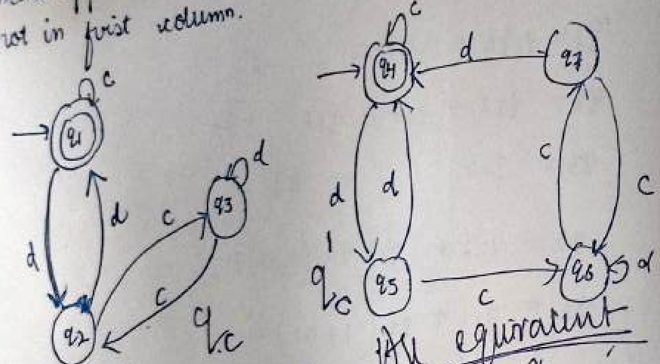
$q_1 = q_1 (0 + 1 (1+01)^* 00) + \Lambda$

$= (0 + 1(1+01)^* 00)^*$

Equivalence of two finite automata

Case 1: If we reach a pair (q, q') such that q is final state of M, q' is a non-final state of M', we terminate the construction & conclude that M & M' are not equivalent.

Case 2: Here constn is terminated when no new element appears in second & subsequent columns which are not in first column.



All equivalent

q, q'	q_c, q'_c	q_d, q'_d
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

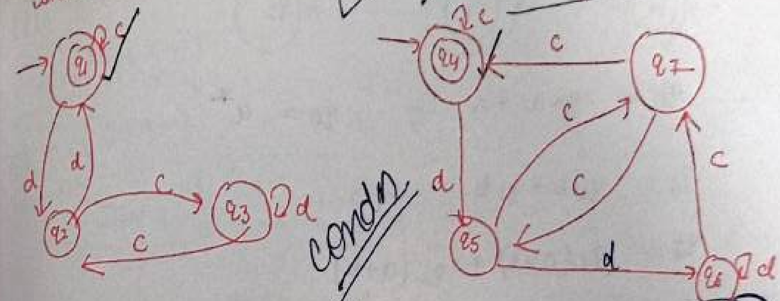
F NF ✗

F F ✓

N NF ✓

We dont get a pair (q, q') where q is a final state, & q' is a non-final state.

Show that automata M_1 & M_2 are equivalent



condn

(q, q')	q_c, q_c'	q_d, q_d'
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)

(q_1, q_6) NF → q_1 is final but q_6 is non final.

M & M' are not equivalent

q_1, q_4

2 or 3 questions

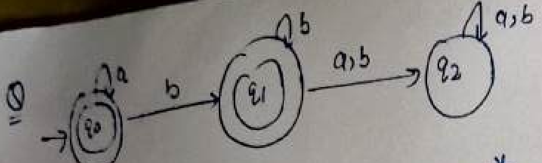
FA: Advantages, Limitation, Application, equivalent

M.I.U ~ first class.

Third Unit

Two first Units

Q



$q_0 = q_0 a + \Lambda \Rightarrow q_0 = a^*$

$q_1 = q_0 b + q_1 b \Rightarrow$.

$q_2 = q_1(a+b) + q_2(a+b)$

$q_1 = a^* b + q_1 b$

$q_1 = q_1 b + a^* b$

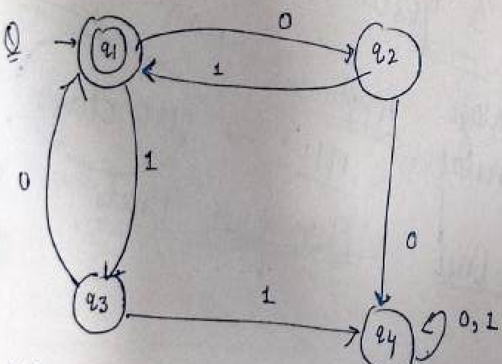
$= (a^* b) b^*$

$q_0 + q_1 = a^* + a^* b b^*$

$= a^* (\Lambda + b b^*)$

$= a^* b^* \quad (\varepsilon + RR^* = R^*)$

Q



Sol^n $q_1 = q_2 1 + q_3 0 + \Lambda$ — (1)

$q_2 = q_1 0$ — (2)

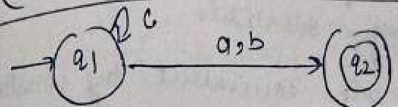
$q_3 = q_1 1$ — (3)

$q_4 = q_3 1 + q_2 0 + q_4(0+1)$ — (4)

$q_1 = q_1 01 + q_1 10 + \Lambda$

$q_1 = q_1(01+10) + \Lambda$

$q_1 = (01+10)^*$



$q_1 = q_1 c$ — (1) $\Rightarrow q_1 = c^* \Lambda \Rightarrow c^*$

$q_2 = q_1(a+b)$

$q_2 = c^*(a+b)$

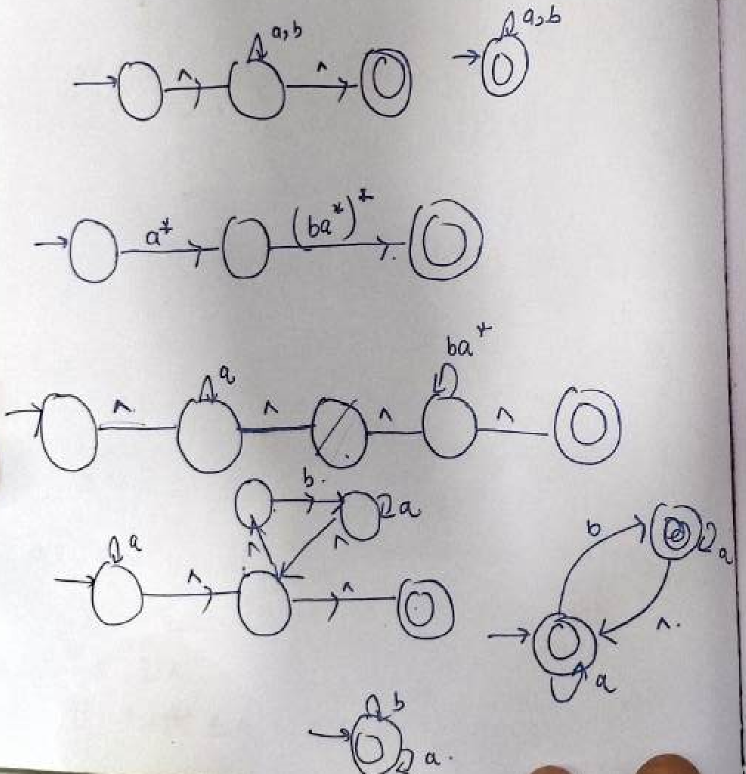
Equivalence of two regular expressions

A & B → are equivalent if & only if they represent same set or if there corresponding finite automata are equivalent or we make them equal by using identities.

Method 1. A & B are equivalent by constructing finite automata's of A & B.

Method 2 equivalence of A & B can be proved with the help of identities

Example 1:- $(a+b)^* = a^*(ba^*)^*$



Q $(0+1)^*01 = 0^*(10^*)^*01$

Sol^n

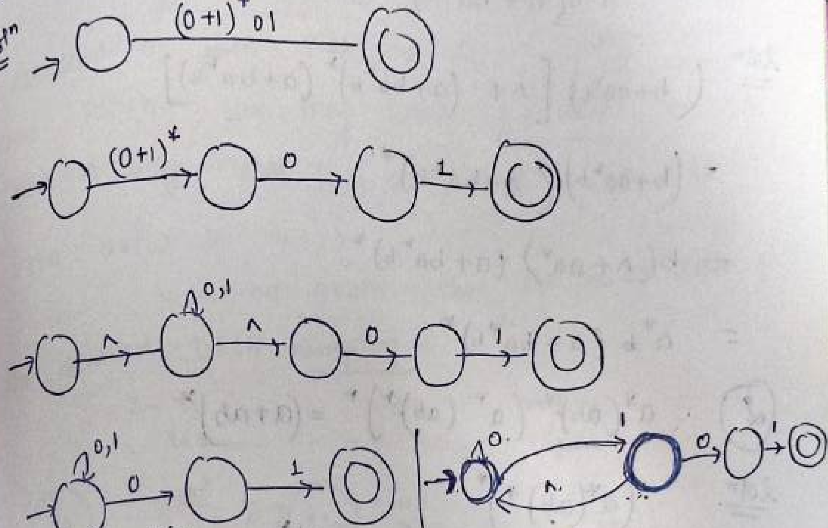
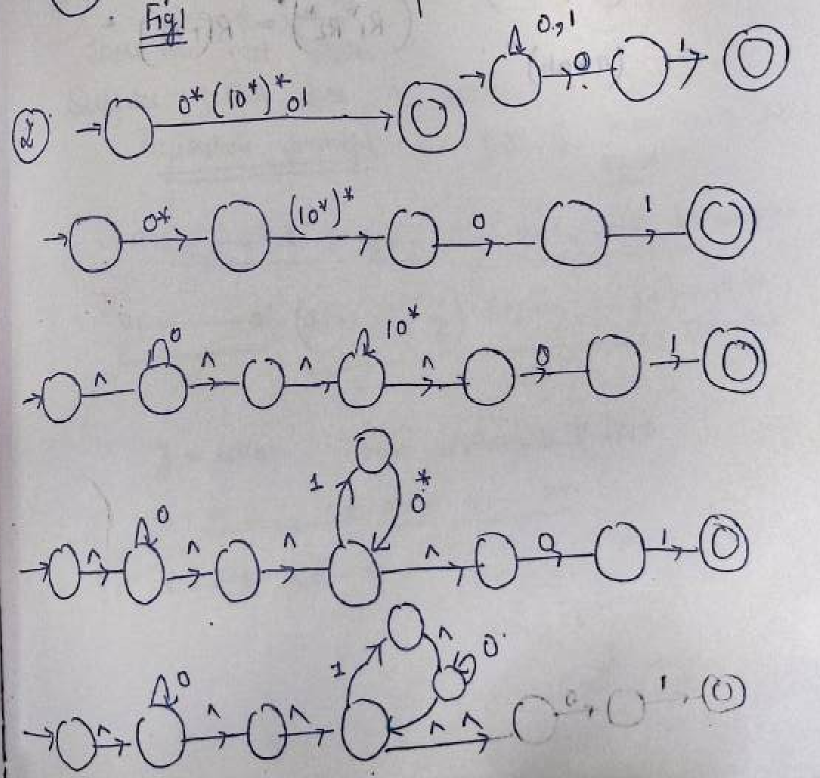


Fig1

(2)



Ex $(b+aa^*b)+(b+aa^*b)(a+ba^*b)^*(a+ba^*b)$

$= a^*b(a+ba^*b)^*$

Soln $(b+aa^*b)\left[\Lambda + (a+ba^*b)^*(a+ba^*b)\right]$

$= (b+aa^*b)(a+ba^*b)^*$

$= b(\Lambda + aa^*)(a+ba^*b)^*$

$= a^*b(a+ba^*b)^*$

Q $\underbrace{a^*(ab)^*}(a^*(ab)^*)^* = (a+ab)^*$

Soln $(a^*(ab)^*)^*$

$(R_1^*R_2^*)^* = (R_1+R_2)^*$

$= (a+ab)^*$

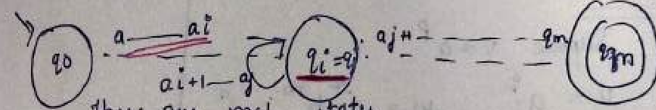
Pumping Lemma

Let L be a regular set. Then there is a constant n such that if z is any word in L, and $|z| \geq n$, we may write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$ for all $i \geq 0$ uv^iw is in L.

n is no greater than the no. of states of the smallest DFSA accepting L.

$z = a_1 - - - - a_m \quad m \geq n.$

$q_0\, q_1 - - - - q_m. \quad i<j$



There are $m+1$ states

Suppose DFSA has n states.

Pigeonhole principle — two of them will be equal.

$a_1 - - a_i,\ a_{j+1} - - a_n$ will be accepted

$\underbrace{a_1 - a_i}_{u}, \underbrace{(a_{i+1} - a_j)^i}_{v} \underbrace{(a_{j+1} - a_m)}_{w}$ will be accepted

$z = uvw.$ ~~for~~ $uv^iw \in L\ \forall\ i \geq 0$

$a_1 a_2 - - a_n\ a_{n+1} - - a_m$

$q_0\ q_2\ q_iq_j\ q_n.$ $|uv| \leq n.$

$uv^iw \in L$

This is useful in showing that certain sets are not regular.

$\{a^n b^n \mid n \geq 1\}$ // Don't follow this method

Soln: Suppose L is regular.

then L will be accepted by FSA

By Pumping Lemma there is n such that

if $z \in L$ and $|z| \geq n$

$z = uvw$ such that $uv^iw \in L$

$a^m b^m \quad m \geq n.$

$\underbrace{aaa}_{u} \mid \underbrace{--}_{v} \mid \underbrace{a\ bbb - - b.}_{w}$

Suppose $v = a^p$.

$u = a^q \qquad w = a^r b^m.$

$a^q \quad p + q + r = m$

$a^q (a^p) a^r \ b^m$

$= a^q (a^p)^i a^r b^m \in L \quad \forall i \geq 0$

$\therefore L$ is not regular

Q2 $L = \{a^{n^2} \mid n \geq 1\}$

a^{n^2}

$\underbrace{uv}\ \mid\ a$

Let L be accepted by DFSA with n states

$|uv| \leq n \qquad |v| \geq 1$

$1 \leq |v| \leq n.$

$|uv^2w| = |uvw| + |v|$

$\leq n^2 + n$

$\{a, a^4, a^9 - a^{n^2}, a^{(n+1)^2} - \ldots$

L is regular ⇒ pumping lemma holds

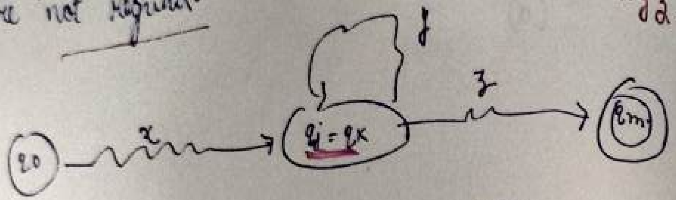
$L = \{a^p b^{m^2} \mid p \geq 1, m \geq 1\} \cup \{b^q \mid q \geq 1\}$

$n. \qquad a - - - b$

(a)

It gives a method of pumping many input strings from a given string. It is used to show that certain sets are not regular.

$w = xyz$.



Let $w = a_1 a_2 \ldots a_m$ $\quad m \geq n$.
↓ no. of states

$\delta(q_0, a_1 a_2 \ldots a_i) = q_i$ for $i = 1, 2, \ldots m$.

Let q_j & q_k (two coincide states).

Then, j & k satisfy the condn $0 \leq j < k \leq n$.

Decompose w into three substrings

$x = a_1 a_2 \ldots a_j$

$y = a_{j+1} \ldots a_k$

$z = a_{k+1} \ldots a_m$.

$\underbrace{a_i \ldots a_j}_{x}\ \underbrace{a_{j+1} \ldots a_k}_{y}\ \underbrace{a_{k+1} \ldots a_m}_{z}$

$|xy| \leq n$

as $k \leq n$, $|xy| \leq n$ and $w = xyz$.

The automaton M starts from initial state q_0. On applying string x, it reaches $q_j (= q_k)$. On applying string y, it comes back to $q_j (= q_k)$. So, after application of y^i for each $i \geq 0$, the automaton is in same state q_j. On applying z, it reaches q_m, a final state. Hence $xy^iz \in L$.

To prove that certain sets are not regular

Step 1 Assume that L is regular. Let n be no of states in corresponding finite automaton. Choose a string w such that $|w| \geq n$.

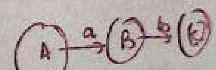
Step 2 Use pumping lemma to write $w = xyz$, with $|xy| \leq n$ & $|y| > 0$.

Step 3 And a suitable integer i such that $xy^iz \notin L$. This contradicts our assumption. Hence L is not regular.

Q Show that $L = \{a^i b^i \mid i \geq 1\}$ is not regular

Soln Step 1 Suppose L is regular. Let n be the no of states in finite automaton accepting L.

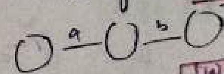
Step 2 Let $w = \cancel{a^i b^i}$.



Then put $n = 1$

$w = ab$.

No of states $\boxed{n = 3}$.



$|w| \leq n$ $\quad \boxed{|w| \geq n}$

Choose w such that $|w| \geq n$ $\quad \boxed{n = 3}$

$w = aabb$ $\quad \boxed{|w| = 4}$

xy^iz

$w = xyz$ $\quad \boxed{|xy| \leq n}$

$= aabb$

$x = a \quad z = bb$

$y = a$

$xy^iz = aa^ibb$

$= abb$

$w = xyz$

$= aabb$

$x = a \quad z = bb$

$y = a$

$|xy| \leq n$

$|2| \leq 3$.

Step 3 $w = xy^iz$ $\quad$ Put $i = 0$

$xz = abb \Rightarrow$ which is not of form $a^n b^n$

Put $y = i = 2$

$xy^iz \Rightarrow xyyz$

aaabb.

$xy^2z \notin L$.

$\therefore$ L is not regular

Q2 Show that $L = \{ a^{i^2} \mid i \geq 1 \}$ is not regular

Soln Step1 Assume that L is regular. Let n be no of states in the corresponding finite automata

Step2 Put $i=1$ $a^{1^2} = a$

Put $i=2$ $a^{2^2} = a^4 = \boxed{aaaa}$

$\boxed{n=5}$

Choose w such that $|w| \geq n$

Put $i=3$

$w = a^{3^2} = a^9 =$ aaaaaaaaa

Choose $x = aa$

$y = aa$

$z = aaaaa$

$|xy| = 4$ $|xy| \leq 5$.

$\boxed{4 \leq 5}$

Step3 $w = xy^iz$ $i=0$

$xz =$ aa aaaaa

$= a^7$

which is not of form a^{i^2} So, given L is not regular.

Q Show that $L = \{ a^p \mid p \text{ is prime} \}$ is not regular

Step1 We suppose L is regular

Step2 Let p be a prime no. greater than n.

$p = 3$. $p = 2, 3, 5, 7, 11$

$a^3 =$ aaa $\Rightarrow$ $\boxed{n=4}$

Choose w such that $|w| \geq n$

~~w=5~~. Put $p=5$

$a^5 \Rightarrow$ aaaaa

$\boxed{|w| = 5}$

Step3 Choose xy^iz $\Rightarrow$ $i=0$

~~$xz =$ $xz =$~~

$xy^0z = xz$

$=$ a aa

$w = xyz$

$=$ aaaaa

Pumping lemma

$x = a$

$y = aa$ $z = aa$.

$|xy| \leq n$.

$3 \leq 4$.

Put $y=1$ xyz $\Rightarrow$ aaaaa.

Put $y=2$ $xyyz =$ aaaaaaa

$xyyyz =$ a aa aa aa aa $\Rightarrow a^9$.

9 is not prime no.

Q Show that $L = \{ww \mid w \in \{a, b\}^*\}$ is not reg

Soln. Let $w = aba$ $n = 4$.

$ww = aba\,aba$

~~n = 7~~ ~~No. of~~ states

Choose ~~this~~ ww

$w = xyz$

$x = ab$ $y = a$ $z = aba$.

$\boxed{|xy| \le n}$

$w = xy^iz$

$= ab.aba$ ⇒ which is not of form ww

Pumping Lemma the term pumping lemma is made up of two words one is "pumping" and second is "lemma". The word pumping means to generate many input strings by pushing a symbol in an input string again & again. The word "lemma" refers to intermediate theorem in a proof. Pumping lemma is used to prove that given language is not regular.

A language is regular if

(1) Language is accepted by finite automata

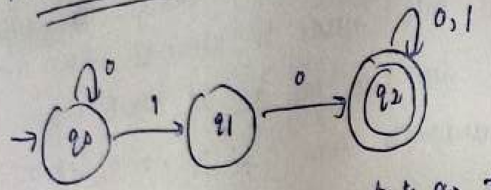
(2) A regular grammar can be constructed to exactly generate strings in language

(3) A regular expression can be " " " "

consider the language $L = \{a^n b^n \mid n \ge 0\}$

Finite automata has very limited memory. To accept this language, the m/c needs to remember how many a's have been seen ~~so~~ far as it reads input. Because finite automata has finite no. of states but no of a's are unlimited. so, m/c needs to keep track of unlimited no. of possibilities But finite automata can't do so because of its memory.

Finite automata with ϵ-Moves



we can't move from state q_0 to q_1 without using input symbol.

Finite automata with $\wedge$ moves allows transition from one state to another without consuming a symbol.

$\wedge$ move is an extension of finite automata that allows moves on empty input

Formal Defn of ϵ-NFA

is five tuple $A = (Q, \Sigma, q_0, F, \delta)$ be a non-deterministic finite automata with ϵ-NFA

$Q \to$ finite set of states

$\Sigma \to$ finite set of input symbols

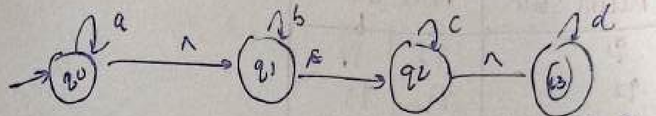
$q_0 \to$ initial state

$q_0 \in Q$

$F \to$ set of final states

δ is the transition fn which takes as input a state and input symbol & returns set of states o/p.

$Q \times \Sigma^* \to 2^Q$ (2^Q is power set of Q)



To find $\wedge$-closure of state q_0

1) add q to E

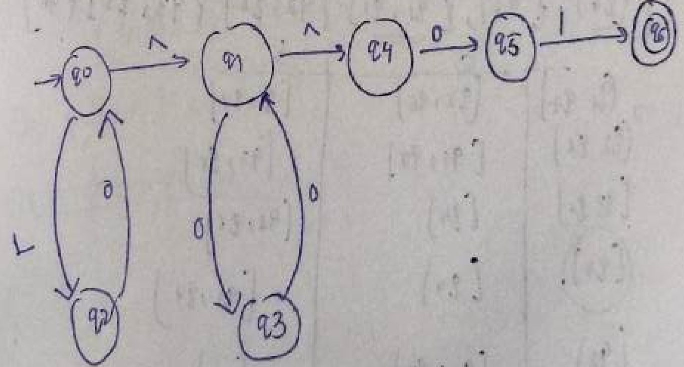
$E = \{q_0\}$

2) find all the states that are reachable from q with label $\wedge$

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

$\delta(q_1, \wedge) = q_2$

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



Minimization

Present State	Next State a	Next State b
→ q1	q2	q6
q2	q1	q3
q3	q4	q2 X
q4	q4	q1 X
q5	q4	q6 X
q6	q7	q5
q7	q6	q7
q8	q7	q4 X

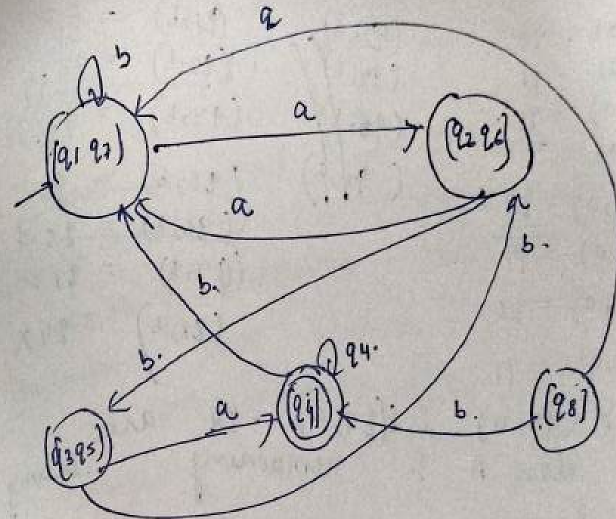
Sol^n $\Pi_0 = \{q_4\}\ \{q_1, q_2, q_3, q_5, q_6, q_7, q_8\}$

$\Pi_1 = \{q_1, q_2, q_6, q_7\}\ \{q_3, q_5\}\ \{q_8\}\ \{q_4\}$

$\Pi_2 = \{\ \{q_4\}\{q_8\}\ \{q_1, q_7\}\ \{q_2, q_6\}\ \{q_3, q_5\}\ \{q_6\}\ \}$

$\Pi_3 = \{\ \{q_4\}, \{q_8\}, \{q_1, q_7\}\ \{q_2, q_6\}\ \{q_3, q_5\}\ \{q_6\}$

	a	b
→ [q1 q7]	[q2, q6]	[q1, q7]
[q2 q6]	[q1, q7]	[q3, q5]
[q3 q5]	[q4]	[q2, q6]
[q4]	[q4]	[q1, q7]
[q8]	[q1, q7]	[q4]



$Q_0 = \{q_1, q_7\}$

$(q_1, a) = (q_2, q_6) = (q_7, b) \Rightarrow$

$(q_7, a) = (q_1, q_7) = (q_7, b)$

$Q_1 = \{q_1, q_2, q_7\}\ q_6\}.$

~~$Q_1 = \{q_1, q_3, q_5, q_2,$~~

Q $\{q_4\}$ $\{q_1, q_2, q_3, q_5, q_6, q_7, q_8\}$
A B

$\delta(q_1, a) = q_2$ ... $(q_1, b) = q_1$
$\delta(q_2, a) = q_1$ ... $(q_2, b) = q_3$
$\delta(q_3, a) = q_4$ × ... $(q_3, b) = q_2$
$(\cancel{q_4, a})$ -
$\delta(q_5, a) = q_4$ × ... $(q_5, b) = q_6$
$\delta(q_6, a) = q_7$ ... $(q_6, b) = q_5$
$\delta(q_7, a) = q_6$ ... $(q_7, b) = q_7$
$\delta(q_8, a) = q_7$. ... $(q_8, b) = q_4$ ×

(q_1, b) (q_2, b) (q_6, b) (q_7, b)

$\delta(q_3, a)$ & $\delta(q_5, a) = q_4$ are belong to class A & remaining belong to class B.

$\{q_4\}_A$ $\{q_3, q_5\}_{B_1}$ $\{q_1, q_2, q_6, q_7, q_8\}$

$\{q_4\}_A$ $\{q_3, q_5\}_{B_1}$ $\{q_1, q_2, q_6, q_7\}_{B_2}$ $\{q_8\}_{B_{21}}$

$\delta(q_3, a) = q_4$ $\delta(q_3, b) = q_2$
$\delta(q_5, a) = q_4$ $\delta(q_5, b) = q_6$

$\{q_4\}$ $\{q_3, q_5\}$ $\{q_2, q_6\}$ $\{q_1, q_7\}$ $\{q_8\}$

$\delta(q_1, a) = q_2$ $\delta(q_1, b) = q_1$
$\delta(q_2, a) = q_1$ $\delta(q_2, b) = q_3$
$\delta(q_6, a) = q_7$ $\delta(q_6, b) = q_5$
$\delta(q_7, a) = q_6$ $\delta(q_7, b) = q_7$

1. What is Computation (TOC)

step-by-step solution to a given problem. eg multiply two numbers dictionary & search for a word. find a word in dictionary.

→ graph reachability problem:→ checking whether there is a path between two vertices in a graph.

Computational devices used to compute our solution

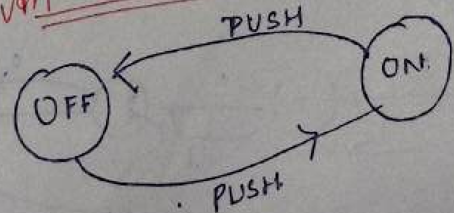
eg. ✓ calculator (computational device)
cell phone (smart phones)
computer, pen & paper

Study computational devices based on resources that use.

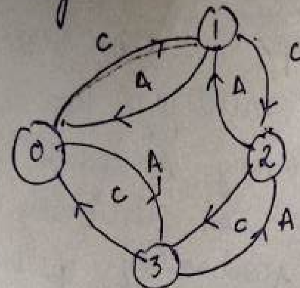
Finite Automata → has finite amount of memory.

Automata is the plural of word "automaton".

eg An electric switch (on or off)



2) Fan regulator



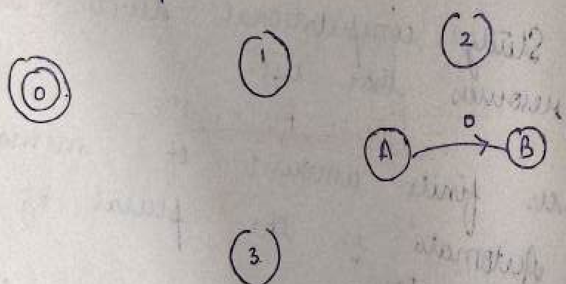
CCACCAC

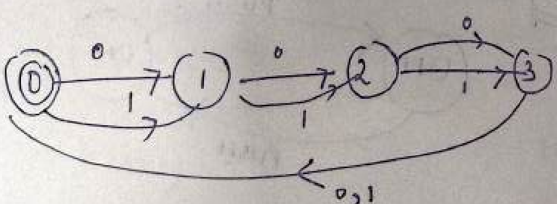
2 operations

4 states

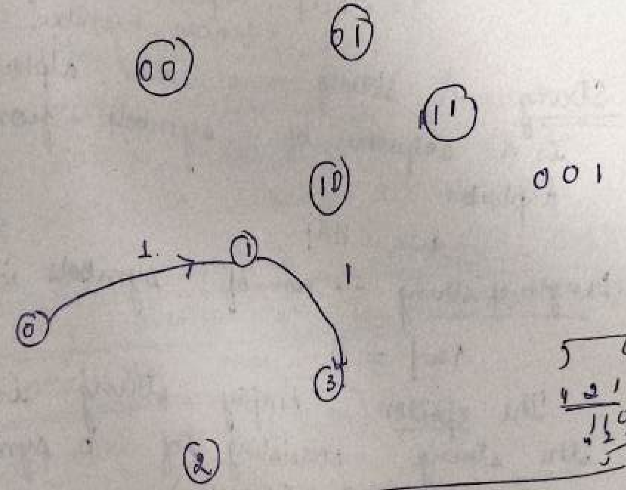
(3) $L = \{x \mid x$ is a binary string divisible by 4$\}$

B	Decimal	Belongs to L
100	4.	✓
110	6	×
1100	12	✓





Observe that set of binary no's divisible by 4 are exactly those that have 00 as a suffix.



No divisible by 4 or not

Definitions and Notations

An **alphabet** is a finite set of symbols. denoted by Σ

$\Sigma = \{0, 1\}$ - alphabet set of binary numbers.

String A string over an alphabet is a sequence of symbols from alphabet

$w = 01101$

Length of string → no. of symbols in string

$|w| = 5$

The **epsilon** → empty string is the string consisting of 0 symbols. denoted as ϵ (epsilon)

Kleen star

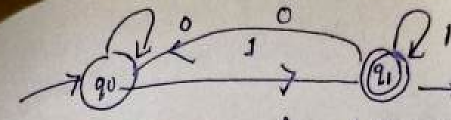
$\Sigma^* = \{w \mid w$ is a string over Σ and length of $|w| = 1\}$

Kleen Σ^* $\bigcup_{i \geq 0} \Sigma^i$

Language → A language L over an alphabet Σ, is a subset of Σ^*.

$L = \{w \mid w$ ~~has~~ ends with a $1\}$

$L = \{1, 01, 011, 101 \ldots$



→ final state

Definition :- A deterministic finite automaton is a 5 tuple

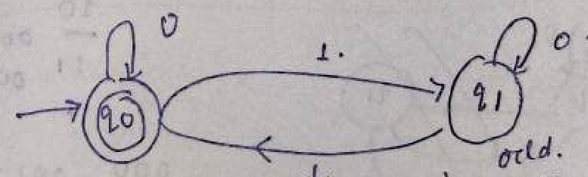
$M = (Q, \Sigma, \delta, q_0, F)$.

accepts a string w if starting at the state q_0, the DFA ends at an accept state on reading the string w.

* DFA M accepts a language $L \subseteq \Sigma^*$ if every string in L is accepted by M and no more.

denoted as $L(M)$

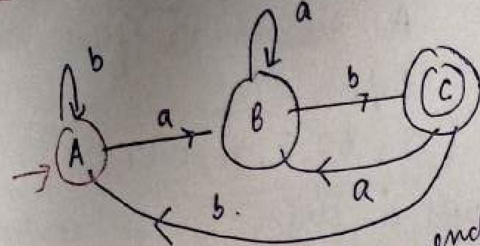
Q. $L_1 = \{w \in \{0,1\}^* \mid w$ has an even no. of 1's



$q_0 \rightarrow$ all strings having an even no of 1's

$q_1 \rightarrow$ all string odd no of 1's

Q2:- $L_2 = \{ w \in \{a,b\} \mid w$ ends with the substring $ab\}$



C → All strings that end with ab

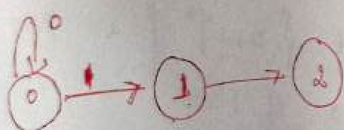
B → ________ a.

A → remaining substrings

Q3 $L_3 = \{ w \in \{0,1\}^* \mid w$ is divisible by 3 $\}$

~~((a+b)(a+b)(a+b))~~

0, 1, 2



000

001

010

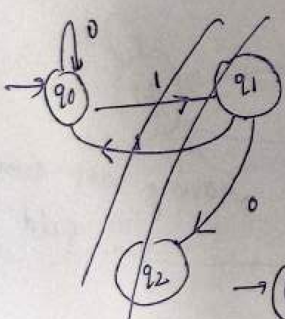
0

1

10 001

11 00001

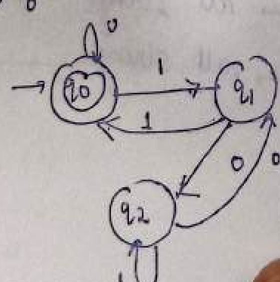
1



000 001

101
421

110
421



001

010
21

011

100

w mod 3 = 0

w mod 3 ≅ 1

w mod 3 ≅ 2

Q4 $L_4 = \{ w \in \{0,1\}^* \mid w$ begins & ends with same symbol $\}$

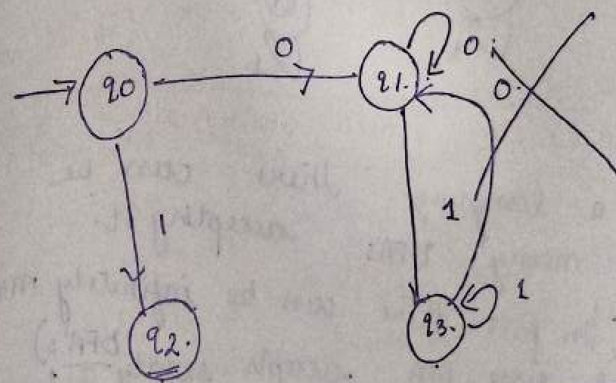
L = { q1 begins with 0 and end with 0 ✓

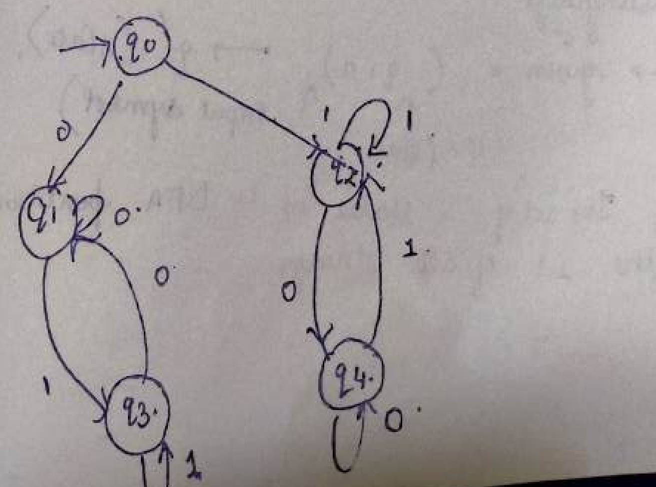
q3 beg — 0 and end with 1.

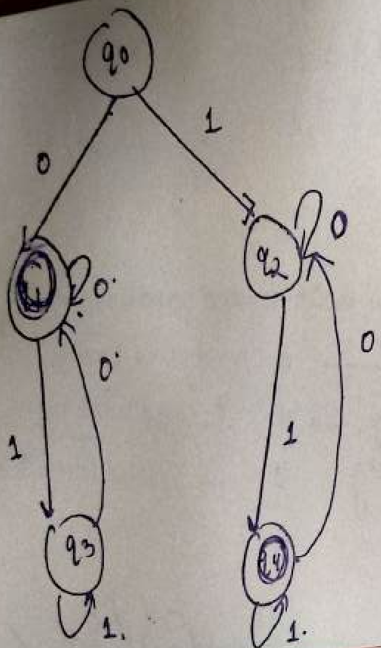
q4 beg → 1 & end with 0

q2 ~~beg~~ 1 —— end with 1.

Z.







$0 \to 0$
$0 - 1$
$1 \to 0$
$1 \to 1$

Deterministic finite automata

For a language there can be many DFAs accepting it.

(In fact there can be infinitely many DFA's)

But every DFA accepts exactly 1 language.

→ given a $(q, a) \longrightarrow q'$ (state).
(state) ↑ input symbol ↑

The set of states in DFA partitions the set of all strings

4) Computation by DFA and Regular operation.

Let $M = (Q, \Sigma, \delta, q_0, F)$ and let w be a string $w = a_1 a_2 \ldots a_n$ where $a_i \in \Sigma$.

we say that M **accepts w** if $\exists$ (there exists) a sequence of states $r_0, r_1, r_2 \ldots r_n$ such that

initial (1) $r_0 = q_0$.

transition cond (2) $r_i = \delta(r_{i-1}, a_i) \; \forall_i = 1, 2, \ldots n$.

accept cond (3) $r_n \in F$.

Notes: The states r_i need not be distinct.

Let $L \subseteq \Sigma^*$ we say that M accepts L if $L = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$

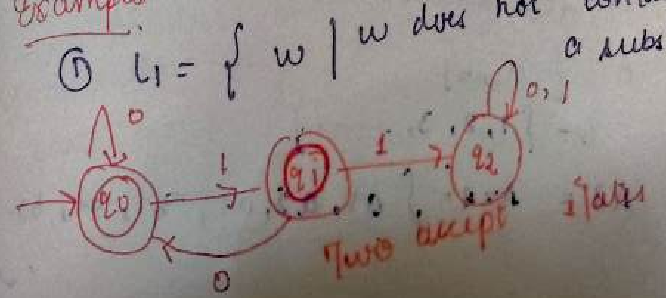
Let $L \subseteq \Sigma^*$ we say that L is regular if there exists a DFA M such that

$$L = L(M).$$

$\therefore L = \{0^n 1^n \mid n > 0\}$ is not regular

Example

(1) $L_1 = \{ w \mid w$ does not contain 11 as a substring.



Dump states from where the automaton can't reach an accept state ✓

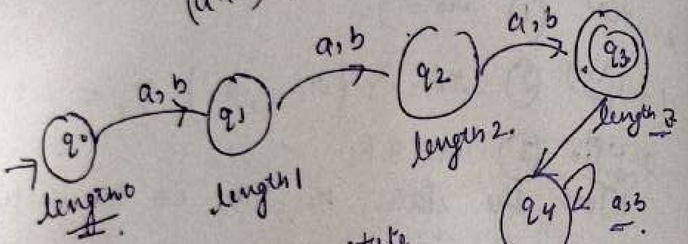
→0110

test → 101 ✓

100 → q_0 ✓

$L_2 = \{x \mid |x| = 3.\}$

$(a+b)(a+b)(a+b)$.



q_4 is a dump state

Regular operations Let $A, B \subseteq \Sigma^*$.

① Union $A \cup B$. $\{x \mid x \in A \text{ or } x \in B\}$.

② Concatenation

$A \cdot B = \{xy \mid x \in A \text{ and } y \in B\}$

also denoted as AB.

③ $A = \{a, b\}$

$B = \{1, 2, 3, \ldots\}$

$AB = A \times B = \{a1, a2, a3, b1, b2, b3 \ldots\}$

(3) Star Operation (unary operation)

$A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and } x_i \in A \text{ for all } i\}$

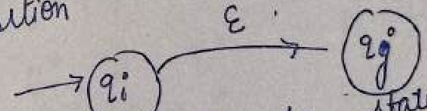
eg. $A = \{10, 001\}^*$.

$A^* = \{\Lambda, 10, 001, 1010, 10001, 00110, 001001.\}$

⑤ Non-Determinism → from a state q, on an input symbol a, the automaton can go to multiple states k $(k \geq 0)$

computation happens simultaneously along each of these paths

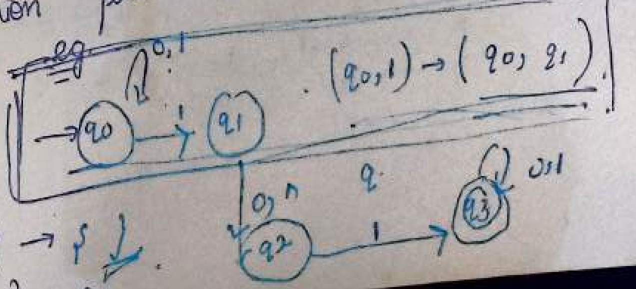
→ has ε-transitions → if there is a ε-transition



then, automaton moves to state q_i and q_j without ~~con~~ reading the next input bit

→ An input is accepted if there is some computation path that leads to an accept state

eg



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

$(q_1, 1) \to \{\ \}$

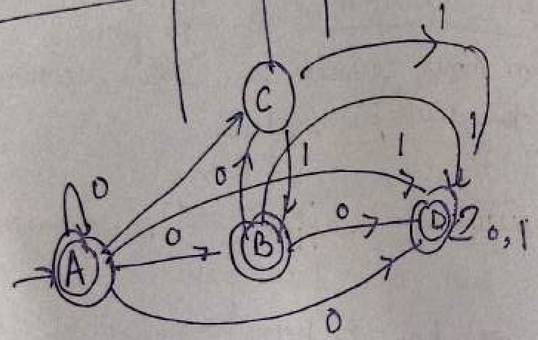
$(q_1, 0) \to q_2$

$(q_1, \varepsilon) \rightarrow \{q_1, q_2\}$ The transitions are labelled with symbols from the set

$$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$$

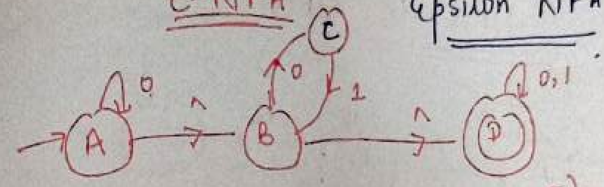
Consider an input $w = 010110$

States	ε	0	1
$\{q_0\}$		$\{q_0\}$	$\{q_0,$



ε-NFA

Epsilon NFA



$(Q, \Sigma, \delta, q_0, F)$.

$\delta: Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$.

Epsilon closure (A) = what are the states

$A \xrightarrow{\varepsilon} \{A, B, D\}$ $A = \{A,$

ε-NFA to NFA

	0	1
$\rightarrow$ A	$\{A, B, C, D\}$	$\{D\}$
B	$\{C, D\}$	$\{D\}$.
C	ϕ	$\{B, D\}$.
D	$\{D\}$	$\{D\}$.

ε^* 0 ε^*

D $D \rightarrow D \rightarrow D$

ε^* 1 ε^*

$D \rightarrow D \rightarrow D$

ε^* 0 ε^*

~~A A, B, D . A~~

~~$\delta(\varepsilon\text{-closure}(A))$~~

ε^* 0 ε^*

$B \rightarrow B \rightarrow C \rightarrow C$

$D \rightarrow D \rightarrow C$

ε^* 1 ε^*

$B \rightarrow B \rightarrow \phi$

$D \rightarrow D \rightarrow D$

ε^* 0 ε^*

$C \rightarrow C \rightarrow \phi \rightarrow \phi$

ε^* 0 ε^*

A A $\rightarrow$ A, B

B C $\rightarrow$ C

D $D \rightarrow D$.

ε^* 1 ε^*

A $\rightarrow$ ϕ

B $\rightarrow$ ϕ

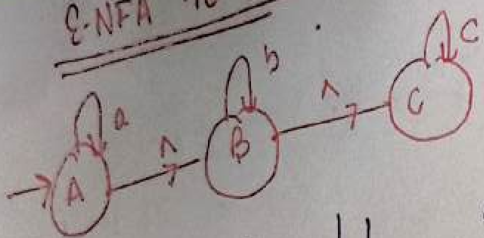
D $\rightarrow D \rightarrow D$

ε^* 1 ε^*

$C \rightarrow C \rightarrow B \rightarrow B$

D.

ε-NFA to NFA



	a	b	c
A	{A,B,C}	{B,C}	{C}
B	{~~A,B,C~~}	{B,C}	{C}
C	φ	φ	{C}

B: ε^* b ε^*

B → B → B, C

C → φ

ε^* a ε^*

B → B → φ → φ

C → φ

ε^* c ε^*

B → B → φ

C → C → C

ε^* a ε^*

A → A → A → A,B,C

B → φ → B,C

C → φ → C

ε^* b ε^*

A → A → φ

B → B → B, C

C → φ

ε^* c ε^*

A → A → φ

B → φ

C → C → C

ε^* a ε^*

C → C → φ → φ

ε^* b ε^*

C → C → φ → φ

Week 2

Lec 6.

Non-Deterministic Finite Automata

Difference between DFA & NFA

DFA	NDFA
① $(q, a) \to$ single state	① $(q, a) \to$ multiple states
② single computation	② multiple computation
③ no ε-transition	③ have Λ-transition
④ Accept if the computation ends at an accept state	④ accept if one of the computation paths end at an accept state

NFA is the five tuple

$N = (Q, \Sigma, \delta, q_0, F)$ $\delta: Q \times \Sigma_\varepsilon \to 2^Q$

power set of Q

Defn We say that N accepts an input $w = a_1 a_2 \ldots a_n$ if we can write w as $w = b_1 b_2 b_3 \ldots b_m$ where $b_i \in \Sigma_\varepsilon$ and there exists a seq. of states $r_0, r_1, \ldots r_m$ (not necessarily distinct) such that

① $r_0 = q_0$ (initial cond)

② $r_i \in \delta(r_{i-1}, b_i)$ for all $i = 1 \ldots m$.

?

punjab police recruitment in /punjab/

at

4th

Sab

3 → Buon

75

77

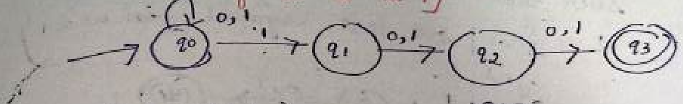
BCA

Age

$q_m \in F$ (acceptance condn)

$L(N) = \{ w \in \Sigma^* \mid N \text{ accepts } w \}$.

Examples → $L_1 = \{ w \in \{0,1\}^* \mid$ 3rd last symbol of w is a 1$\}$



$\delta(q_0, 1110) \vdash (q_0, q_1)$ 1010

$(q_0; 110)$ $(q_1, 110)$

$\{q_1, 10)$

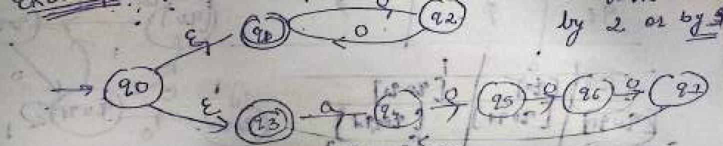
$\{q_2, 0)$

q_3

show that there exists a DFA for L.

$L_K = \{ w \in \{0,1\}^* \mid$ the Kth last bit is 1$\}$.

Example 2. $L_2 = \{ w \in \{0\}^* \mid |w|$ is divisible by 2 or by 3$\}$

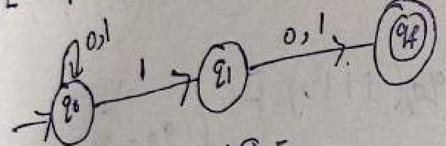


(7) lec Equivalence of NFA and DFA, Closure properties

Every DFA is also an NFA.

Theorem $N = (Q, \Sigma, \delta, q_0, F)$ is NFA. there exists a DFA $D = (Q', \Sigma', \delta', q_0', F')$ such that $L(N) = L(D)$.

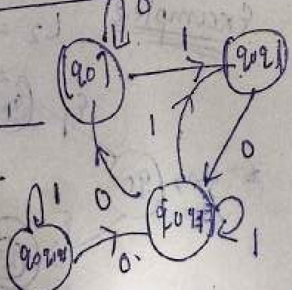
$L = \{ w \mid \text{2nd last symbol in } w = 1 \}$



$Q \; \delta$ -

Equivalent DFA

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



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

Regular Operations

Theorem:- If L_1 and L_2 are regular then $L_1 \cup L_2$, $L_1 \cdot L_2$ and L_1^* are regular.

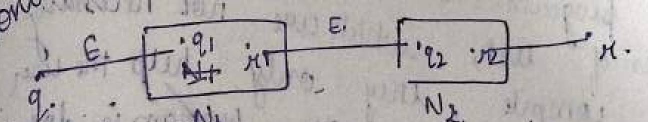
For a regular language, we can assume that there is NFA accepting it with a unique accept state.

Language	NFA	start state	accept state
L_1	N_1	q_1	r_1
L_2	N_2	q_2	r_2

$L_1 = L(N_1)$

$L_2 = L(N_2)$ $\quad L_1 \cup L_2$

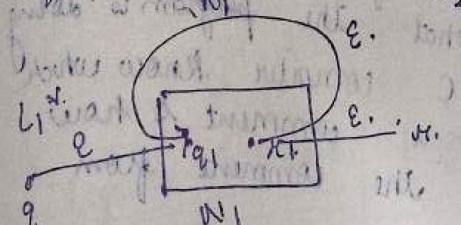
Concatenation



L_1

L_2

L_1^*



Lec 8

Regular Expressions

are an algebraic way to represent languages

representation of languages

① → many applications in text editors

the way the word application or the s/w goes about searching for a word in your document is by regular expression

② compiler design → when we write a C program, we put comments in C program. Comments are basically pieces of text, which are not necessary to compile, they only help the user understand what the program is doing

how does the C compiler know which part of text is a comment & how does it remove the comment from main program

→ comment → by putting comment is by giving slash

200 stay string 1^*

250 2° 6d 600

→ Search your entire program.

③ Examples of Regular Expression

Regular Expression	Language
(1) 0	$\{0\}$
(2) ε	$\{\varepsilon\}$
(3) $0 \cup 01$	$\{0, 01\}$
(4) 1^*	$\{\wedge, 1, 11, 111 \ldots\}$
(5) $(0 \cup 01) 1^*$ $(0+01) 1^*$	$\{0, 01, 01, 011, 01111 \ldots$

It is a finite expression and capable of generating infinite language.

$A \xrightarrow{a} B \xrightarrow{b} C$

$\{\boxed{ab}\}$

ab. $L = \{ab\}$

$(a+b)^* = \{\wedge, a, b, \ldots\}$

Defn R is said to be a regular expression (RE) if R has one of the following forms :-

① Regular Expression	Language	Comment
1) ϕ	$\{\}$	
2) ε	$\{\varepsilon\}$	
3) a	$\{a\}$	$a \in \Sigma$.
4) $R_1 \cup R_2$	$L(R_1) \cup L(R_2)$	
5) $R_1 \cdot R_2$	$L(R_1) \cdot L(R_2)$	
6) R_1^*	$(L(R_1))^*$	
7) (R_1)	$L(R_1)$	

Note for every regular expression R there is a unique language L(R) corresponding to it but the converse is not true

Precedence of the symbols

(1) ( )
(2) *
(3) .
(4) +

$\phi^* = \{\varepsilon\}$

RE	Language
① 01	{01}
② 01 + 1	{01, 1}
③ $(01 + \Lambda)1$	{011, 1}
④ $(0+10)^*(\varepsilon+1)$	{Λ, 1, 0, 01, 10, 101, 00, 001, 010, 0101, 100, 1001, 1010, 10101 ---}

Examples (lang) → RE

Language	Regular Expression
① {w | w has a single 1}	0^*10^*
② {w | w has atmost a single 1}	$(0^* + 0^*10^*)$
③ {w | |w| is divisible by 3}	$((0+1)(0+1)(0+1))^*$
④ {w | w has a 1 at every odd position and length of |w| is odd}	$1((0+1)1)^*$

Algebraic properties of RE

lec 9

① (a) $R_1 + (R_2 + R_3) = (R_1 + R_2) + R_3$

(b) $R_1 (R_2 R_3) = (R_1 R_2) R_3$

② (a) $(R_1 + R_2) \cdot R_3 = R_1 R_3 + R_2 R_3$

(b) $R_1 (R_2 + R_3) = R_1 R_2 + R_1 R_3$

Commutativity

③ $R_1 + R_2 = R_2 + R_1$ (only + operation is commutative)

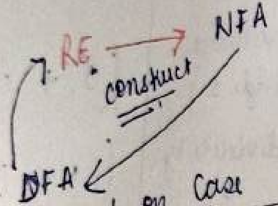
④ (a) $R_1 + \phi = R_1$

(b) $R_1 \cdot \varepsilon = R_1$

⑤ $(R_1^*)^* = R_1^*$

Theorem A language L is regular if and only if there is a regular expression

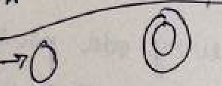
$L = L(R)$



Proof is based on Case Analysis

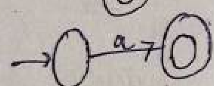
RE — NFA

1. ϕ

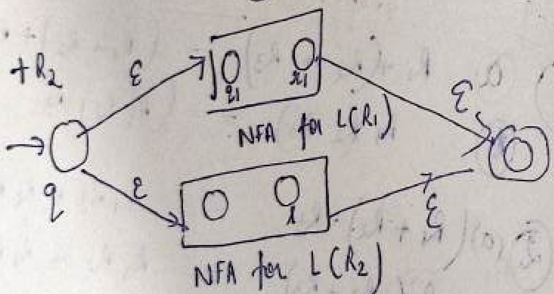


2. ε

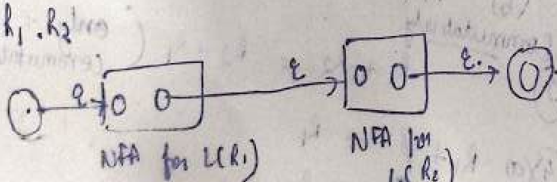
3. a



4. $R_1 + R_2$

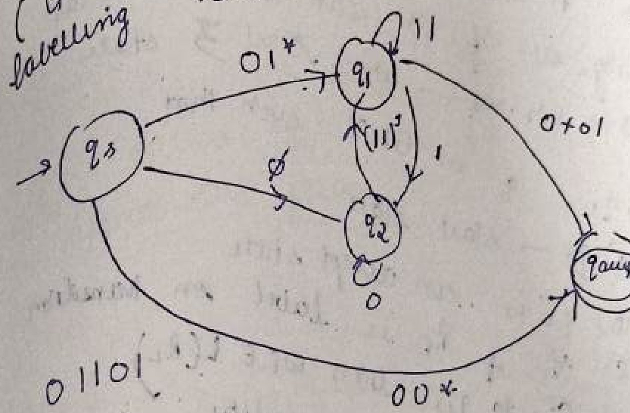


5. $R_1 . R_2$ (concatenation)



R_1^*

A generalized non-deterministic finite automaton (GNFA) is a NFA that has regular expressions labelling its transitions.



01101

$q_s \xrightarrow{011} q_1 \xrightarrow{01} q_{accept}$

$q_s \xrightarrow{0} q_1 \xrightarrow{11} q_1 \xrightarrow{01} q_{accept}$

(2) → 10 →, no way of going from q_s to q_{accept} forming string 10.

Lec10 Converting a DFA to RE

a GNFA is an NFA with the transitions being labelled by regular expression.

Defⁿ A GNFA is said to accept a string w if w can be written as $w = w_1 \cdot w_2 \cdots w_K$ and $\exists$ states $q_0, q_1 \ldots q$ in GNFA such that

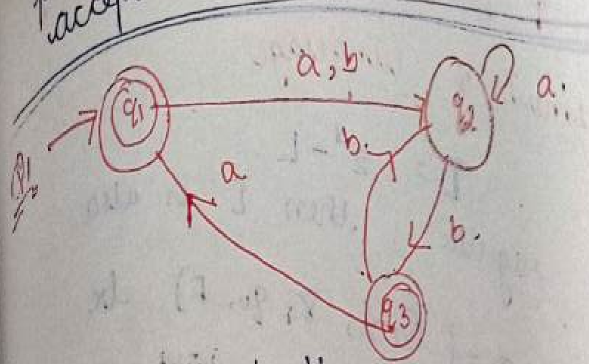
(a) $q_0 \rightarrow$ start state

(b) q_K is an accept state

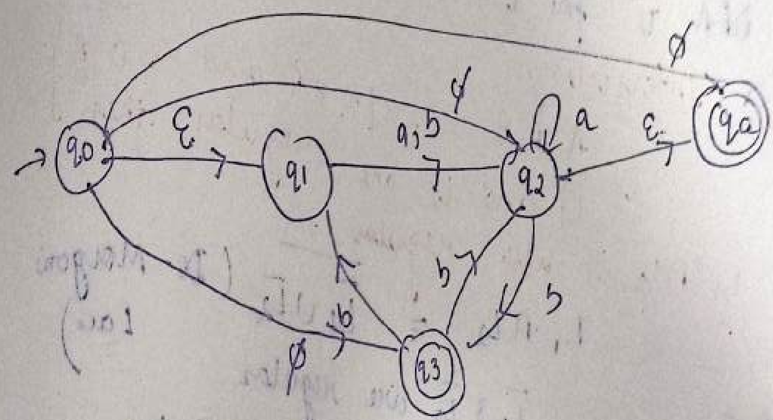
(c) $\forall_i$ if R_i is label on transition q_{i-1} to q_i, then $w_i \in L(R_i)$

Without loss of generality, we may assume the following:

1. GNFA has a unique accept state
2. There are no incoming transitions to the start state and no outgoing tx from the accept state
3. There are transitions from start state to every other state and from every state to the accept state.
4. There is a transition between every pair of states that are not the start/accept state.



1. Convert the DFA to GNFA to appropriate form



Week3 Lec 11

Closure properties of Regular Languages

(1) complement of language

$$\bar{L} = \Sigma^* - L$$

If L is regular then $\bar{L}$ is also regular. Let $D = (Q, \Sigma, \delta, q_0, F)$ be some DFA for L. We construct a DFA D' for $\bar{L}$ where, $D' = (Q, \Sigma, \delta, q_0, Q-F)$

(2) Intersection

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

If L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular

$$L_1 \cap L_2 = \overline{\bar{L_1} \cup \bar{L_2}} \quad \text{(De-Morgan's Law)}$$

$A \cap B = \overline{\bar{A} \cup \bar{B}}$

$\bar{L_1}$ & $\bar{L_2}$ are regular

$\rightarrow \bar{L_1} \cup \bar{L_2}$ is also regular

$\Rightarrow \overline{\bar{L_1} \cup \bar{L_2}}$ is regular

(3) Set difference :-

$$A - B = \{x \in A \mid x \in A \text{ and } x \notin B\}$$

$$A - B = A \cap \bar{B}$$

If L_1 & L_2 are regular then $L_1 - L_2$ is also regular.

$L_1 - L_2 = L_1 \cap \bar{L_2}$

$L_2 - L_1 = L_2 \cap \bar{L_1}$ is also regular

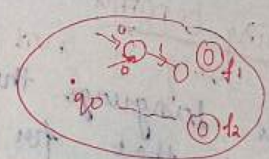
Reversal of Language

Let $w = a_1 a_2 \dots a_n$ be a string then reverse of w is a string $rev(w) = a_n a_{n-1} \dots a_1$

for a language $L \subseteq \Sigma^*$

$$rev(L) = \{w \in \Sigma^* \mid rev(w) \in L\}$$

If L is regular then $rev(L)$ is also regular. Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA for L.



Homomorphism

Let Σ, Γ be two alphabets

homomorphism $h : \Sigma \rightarrow \Gamma^*$ (every symbol in Σ it gives a string in Γ)

$w = a_1 a_2 \dots a_n$ be a string in Σ^*

Let $w \in \Sigma^*$ then $h(w) =$

$$h(w) = h(a_1) \cdot h(a_2) \dots h(a_n)$$

Let $L \subseteq \Sigma^*$

$$h(L) = \{h(w) \in \Gamma^* \mid w \in L\}$$

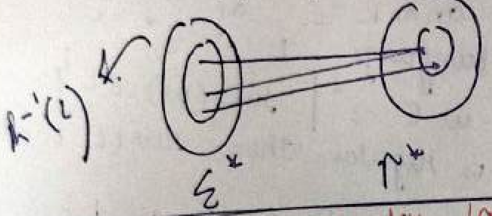
If L is regular then $h(L)$ is also regular.

Inverse Homomorphism

$h : \Sigma \to \Gamma^*$ is a homomorphism

$\to L \subseteq \Gamma^*$, define

$$h^{-1}(L) = \{ w \in \Sigma^* \mid h(w) \in L \}$$



Lec 12

Non-regular Languages

which requires some sort of counting.

Pumping Lemma

To prove that lang is not regular

If L is a regular language there exists $p \geq 0$, such that for all strings w where $|w| \geq p$ there exists a partition

$|w| \geq p$.
$|w| = xyz$
$|xy| \leq p$
and $|y| > 0$.
& for $i \geq 0$,
$xy^iz \in L$

$A \Rightarrow B$

then $\sim B \Rightarrow \sim A$ (Contrapositive form).

two is even

Example

$$L = \{ 0^n 1^n \mid n \geq 0 \}$$

Soln ~~Let p =~~

$\boxed{n=3}$ $w = 0011$ $n = 2$.

$w = 000111$ $\boxed{p = 5}$.

$w = xyz$. $|w| = 6$. $p = 5$.

$w = 000111$ $|xy| \leq p$. $|xy| \leq 5$

$x = 00$

$y = 01$

$z = 11$

$w = \underbrace{00}_{x}\,\underbrace{01}_{y}\,\underbrace{11}_{z}$

$xy^iz =$ $i=0 \Rightarrow xz = 0011$

$i=1 \Rightarrow xyz = 000111$

$i=2$ $xy^2z = 00\,0101\,11$

$xy^2z \notin L$.

Lec 13 More examples of non-regular languages

(1) $L_1 = \{0^n 1^n \mid n \geq 0\}$ is not regular

(2) $L_2 = \{a^l b^m c^n \mid l+m \leq n\}$

$l = 1$
$m = 1$
$c = 2$

~~{a~~

(3) $L_3 = \{ w \in \{0,1\}^*$ no of 0's = no of 1's in w $\}$

$L \cdot (0^* 1^*) \cap L_3 =$

(4) $L \cap L' = L''$. If L is regular and L' is not regular it does not imply that L'' is not regular

(5) If L and L' are non-regular even then L'' may be regular

*

$L_4 = \{0^p \mid p$ is prime$\}$ is not regular

$\{2, 3, 5, \ldots \ldots$

$01|01$ $L_1 \cup L_2$ (6)

$(01)^*$ $(L_1 \cup L_2)^2$

$01|01|01$

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

$0+11$ $\{0, 11\}$ $\Sigma^2 = \{aa,$ (4)

$\{0^*, 1.0^*\} \cup \{0^* 1^* 0\}$ 2^2 $4^4 = 256$

$00|00|01$

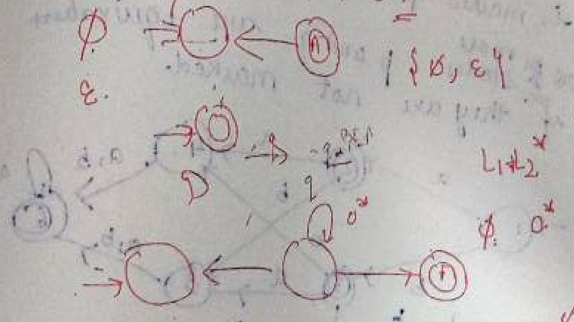
$L_1 L_2^* \cup L_1^*$ $\{1\,00\,101, 000\}$ L_1 L_2

$\phi\, 0^* \Rightarrow \phi$

$\phi \cup \phi^*$ $111\,0\,101$

ϕ 101

ε $\{\phi, \varepsilon\}$



$L_1 L_2^*$

$\phi \cdot 0^* \to \phi$

$\phi \cup \phi^*$

$\phi^* \to \varepsilon$ $\boxed{\phi \cup \phi^2 \cup \varepsilon}$ $\phi, \{\varepsilon\}$

NDF → q plus

$L_1 = \{4, 5, 6, 7\}$ ϕ

$L_2 = \{0, 1, 2\}$

(9)

Lec14

DFA minimization

DFA minimization algorithm that produces a minimal DFA from given DFA.

Algorithm

Input: $D = (Q, \Sigma, \delta, q_0, F)$

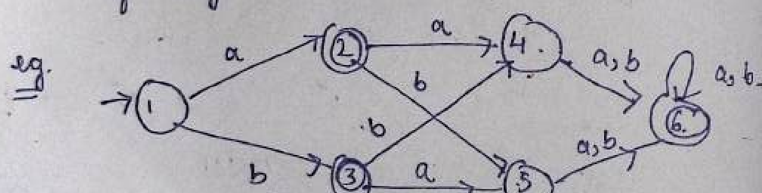
(1) Construct a table of all $\{p, q\}$ pairs where p, q are $\in Q$

(2) Mark a pair $\{p, q\}$ if $p \in F$ and $q \notin F$ or vice versa

(3) Repeat the following until no more pairs can be marked.
Mark $\{p, q\}$ if $\{\delta(p,a)$ & $\delta(q,a)\}$ is marked for some $a \in \Sigma$.

4) Two or more states p and q are equivalent if they are not marked.

eg.



	a	b
1	2	3
2	4	5
3	5	4
4	6	6
5	6	6
6	6	6

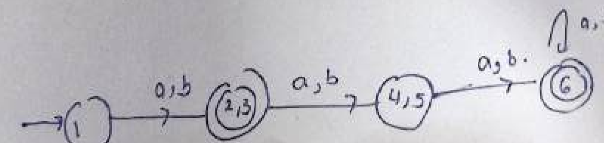
2,3,6

	1	2	3	4	5
2	X				
3	X				
4		X	X		
5		X	X		
6	X			X	X

$\Pi_0 = \{1, 4, 5\}, \{2, 3, 6\}$

$\Pi_1 = \{1, 4, 5\}; \{2, 3\} \{6\}$

$\Pi_2 = \{1\} \{4, 5\}, \{2, 3\}, \{6\}$



Minimum Automata

State / Σ	0	1
→ q0	q1	q5
q1	q6	q2
q2	q0	q2
q3	q2	q6
q4	q7	q5
q5	q2	q6
q6	q6	q4
q7	q6	q2

$\Pi_0 = \{\{q_2\}, \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}\}$

$\Pi_1 = \{q_2\}, \{q_0, q_4, q_6\} \{q_3, q_5\} \{q_1, q_7\}$

$\Pi_2 = \{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_3, q_5\} \{q_1, q_7\}$

$\Pi_3 = \{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_3, q_5\}, \{q_1, q_7\}$

Σ	a	b
q0	q1	q0
q1	q0	q2
q2	q3	q1
q3	q3	q0
q4	q3	q5
q5	q6	q4
q6	q5	q6
q7	q6	q3

$\Pi_0 \ \{q_3\} \ \{q_0, q_6\} \ \{q_1, q_5\} \ \{q_2, q_4\} \ \{q_7\}$

q0

q1

q2

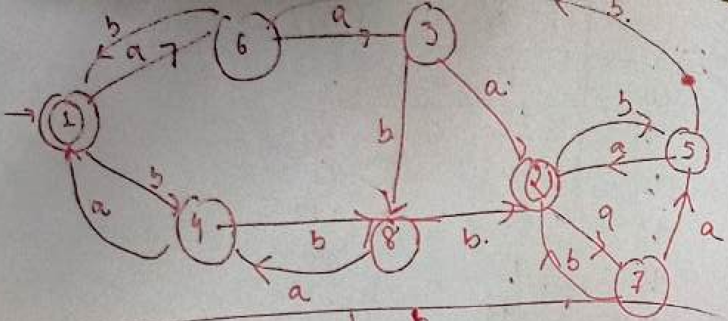
q3

q4

q5

q6

q7

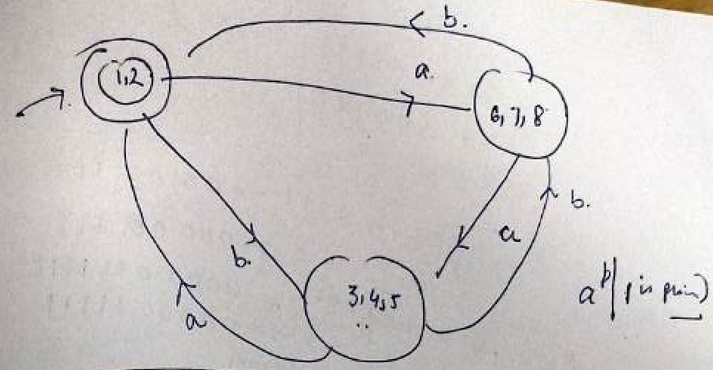


State	a	b
1	6	4
2	7	5
3	2	8
4	1	8
5	2	6
6	3	1
7	5	2
8	4	2

$\Pi_0 = \{1,2\}\ \{3,4,5,6,7,8\}$

$\Pi_1 = \{1,2\}\ \{3,4,5\},\ \{6,7,8\}$

$\Pi_2 = \{1,2\}\ \{3,4,5\},\ \{6,7,8\}$



a^b / (in plain)

Lec15 Introduction to CFGs

Recognize a larger class of languages than regular languages.

→ Applications in programming languages & compiler design

→ Terminal symbols. similar to the alphabet symbols.

→ Variable symbols → can be replaced with a string of variables & terminals.

Production Rules →.

Start variable → starting point of the computation.

$S \to 0S1$
$\to 00S11$
$\to 0011$

eg

Terminals $\{0,1\}$

Variables $\{S\}$

Start Variable :- S.

$S \to 0S1$

$S \to \Lambda$.

$\to 00S11$

$\to 000S111$

$\to 000111$

$w = 0^5 1^5$

$S \to 0S1 \to 00S11 \to 000S111$

$\to 0000S1111$

Hence, w is accepted. $\to 00000S11111$

$\to 0000011111.$

Every string of the form

$0^n 1^n \mid n \geq 0$ is accepted by grammar

Example 2. Terminals:- $\{0, 1\}$

Variables:- $\{S, A, B\}$

$S \to ASB \mid \Lambda$

$A \to 0$

$B \to 11$

$S \to ASB$

$\to 0SB$ $\longrightarrow$ $0SB$

$\to 0B$ $\to 0ASBB$

$\to 011$ $\to 0ABB$

$\xrightarrow{B \to 11} 0A11B$

$L = \{0^n 1^{2n} \mid n \geq 0\}$ $\xrightarrow{B \to 11} 0A1111$

$\to 001111$

Lec 16

Examples of CFGs, Reg

Defn - A CFG is a 4 tuple

$(V, \Sigma, P, S) \longrightarrow$ Start variable

Variable, Input alphabet

$\to S$

$P \subseteq V \times \{V \cup \Sigma\}^*$

Let $A \to w$ be a production rule.

Let $u, v \in \{V \cup \Sigma\}^*$

Then we say that the string uAv yields uwv in one step

For

$u \overset{*}{\Rightarrow} v$ (in zero or more steps)

$L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$

A language L is said to be a CFL if there exists CFG,

$L = L(G)$

eg $L = \{a^n b^m \mid n \geq 0, \quad n \leq m \leq 2n\}$

$S \to ASB \mid \Lambda$

$A \to a$

$B \to bb \mid b$

(2) $L_2 = \{w \in \{0,1\}^* \mid$ no. of 0's in w $=$ no. of 1's in w.

$w = 0 \ldots 1 \ldots$ $S \to 0S1S \mid 1S0S \mid \Lambda$

(3) Language of Balanced parenthesis
→ string over '(' and ')' such that they are balanced and well matched.

(( )) ( )

( ) ( ) ( )
11 22 33

$L_3 = \{ w \in \{ (,) \}^* \mid w \text{ is balanced} \}$

$S \to (S) \mid SS \mid \wedge$.

$S \to SS$
$\to (S)S$
$\Rightarrow (S)S$
$\Rightarrow ((S))S$
$\Rightarrow (())S$
$\Rightarrow (())(S)$
$\Rightarrow (())()$

Regular languages are context free

Let L be a regular language.
$\exists$ a DFA $D = (Q, \Sigma, \delta, q_0, F)$
such that $L = L(D)$

Construct a CFG.
$V = \{ R_i \mid q_i \in Q \}$

$R_i \to a R_j$
$q_i \in F$, then add the rule
$q_i \to \wedge$.

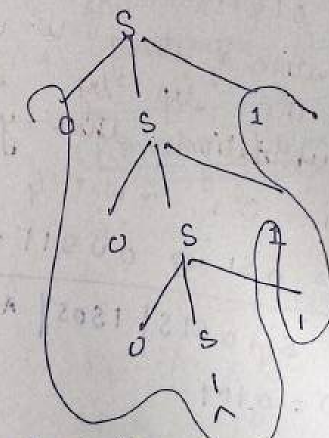
Lec 17 Parse trees and Ambiguity

Let G be a CFG and w be a string $\in L(G)$. A parse tree of w with respect to G is a rooted, ordered tree that represents the derivation of w. syntactic structure of w

eg $G_1 \longrightarrow S \to 0S1 \mid \wedge$.

$w = 000111$

000111.



Some properties
→ Every internal node is a variable
→ Every leaf node is either a terminal or ε.
If it is $\wedge$, then it is the only child of its parent.

Concatenation of the leaves of parse tree from left to right gives the string.

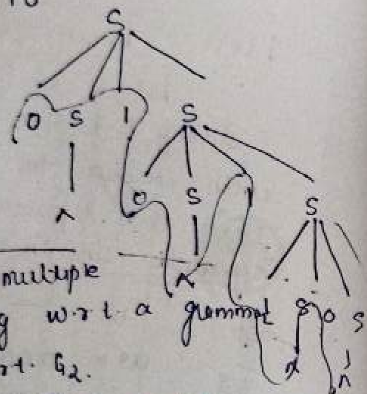
$S \to 0S1S \mid 1S0S \mid \varepsilon$.

$w = 010110$

$S \to 0S1S$

$\Rightarrow 01S0S$

$\Rightarrow 01$.



There may exist multiple parse trees for a string w.r.t a grammar

eg 010110 w.r.t. G_2.

The derivation of string w w.r.t a grammar G is the step by step (seq. of substitutions that yields w).

eg $w = 0^2 1^2$ w.r.t G

$S \to 0S1 \Rightarrow 00S11 \Rightarrow 0011$.

$G_2 \Rightarrow S \to 0S1S \mid 1S0S \mid \Lambda$

$w = 0101$

$S \to 0S1S$

$\Rightarrow 01S$

$\Rightarrow 010S1S$

$\Rightarrow 0101$

$S \to 0S1S$

$\Rightarrow 0S10S1S$

$\Rightarrow 0101$.

$S \to 0S1S$

$\Rightarrow 01S0S1S$

$\Rightarrow 0101$

A leftmost derivation of w. w.r.t G is a derivation where in each step the left most variable of a string gets replaced.

eg. derivation 1 & 3

A string $w \in L(G)$ if it has two leftmost derivations for same string we say a grammar G is ambiguous if $\exists$ some ambiguous string $\in L(G)$.

Lec 18 **Chomsky Normal Form**

A CFG $G = (V, \Sigma, P, S)$ is said to be in Chomsky Normal form (in CNF) if every production rule has

1) $A \to BC$ where $B, C \to$ variables

2) or $A \to a$ where $a \in \Sigma$.

3) S does not appear on r.h.s of any rule.

4) $S \to \Lambda$ may be present depending on whether $\varepsilon \in L(G)$ or not

Q Converting a CFG $G = (V, \Sigma, P, S)$ to a in CNF.

Removing Null

$A \to \Lambda$

$B \to uAv$

$C \to u_1 A u_2 A u_3$

9

$B \to uAv \mid uv$

$C \to u_1 A u_2 A u_3 \mid u_1 A u_2 u_3 \mid u_1 u_2 A u_3 \mid u_1 u_2 u_3$

Removing unit productions

① $A \to B$
$B \to u$

$B \to u$
$A \to u$

④ shortening the RHS

$A \to u_1 . u_2 \ldots u_K \quad (K \geq 3)$

③ Add variables

$A \to uv$
$A \to U_1 V_1$

$U_1 \to u$
$V_1 \to v$

---

eg Q1:
$S \to ASB$
$A \to aASA \mid a \mid \Lambda$
$B \to SbS \mid A \mid bb$

Soln

$S_0 \to S$
$S \to ASB$
$A \to aASA \mid a \mid \Lambda$
$B \to SbS \mid A \mid bb.$

Remove Null Transitions

$A \to \Lambda$

$S_0 \to S$
$S \to ASB \mid SB$
$A \to aASA \mid aSA \mid aAS \mid aS \mid a$
$B \to SbS \mid bb \mid \Lambda \mid A$

$B \to \Lambda$.

$S_0 \to S$
$S \to ASB \mid AS \mid SB$
$A \to aASA \mid aSA \mid aAS \mid aS \mid a$
$B \to SbS \mid bb \mid A.$

Remove Unit Productions

$S_0 \to ASB \mid AS \mid SB$
$S \to ASB \mid AS \mid SB$
$B \to SbS \mid bb \mid aASA \mid aSA \mid aAS \mid aS$

Non-CFLs, pumping Lemma

$w \in L$

$w = uv^i xy^i z$

If max. degree of T is d and height of T is h, then $|w| \le d^h$.

Examples of non CFLs

$w = uvxyz$

$|vxy| \le p$ & $|vy| > 0$.

$L = \{a^n b^n c^n \mid n \ge 0\}$

(2) $L = \{ww \mid w \in \Sigma^*\}$

$S \to AB \mid BA \mid A \mid B$

$A \to aAa \mid aAb \mid bAa \mid bAb \mid a$

$B \to aBa \mid aBb \mid bBa \mid bBb \mid b$.

$S \to AB$

$S \to aAaB$

$S \to aaaaBa$

$\Rightarrow aaaaba$

aaaba

abaaa

$S \to BA$

$\to aBaA$

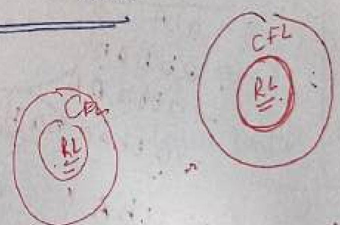
$\Rightarrow abaa$

aaba

(1) Union of R and CFL $\to$ CFL

(2) Inter $R \cap CFL \to CFL$

(3) RL . CFL $\to$ CFL.



$|vy| \le n$

$0^* 1^*$

$L_1 = a^* b^*$

$L_2 = a^n b^n$

u v x y z
ab ab ba

$a^n b^n$

$uv^i xy^i z$

abb

aaabbb

(a) a ba

a a ba

X

eg ab ab ba

$w = uvxyz$

$w = uv^i xy^i z$

$\boxed{n = 7}$

$|w| \ge$

$|w| = 4$.

$|w| \ge n$.

a b a x y z
u v
ab ab ba

w =

abab $ ba.
u v x y z

~~abab$ba~~

uxz ⇒ aabba

uv^ixy^iz

90/45
77

a bbababB.$

S ⇒ ab baB

S ⇒ ab b a bb..

S → abA
A → baB
B → aA | bb

S → abA
→ abbaB
→ abbaaA
⇒ abbaa baB
⇒ abbaa babb

S → abA
A → baB
B → aA | bb

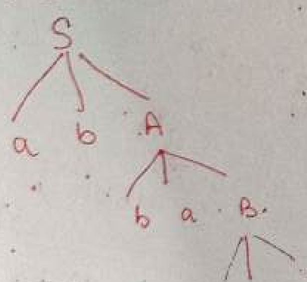
① 90 ✓
② 77 ✓
48

S → abA
⇒ abbaB
→ abbaaA
→ abbaa babb

c

$0^n \$ 0^n 0^n \$$

S → abA



$0^n 1^n 0^n 1^n \mid n \geq 0$

$a^n b^n a^n b^n$

Lec 21

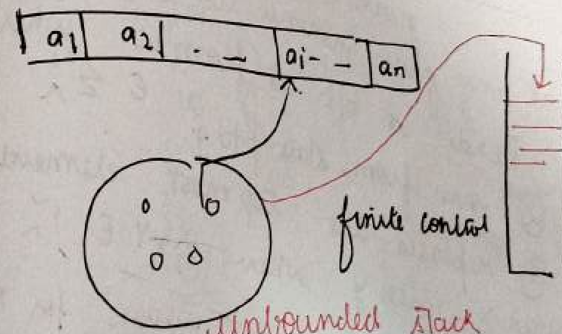
PDA

ε-NFA + stack

Push operation → adds an element to top of stack

Pop → removes the top most element

A pushdown automaton has a input tape, a finite set of states and an (un)bounded stack → the stack can store an arbitrary amount of info



Unbounded stack

PDA

At any given point of time, the pushdown automaton (PDA) → is in some state p, reads some input symbol, a_i can access the top most element of stack. (say X)

A PDA at this point,
can move from state
p to q.

→ can change the top most
element of the stack from X to Y.

$(p, a_i, X) \to (q, Y)$

We allow the stack to have a
different alphabet (usually denoted as Γ)

given an input w the PDA
at any given instance does the following

① Reads a bit a_i from w where $a_i \in \Sigma_\wedge$
② goes from state p to q.
③ replaces the top most element of
stack X with Y where $X, Y \in \Gamma_\wedge$

Q { What does it mean for $\wedge$ to be $\wedge$
Pushing Y onto stack }

Q What does it mean for Y to be $\wedge$
Popping X from stack

22nd

Formal Defn of PDA

7- 6-sixple $(Q, \Sigma, \Gamma, \delta, q_0, F)$

Q → finite set of states
Σ → input alphabet
Γ → stack alphabet
$\delta : Q \times \Sigma_\wedge \times \Gamma_\wedge \to 2^{Q \times \Gamma_\wedge}$
q_0 → start state
F → accept states.

Transition function of PDA

Input

① state
② $a_i \in \Sigma_\wedge$ (input symbol)
③ $X \in \Gamma_\wedge$ (symbol at top of stack)

O/P

pairs of (q, Y), $q \in Q$, $Y \in \Gamma_\wedge$

Due to non-determinism of PDA,
there can be multiple transitions on
the same tuple (p, a, X)

$$p \xrightarrow{a_i\, X \to Y} q$$

$P = (Q, \Sigma, \Gamma, \delta, q_0, F)$
is said to accept a string $w \in \Sigma^*$,
if there exists

① a sequence of symbols.
$a_1, a_2 \ldots\ldots a_m \; (\in \Sigma_\wedge)$

② states $r_0, r_1 \ldots\ldots r_m \in Q$

③ strings $s_0, s_1, \ldots s_m \in \Gamma^*$

such that

(i) $w = a_1 a_2 \ldots\ldots a_m$

(ii) $r_0 = q_0$ and $s_0 = \wedge$ (initial condn)

(iii) if $(r_i, y) \in \delta(r_{i-1}, a_i, x)$

then $s_{i-1} = xt$ and

$s_i = yt$ for some $t \in \Gamma^*$ and

$x, y \in \Gamma_\wedge$

(iv)

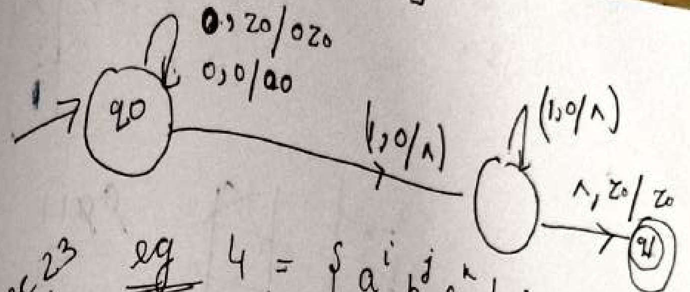
After ith step

Before the ith step

Example

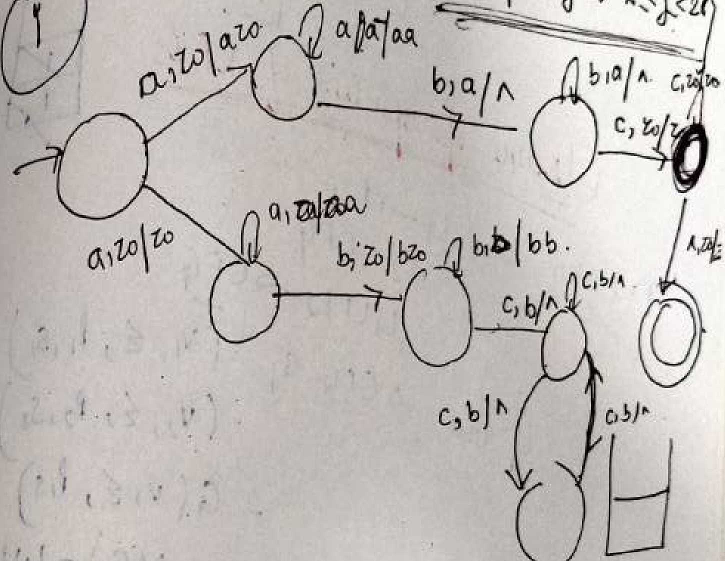
$L = \{0^n 1^n \mid n \geq 0\}$

$L = \{0^n 1^n \mid n \geq 0\}$



Lec 23

eg 4 $= \{a^i b^j c^k \mid i = j \text{ or } k \leq j \leq 2k\}$

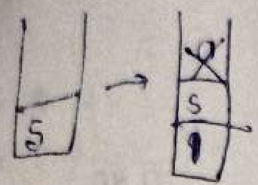


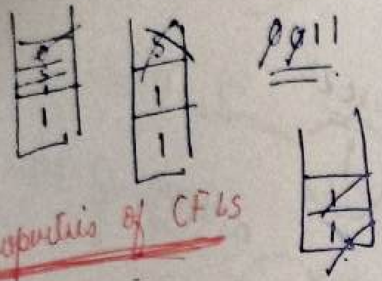
② $L_2 = \{w \in \{a, b\}^* \mid w = rev(w)\}$.

abbba

abbba

$S \rightarrow 0S1, \; S0S11$





$pq11$

Closure properties of CFLs

① Union $\rightarrow$ $L_1 \rightarrow CFG$

$L_2 = CFG$

$L_1 \cup L_2 \rightarrow CFG.$

L_1 via a CFG $G_1 = (V_1, \Sigma, P_1, S_1)$

$= (V_2, \Sigma, P_2, S_2)$

$= G(V, \Sigma, P, S)$

$\Rightarrow$ st $L(G) = L_1 \cup L_2.$

② Concatenation

③ Star $\quad L(G) = L_1^*$

④ Reverse $\quad L(G') = rev(L(G))$

CFLs are closed under homomorphisms and inverse homomorphisms.

Intersection $\rightarrow$

$L_1 \rightarrow CFG$

$L_2 \rightarrow CFG$ ✓ $L_1 \cap L_2$

$L_1 \rightarrow CFG$

$L_2 \rightarrow RG.$

$L_1 \cap L_2 \rightarrow CFG.$

$L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$

$L_2 = \{a^n b^m c^m \mid n, m \geq 0\}$

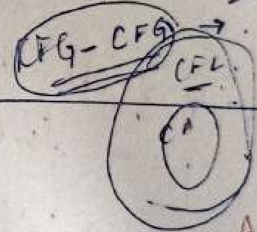
$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ is not CFG.

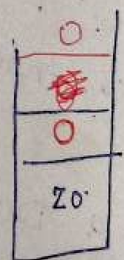
CFL are not closed under intersection

$a^i b^j \; i > j$

Complement: $\quad A \cap B = \overline{\bar{A} \cup \bar{B}}$

set difference $\quad CFG - CFG \rightarrow$





$aabbcc$

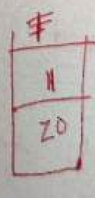
$a a b b c c$

$a^{n+1} b^{n+1} c^{n+1}$

$a^{n+1} b^{n+1} 0$

$0, Z0/0$

$1, Z0/1$



0

Deterministic Push Down Automaton

A deterministic Push Down Automaton (DPDA) is a six tuple seven

$$(Q, \Sigma, \delta, q_0, Z_0, \Gamma, F)$$

such that for all $q \in Q$, $a \in \Sigma$ and $X \in \Gamma$, $\delta(q, a, X)$ has at most one element

$q \in Q$, $X \in \Gamma$, if $\delta(q, \varepsilon, X) \neq \phi$

A language L is said to be a deterministic context-free language if $\exists$ a DPDA M such that $L = L(M)$

$L = \{0^n 1^n \mid n \geq 0\}$ is a DCFL

---

eg of CFL that is not a DCFL

$\{w \in \{a, b\}^* \mid w = rev(w)\}$

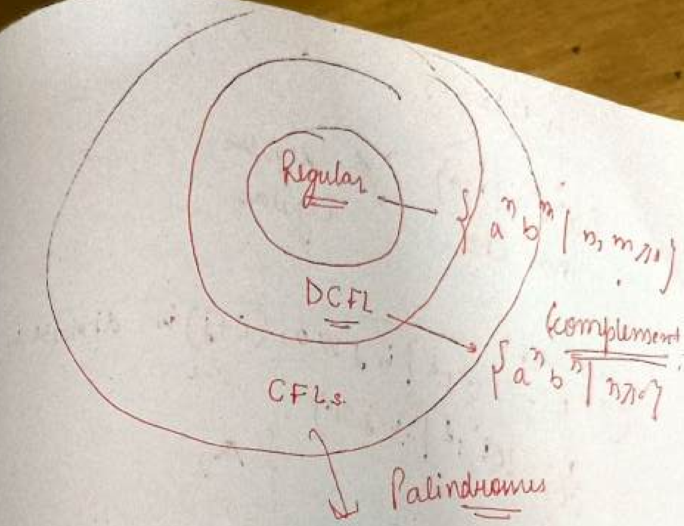
$L = \{a^n b^n c^n \mid n \geq 0$ is not a CFL

$\bar{L}$ is a CFL

It is a not a DCFL.

DCFL are closed under complementation

DCFLs are not closed under union, intersection.



$\bar{L}$ = Consider the regular language $L(a^* b^* c^*)$

$L \cap L(a^* b^* c^*) =$

$L = \{w \in \{a, b, c\}^*$

Consider the regular language, $L(a^* b^* c^*)$

$L \cap L(a^* b^* c^*)$ ← regular

$\{a^n b^n c^n \mid n \geq 0\}$ not a CFL

L is not CFL.

$L = \{w \in \{a, b, c\}^* \mid$
no of a's in w
no of b's in w
no of c's in w

$L_1 \cap L_2 = L_3 \longrightarrow \phi$

$\{a^n b^n c^n \mid n \geq 0\} \longrightarrow \phi$ regular

not CFL

regular

CFL

(2) $A = \{0^i 1^j \mid i, j \geq 0, (i+j) \text{ is divisible by } 2\}$

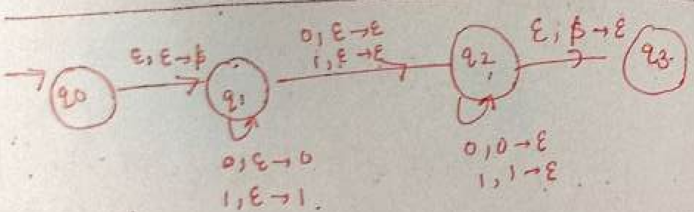
$2+2$

$B = \{0^i 1^j \mid i, j \geq 0\}$ CFL

$\{A\}$

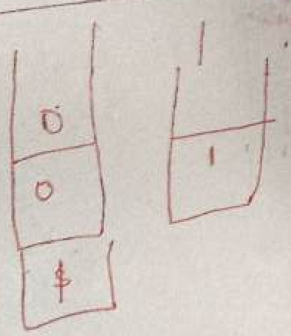
$B - A$

CFL $\rightarrow$ CFL



0110   10:01

0110



$S \rightarrow a S_1 b S_3 c \mid a S_4 b S_2 c$

$S_1 \rightarrow a S_1 b \mid \Lambda$

$S_2 \rightarrow b S_2 c \mid \Lambda$

$S_3 \rightarrow S_3 c \mid \Lambda$

$S_4 \rightarrow S_4 a \mid \Lambda$

$S \rightarrow a a S_1 b b S_3 c c$

$\rightarrow a a b b c c$

$a S_4 a b b S_2 c \quad S_2 c$

$a a b b c c \quad a$

$S \rightarrow a b c \mid$

$A = \{a^i b^j \mid i > j\} \rightarrow$ CFG

$B = \{b^k a^l \mid k > l\}$ CFG.

$a^2 b, a^4 b, a^3 b^2.$

A $\{$ aab, aaaab, aaabb...

aabbba, aaaabba

B $\{$ bba, bbba... $\}$.

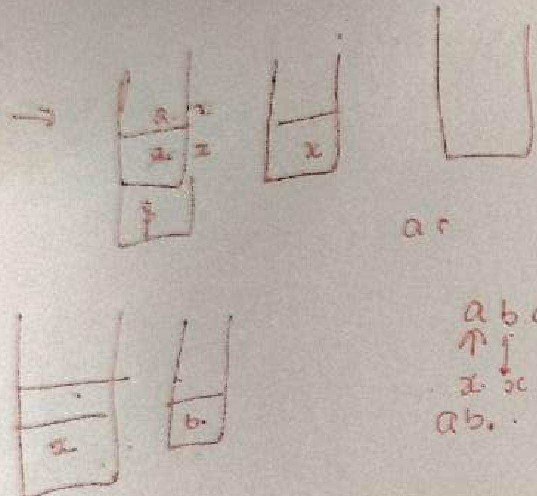
$A = \{ \rightarrow ww$ is not CFL

compl is CFL

$A.\ ww^r w^r w$ X NPA

$B.\ ww^r x x^r$

$C = \{a^i b^i a^k b^k$ abba baab.



ar

abcc

x. x

ab.

$ww^r \# ww^r$

abba # abba

$a^n b^n b^m c^m$ $ww^r \# ww^r$

00 01 11 10

$L_1 \ \& \ L_2 - L_1 \subseteq L_2.$

$a^2 b^2 \subseteq a^n b^n$ X

$a^n b^n \subseteq (a+b)^*$

$a^n b^n \subseteq a^*.$

$S \rightarrow A \mid Sb \mid a \mid b$

$A \rightarrow aS \mid Sb.$

$S \rightarrow A$

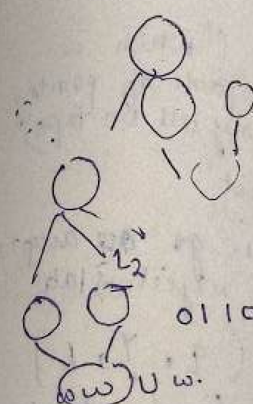
$\rightarrow aS$

$\Rightarrow aSb$

$\Rightarrow a$ $a^5 b^5$

$S \rightarrow Sb$

$\rightarrow Sbb$

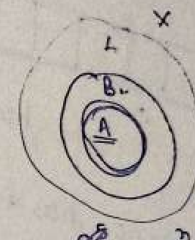


L_2

0110

$ww \cup w.$

$L_3 - L_1$

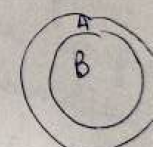


$A \cap B \rightarrow CFL$

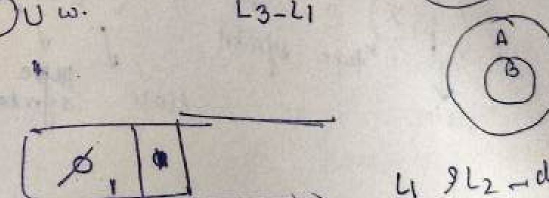
$A \cup B \rightarrow CFL$

$a^n b^n$

$A \cap B \rightarrow$



$A \cup B$

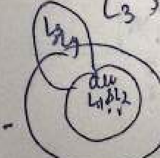


q_0

$L_1 \ \& \ L_2 \rightarrow$ decidable

$L_3 \ \& \ L_4 \rightarrow$ any

$L_3 - L_1$



$(L_1 \cup L_2) - L_3.$

$L_1 - (L_3 \cup L_4)$

$L_3 - L_1$

$L_3 - L_1 -$

Turing Machines

Finite Automata: finite control + no memory

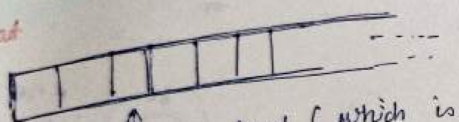
Push Down Automata → finite control + stack

Turing M/c → finite control + tape

Tape → infinite data structure

A finite control + tape.

Tape is unbounded and any cell of the tape can be accessed by using a tape head



↑ tape head (which is capable of pointing any cell on tape)

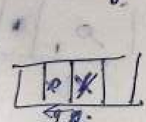
Ff

A turing machine has a designated accept and reject

① which never halt → neither go to accept or reject state.

$$\delta(p, X) \longmapsto (q, Y, L)$$

p → state, X → tape symbol; q → state, Y → tape symbol, L → head moves to left



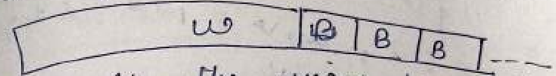
$$\delta : Q \times \Gamma \to Q \times \Gamma \times (L, R)$$

Input alphabet, Σ is a subset of tape alphabet.

$$\Sigma \subseteq \Gamma$$

There is a blank symbol $\sqcup \in \Gamma - \Sigma$.

Initially the tape contains the input w and the remaining cells are filled up with symbol $\sqcup$



If the input head tries to move to the left of leftmost cell of tape then it stays at its current position.

Defn: A turing machine (TM) is the tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$

- Q → set of finite states
- Σ → Input alphabet.
- Γ → tape alphabet
- δ → transition function. $\delta : Q \times \Gamma \to Q \times \Gamma \times (L, R)$
- q_0 → initial state
- q_A → accept state
- q_R → reject state

More on Turing Machines

A configuration of a Turing M/c M with respect to an input w. is a snapshot of the machine consisting of

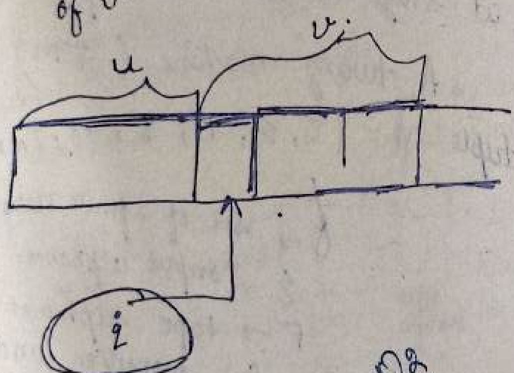
(1) The current state ✓
(2) The current tape contents ✓
(3) The position of tape head. ✓

We represent a configuration

uqv

$q \in Q,\ u, v \in \Gamma^*$ s.t.

→ q is the current state
→ string uv is the current contents of tape
→ tape head points to the first symbol of v.



Q8 //

Start configuration

$q_0 w$ → where w is input

accept configuration $u q_A v$ where $u, v \in \Gamma^*$

→ $u q_R v$, $u, v \in \Gamma^*$

$q_A \neq q_R$ are never same.

$a^n b^n$ $a^* b^*$

M halts on w if either M accepts w or M rejects w.

M is said to be a halting turing machine if $\forall w \in \Sigma^*$ M halts on w. (accept or reject) 25/.

The language of TM, M.

$(12)^* 0 H$ →

$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$

A language is Turing recognizable if ∃ a Turing Machine st $L = L(M)$

A lang is turing decidable (or just decidable) if ∃ a halting TM such that $L = L(M)$.

$(x+y)^* y (a+ab)^*$

$0 \to 1$

$1 \to$ ~~y~~

y, a

$\{a^n b^n c^n \mid n \geq 0\}$

Description of Turing Machine

checks whether the input is of the form $a^* b^* c^*$. If not then reject.

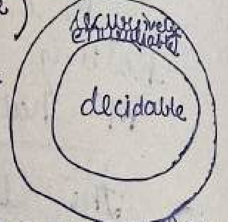
(2) X X X X X X

Non-deterministic Turing Machine

halting!

decidable (also recursive)

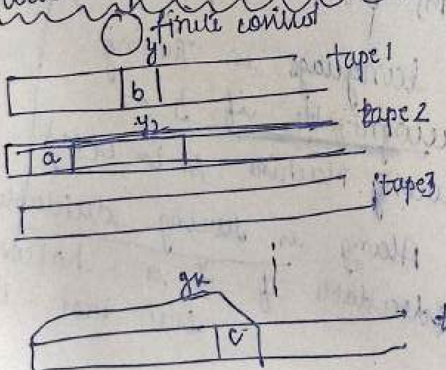
→ Turing recognizable language (recursively enumerable languages)



Decidable ⊆ REL

Variants of Turing Machine

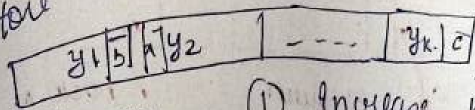
(1) Multi-tape Turing Machine



K-tape Turing Machine ≡ 1 tape Turing

Idea We simulate the K-tapes Machine via a single tape 1-tape Turing Machine

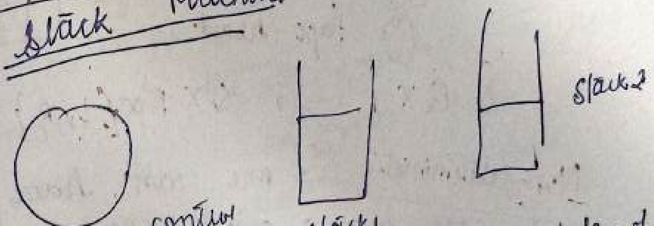
Store the contents of all the tapes.



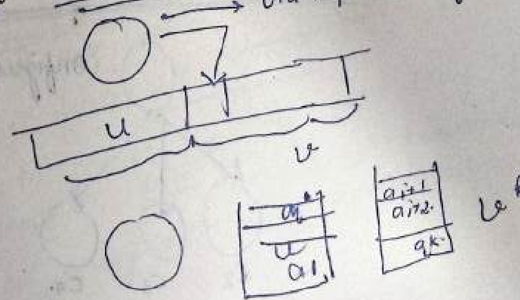
Challenges. (1) Increase in length of string on a tape.

(2) Multiple tape heads. for every symbol $a \in \Gamma$, add a symbol $\bar{a}$

2 Stack Machine



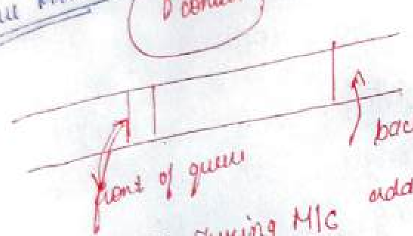
2 stack machines are subclass of one tape machines → one tape turing m/c



2 stack = TM $\frac{25 \times 685}{100}$

Queue Model

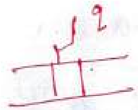
finite control



add and delete

Queue = Turing M/C

Conf
1. state
2. tape contents
3. tape head.

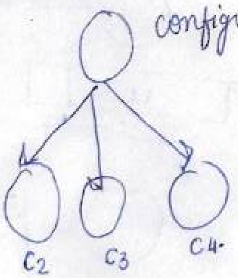


$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

Non-deterministic → one can have multiple transitions from a given configuration

$\boxed{\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}}$

configuration 1



eg $L = \{ 0^t \mid t$ is a composite number $\}$

Write down n_1 no. of 0's and n_2 number of 1's.

check if $n_1 \times n_2 = t$ $2 \leq n_1, n_2 < t$

Configuration graphs

A configuration is a tuple of form.

(state, tape contents, position of tape head)

Turing machine M w.r.t. an input x

A configuration graph of M on input x (denoted as $G_{M,x}$) is a graph whose vertices are the configuration of G w.r.t x and there is an edge from configuration C_1 to C_2 if the turing machine M can go from C_1 to C_2 in one step

→ graphical representation of the computation of M on x.

A computation path is a path in G_{Mx} from start configuration.

M accepts x if and only if $\exists$ computation path in G_{Mx} from the start configuration to accept configuration

Properties of configuration graph

configuration graph is defined w.r.t a TM and an input.

1. → If M is deterministic then out degree of every vertex in $G_{M,x}$ is ≤ 1
2. If M is non-deterministic then out degree can be arbitrary

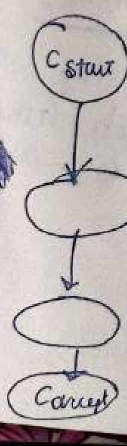
Out degree of accept & reject configuration are zero

Technically $G_{M,x}$ can be infinite but if the size of tape is bounded then $G_{M,x}$ is finite

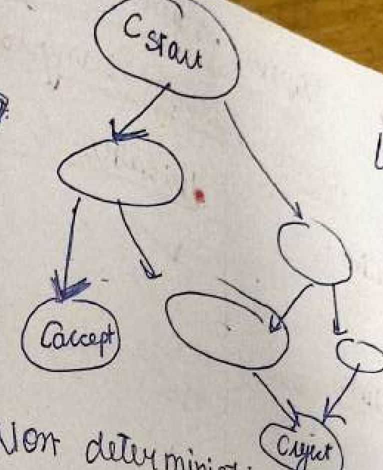
→ Although every configuration is a part of $G_{M,x}$

$G_{M,x}$ has a unique start and accept configuration

Example of configuration graph



Deterministic Turing M/c



$L_1 \to$ decidable
$L_2 \to$ true

L and $\bar{L}$ true

$\frac{M_1}{M_2}$

Non deterministic Turing Machine

Theorem The class of languages accepted by deterministic and non-deterministic TMs are equal

DO a BFS of $G_{M,x}$

Maintain a queue data structure to store the visited configurations of $G_{M,x}$

if an accept configuration is encountered then halt and accept.

→ ~~reject conf is~~

If entire $G_{M,x}$ is scanned without seeing an accept configuration then halt & reject

Lec 30 Closure properties of decidable and Turing Languages

Operation	decidable	Turing
(1) Union:	✓	✓
(2) Concatenation	✓	✓
(3) Star	✓	✓
(4) Intersection	✓	✓
(5) Complement	✓	✗

(1) L_1 & L_2 are decidable via halting turing machines, M_1 & M_2

Yes, $L_1 \cup L_2$ is decidable

M →
Input x
Simulate M_1 on x. If it accept then accept
else simulate M_2 on x,
if M_2 accept then accept
else reject

$L(M) = L(M_1) + L(M_2)$

Turing Recognizable language

Input x

(1) → simultaneously run M_1 & M_2 on input x
(run M_1 for one step on x, run M_2 for one step on x /

(2) If either M_1 or M_2 accepts x, then accept if both reject then reject

L_1 & L_2 are TR →

$L(M) = L(M_1) + L(M_2)$

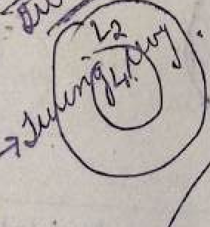
$L_1 \cup L_2$

L_1 → recursive lang
L_2 → recursive enumerable but not recursive
$\bar{L}^1$ is recursive

$\bar{L}^1$ — $L_1 \cap L_2$

L

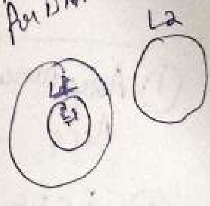
L_1 is decidable
L_2 → Turing lang



$L_2 - L_1$ → Turing lang

$L_1 \cap L_2$: ✓

Acceptance problem for DFA ✓
Emptiness problem for DFA ✓



$\bar{L_1}$ → decidable
$\bar{L_2}$ → undecidable

$L_1 - L_2$ →

Decidability,

Acceptability { (1) A DFA (A_{DFA}) is decidable

Equivalence (2) A NFA is also decidable

(3) ARE is also decidable

(1) $Empty_{DFA} = \{ \langle D \rangle \mid L(D) \text{ is } \phi \}$ is also decidable

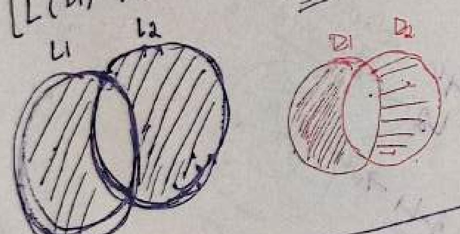
$Empty_{NDFA}$ → decidable

$EQ_{DFA} = \{ \langle D_1, D_2 \rangle \mid D_1 \,\&\, D_2 \text{ are DFA and } L(D_1) = L(D_2) \}$ is also decidable.

$[L(D_1) \setminus L(D_2)] \cup [L(D_2) \setminus L(D_1)] = \phi$

$D_2 - D_1$

$D_1 - D_2$



(1) Acceptance of CFG is also decidable.
A_{CFG} → decidable

(2) E_{CFG} → empty $\{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \phi \}$

(3) $EQ_{CFG} = \{ \langle G_1, G_2 \rangle \mid G_1 \text{ and } G_2 \text{ are CFG } L(G_1) = L(G_2) \}$

not decidable

Equivalence of CFG is undecidable → not decidable

$L_1 \cap L_2$

$L_1 - L_2$

E_{CFG} → Turing Recognizable

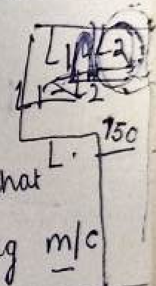
EQ_{CFG} → Turing Recognizable

False.

$\frac{100}{25}$ $\frac{7}{25} \times 100$

$\frac{90+77+48+48+59}{5}$

Undecidability → There are languages which can't be computed during Turing machines. These problems are known as undecidable problems.



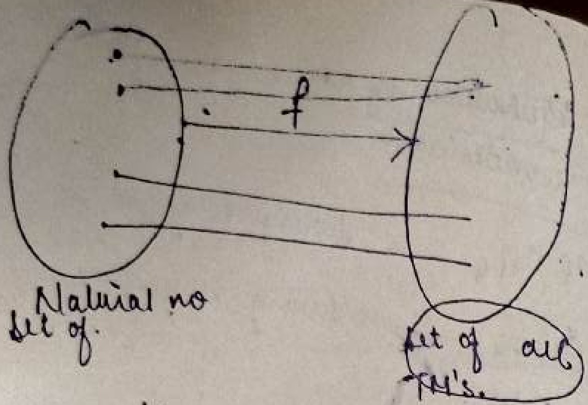
Church Turing Thesis - Anything that can be computed by a Turing m/c

A Turing Machine can also be represented using a finite string consisting of symbols from some finite alphabet.

$\frac{7 \times 100}{25}$

Every Turing m/c maps to natural no.

If a natural no does not map to the Turing Machine by earlier representation then we map it to the Turing m/c that accepts all inputs



f is onto function

jth natural number

Let M_j denote the jth TM

Let s_j be the jth binary string

Language

Non-Turing Recognizable Language

Diagonal $L_d = \{ s_j \mid s_j \notin L(M_j) \}$ — jth Turing M/c

$a_{ij} = 1$ if M_i accept s_j
Otherwise 0

	s_1	s_2	s_3	- - -
M_1	1 2 0	0	0	
M_2	1	0	0	
M_3	0	1	0 2 1	
⋮				

Ld is a T.M.

⇒ ∃ a TM M_i s.t $L_d = L(M_i)$

$s_i \in L_d$

$s_i \notin L(M_i)$

⇒ M_i does not accept s_i

M_i accepts s_i ..

not Turing Recognizable

An undecidable language

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$

encoding of Turing m/c M

undecidable language

Turing Recognizable

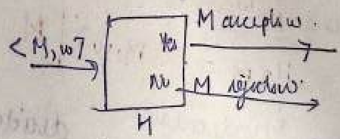
$A_{T.M}$ is turing recognizable

If ... undecidable language

Undecidability

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \}$

Suppose A_{TM} is decidable, there exists a Halting Turing M/c H such that $A_{TM} = L(H)$



Construct a Turing Machine N as follows:

$L_1 \subset L_2$

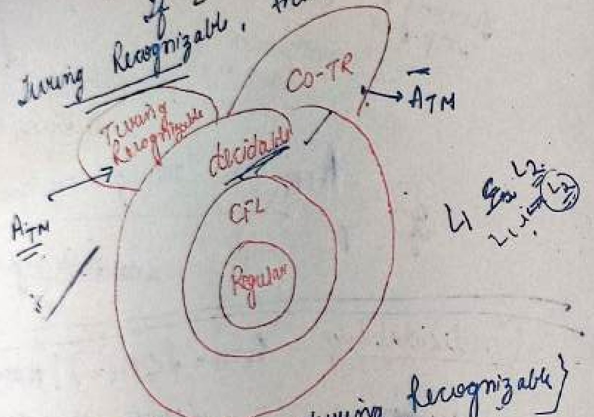
A_{TM} is undecidable

A_{TM} is Turing Recognizable.

$\overline{A_{TM}} \to$ not turing recognizable.

$\{\langle M, w\rangle \mid M$ does not accept $w\}$

If L is undecidable and $\overline{L}$ is Turing Recognizable, then L is not T.R.



CO-TR = $\{L \mid \overline{L}$ is turing Recognizable$\}$

$H_{TM} \to$ turing Recognizable

Halting problem

$H_{TM} = \{\langle M, w\rangle \mid M$ halts on $w\}$

undecidable

decidable $M_1 \cap M_2$. $L_1 - L_2$. functioning

$L_1 \to$ decidable

$L_2 \to$ turing Recognizable

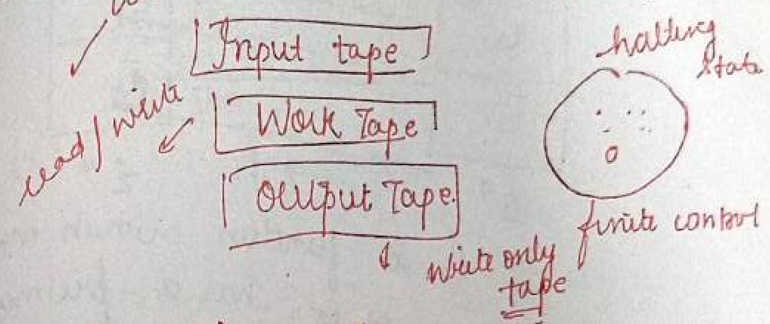
$L_2 - L_1$

$\overline{\emptyset} \cap \overline{L_2} \to \overline{L_2}$

$L_1 \cap L_2$

$L_1 = (\Sigma^* - L_2)$

Reduction → converting one problem into easier problem



$f: \Sigma^* \to \Sigma^*$

Boolean function

Languages = Boolean functions

$f: \Sigma^* \to \{0, 1\}$

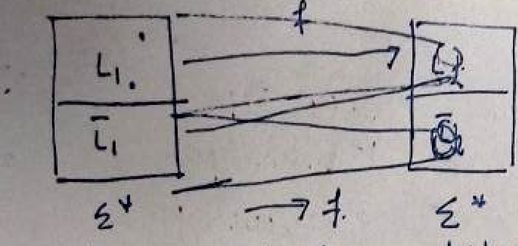
TM with O/P is a TM that has input, work and output tapes as shown s.t. for $\forall x \in \Sigma^*$, the Turing m/c computes a string $y \in \Sigma^*$ before entering the halt state.

→ A function $f: \Sigma^* \to \Sigma^*$ is said to be computable iff $\exists$ a TM with output M, s.t. $\forall x \in \Sigma^*$, M halts with $f(x)$ written on its output tape.

Defn $L_1, L_2 \subseteq \Sigma^*$

We say that L_1 reduces to L_2 (denoted as $L_1 \le_m L_2$) if $\exists$ a computable function f such that $\forall x \in \Sigma^*$,

$x \in L_1 \iff f(x) \in L_2$

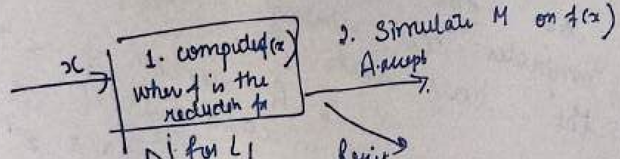


f is a function which means not all element in L_2 or $\bar{L_2}$ has a preimage.

→ many to one $\leq_m$.

Proof Property Let $L_1 \leq_m L_2$ and L_2 is decidable then L_1 is also decidable.

Proof ∃ a halting Turing Machine – $L_2 = L(M)$.



① If $L_1 \leq_m L_2$ and L_1 is undecidable then L_2 is also undecidable

② If $L_1 \leq_m L_2$ and L_1 is not Turing Recognizable then L_2 is also not TR.

Key:- $L_1 \leq_m L_2$ then L_2 is at least as hard as L_1

App

$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \phi\}$

I Encoding is undecidable

We will show that

$\overline{A_{TM}} \leq_m E_{TM}$.

$L_1 \subseteq L_2$.

Week 8 Lec

Rice Theorem undecidability of an infinite set of languages.

A property of languag is a fx

$P = \{\text{set of all languages}\} \to \{0, 1\}$

$P(L) = 1$ 1 → language satisfy property

$P(L) = 0$ 0 → ,, does not satisfy the property

eg of properties

→ L has the string 0110

→ L is empty

→ L has 1000 strings.

Non-trivial property → P is said to be a non-trivial property of languages of TM if ∃ TM₁ and M2 such that

$P(L(M_1)) = 1$ & $P(L(M_2)) = 0$

Rice's Theorems Let P be a non-trivial property of languages of TM, then the language

$L_P = \{ \langle M \rangle \mid P(L(M)) = 1 \}$

whose language satisfy the property is undecidable

Proof Case 1 → $P(\phi) = 0$

P is a non-trivial property of language of TM ∃ a TM N s.t. $P(L(N)) = 1$.

Using this, we will show that $A_{TM} \leq_m L_P$

If A_{TM} is undecidable then L_P is undecidable.

The reduction f.

$\langle M, w \rangle \xrightarrow{f} \langle M' \rangle$

Input: $\langle M, w \rangle$

(1) Design a Turing M' that on input x does the following (M' has a description of N hardcoded into its description together with M & w)

Simulate M on w.

→ If M rejects w then Reject

→ If M accept w then simulate N on x. If N accept x then accept. If N rejects then Reject

Output $\langle M' \rangle$

M accept w ⇒ $L(M') = L(N)$

⇒ $P(L(M')) = 1$

M does not accept w ⇒ $L(M') = \phi$.

⇒ $P(L(M')) = 0$.

Therefore $\boxed{A_{TM} \leq_m L_P}$

Case 2 $P(\phi) = 1$

we will "reduce" this to Case 1

Consider the complement property

$\boxed{\bar{P}(L) = 1 - P(L)}$ if $P(L) = 0$, $P(L) = 1$.

since $P(\emptyset) = 1$, $\bar{P}(\emptyset) = 0$.

$A_{TM} \leq_m L_{\bar{P}} \Rightarrow \bar{A}_{TM} \leq_m \bar{L}_{\bar{P}}$

observe that $\Rightarrow$

$\bar{L}_{\bar{P}} = \{ \langle M \rangle \mid P(L(M)) = 0 \}$
$= \{ \langle M \rangle \mid P(L(M)) = 1 \}$.

$\boxed{\bar{A}_{TM} \leq_m L_P}$.

A_{TM} & $\bar{A}_{TM}$ are undecidable

then L_P is also undecidable.

Application

$\{ \langle M \rangle \mid L(M)$ is finite$\}$ is.
regular.
contains the empty $\varnothing$

are undecidable

Complexity Theory

Decidable Languages. $\rightarrow$ We will assume that all our Turing Machines are halting turing machines.

Let M be a deterministic TM

*Defn. The running time of M is said to be $t: N \rightarrow N$ if $\forall x \in \Sigma^*$, M halts in at most $O(t(|x|))$ steps.

The space required by M is said to be $s: N \rightarrow N$ if $\forall x \in \Sigma^*$, M uses at most $O(s(|x|))$ cells

in its work tape

We assume the 3-tape model of TM and count the space used in the work tape only

$\longrightarrow$ We always count the amount of resource used as a function

Since, we use the O-notation we ignore multiplicative constants and lower order terms

Time and space complexity classes

Let $t: N \rightarrow N$.

Time $TIME(t(x)) = \{ L \mid \exists$ a det TM having running time $t(x)$ such $L = L(M)\}$.

$s: N \rightarrow N$

Space $(s(x)) = \{ L \mid \exists$ a det. TM M requiring $s(x)$ space, such that $L = L(M)\}$

Time & space of non-deterministic TMs

Let N be a non-deterministic TM

The running time of N is said to be $t: N \rightarrow N$ if $\forall x \in \Sigma^*$, every computation path in N halts in at most $O(t(|x|))$ steps

space $\rightarrow$ at most $O(s(|x|))$ cells

$NTIME(t(x)) = \{ L \mid \exists$ a NDTM N having running time $t(n)$ s.t. $L = L(N) \}$

NSPACE $\{ L \mid \exists$ a NDTM N using $s(n)$ space such that $L = L(N) \}$

classes P and NP

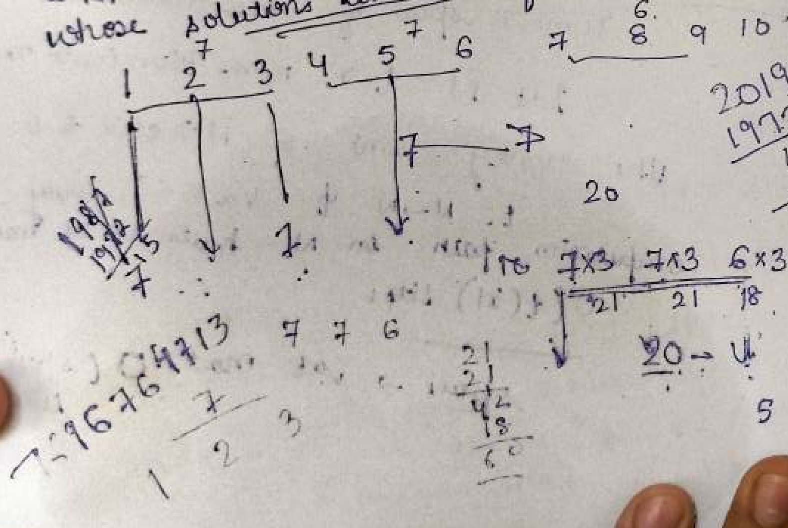
$$\boxed{P = \bigcup_{k>0} Time(n^k)}$$ eg. $(\phi, n, n^2, \ldots)$

Polynomial

$$\boxed{NP = \bigcup_{k>0} NTIME(n^k)}$$

* P is widely recognized as the class of problems having an efficient soln.

* NP is said to be the class of problems whose solutions can be verified efficiently



2019
1973
47

More on the class NP.

$$P = \bigcup_{k>0} Time(n^k)$$

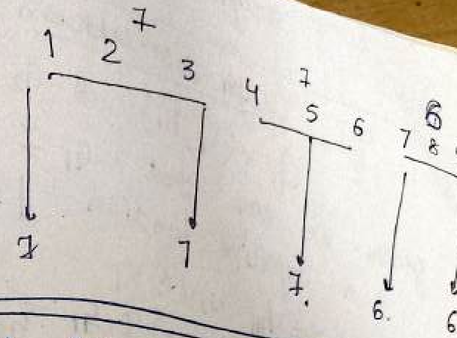
Matrix Multiplication ✓
sorting an array ✓
computing minimum spanning tree
shortest path ✓

$$NP = \bigcup_{k>0} NTime(n^k)$$

Alternate $L \subseteq \Sigma^*$ is said to be in NP if $\exists$ constants $c>0$ and $k>0$ and deterministic polynomial TM V s.t. $\forall x \in \Sigma^*$,

$$x \in L \iff \exists y \in \Sigma^*, |y| \le c x^k \quad V(x,y) = 1$$

V (verifier, deter polynomial m/c that takes 2 strings and input x and 20



y_i certificate

eg. Graph Isomorphism

$G_1 = (V_1, E_1)$ & $G_2 = (V_2, E_2)$ be 2 graphs. We say that $G_1 \cong G_2$ if $\exists$ a bijective function $f : V_1 \to V_2$ s.t.

~~u, v ∈ V~~ $\forall\ u, v \in V_1$

$(u, v) \in E_1 \Rightarrow (f(u), f(v)) \in E_2$

non-deterministic

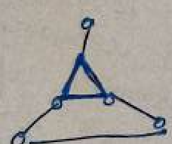
1. Graph Isomorphism problem $\to \boxed{NP}$

$GI = \{ \langle G_1, G_2 \rangle \mid G_1 \cong G_2 \}$

NP algorithm for GI

2. Clique problem

$\{ \langle G, k \rangle \mid G$ has a complete graph of size $\geq k \}$

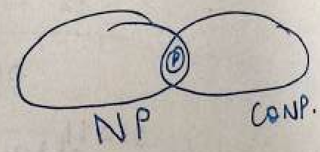


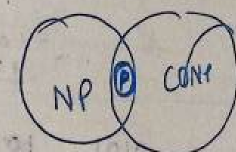
CO NP (class of problem L such that $\{ L \mid \bar{L} \in NP \}$

Imp

Is $\boxed{NP = CONP \to ?}$ (open)

$(NP = P ?)$ open





P $\to$ then $NP \cap CONP$

if in P.

NP-completeness → looking at hardest problems in NP.

$\Sigma^* - \{x \in \Sigma^* \mid |x| > 0\}$

Λ - ✓

01 0101
12 1234.

00000
12345

01011 $\boxed{3-2=1}$
2 3

GNFA.

-101
421

0101 to 011

$3m+2$

NFA → DFA.

$\boxed{\text{DFA} \to \text{NFA}}$ ✓ $3\times0+2=$.

$0^*(1(01^*0)^*1)^*0^*0$. DFA →

10101   010
421   21

110
421

$q_0 = q_0\Lambda + q_0 0 + \Lambda$

$q_1 = q_1 0 + q_0 1$

$q_0(0+\Lambda)+\Lambda$.   RP $q_1 = q_1 0 + 0^* 1$

$q_0 0+\Lambda \Rightarrow 0^*$ $q_1 = q_1$. 0^*10^*

$(0^*1)0^*$

$S \to AB$

$A \to aa \mid ab \mid ba \mid bb$

$B \to aBa \mid bBb \mid C$

$C \to aa \mid ab \mid ba \mid bb$.

$S \to AB$.

$\to ba\, bBb$.

$\Rightarrow bab\, aBab$

$\Rightarrow babab bab$ ✓

$S \to AB$
$\to ab$.

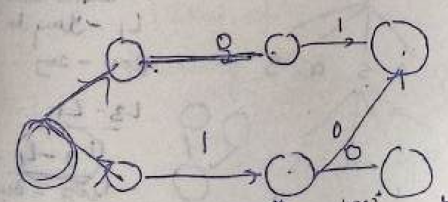
$\pi_0 = \{\cancel{q_0},$

	a	b
→q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_4.	q_0.
q_3	q_1.	q_2
q_4	q_4	q_4

$\Pi_0 = \{q_0, q_3\}\ \{q_1, q_2, q_4\}$

$\Pi_1 = \{q_0, q_3\}\ \{q_1\}\ \{q_2\}\ \{q_4\}$

$\{ q'$ 4 states



a^* $b(a a)^*$ bb

$S \to aS \mid bT \mid \Lambda$

$T \to aaT \mid bT \mid \Lambda$

$(aa)^*\ b^*$

$a^* b^n$

$S \to aSb \mid \wedge$

$S \to aA \mid bB$

$A \to aA \mid bB \mid \wedge$

$B \to bB \mid aA \mid \wedge$

b^*

3:30

6:30

2 to 5:30

$S \to aSB \mid aAb$

$A \to aA \mid bA \mid \wedge$

$L_1 \subseteq L_2$

Recognizing m/c → Decidable

$L_1 \leq_m L_2$

Turing $\leq_T$

$L = a^n b^n$

$L_1 \leq_m L_2$

$L_2 \leq L_3$

$\leq_p$ 3 SAT

L' is regular →

$L_1 \leq_m$

$L \leq$

$L_1 \subseteq L_2$

$a^* \cancel{b}$ $\quad a^n b^n$ $\cancel{E}$

$a^2 b^2 \subseteq a^n b^n$ X False

$a^n b^n \subseteq (a+b)^*$ → False

$a^n b^n \subseteq a$

not decidable

Reg.

Dec

L_1 L_2

decidable

$(L_1 \cup L_2) - L_3$

$L_1 - L_3 \cup L_2 - L_3$

Dec U

$L_1 - (L_3 \cup L_4)$

L_1 - Turing Rec.

Dec - Tury

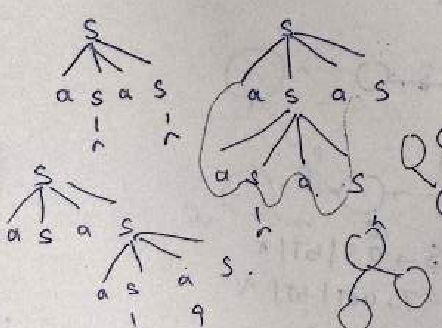
$L_3 - L_1$

$(L_3 - L_1)$ →

Reg - Decidable

L_1 is turing in m/c

3 · $S \to aSaS \mid \wedge$



490

Raw Material

ETHC.

Tower Wiggers

Tower Supply

ODP

Switch board → B

3 Phase Main switch Board

One Phase. Yellow Clamp

Red

blue

Fuse

Main Power

Power

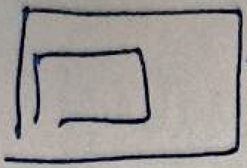
VS

connected to the control

land of Drilling M/C

Cutting Tool

CNC Drill



→ computerized screen Reading

~~Empty~~ Dry Run Drilling M/C

↓ clockwise Drill Rotate

409 → voltage Dummy

Voltage

~~current~~

Clamp Meter → ~~voltage~~

Power Logger

Motor → 3,000

CNC Drill M/C

Model 8054-M04025

AMI - 251 DX

Unit

Strings and alphabets - Basics of strings, alphabets & languages, operations on languages, chomsky Classification of languages.] Symbol: smallest building block in TOC.

Alphabet An alphabet is a finite, non-empty set of symbols. These are the input symbols from which strings are constructed by applying certain operations.

We use symbol Σ to denote the alphabet set.

eg. $\Sigma = \{0, 1\}$ → for binary sequences.

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ for decimal no's

$\Sigma = \{a, b, c, — z\}$, for character string in lowercase

$\Sigma = \{A, B, C, — Z\}$, for character string in uppercase.

Strings It is a finite sequence of symbols selected from some alphabet. It is generally denoted as w.

eg. for alphabet $\Sigma = \{0, 1\}$ ⇒

$w = 010101$ is a string.

eg. 'cabcad' is a valid string on alphabet set $\Sigma = \{a, b, c, d\}$.

Length of string It is the number of symbols present in string. denoted by $|w|$

eg if $w =$ 'cabcad'

- $|w| = 6$

if $|w| = 0$, it is called an empty string (denoted by λ or ϵ)

Kleene Star Σ^* – is the infinite set of all possible strings of all possible lengths over Σ including λ.

Representation $\Sigma^* = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \cup$ ...

eg $\Sigma = \{a, b\}$, $\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb \ldots$

Kleene Closure / Plus Σ^+ The set Σ^+ is the infinite set of all possible strings of all possible lengths over Σ excluding λ.

Representation $\rightarrow \Sigma^+ = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \ldots$

$\Sigma^+ = \Sigma^* - \{\lambda\}$

a^+ a^* $= \{\lambda, a, aa$

eg $\Sigma = \{d\}$.

$\Sigma^+ = \{d, dd, ddd\}$ $\{a, aa$

$\Sigma^* = \{\lambda, d, dd, ddd \ldots\}$.

Concatenation of strings

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

$L_1 =$ strings of length 3

If $w_1 = \{a, b, c\}$
$w_2 = \{d\}$

ie two sets of strings, then on concatenation

$w_1 \cdot w_2 = \{ad, bd, cd\}$

where '$\cdot$' is a concatenation operator.

Languages A language is collection of all possible strings subset of Σ^* for some alphabet Σ. It can be finite or infinite

eg. If the language takes all possible strings of length 2 over $\Sigma = \{a, b\}$

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

→ It a set of string of which are chosen from some Σ^*, where Σ → particular alphabet.

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

$\Sigma^* = \{\lambda, a, b, aa, bb, ab, ba, aaa, bbb \ldots$

Then $L = \{a, aa, aab\}$. is a language on Σ.

Finite language → having finite no. of sentences.

$L = \{a^n b^n : n \geq 0\}$ is also language on Σ.

finite set of symbols

Language

symbols

$\Sigma = \{a, b, c, 0, 1, 2, 3\}$ (a, b)

The strings aabb, aaaabbbb are in the language L, but string abb is not in L.

This language is infinite.

Symbol
Alphabet
Language
String.

language is a collection of strings.

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

L_1 = Set of all strings of length 2

= {aa, bb, ab, ba}.

↓ $L_1 \rightarrow$ finite language.

L_2 = set of all strings of length 3.

= { aaa, aab, aba, abb, baa, bab, bba, bbb}

L_3 = set of all strings where each string starts with a

= { a, aa, ab, aaa, aab, aba, ... }.

L_3 is infinite

Language can be finite or infinite.

Powers of Σ $\quad \Sigma = \{a, b\}$

Σ^1 - set of all strings over Σ of lengths exactly 1

$= \{a, b\}$

Σ^2 = set of all $\Rightarrow \Sigma' \Sigma$.

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

set of all strings of length '2'.

$\Sigma^3 = \Sigma\Sigma.\Sigma = \{a,b\}\{a,b\}\{a,b\}$

$|\Sigma^3| = 8$

$|\Sigma^n|$ = n length string.

Σ^0 = set of all strings of length 0.

$\Sigma^0 = \{\epsilon\}$

$\boxed{|\epsilon| = 0}$ $\therefore$ $\Sigma = \{a, b\}$.

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

$\Sigma = \{\epsilon\} \cup \{a, b\} \cup \{aa, bb, ab, ba\}$ ———

Infinite set of all strings possible over {a, b}. (strings)

$L_1 \subseteq \Sigma^*$ → all languages are subsets of ~~string~~ Σ^* defined over Σ

$L_2 \subseteq \Sigma^*$

$L_3 \subseteq \Sigma^*$. No. of languages possible → infinite

Consider C programming language

Σ = { a, b, -- z, A, B -- Z, 0, 1 -- 9, +, * -- }.

finite set of symbols.

alphabet → finite

```
void main ()
{ int a, b;
  :
}
```

Program in C

but in TOC it is a string

Program is a string

C → programming language = set of ~~of~~ all "valid" programs.

any no of valid programs

C is infinite.

{ P_1, P_2, P_3 —— }, —— P_n

Given Language L and string S find whether string is valid.

L is finite.

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

$L_2 = \{aa, ab, ba, bb\}$ → language is finite

$S = aaa$.

$L_2 = \{a, aa, aaa, ab, —\}$ language is infinite

$S = baba$.

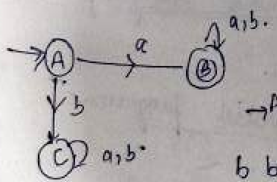
L → find a finite representation of language using which if we have given a string, we can find whether string in of language or not present in language.

$S \rightarrow F \rightarrow$ Yes / No

$S \rightarrow FA \rightarrow$ Y / N.

L_1 = set of all strings which starts with a

$= \{a, aa, ab, aaa, —\}$.



a a b

$\rightarrow A \rightarrow B \rightarrow B \rightarrow B$

b b a

$A \rightarrow C \rightarrow C \rightarrow C$

string is not accepted

Operations on Languages

The usual set of operations:-

Complement The complement of language is defined with respect to Σ^*.

The complement of L is $\boxed{\Sigma^* - L}$

$\Sigma^* = \{\lambda, a, b, ab, ba, aa, aba, ---\}$

The complement of $L = \{a, ba\}$

$\bar{L} = \{\lambda, b, ab, aa, bb, aaa —\}$

Reverse The reverse of language is set of all string reversals.

$L^R = \{w^R : w \in L\}$

eg. $\{ab, aab, baba\}^R = \{ba, baa, abab\}$.

$L = \{a^n b^n : n \geq 0\}$

$L^R = \{b^n a^n : n \geq 0\}$

$= \{a^n b^n : n \geq 1\}$

(2) **Union**

Concatenation - of two languages L_1 and L_2 is set of all strings obtained by concatenating any element of L_1 with any element of L_2.

$L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

eg $\{a, ab, ba\}\{b, aa\}$

$\{ab, aaa, abb, abaa, bab, baaa\}$

$L^n = \underbrace{LLL\text{---}}_{n}$

$\{a,b\}^3 = \{a,b\}\ \{a,b\}\ \{a,b\}$

$= \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

$= \{aaa, aab, aba, abb,\ \ \{aa, ab, ba, bb\} \times \{a,b\}$

$L^0 = \{\lambda\}$

$\{a, bba, aaa\}^0 = \{\lambda\} = \{\cancel{aaa, aab, aba, abb, baa}$

$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$.

$\alpha A \beta = \alpha \gamma \beta$.

$\alpha, \beta \in V^*$

$A \in N,\ \gamma \in V^+$

$A \to \alpha$. $A \in N$. $\alpha \in V^*$.

$A \to aB$ $A, B \in N \to$ Non terminals

$A \to b$ $a \in T$

$b \in T \cup \{\epsilon\}$

2016

Chomsky classification of Languages.

(10).

According to Noam Chomsky, there are four Types of grammars - 1956 → 5 marks

Type 0 — Unrestricted grammar — Turing M/C

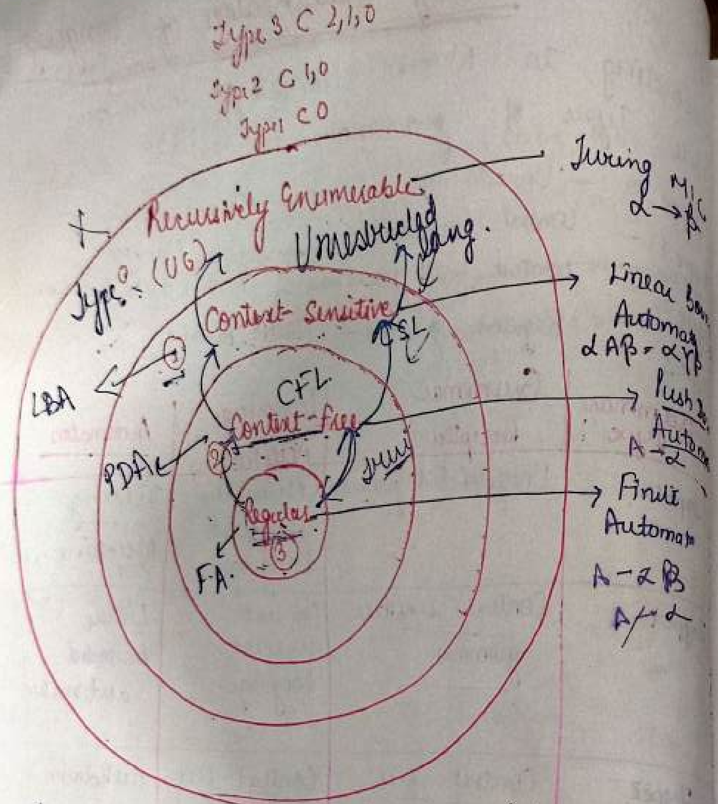
Type 1 — context sensitive — Linear Bounded

Type 2 — context free grammar — Push Down Automata

Type 3 — Regular grammar — finite Automata

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively Enumerable languages	Turing Machine TM
Type 1	Context sensitive grammar	Context-sensitive language	Linear bounded automaton LBA
Type 2	Context-free grammar	Context-free language	Pushdown Automaton PDA
Type 3	Regular grammar	Regular language	Finite state Automaton

TM > LBA > PDA > FA



Type0 grammars generate recursively enumerable languages. The productions have no restrictions. They generate the languages that are recognized by Turing machine.

The productions can be in the form of $\alpha \to \beta$ where α is a string of terminals and non-terminals with at least one non-terminal and α can't be null. β is a string of terminals and non-terminals.

$S \to ACaB$

$Bc \to acB$

$CB \to DB$

$aB \to Db$

eg of unrestricted languages is natural language.

$\alpha \to \beta$

$\alpha \in (V+T)^+$

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

eg. $aAb \to bB$

$\alpha \to \beta$

Decision

$aAb \to bb$ not allowed

$\alpha \to \beta$

$|\alpha| \le |\beta|$

$\alpha, \beta \in (V+T)^+$

eg $aAb \to bbb$

$\alpha \to \beta$

$\alpha A \beta \to \alpha \gamma \beta$

Type1. generate context-sensitive languages.

These grammars have rules of form $\boxed{\alpha A \beta = \alpha \gamma \beta}$

with A → a non-terminal

α, β & γ strings of terminals and non-terminals

Strings α & β may be empty, but γ must be non-empty.

eg $\alpha A \beta = \alpha \gamma \beta$

$AB \to CD.B$

$AB \to CDB$

$AB \to CdGB$

$ABcd = abCDBcd$

$B \to b$

$\alpha A \beta = \alpha \gamma \beta$

$ABcd \to abCDBcd$

eg most programming languages

Context-free grammars

A context-free grammar is one whose production rules are of form

$A \to \alpha$

where $A \to$ single non-terminal (V)

$\alpha \to$ combination of terminals and non-terminals.

eg. simple programming languages

$A \in N$.

$\alpha \in (T \cup N)^*$

$S \to Xa$
$X \to a$
$X \to aX$
$X \to abc$.
$S \to AB$
$A \to a$
$B \to b$.

$X \to \varepsilon$.

$A \to aBC$.

$A \to \alpha$

$\alpha \to (V+T)^*$

eg $A \to \varepsilon$
$A \to BCD$

Regular Grammar

Right linear with rules of form $A, B \in V$, $x \in \Sigma^*$

$A \to \alpha B$
$A \to \alpha$

$A \to xB \mid x \to$ RL
$A \to Bx \mid x$

A & $B \to$ single non-terminal symbols
$\alpha \to$ terminal symbol.

Left linear with rules of form

$A \to B\alpha$
$A \to \alpha$

eg pattern matching languages (regular expression)

$A \to \varepsilon$
$A \to a$
$A \to aX$

$S \to \varepsilon$ is allowed if S does not appear on right side of any rule

$aAb \to bb$

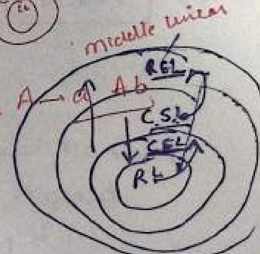
Left context of A — Right context of A

$\varepsilon A \varepsilon$

Regular grammar

$S \to aS \mid b$
$S \to aS \mid \varepsilon$
$S \to Sa \mid b$
$A \to \varepsilon$
$A \to ba$

$\alpha A \beta \to \alpha \gamma \beta$ (Left, Right)



eg. $A \to aAb$

Superset CFL but not subset RL ✓
CSL but not REL ×
REL but not RL ✓
RL but not CSL.

$aAb \to abB$

$A \to aB$.

$A = \alpha B$

Left Linear if all productions are in the form $A \to B\alpha$ or $A \to \alpha$, where $A, B \in V_N$, $\alpha \in \Sigma^*$.

eg. $A \to Aa \mid Bb \mid b$

R.L. $A \to aA \mid aB \mid b$.

Identify types of ...

① $S \to aS|a$ (Right Sensitive)
(3, 2, 1, 0)

② $S \to AB$ Type 3 (✗) (Alan Turing)
$A \to a$ → Type 2
$B \to b$ → Type 2
(2, 1, 0)

③ $S \to aSb \mid bSb \mid a \mid b$
$S \to aSb$ Type 2

④ $S \to aS \mid bS \mid \Lambda$
(3, 2, 1, 0)

a+b*c

Without o/p → Input [M/C] — o/p Transducers
Acceptors (Yes/No) Numerically
↓ DFA NDFA

Power (how tough the m/c can accept)
RL < CFL < CSL < REL

Type 0 ① $\phi A \psi \to \phi \alpha \psi$
$A \to$ variable
ϕ - Left context
$\psi \to$ Right context.

② Type 1 $\phi A \psi \to \phi \alpha \psi$
$|\phi A \psi| \leq |\phi \alpha \psi|$ as $\alpha \neq \Lambda$

$\alpha \to \beta$. $|\alpha| \leq |\beta|$,
eg. $BC \to CB$

$S \to aS|a$
3, 2, 1, 0

$S \to AB$ ⌉ 2, 1, 0
$A \to a$
$B \to b$

$S \to aSb \mid bSb \mid a \mid b$.
$S \to aSb$.

$S \to aS|a$.
$S \to AB$ → Type.
$A \to a \to$ Type.

(1) Complementation

Let L be a lang over an alphabet Σ, the complementation of L denoted by $\bar{L}$

$\boxed{\bar{L} = \Sigma^* - L}$

(2) Union Let L_1 & L_2 are lang over an alphabet Σ, union of L_1 & L_2 denoted by $L_1 \cup L_2$

is $x \mid x$ is $\boxed{L_1 \cup L_2}$.

(3) Intersection Let L_1 & L_2 be lang over an alphabet Σ Intersection of L_1 & L_2 denoted by

$L_1 \cap L_2$. $\{x \mid x$ is in $L_1 \cap L_2\}$

(4) Concatenation Let L_1 & L_2 be lang over an alphabet Σ. The concatenation of L_1 & L_2.

$L_1 \cdot L_2$ $\{w_1 \cdot w_2 \mid w_1$ is in L_1 w_2 is in $L_2\}$

(5) Reversal Let L be a lang over an alphabet Σ

Reversal of L

L^R is $\{w^R \mid w$ is in L

(6) Kleen closure

Let L be a lang over an alphabet Σ the kleen closure of L denoted by L^*. is $\{x \mid$ for an integer $n \geq 0\}$.

$\boxed{x = x_1 x_2 \text{———} x_n}$.

& $x_1 x_2$ ——— x_n are L.

eg $a^* = \{\wedge, a, aa, aaa - -$ ——— $\}$

Basic Mathematical Notation & Techniques,

finite state systems

finite Automata - DFA & NDFA

finite Automata with ε-moves

* Regular languages & regular expressions

Equivalence of NFA & DFA, Minimization of DFA, Moore & Mealy M/c.