Alphabet

$\Sigma = \{a, b, c, \ldots, z\}$

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

Strings : Finite sequence of symbol drawn from alphabet. Has a length

$w, x, y, z \rightarrow$ string

$x = 0110$

$|x| \rightarrow$ length of a string = no. of symbol in it.

Empty string: Has no symbols in it (λ / ϵ).

String of infinite sequence: 0100 . . .

$\Sigma^* \Rightarrow$ Set of all strings over Σ

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

$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\} \rightarrow$ Countable

Sets
- Countable
  - finite: $\{a, b\}$
  - Infinite: $\mathbb{N}$
- Uncountable

Σ^* is countable $\rightarrow$ $f: \mathbb{N} \rightarrow \Sigma^*$

~~with~~ which is on-to. and one to one (Bijective).

Lexicographic ordering

0	1	ε	
1	2	0	
2	3	1	
3	4	0	0
4	5	0	1
5	6	1	0
6	7	1	1
7	⋮	⋮	
⋮			

$f(x) = \{$ binary representation of $x+1$ discarding leading 1 $\}$

bijective function $|\mathbb{N}| = |\Sigma^*|$

Language : Finite or infinite set of strings over Σ

$$L \subseteq \Sigma^*$$

In general any subset of Σ^* is a language.

A is a language over Σ

$$A \subseteq \Sigma^*$$

Empty Language : ϕ

twin prime conjecture

How many Language over alphabet Σ

$$L \subseteq \Sigma^*$$

no. of Language over Σ is $2^{|\Sigma^*|} = P(\Sigma^*)$

$$L \in P(\Sigma^*)$$

- The power set of Natural numbers is uncountable

$$f : \mathbb{N} \rightarrow P(\mathbb{N})$$

⇒ Assume $P(\mathbb{N})$ is countable.

So, there exists an ~~one~~ onto function of the form

$f : \mathbb{N} \rightarrow P(\mathbb{N})$

let us define a subset of natural no. as follows

$$S = \{x \in \mathbb{N} : x \notin f(x)\}$$

$\mathbb{N}$	$P(\mathbb{N})$
0	ϕ
1	$\{1\}$
✓2	$\{3, 4, 5\}$
✓3	$\{1, 2, 4\}$
4	$\{1, 4, 5, 7\}$
⋮	⋮

$S \subseteq \mathbb{N}$

$S \in P(\mathbb{N})$

let, m be a number

Such that $f(m) = S$

$\therefore$ $m \notin f(m)$

$\therefore$ $m \notin S$

but $S = \{x \in \mathbb{N} : x \notin f(x)\}$

$\therefore$ $m \in S$

So we come to an contradiction.

- The set of all languages over Σ is uncountable.

$g : P(\mathbb{N}) \rightarrow P(\Sigma^*)$ onto and one to one

we have seen,

$f : \mathbb{N} \rightarrow \Sigma^*$ an onto and one to one

for $A \subseteq \mathbb{N}$,

$g(A) = \{ f(n) : n \in A \}$

$A = \{1, 2, 3\}$

$g(A) = \{ f(1), f(2), f(3) \}$

[Inverse function exists when mapping is bijective]

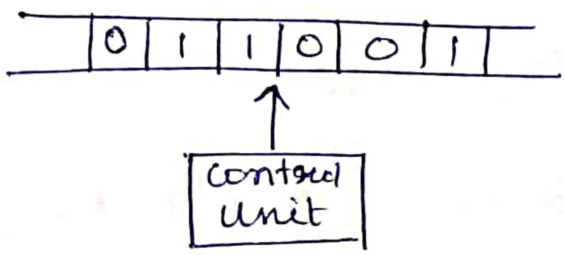
if inverse of $g(A)$ exist then the mapping is bijective.

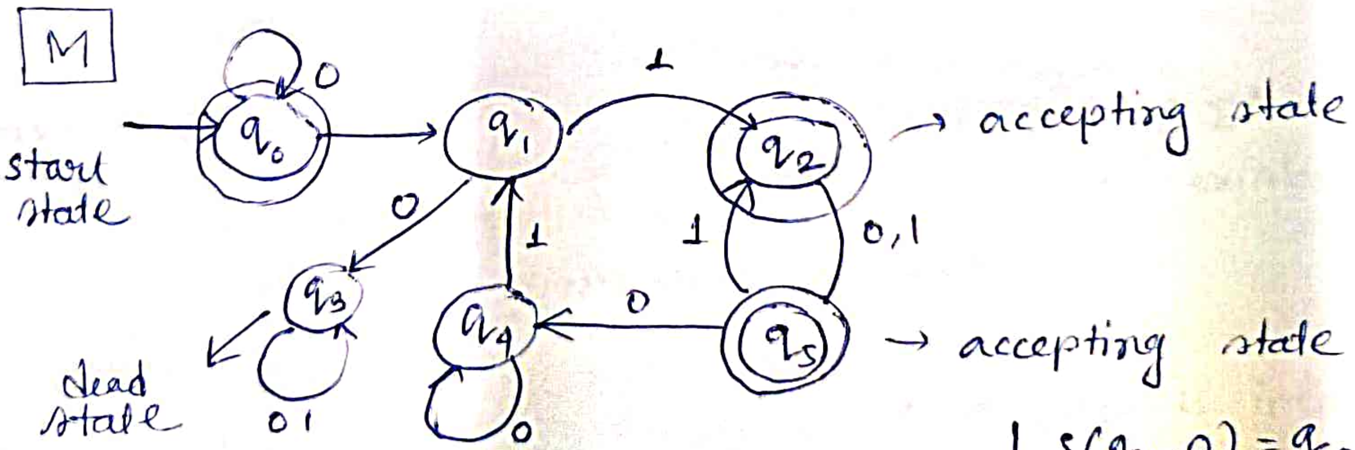
Deterministic Finite Automata (DFA)
Finite state machine.

Definition: A DFA is a 5 tuple

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

Q = Finite non-empty set of states
Σ = is an alphabet
δ = is a transition function $\delta : Q \times \Sigma \to Q$
q_0 = start state, $q_0 \in Q$
F = subset of sates, accepting states / final state
$F \subseteq Q$.



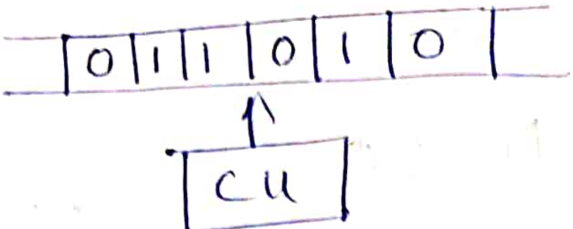


$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$
$\Sigma = \{0, 1\}$

$F = \{q_0, q_2, q_5\}$

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

$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \to q_4$



- movement one directional
- Cannot re-write the symbol in the input

$$L(M) = \{ x \in \Sigma^* \mid x \text{ is accepted by } M \}$$

$\hat{\delta} \mid \delta^*$

Extended transition Function :

$\Sigma^*(q, \epsilon) = q \quad \forall q \in Q$

$\delta^*(q, aw) = \delta^*(\delta(q,a), w) \quad \forall q \in Q, a \in \Sigma$ and $w \in \Sigma^*$

1101

$\delta^*(q_0, 1101)$

$= \delta^*(\delta(q_0, 1), 101)$

$\delta^*(q_1, 101) = \delta^*(\delta(q_1, 1), 01)$

$\delta^*(q_2, 01) = \delta^*(\delta(q_2, 0), 1)$

$\delta^*(q_5, 1) = \delta^*(\delta(q_5, 1), \epsilon) = q_2$

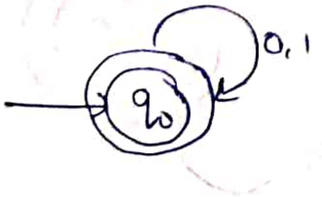
$L(M) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \in F \}$

Definition: Let Σ be an alphabet and let $A \subseteq \Sigma^*$ be a language A is regular if there exists a DFA M such that $A = L(M)$.

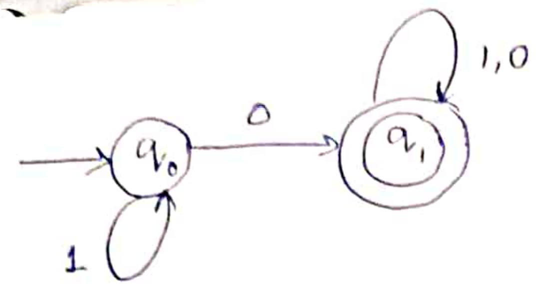
?? $L = \{0^n 1^n \mid n \geq 0\}$ → No DFA → language is infinite

$L = \{0^n 1^n \mid n \leq 9\}$ → DFA exists

If a language is finite, it is regular.



No of DFA is ~~finite~~ countable, as we can classify DFAs with no. of states. Like, DFA with 1 state, 2 state and the no. is countable.



q_0 0
q_1 1

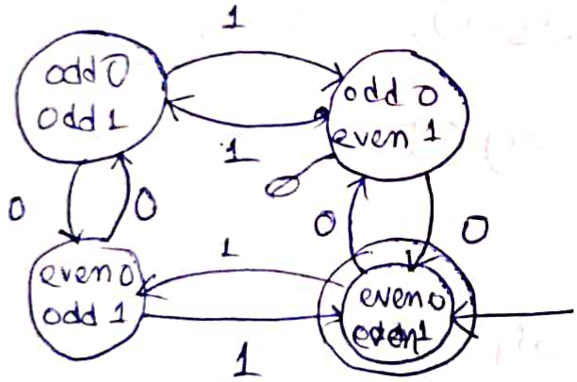
0c1c0cc1c1c1

q_0 q_1 q_0

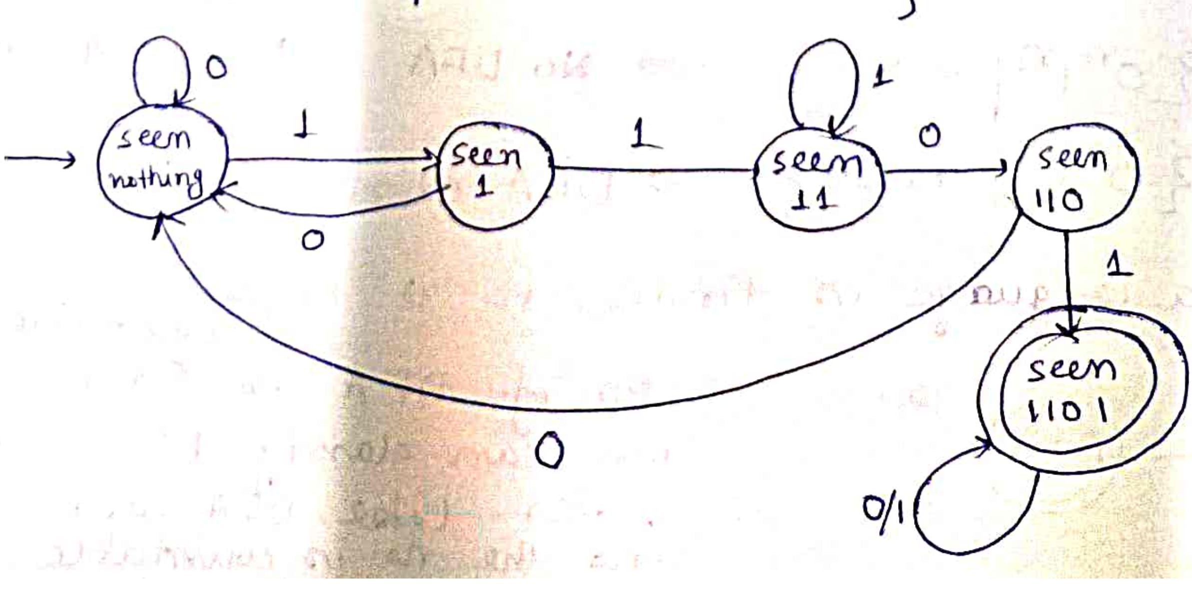
$L_1 = \{ w \in \{0,1\}^* \mid w$ contains even no. of zeros and even no. of ones $\}$

~~1001~~ 1011/0001

odd 1
odd 0



$L_2 = \{ w \in \{0,1\}^* \mid w$ contains $1101 \}$



Proof : $L(M) = L_2$

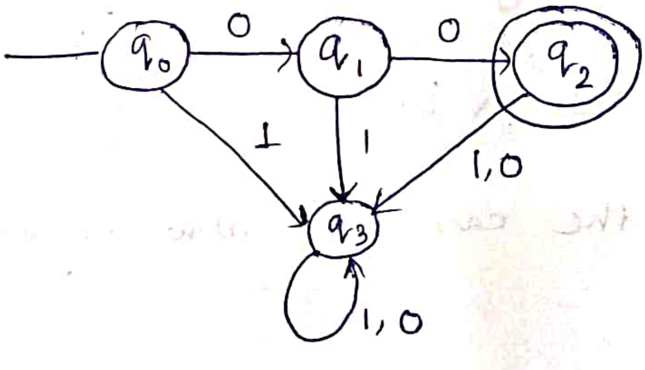
$A = B$ when $A \subseteq B$, $B \subseteq A$

1. Pick a string from the language and fit it into the machine, if the ~~language~~ string is accepted by M, $L_2 \subseteq L(M)$

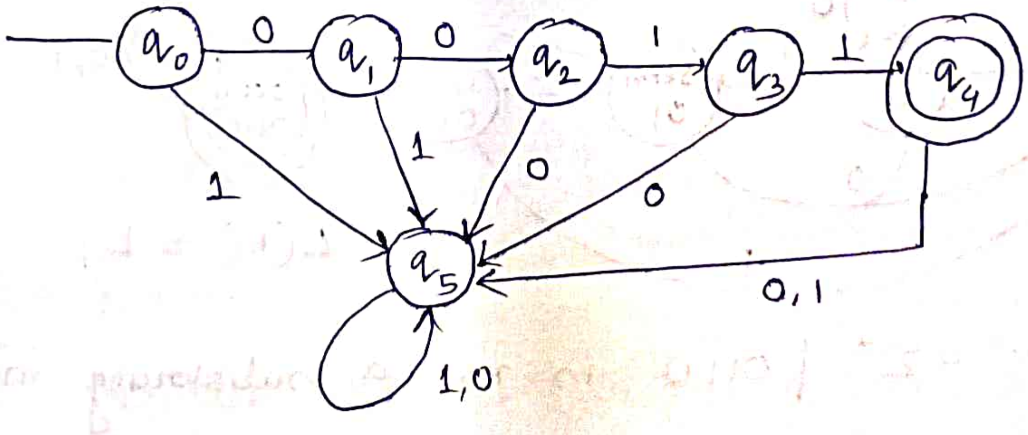
2. ~~Chech~~ Check if the language of the machine has the characteristic of L(M). If same as L_2, $L(M) \subseteq L_2$

$\therefore L(M) = L_2$

$L_2 = \{00, 0011\}$



$L = \{00\}$

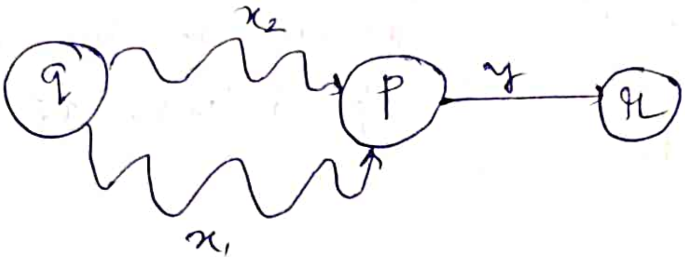


$L = \{0011\}$

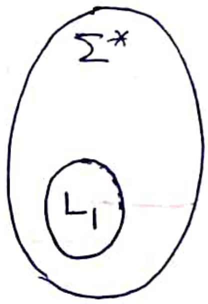
$$\hat{\delta}(q_0, x_1) = \hat{\delta}(q_0, x_2)$$

$$\hat{\delta}(q, x_1) = \hat{\delta}(q, x_2) = P$$

$$\hat{\delta}(q, x_1 y) = \hat{\delta}(q, x_2 y) \qquad \forall y \in \Sigma^*$$



$L_1 \subseteq \Sigma^*$



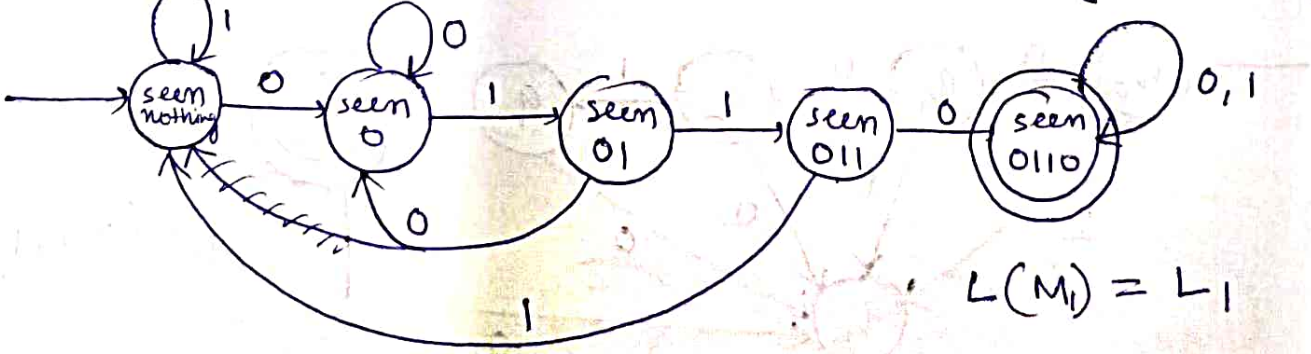
Complement of L_1 denoted by $\overline{L_1}$

$$\overline{L_1} = \Sigma^* - L_1 = \Sigma^* \setminus L_1$$

- If L_1 is regular, is it the case $\overline{L_1}$ also regular?

$L_1 = \{ w \in \Sigma^* \mid 0110 \text{ as substring in } w \}$

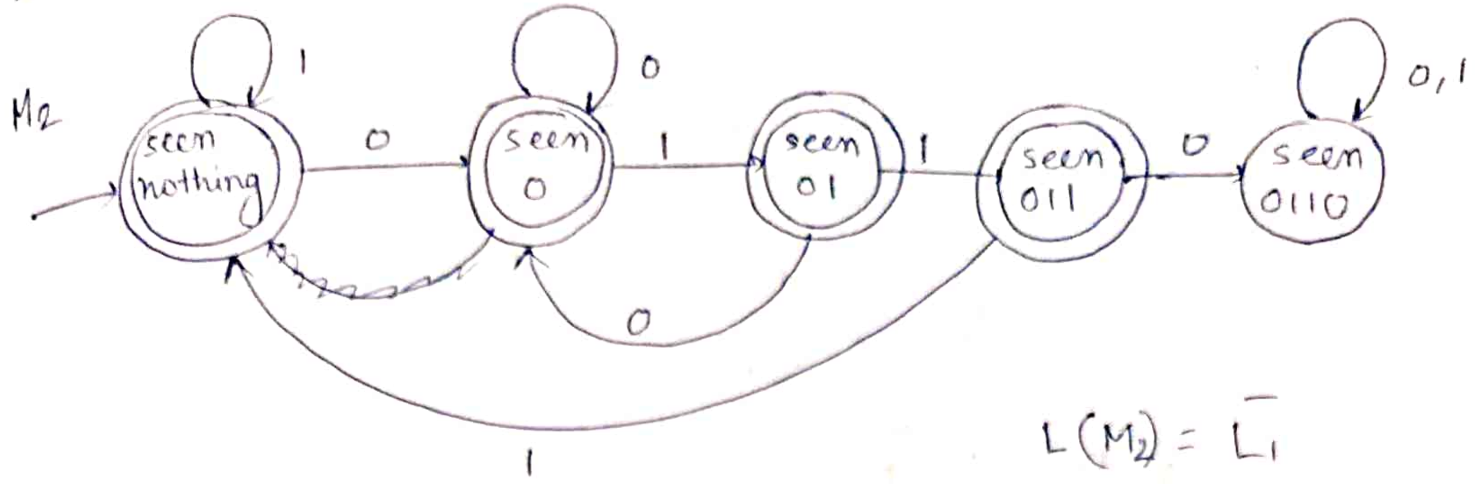
M_1



$L(M_1) = L_1$

$\overline{L_1} = \{ \omega \in \Sigma^* \mid 0110 \text{ is not a substring in } \omega \}$

M_2



$L(M_2) = \overline{L_1}$

** Complement of a regular language is also regular.

i.e. Regular language is closed under complementation

Union :

Suppose L_1 & L_2 are two regular language, i.e there exists M_1 & M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$.

- Is $L_1 \cup L_2$ regular ?

$$M_1 = \{Q_1, \Sigma, \delta_1, q_{01}, F_1\}$$

$$M_2 = \{Q_2, \Sigma, \delta_2, q_{02}, F_2\}$$

$$M = \{Q, \Sigma, \delta, q_0, F\}$$

$$Q = Q_1 \times Q_2$$

$$Q = \{(p, q) \mid p \in Q_1, q \in Q_2\}$$

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

$\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$

$P_1 = \delta_1(P, a)$
$q_1 = \delta_2(q, a)$

$q_0 = \{(q_{01}, q_{02}\}$

$F = \{(P, q) \mid P \in F_1 \text{ or } q \in F_2\}$

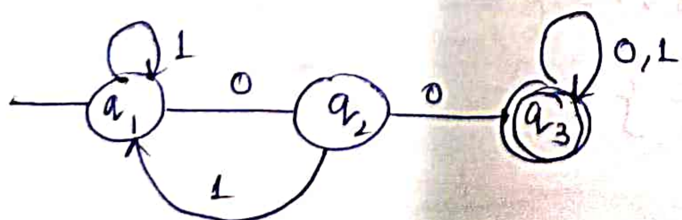
$F = \{(P, q) \mid P \in F_2 \ \& \ q \in F_2\} \rightarrow$ For $L_1 \cap L_2$

** Regular language is closed under union and intersection.

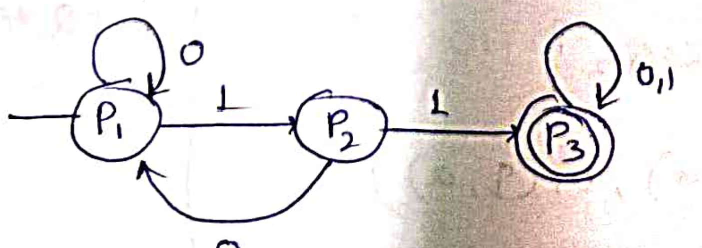
H.W. Design this machine.

Tabular.

states	0	1
(P_{01}, q_{01})	(P_1, q_1)	(P_1, q_3)
( )		
( )		



L_1



L_2

Q1) $L_1 = \{ x \in \{0, 1, 2\}^* \mid$ the sum of digit in x is 2 mod 3 $\}$

Example :

2101012   sum of digit : 7   7 mod 3 ≡ 1 mod 3 ✗

12101012   Sum of digit : 8   8 mod 3 ≡ 2 mod 3 ✓

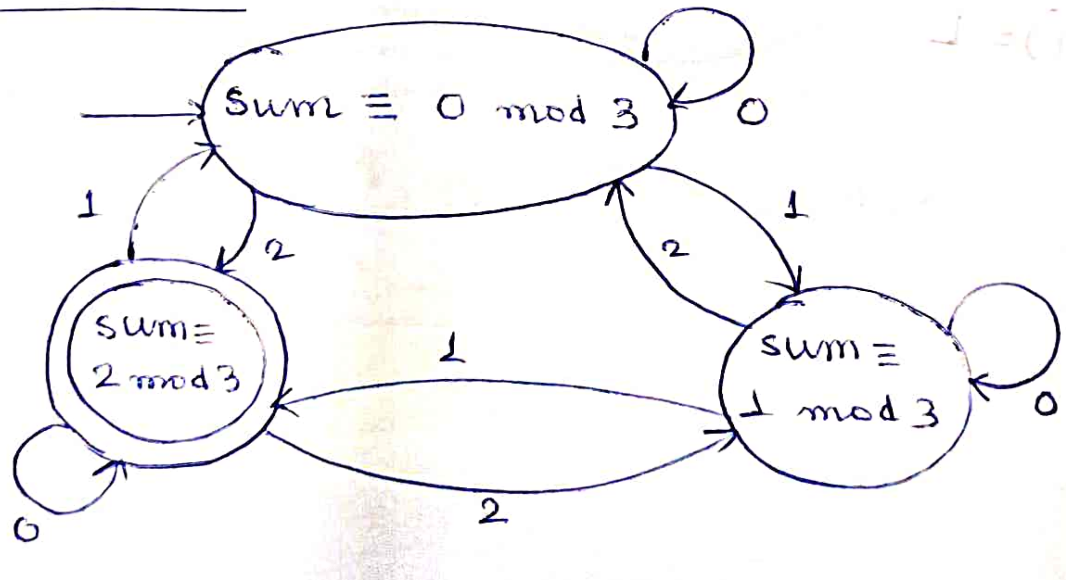
We can't store sum in the states, as then there will be ∞ states.

So we'll store $\boxed{\text{sum mod 3}}$ in the states.

So we will need 3 states to store 3 possiblities 0 mod 3, 1 mod 3 and 2 mod 3.

[Generalizing, n states to store 0 mod n, 1 mod n, ... (n-1) mod n]

- State diagram



Q.2) $L_2 = \{x \in \{0,1\}^* \mid$ the number whose binary representation is x, that number is 2 mod 3 $\}$

[Appending a zero to binary form, multiplies its decimal form by 2]

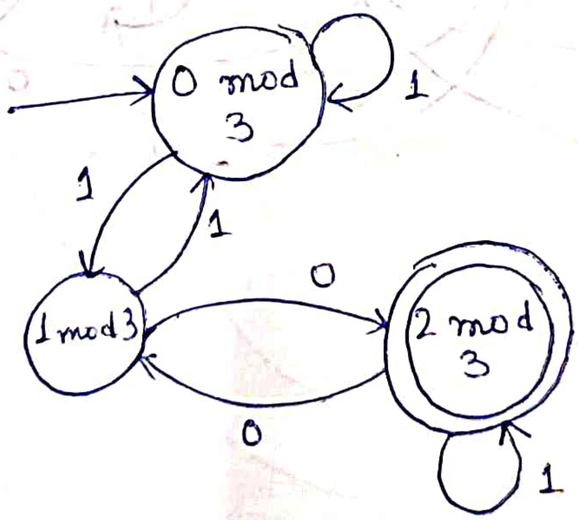
i.e. $(11)_2 \equiv (3)_{10}$

$(110)_2 \equiv (6)_{10}$

Appending a 1 to binary form, makes n in decimal form $\rightarrow$ $2n+1$

i.e $(11)_2 \equiv (3)_{10}$

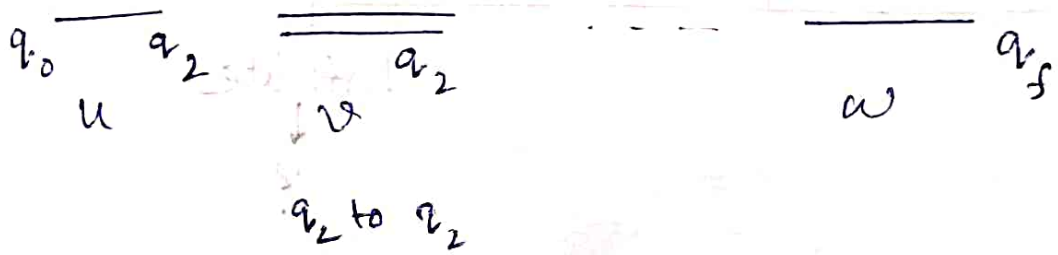
$(111)_2 \equiv (7)_{10}$



$L = \{ 0^n 1^n \mid n \geq 0 \}$

$q_0 \quad q_1 \quad q_2 \quad q_3 \quad q_4 \quad q_5 \quad q_f$

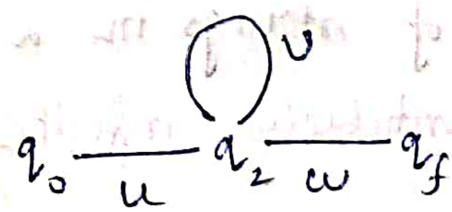
$x = \times \quad \times \quad \times \quad \times \quad \times \quad \times$ string



* $\boxed{\text{pumping}}$

If $uvw \in L$

then $uvvw \in L$

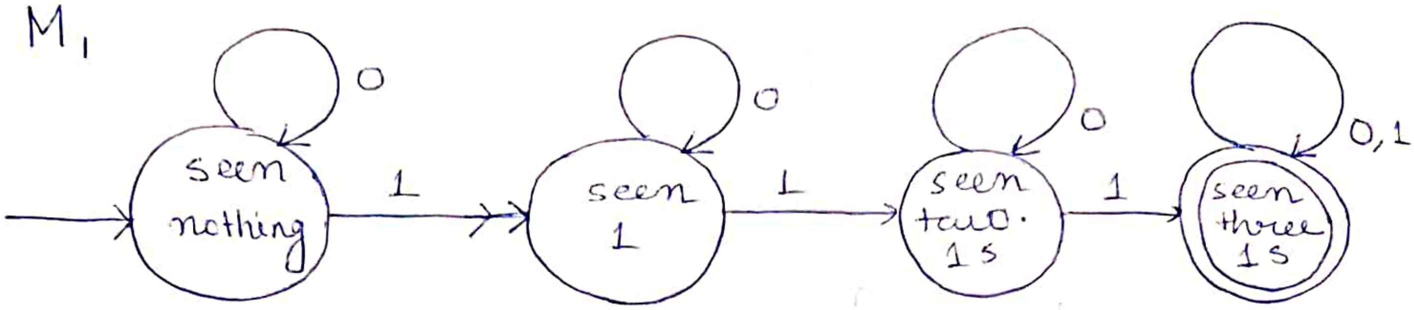


$uv^iw \in L$ ~~for~~ $\forall i \geq 0$

* If an infinite language is regular it must have pumping property.

1. (a) $L_1 = \{ w \in \{0,1\}^* \mid w \text{ contains atleast three } 1s \}$

M_1



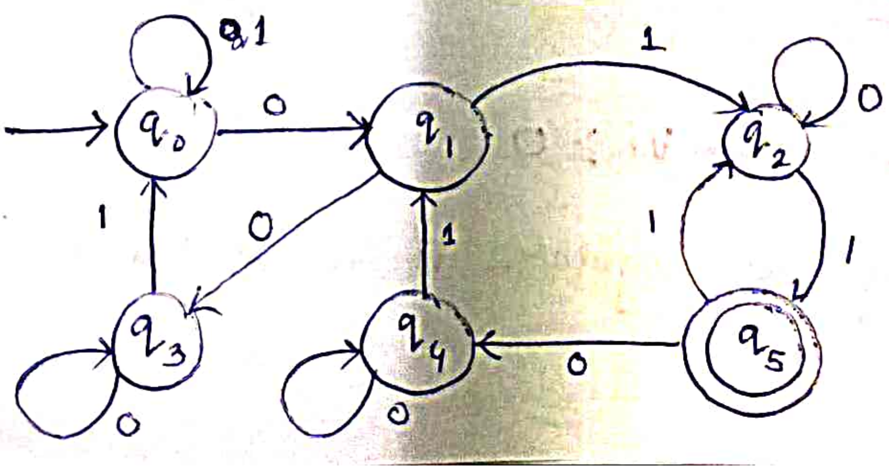
$L(M_1) = L_1$

Language

Finite → Regular

Infinite → ?

No. of strings in a Infinite language is ~~either~~ countablely infinite.



$$x = \overset{r_0}{} a_1 \overset{r_1}{} a_2 \overset{r_2}{} a_3 \overset{r_3}{} a_4 \overset{r_4}{} a_5 \overset{r_5}{} \qquad x \in (0,1)^*$$

r_i = state sequence $\in Q$
a_i = input signal.

$r_0 = q_0$
$r_5 = q_5$

$x = 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1$

$\underbrace{q_0}_{u} \ \underbrace{q_1 \ q_2 \ q_5 \ q_4 \ q_1}_{v} \ \underbrace{q_2 \ q_5}_{w}$

$u = 0 \quad , \quad v = 1101 \quad , \quad w = 11$

$0 \underline{1101} \ \underline{1101} \ 11$ ✓

011 ✓

$0\underline{1101} \ \underline{1101} \ \underline{1101} \ 11$ ✓

Loop must appear in the first n symbols of the string, where n is the no. of states in the DFA.

<u>Pumping Lemma</u>: If L is regular then

- $\exists$ integer $p \geq 1$.
- $\forall$ $s \in L$ with $|s| \geq p$
- $\exists$ strings uvw, satisfy $s = uvw$
  - $|v| \geq 1$
  - $|uv| \leq n$
- $\forall$ integer $i \geq 0$, $uv^iw \in L$

P = pumping lemma cont.

Contrapositive version :

L is regular $\rightarrow$ pumping property

$\neg$ pumping property $\rightarrow$ $\neg$ L is regular

$$\left.\begin{array}{l} \neg(\forall_x A(x)) = \exists_x(\neg A(x)) \\ \neg(\exists x B(x)) = \forall_x(\neg B(x)) \end{array}\right\}$$

If
- $\forall$ integer $p \geq 1$
- $\exists$ string $s \in L$ with $|s| \geq p$
- $\forall$ strings uvw satisfying $s = uvw$
  - $|uv| \leq p$
  - $v \geq 1$
- $\exists$ integer $i \geq 0$, $uv^iw \notin L$

Then, the Language L is not regular.

Q.1. $L = \{0^n 1^n \mid n \geq 0\}$

let K be the states.

let a string be $0^K 1^K$.

$\therefore |uv| \leq K$

$\therefore uv = \underbrace{000\ldots0}_{K}$

let, $v = \underbrace{00\cdots0}_{m}$

if $i=2$

$\therefore uv^iw = uvvw$

$\therefore \quad = \underbrace{00\cdots0}_{K}\ \underbrace{00\cdots0}_{m}\ \underbrace{11\cdots1}_{K}$

$K+m \neq K \quad \therefore$ L is not regular.

Q.1. $L_2 = \{ 0^{n^2} \mid n \geq 0 \}$

let, states K.

~~let a 0^{K^2} be~~ let a string 0^{K^2}.

$|uv| \leq K$

$\therefore uv = \underbrace{00\cdots0}_{K}$

$\therefore |w| = K^2 - K$

$v = \underbrace{00\cdots0}_{m}$

if $i=2$.

$uv^iw = uvvw$

$= \underbrace{00\cdots0}_{K}\ \underbrace{00\cdots0}_{m}\ \underbrace{00\cdots0}_{K^2-K}$

$K+m+K^2-K = K^2+m$ [worst case $m=K$; $K^2 < K^2+K < (K+1)^2$; $\therefore K^2+m$ is not perfect square.]

$\therefore$ Not regular.

1.

(a) $L_1 = \{ w c w \mid w \text{ in } \{0,1\}^* \}$

let K be the states in the DFA that accepts L_1, i.e pumping lemma const.

let a string be, $0^{K/2} 1^{K/2} 0^{K/2} 1^{K/2}$

Now, according to pumping lemma,

$|uv| \leq K$

$|v| \geq 1$

$\therefore \quad uv = 0^{\lceil K/2 \rceil} 1^{\lceil K/2 \rceil}$

$= \underbrace{00 \ldots 0}_{K/2} \underbrace{11 \ldots 1}_{K/2}$

$\therefore$ let, $|v| = m \leq \lceil K/2 \rceil$

$\therefore \quad v = \underbrace{11 \ldots 1}_{m}$

$\therefore \quad w = 0^{K/2} 1^{K/2}$

$= \underbrace{00 \ldots 0}\, \underbrace{11 \ldots 1}$

According to pumping lemma, if $uvw \in L_1$, $uv^iw \in L_1, \forall i \geq 0$.

If we take, $i = 0$

$uv^0w = 0^{\lceil K/2 \rceil} 1^{(\lceil K/2 \rceil - m)} 0^{\lceil K/2 \rceil} 1^{\lceil K/2 \rceil}$

Clearly, $uv^0w \notin L_1$.

$\therefore$ L_1 does not satisfy pumping ~~lemma~~ property. Hence, it is not ~~regural~~ regular.

(b) $L_2 = \{0^i 1^j \mid i > j\}$

let, K be the states in the DFA that accepts L_2.

let a string be $0^K 1^{(K-1)}$

Now, according to pumping lemma,

$|uv| \leq K$

Let's take $uv = 0^K$

again, $|v| \geq 1$

let's take $v = 0^m$ & $m \geq 1$

and, $w = 1^{(K-1)}$

According to pumping lemma, if $uvw \in L_2$,

$uv^i w \in L$, $\forall i \geq 0$

If we take $i = 0$,

$uv^0 w = 0^{(K-m)} 1^{K-1}$

now, as we have taken $m \geq 1$

$(K-m) \leq K-1$

$\therefore$ $uv^0 w \notin L_2$

$\therefore$ L_2 does not satisfy pumping property.

Hence L_2 is not regular.

(c) $L_3 = \{ w \text{ in } \{0,1\}^* \mid w = w^R \}$

$w = w^R$, this suggests that the strings are palindrome.

let, K be the no. of states in the DFA that accepts L_3 i.e pumping lemma const.

Now, let's take a string $x = 0^K 1 1 0^K$

let $x = uvw$

For this ~~to~~ language to have pumping property

$|uv| \leq K$

let, $uv = 0^K$

and, $|v| \geq 1$

let, $v = 0^m$

$\therefore w = 110^K$

Now, if $uvw \in L_3$, $uv^i w$ should also belong to L_3 for all $i \geq 0$.

let, $i = 2$,

$\therefore x' = uv^2 w$

$= 0^K 0^m 1 1 0^K$

Clearly, $x' \neq x'^R$

$\therefore x' \notin L_3$

$\therefore$ This language does not satisfy pumping property.

$\therefore$ ~~This~~ L_3 is not regular.

(d) $L_4 = \{(10)^p 1^q \mid p, q \in N, p \geq q\}$

let, k be the pumping lemma const.

let, take a string $x = (10)^K 1^K$

let, $x = uvw$.

For language to have pumping property.

$|uv| \leq K$ & $|v| \geq 1$

Now, ~~for~~ if K is even,

$uv = (10)^{K/2}$

$v = (10)^m$, $1 \leq m \leq K/2$

if, K is odd,

$uv = (10)^{\frac{K-1}{2}}$

$v = (10)^m$, $1 \leq m \leq \frac{K-1}{2}$

$w = 1^K$

Now. uv^0w should also belong to L_4.

for, first case,

$uv^0w = (10)^{K/2 - m} (10)^{K-(K/2-m)} 1^K$

for 1st case,

$\therefore$ $w = (10)^{K-K/2} 1^K$

for 2nd case,

$w = (10)^{K-(\frac{K-1}{2})} 1^K$

$\therefore$ For, 1st case,

$uv^0w = (10)^{K/2-m} (10)^{K-(\frac{K}{2})} 1^K$

$= \cancel{(10)^{\frac{2K+1-2m}{2}} 1^K} = (10)^{K-m} 1^K$

Surely, $K - m < K$

$\therefore\ uv^0w \notin L_4$

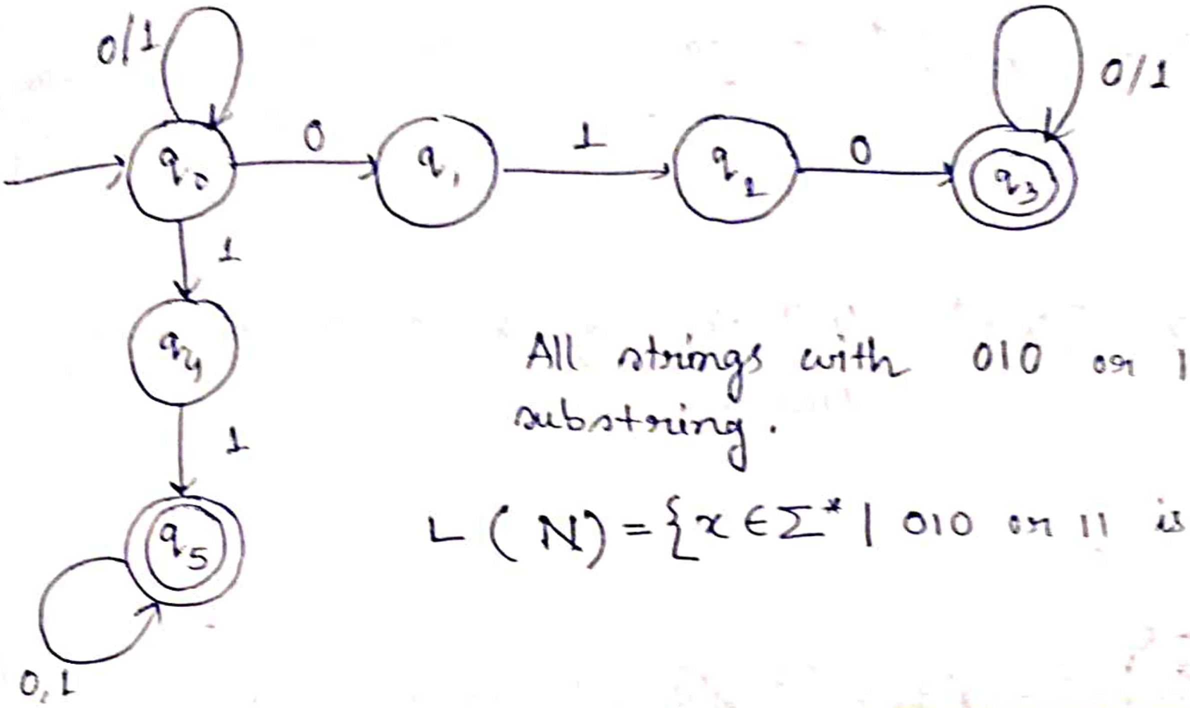
For, ~~case~~ 2nd case,

$uv^0w = (10)^{\frac{K-1}{2} - m} (10)^{K - \frac{K-1}{2}} 1^K$

$= (10)^{K-m} 1^K$

$\therefore\ uv^0w \notin L_4$.

As, this languag L_4 does not satisfy pumping property, L_4 is not regular.



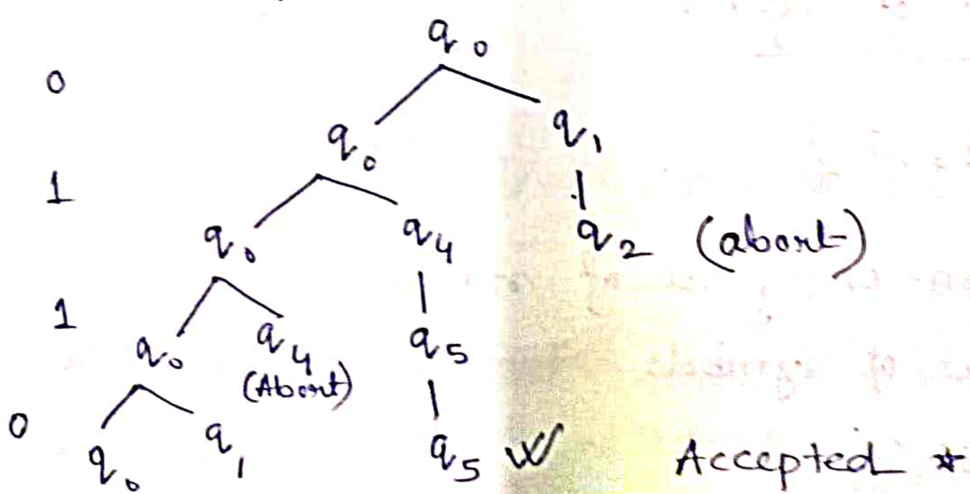
All strings with 010 or 11 as substring.

$L(N) = \{x \in \Sigma^* \mid 010 \text{ or } 11 \text{ is a substring}\}$

- $\boxed{\text{Nonditerministic Finite Automata}}$

If any path leads to finite state $\rightarrow$ accepted.

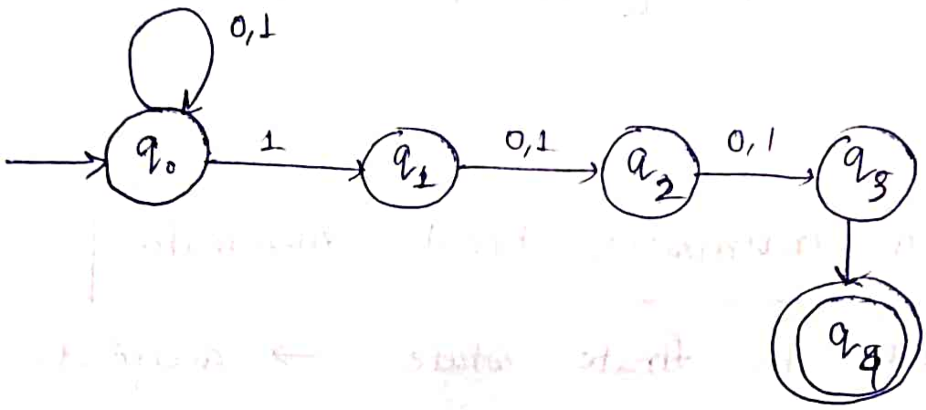
0 1 1 0



- DFA $\delta(p, a) = q \in Q$
- NFA $\delta(p, a) = R \subseteq Q$.

$L(N) = \{ x \in \Sigma^* \mid x$ can take the machine from initial state to its final state$\}$

$L = \{ x \in \{0,1\}^* \mid$ 4 th bit of x from the right end is $1\}$



1 000
1 0 0 1
1 0 1 0
1

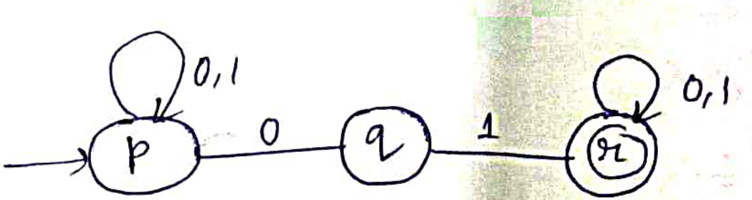
Formal definition of NFA :

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

Q : Finite non-empty set of states

Σ : Finite set of symbols

$\delta : Q \times \Sigma \rightarrow P(Q)$



$\delta(p,0) = \{p, q\}$

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

$\delta(q, 0) = \phi$
$\delta(q, 1) = \{r\}$
$\delta(r, 0) = \{r\}$
$\delta(r, 1) = \{r\}$

$\delta^*(q, x) \longrightarrow$ DFA

NFA $\rightarrow$ Inductive definition of δ^*

$\delta^*(p, \epsilon) = \{p\}$ [base case]

let, $\delta^*(p, x) = R \subseteq Q$

$$\delta^*(p, xa) = \bigcup_{r \in R} \delta(r, a)$$

$L = \{x \in \Sigma^* \mid \delta^*(q_0, x) = R, \ F \subseteq R\}$ X

$L = \{x \in \Sigma^* \mid \delta^*(q_0, x) \cap F \neq \phi\}$

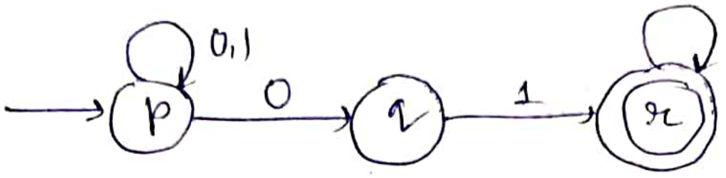
- Class of language accepted by DFA is a subset of class of language accepted by NFA

DFA is a sub case of NFA.

For every ~~DFA~~ NFA there exists a DFA accepting the same language.

* Proof by construction:

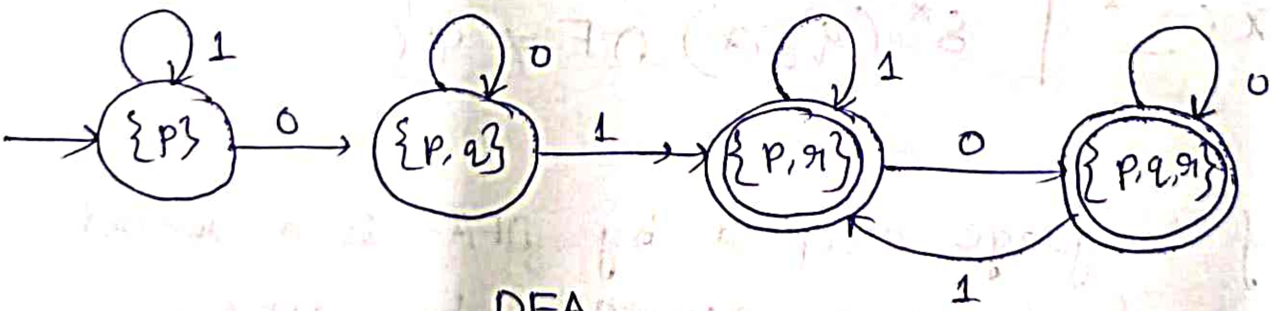
Construct a DFA from the NFA



NFA

	0	1
→ {p}	{p, q}	{p}
{p, q}	{p, q}	{p, r}
{p, r}	{p, q, r}	{p, r}
{p, q, r}	{p, q, r}	{p, r}

$F = \{ \{p, r\}, \{p, q, r\} \}$

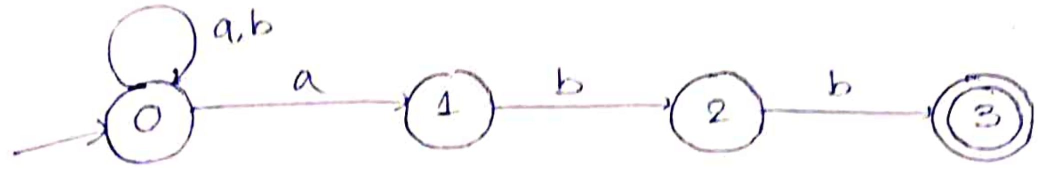


DFA

∴ Class of language accepted by NFA is a subset of class of language accepted by DFA

∴ L accepted by NFA = L accepted by DFA

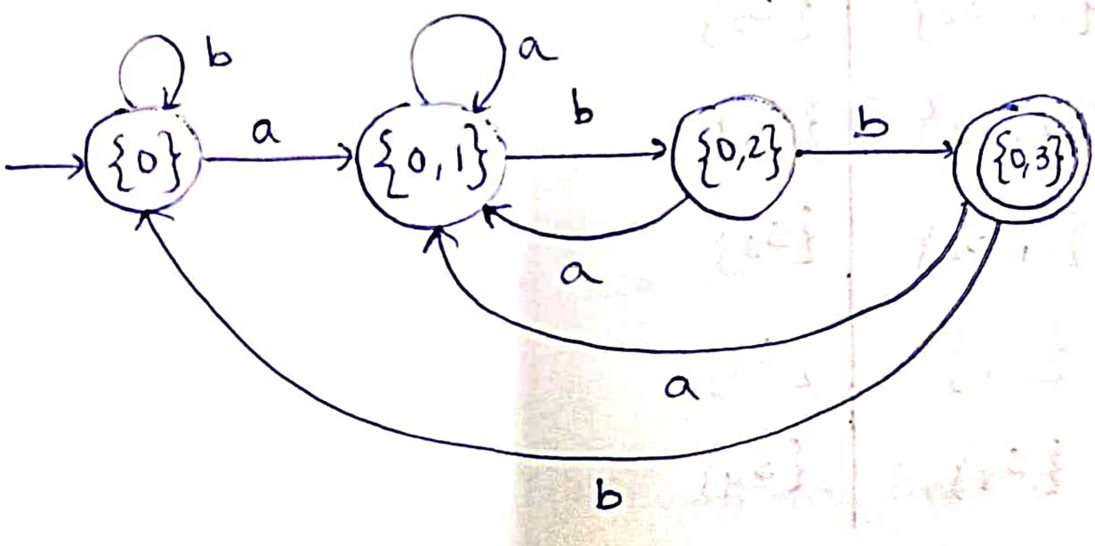
ll

1\. 

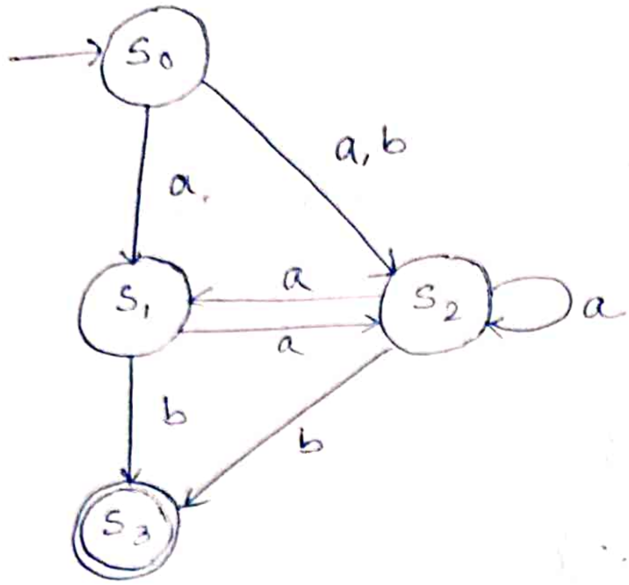
$L(N) = \{ x \in \{a,b\}^* \mid x \text{ ends with } abb \}$

	a	b
{0}	{0,1}	{0}
{0,1}	{0,1}	{0,2}
{0,2}	{0,1}	{0,3}
{0,3}	{0,1}	{0}

DFA :

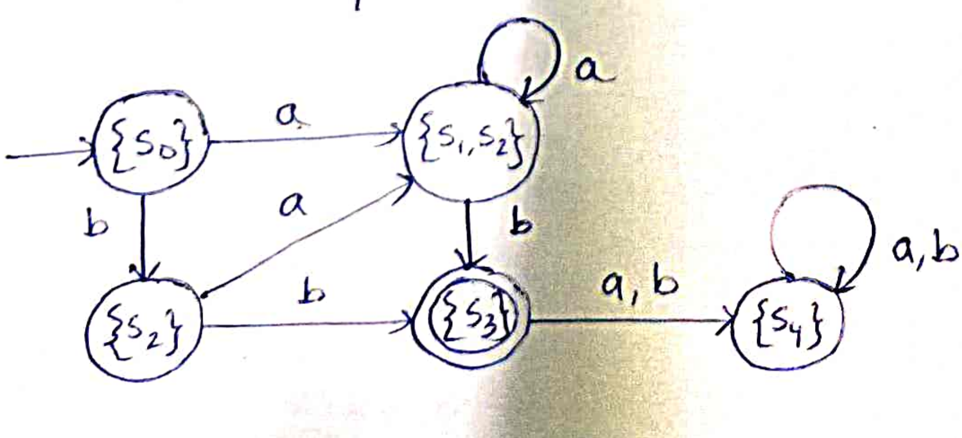


#3



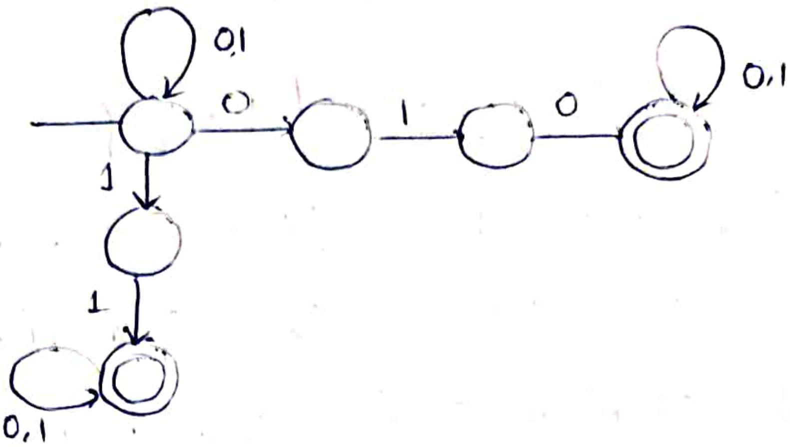
$$L(N) = \{ x \in \{a,b\}^* \mid x = a^{(i+1)}b \text{ or } x = b a^i b, \forall i \geq 0 \}$$

	a	b
$\{s_0\}$	$\{s_1, s_2\}$	$\{s_2\}$
$\{s_1, s_2\}$	$\{s_1, s_2\}$	$\{s_3\}$
$\{s_2\}$	$\{s_1, s_2\}$	$\{s_3\}$
$\{s_3\}$	$\{s_4\}$	$\{s_4\}$
$\{s_4\}$	$\{s_4\}$	$\{s_4\}$



$$L = \{ x \in \{0,1\}^* \mid x \text{ has } 010 \text{ \& } 11 \text{ as a substring} \}$$

0110010101



Suppose,

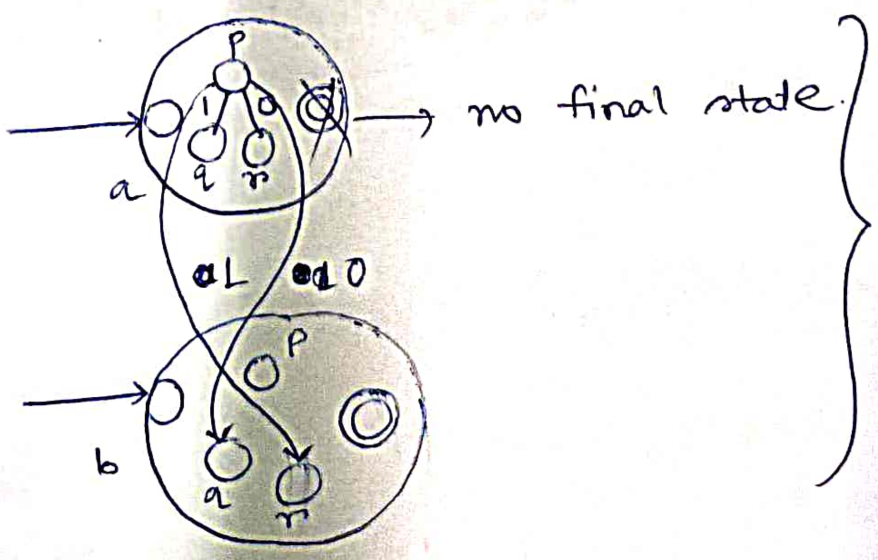
L is regular. $\exists$ exist DFA M | L(M) = L

$L_1 = \{ y \in \{0,1\}^* \mid$ there is an $x \in L$, exactly one bit of which is flipped to obtain $y \}$

$L = \{01, 010\}$

$L_1 = \{00, 11, 110, 000, 011\}$

* L_1 should also be regular



DFA M

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

NFA M_1,

$M_1 = \{Q_1, \Sigma, \delta_1, q_0, F_1\}$

$Q_1 = \{ (q,a), (q,b) \mid q \in Q \}$ $\quad 2 \times |Q|$

$\delta_1((q,a),0) = \{ (\delta(q,0),a), (\delta(q,1),b) \}$

$\delta_1((q,a),1) = \{ (\delta(q,1),a), (\delta(q,0),b) \}$

$\delta_1((q,b),0) = \{ (\delta(q,0),b) \}$

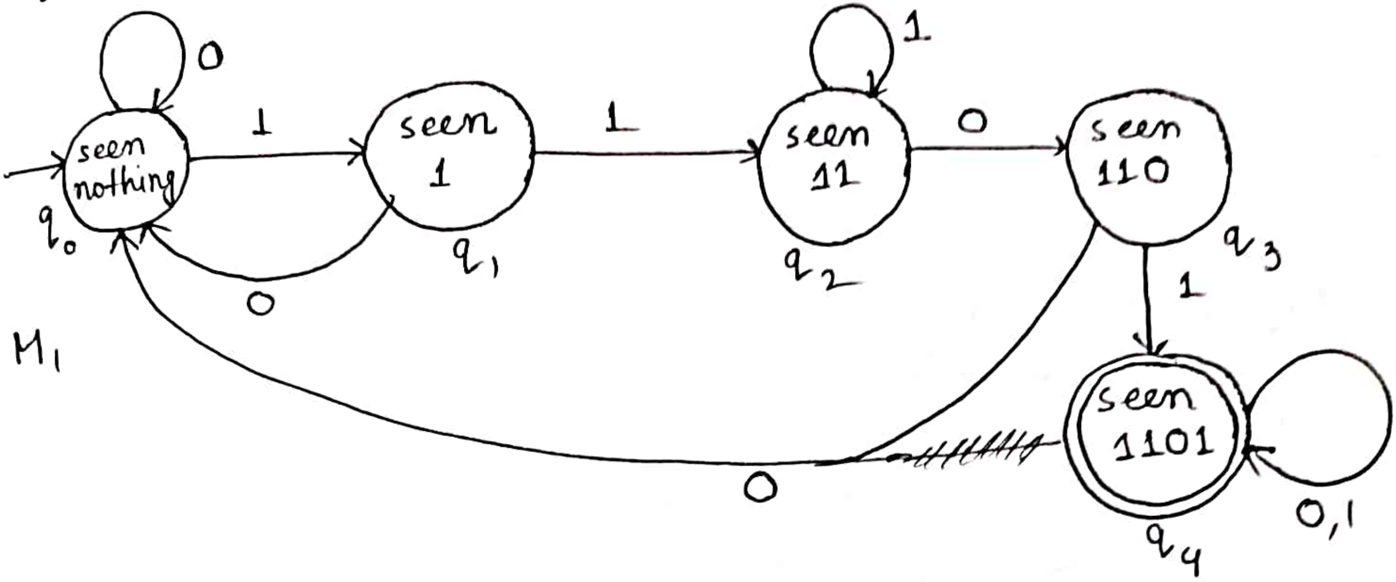
$\delta_1((q,b),1) = \{ (\delta(q,1),b) \}$

$q = \cancel{\{q_0, a\}}\ (q_0, a)$

$F_1 = \{ (q,b) \mid q \in F \}$

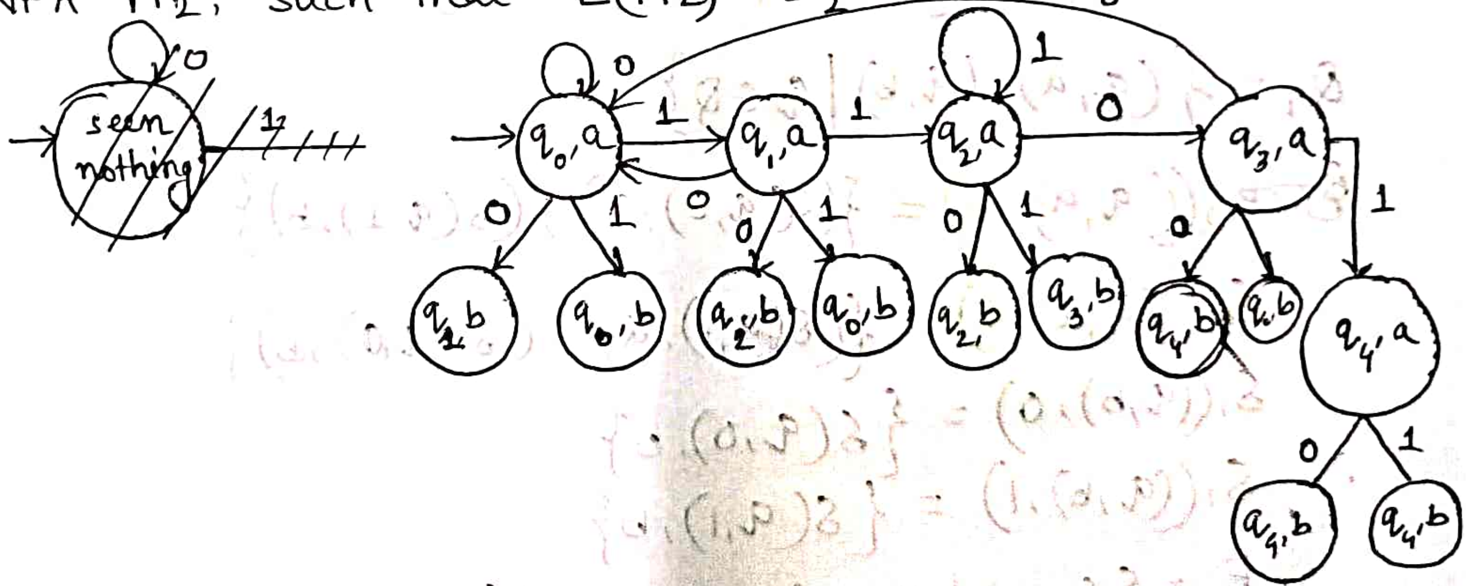
1. $L_1 = \{x \in \{0,1\}^* \mid x$ contains 1101 as a substring$\}$

DFA M_1, such that $L(M_1) = L_1$

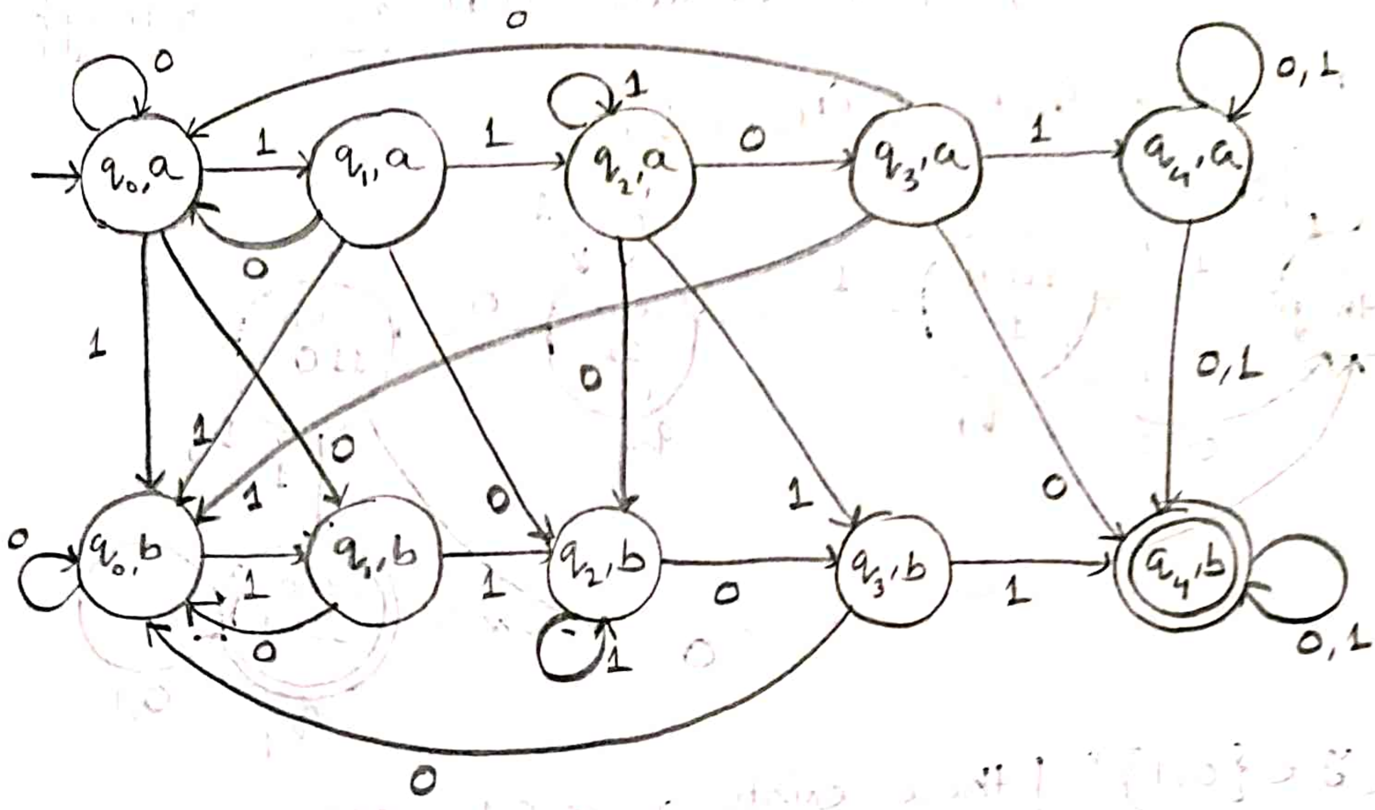


$L_2 = \{y \in \{0,1\}^* \mid$ there exists a $x \in L_1$, exactly one bit of which is flipped to obtain $y\}$

NFA M_2, such that $L(M_2) = L_2$.



NFA M_2 such that $L(M_2) = L_2$



Say, DFA $M_1 = \{Q, \Sigma, \delta, q, F\}$

NFA $M_2 = \{Q_1, \Sigma, \delta_1, q_1, F_1\}$

$Q_1 = \{(q,a), (q,b) \mid q \in Q\}$

$\delta_1((q,a),0) = \{(\delta(q,0),a), (\delta(q,1),b)\}$

$\delta_1((q,a),1) = \{(\delta(q,1),a), (\delta(q,0),b)\}$

$\delta_1((q,b),0) = \{\delta(q,0),b\}$

$\delta_1((q,b),1) = \{\delta(q,1),b\}$

$F_1 = \{(q,b) \mid q \in F\} = \{(q_4,b)\}$

$q_1 = \{(q_0,a)\}$

	0	1
$\rightarrow (q_0, a)$	$\{(q_0, a), (q_1, b)\}$	$\{(q_1, a), (q_0, b)\}$
(q_1, a)	$\{(q_0, a), (q_2, b)\}$	$\{(q_2, a), (q_0, b)\}$
(q_2, a)	$\{(q_3, a), (q_2, b)\}$	$\{(q_2, a), (q_3, b)\}$
(q_3, a)	$\{(q_0, a), (q_4, b)\}$	$\{(q_4, a), (q_0, b)\}$
(q_4, a)	$\{(q_4, a), (q_4, b)\}$	$\{(q_4, a), (q_4, b)\}$
(q_0, b)	$\{(q_0, b)\}$	$\{(q_1, b)\}$
(q_1, b)	$\{(q_0, b)\}$	$\{(q_2, b)\}$
(q_2, b)	$\{(q_3, b)\}$	$\{(q_2, b)\}$
(q_3, b)	$\{(q_0, b)\}$	$\{(q_4, b)\}$
(q_4, b) (final state)	$\{(q_4, b)\}$	$\{(q_4, b)\}$

Questions about DFA

1. Does a DFA M accept any string at all?

Algo 1) Do a DFS from initial state to see if one of the final state is reachable.

Algo 2) If the DFA has N states, then the DFA accepts anything if it accepts a string $< n$

take those string one by one and feed them to the DFA M to see if it is accepted.

strings $< n \rightarrow \epsilon + 2^1 + 2^2 + 2^3 + \dots + 2^{n-1}$

assuming, $\Sigma^* = \{0, 1\}$

2. Does a DFA M accepts a given input string w?

Algo. 1. Write a program to emulate the behaviour of the DFA ~~and~~

2. Feed the string w to the program and see if it takes you to any of the final states and answer accordingly.

Claim: If a DFA M accepts anything it must accept a string of length $< n$.

(contradiction)

proof: There exists a ~~stat~~ a shortest accepted string of length $>= n$

Then the states encountered in processing w must repeat somewhere by PHP.

~~Short~~ Short circuit the loop to get a string to get a string shorter than w that is accepted by DFA M, contradicting the assumption

3. Does a DFA M accepts all strings? assuming $\Sigma = \{0,1\}$ or, Is $L(M) = \Sigma^*$?

Algo 1. Search to see if there is a path to non accepting state and give the opposite answer.

Algo 2: Transform M into a machine M' such that $L(M') = (L(M))'$

if, $L(M') = \epsilon$

then M accepts all strings.

if $L(M')$ is not empty, then give opposite answers.

4. Do $L(M_1)$ and $L(M_2)$ have any string in common?

or. Is $L(M_1) \cap L(M_2) \neq \phi$?

<u>Algo</u>: Design a DFA M_3 such that

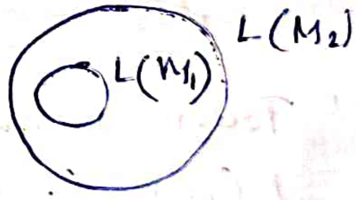
$$L(M_3) = L(M_1) \cap L(M_2)$$

Ask if M_3 accepts any string or not i.e. $L(M_3) = \in$ or not.

And give the same ~~was~~ answer.

5. Is $L(M_1) \subseteq L(M_2)$?

$L(M_1) \subseteq L(M_2)$ iff $L(M_1) \cap L\overline{(M_2)} = \phi$



- get a DFA M_3 such that

  $L(M_3) = \cancel{L(M_1) \cap}\ L\overline{(M_2)}$

- Get another DFA M_4, such that

  $L(M_4) = L(M_1) \cap L(M_3)$

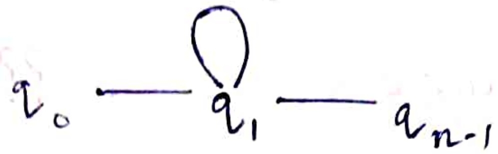
- Ask $L(M_4)$ is empty.
- Give same answer.

6. Is $L(M_1) = L(M_2)$?

- $L(M_1) \subseteq L(M_2)$ or not
- $L(M_2) \subseteq L(M_1)$ or not
- say yes if both say yes.

7. Is $L(M)$ a finite set ?

- M has some finite no. of states, let it be n.
- Then check there exists a string $|x| \geq n$
- that is accepted by M.



<u>Claim 1</u> If M accepts even one string of N (no. of states) then L(M) is infinite.

- Because we can pump to generate more strings in the language.

<u>Claim 2</u> If L(M) is infinite then M accepts at least one string x such that

$$N \leq |x| \leq 2n$$

Algo.
- try all strings ~~Boo~~ of length $n \ldots 2n-1$
- if any of them are in L(M) the L(M) is infinite.
- if ~~any of them are~~ all of them are not in L(M) then it is finite.

Proof Claim 2 :

$$n \leq |x| < 2n$$

- L (M) is infinite if includes a string length $\geq 2n$.
- Choose one x of minimum length $\geq 2n$
- Apply pumping lemma to x until
  $x = uvw$, $uv \leq n$, $|v| \geq 1$
- pumping down. uw is $L(M)$
- Show that $n \leq uw < 2n$. ??

$uw \geq n$
because $|uvw| = |x| \geq 2n$ and $v \leq uv \leq n$

Regular language

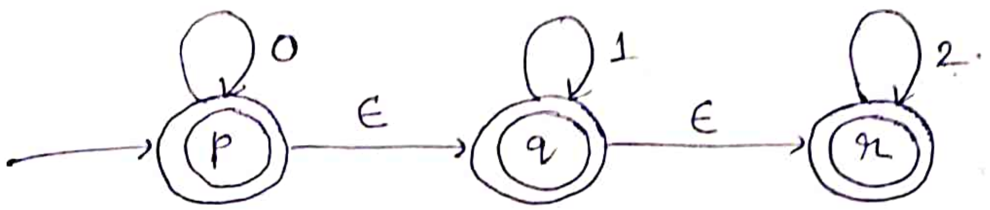
1. M/c counterpart
2. Regular expression
3. Regular grammer

PDA
Push Down Automata

M/c
grammer

Tuning M/C

NFA's with ε transition



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

$\epsilon\epsilon\; a_1 \epsilon\, a_2 \epsilon\, a_3 \epsilon\epsilon\, a_4 \;\ldots\; a_n \epsilon \qquad a_i \in \Sigma$

$0 0 1 1\; 2 \in L(M)$

$2 1 1 \notin L(M)$

$L(M) = \{ 0^i\, 1^j\, 2^k \mid i, j, k \geq 0 \}$

- If a language accepted by an NFA iff it is accepted by a DFA.

▷ True for NFA with ε ? → Yes

$p_i \in Q$

- ϵ - Closure $(p_i) = \{ q_j \mid q_j$ is reachable from p_i using 0 or more ϵ transition $\}$

[state]

ϵ - Closure $(p) = \{p, q, r\}$

ϵ - closure $(q) = \{q, r\}$

ϵ - closure $(r) = \{r\}$

ϵ - closure (P), when $P \subseteq Q$ [set of state]

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

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

$\delta : Q \times (\Sigma \cup \epsilon) \rightarrow P(Q)$

$\hat{\delta} : Q \times \Sigma^* \rightarrow P(Q)$

$\hat{\delta}(p, x) = \{$ the set of all states M can reach from p on input $x \}$

$\hat{\delta}(p, \epsilon) = \epsilon\_closure(p)$

$\hat{\delta}(p, 0) = R \subseteq Q$

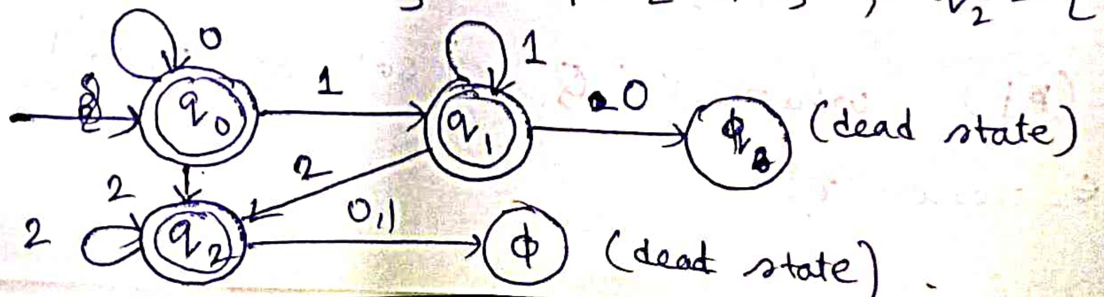
$\hat{\delta}(p, 01) = \epsilon\_closure(R')$ where $R' = \bigcup_{r \in R} \delta(r, 1)$

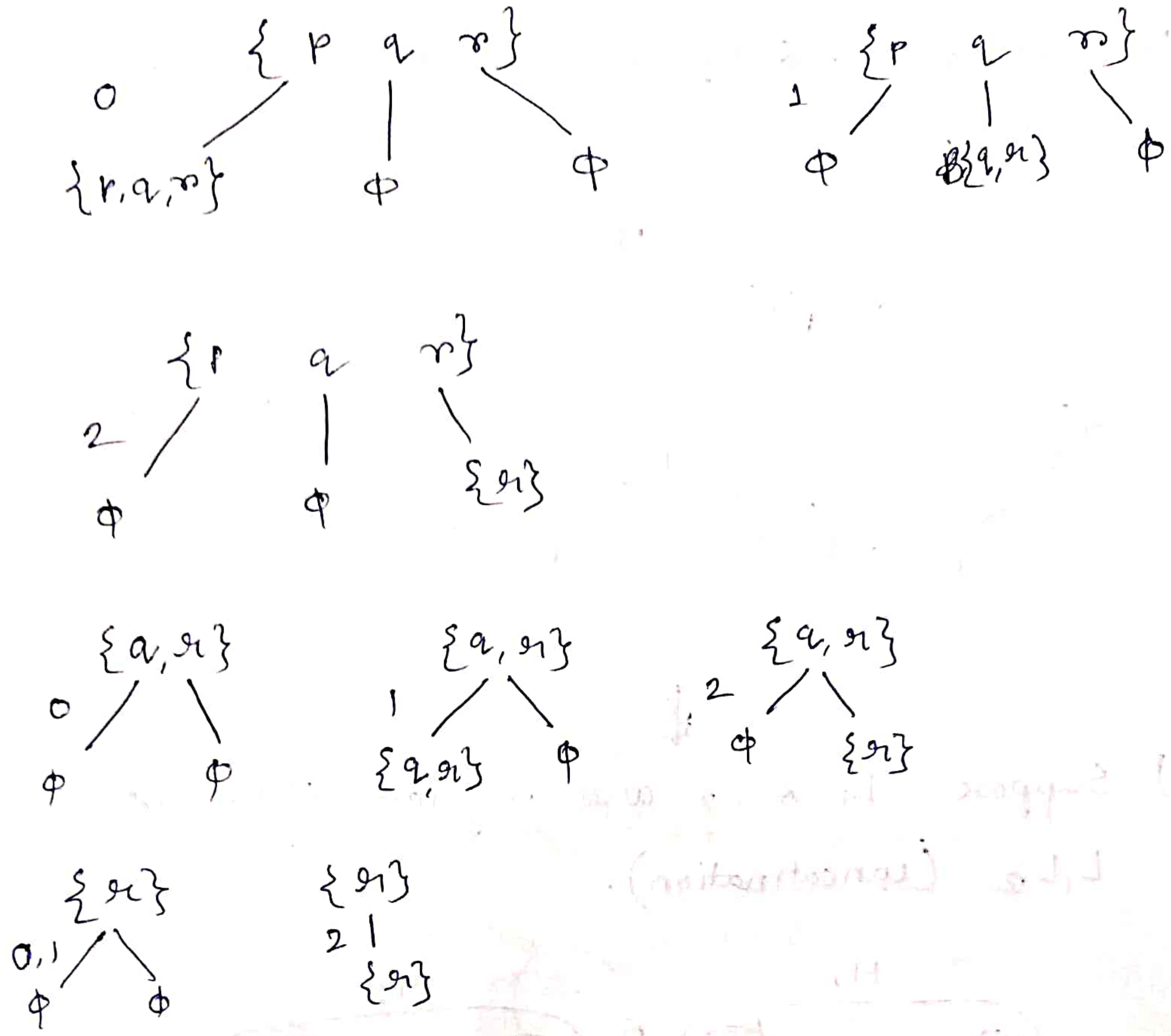
$\hat{\delta}(q, w, a) = \epsilon\_closure(P)$ where $P = \{ p \mid$ for some r in $\hat{\delta}(q, w)$, p is in $\delta(r, a) \}$

ϵ-NFA to DFA

initial state of DFA = ϵ- closure of initial state of NFA.

$q_0 = \{p, q, r\}$, $q_1 = \{q, r\}$, $q_2 = \{r\}$

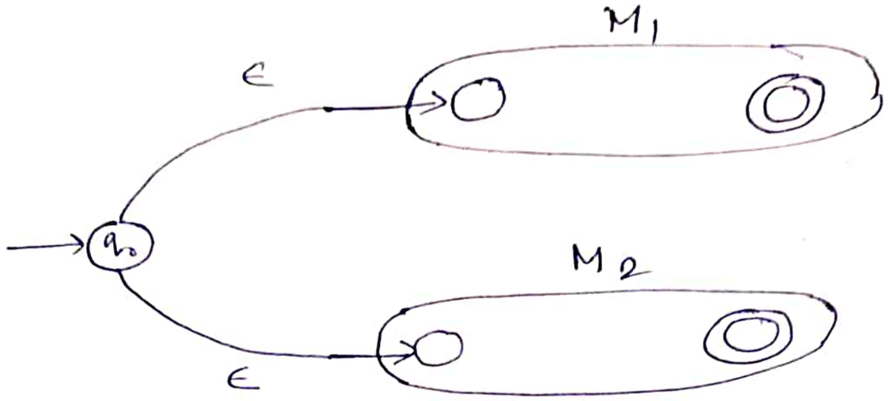




** $L = \{ 0^i 1^j 2^k \mid i = j = k \}$ $\longrightarrow$ No DFA/NFA possible

Counting is involved and it can go upto ∞.

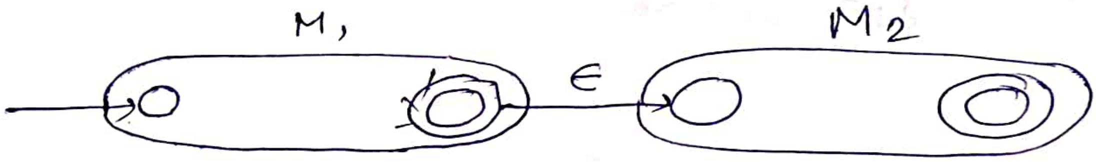
▷ Suppose L_1 & L_2 are regular then so is $L_1 \cup L_2$



done !!

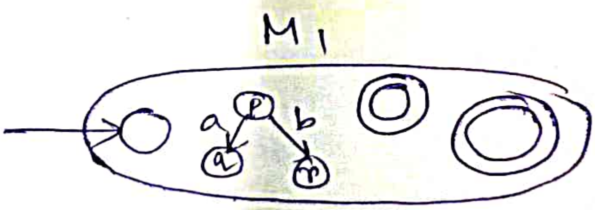
&

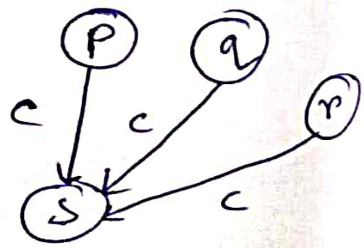
▷ Suppose L_1 & L_2 are regular then so is L_1L_2 (concatination).



▷ Suppose L is regular then so is L^R

$$L^R = \{x \in \Sigma^* \mid x^R \in L\}$$

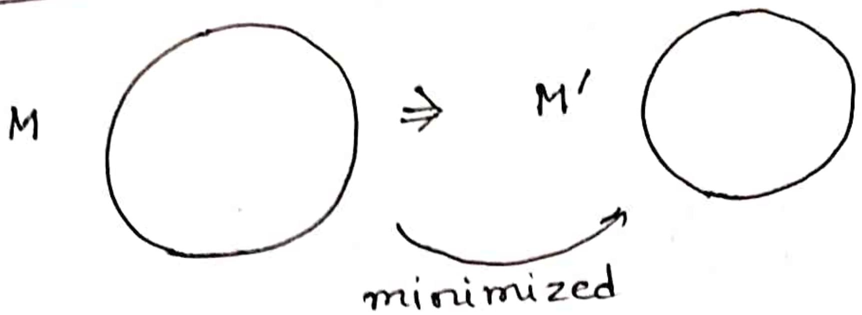




~~NFA~~ DFA

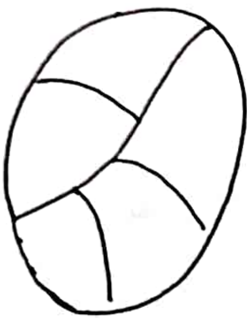
Minimization of DFA and Myhill Nerode theorem

$L(M) = L(M')$



The theorem says that, there exists unique best DFA to accept a given regular language.

DFA as algorithm to solve membership problem.



equivalence classes.

No. of equiv. classes reduced by an equiv. relation is called index of that relation.

$$P_1 \cup P_2 \cup P_3 \cdots \cup P_n = S$$

$$P_i \cap P_j = \phi \quad , \text{if } i \neq j$$

Index could be finite or infinite.

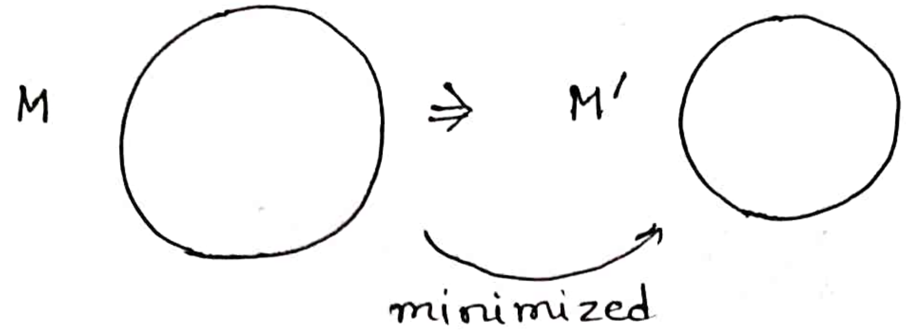
Infinite $\rightarrow$ $x = xy$.

Regular language L

$M = \{Q, \Sigma, \delta, q_0, F\}$

$L(M) = L \quad , \quad L \subseteq \Sigma^*$

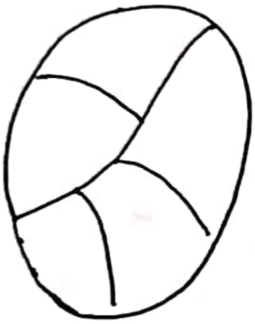
Minimization of DFA and Myhill Nerode theorem



$L(M) = L(M')$

The theorem says that, there exists unique best DFA to accept a given regular language.

DFA as algorithm to solve membership problem.



equivalence classes.

No. of equiv. classes reduced by an equiv. relation is called index of that relation.

$$P_1 \cup P_2 \cup P_3 \cdots \cup P_n = S$$

$$P_i \cap P_j = \phi \quad , \text{if } i \neq j$$

Index could be finite or infinite.

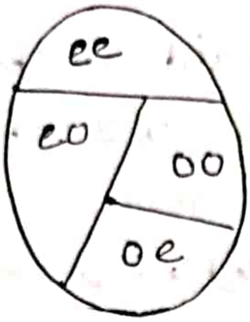
Infinite → $x = xy$.

Regular language L

$M = \{Q, \Sigma, \delta, q_0, F\}$

$L(M) = L \quad , \quad L \subseteq \Sigma^*$

$x \; R_M \; y$ if $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$



even 1 even 0
even 1 odd 0
odd 1 even 0
odd 1 odd 0

1) $x \; R_M \; x \quad \forall x$
2) $x \; R_M \; y \Rightarrow y \; R_M \; x$
3) $x \; R_M \; y \;\&\; y \; R_M \; z \Rightarrow x \; R_M \; z$

__Proof:__

1) $x \; R_M \; x$

$$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, \cancel{y}\, x)$$

2) $x \; R_M \; y$

$$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) = p$$

$$\therefore \hat{\delta}(q_0, y) = \hat{\delta}(q_0, x)$$

$$\Rightarrow y \; R \; x$$

3) $x \; R_M \; y \;\&\; y \; R_M \; z$

$$\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y) = p$$

$$\hat{\delta}(q_0, y) = \hat{\delta}(q_0, z) = p$$

$$\therefore \hat{\delta}(q_0, x) = \hat{\delta}(q_0, z)$$

$$\therefore x \; R_M \; z$$

3 Properties of equivalence relation :

1) Right invariant :

An equiv. relation is said to be right invariant if $x R y \Rightarrow xz \; R \; yz \quad \forall z \in \Sigma^*$

$$\hat{\delta}(q_0, xz) = \hat{\delta}(\hat{\delta}(q_0, x), z)$$
$$= \hat{\delta}(\hat{\delta}(q_0, y), z)$$
$$= \hat{\delta}(q_0, yz)$$

2) Finite index :

Partitions induced by the equiv. relation is finite in number.

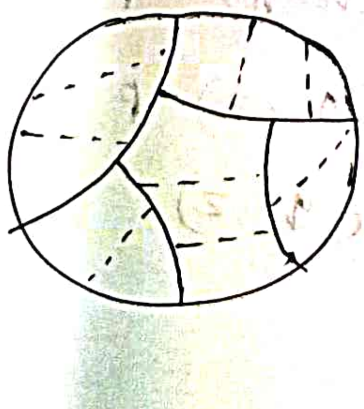
Index = states $\rightarrow$ finite

3) Refinement of a relation :

R_1 & R_2 on same ~~state~~, set, R_1 is a refinement of R_2, iff, $x R_1 y \Rightarrow x R_2 y$

R = being in the same hostel ———

R' = being in the same wings - - -



Myhill Nerode theorem :

The theorem says following statements are equivalent

1. L is accepted by a finite automata.
2. L is the union of the some of the equivalent classes of a right invariant equivalence relation of finite index.
3. R_L is of finite index.

$x R_L y$, iff both xz & yz in L or neither of them are in L. $\forall z \in \Sigma^*$.

$L = \{x \in (01)^* \mid x$ has even 0 and odd 1$\}$

$1111 \in L \rightarrow 111\underline{0} \notin L$

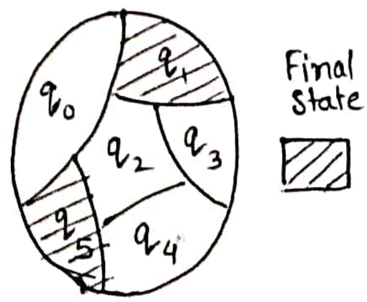
$~~0~~0111 \notin L \rightarrow 0111\underline{0} \in L$ $\quad z = 0$

$\therefore$ 111 and 0111 are not related.

1 $\Rightarrow$ 2

Take R_M as defined,

i) R_M is equivalence relation
ii) R_M is right invariant
iii) R_M is also of finite index



L is the union of the equivalent clases corresponding to finial states. $F = \{q_1, q_5\}$

$\therefore \quad L = q_1 \cup q_5$