THOC

Congruence mod n

$a \equiv_n b$ ie $(a-b)$ is an integer multiple of n

For eg $a = 5$, $b = 1$, $n = 4$

$a \equiv_4 b$

$\therefore a - b = n * k$

Just eg $4 = 4 * k$ $\quad k = 1$

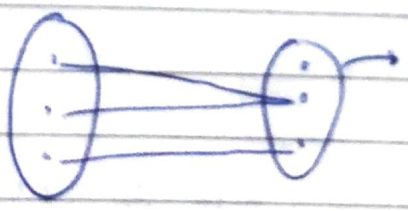
Eg. Find pairs of a,b if $n = 3$

Sol^n $a \equiv_3 b$

$$R = \{(a,3), (a,c), (1,4), (4,1), \ldots\}$$

3 equivalence classes we will get as we have $(a-b)$ %

$\Rightarrow$ Onto

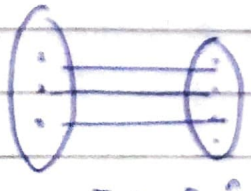


codomain ... set of all possible outputs

Not onto because ~~codom~~ range ≠ codomain

i.e. some elements of range are missing

$\Rightarrow$ Into



For one value of x only one value of y should be in codomain

Into fun^

Date:

eg. $f: R \rightarrow R \qquad f(x) = x^2$

	Into	Onto
$f: R \rightarrow R^+$	No	yes
$f: R^+ \rightarrow R$	Yes	No
$f: R^+ \rightarrow R^+$	Yes	Yes

$a^k = aaa \ldots a \qquad a \in \Sigma \qquad L \subseteq \Sigma^*$
$x^k = xxx \ldots x \qquad x \in \Sigma^*$
$\Sigma^k = \Sigma\Sigma\Sigma \ldots = \{x \in \Sigma^+ \mid |x| = k\}$

$L^k = LLL \ldots L$

$a^0 = \Lambda \qquad \Sigma^0 = \{\Lambda\}$
$x^0 = \Lambda \qquad L^0 = \{\Lambda\}$

⇒ Concatenation

Apply in strings as well as languages (set of strings)

$x \qquad xy$
y

$L_1 \cdot L_2 \subseteq \Sigma^*$

$L_1 L_2 = \{xy \mid x \in L_1 \ \& \ y \in L_2\}$

e.g. $L_1 \quad L_2$
$\{a, b\} \ \{c, d\} = \{ac, ad, bc, bd\}$

e.g. $L_1 = \{ab, bab\}^{*\ (0 \text{ to } \infty)} \cup \{b\}\{bb\}^{*\ (0 \text{ to } \infty)}$

$L_2 = \{ x \in \{a,b\}^* \mid n_a(x) \geq n_b(x) \}$ (number of a, number of b)

Soln

$L_2 = \{ab, aab, aabb, aaab, aba, ba, \ldots\}$

$L_1 = \{ab, bab, abbab, babab, ababab, \ldots\}$ (neglecting $\{b\}\{bb\}$)

$\{b, bbb, abbbb, \ldots\}$

$\{b, bbb\}$ ← neglecting $\{ab, bab\}^*$

$abbbb$ ← mixing ab & bbb

* represents arbitrary time concatenate ab & bab

e.g. $L = \{ byb \mid y \in \{a,b\}^* \}$

Soln $L = \{bb, babb, baaab, bbbb, \underline{bababb}, \ldots\}$

e.g. either start with b or end with b

$L = \{ by \cup yb \mid y \in \{a,b\}^* \}$

⇒ Recursive Def^n

For eg. $n! = \begin{cases} 1 & n = 0 \\ n \cdot (n-1)! & n > 0 \end{cases}$

Recursive def^n of L^*

1) $\wedge \in L^*$
2) For any $x \in L^*$ & $y \in L^*$, $xy \in L^*$
3) No string other than (1) & (2) will come in L^*

Eg. Recursive def^n of Palindrome strings

Data
1) $\wedge \in L^*$
2) For any $x \in L^*$ & $y \in L^*$, $xy = yx$
3) No

Soln
1) $\wedge \in Pal$ {Null string}
2) For any $a \in \Sigma$, $a \in Pal$ {Single character}
3) For any $x \in Pal$ & $a \in Pal$, $axa \in Pal$
4) No string is in pal unless it can be obtained by using 1, 2 & 3.

Q1 The set ω of all strings of the form 0_1^{ij} where $i \geq 2j$

Q2 The set V of all strings of the form 0_1^{ij} where $j \leq i \leq 2j$

Q.

⇒ ~~Structural Induction~~

* Regular Language

→ A regular language over an alphabet Σ is one that can be obtained from given basic languages using the operations Union, Concatenation & kleene.

$L_1 = \{a\}$, $L_2 = \{b\}$
$L_1L_2 = \{ab\}$ $L_1 \cup L_2 = \{a, b\}$ } These are regular languages

→ A regular expression is an algebraic repre^n of regular language.

Lang	R.E.	
$\{\wedge\}$	$\wedge$	$\{\} \rightarrow$ remove.
$\{0\}$	0	$\{\} \rightarrow ()$
$\{001\}$	001	$\cup \rightarrow +$
$\{0\} \cup \{1\} = \{0,1\}$	0+1	
$\{0,10\}$	0+10	
$\{1,\wedge\}\{001\}$	$(1+\wedge)001$	
$\rightarrow \{1\} \cup \{\wedge\}\{001\}$		
$\{110\}^*\{0,1\}$	$(110)^*(0+1)$	
$\{10, 111, 11010\}^*$	$(10+111+11010)^*$	
$\{0,10\}^*(\{11\}^* \cup \{001, \wedge\})$	$(0+10)^*((11)^* + 001 + \wedge)$	
$\{\emptyset\}$	$\emptyset$	
$L_1 \cup L_2$	$\gamma_1 + \gamma_2$	
$L_1 \cdot L_2$	$\gamma_1\gamma_2$	
L_1^*	γ_1^*	

Eg. $(0+1)^* 01 (0+1)^* + 1^*0^* = (0+1)^*$

Are they equal or not.. yes equal

Soln $(0+1)^* = \{\wedge, 0, 1, 00, 11, \ldots\}$

Q Strings of even length $\Sigma = \{0, 1\}$

$L = \{00, 01, 10, 11, 0000, \ldots\} = \{00, 01, 10, 11\}^*$

$RE = (00 + 01 + 10 + 11 + 0000 + \ldots) = (00+01+10+11)^*$

Q Strings with an odd number of 1's

1. $(0^*10^*)(0^*10^*1)^*0^*$
2. $(0^*10^*1)^*(0^*10^*)$
3. $(0^*1\,0^*)(0^*10^*10^*)^*$

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

Soln

Also check that null string must also be accepted in the answer.

$L = \{0\}^*\{1, 111, 11111, \ldots\}$

Ans: $0^*10^*(10^*10^*)^*$

$1(00+11)^*$ ⟶ It does not contain all the language

Q Strings ending in 1 & not containing 00, $\Sigma = \{0, 1\}$

$(1+01)^+$

Q R.E. for identifier of C language. $\Sigma = \{a, \ldots z, A, \ldots Z, 0, \ldots 9, \_\}$ (_ : underscore)

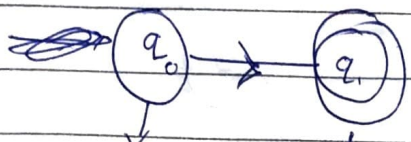
int temp ✓
int 2tempX
int _temp ✓
int temp_ ✓

$l = \{a, \ldots z, A, \ldots Z\}$

$d = \{0, \ldots 9\}$

RE. = $(d + \_)(d + d + \_)^*$

FINITE AUTOMATA

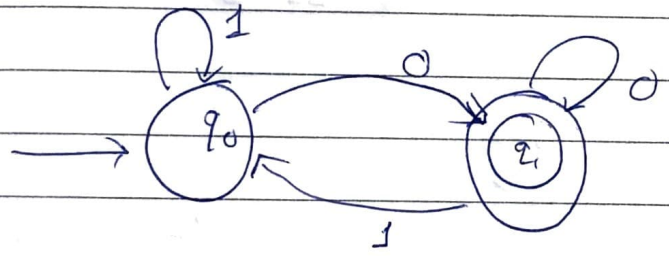


Initial state

Final State/ Accept state

input State

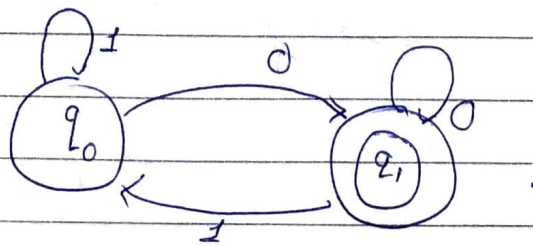
Eg.



initial State → q_0

If 01 is input the ● for 0 $q_0 \rightarrow q_1$
1 $q_1 \rightarrow q_0$

Eg.



1^+0^+

It must end with 0 to achieve q_1

0, 10, 010, 1000

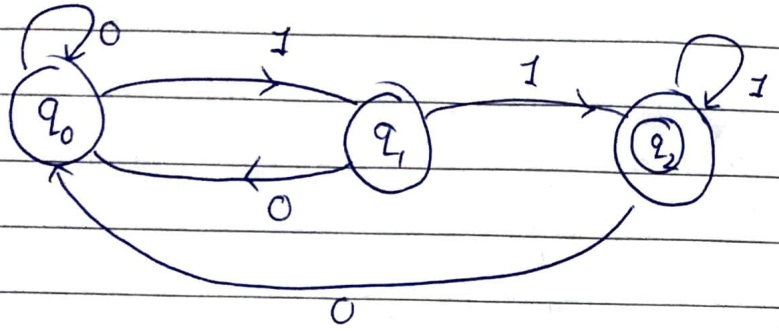
Ans. $(0+1)^*0$
OR
$(0^*1^*)^*0$

Exactly 2 0's ⟶ $1^*01^*01^*$

At least 2 0's ⟶ $(0+1)^*0(0+1)^*0(0+1)^*$

Eg. (q) This is final automata with final state = initial state & it will accept null string

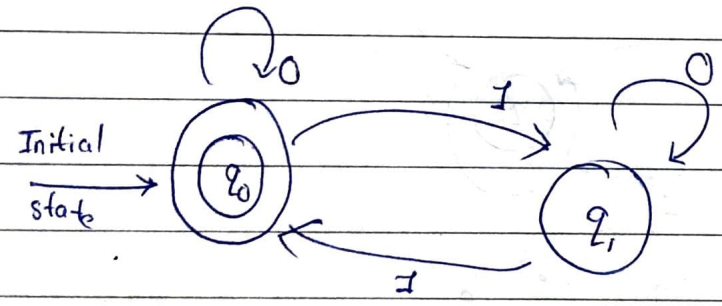
Eg.



~~Language~~ Strings should end ~~11~~ with 11

R.E. = $(0+1)^*.11.$

Eg.



{ ∧, 11, 0, 101, ... }

↓

even number of 1s

∴ R.E. = ~~$0^*10^*10^*$~~ ~~$(00+11)^*$~~ $(0^*(10^*1)^*0^*)^*$

OR

$(0^*10^*10^*)^*$

⇒ Def^n of finite Automata or FSM (Finite State machine)

It is a 5-tuple

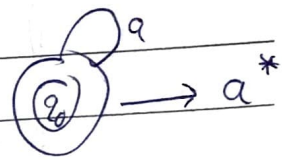
1) Q is a finite set $Q = \{q_0, q_1\}$

2) Σ is an alphabet of input symbols

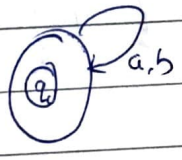
3) $q_0 \in Q$

4) $A/F/q_f \subseteq Q$] → More than one final state

5) Transition fun^n $\delta : Q \times \Sigma \longrightarrow Q$

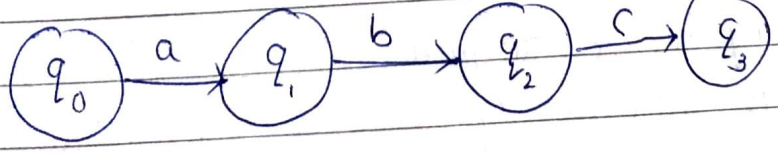
Eg. $L_1 = \{\wedge, a, aa, aaa, \ldots\}$ → Ans → 

Eg. $L_2 = \{\wedge, a, b, ab, bb, aa, \ldots\}$ 

sol^n 

Extended transition fun^n

$\delta^* : Q \times \Sigma^* \longrightarrow Q$

Eg. 

$\delta^*(q_0, abc) = \delta(\delta^*(q_0, ab), c)$

$= \delta(\delta(\delta^*(q_0, a), b), c)$

$= \delta(\delta(\delta(\delta^*(q_0, \wedge), a), b), c)$

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

$$= \delta(\delta(q_2, ab), c)$$

$$= \delta(q_3, abc)$$

Q Let $M = (Q, \Sigma, q_0, A, \delta)$ be an FA. We define the fun^n

$$\delta^* : Q \times \Sigma^* \longrightarrow Q$$

as follows

1) For any $q \in Q$, $\delta^*(q, \lambda) = q$

2) For any $q \in Q$, $y \in \Sigma^*$ & $a \in \Sigma$

$$\delta^*(q, ya) = \delta(\delta^*(q, y), a)$$

⟹ Acceptance of an FA

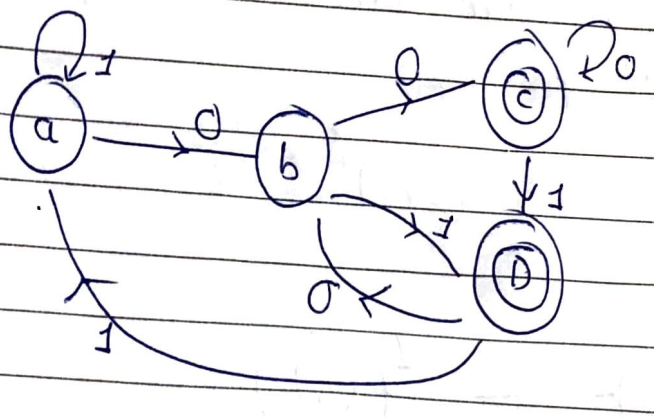
Let $M = (Q, \Sigma, q_0, A, \delta)$ be an F.A. A string $x \in \Sigma^*$ is accepted by M if $\delta^*(q_0, x) \in A$

If a string is not accepted, we say it is rejected by M. The language accepted by M or the language recognized by M, is the set

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

if L is any language over Σ, L is accepted or recognized by M ~~iff~~ iff $L = L(M)$

Eg

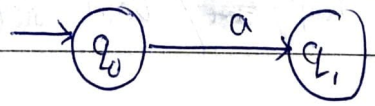


Eg FA, binary strings divisible by 3

⇒ Types of automata

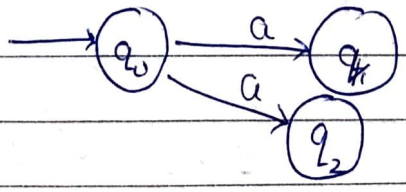
1) DFA (Deterministic)
2) NFA (Non-deterministic)

→ ~~Det~~ DFA mean that for transition fun^n we have clearly possible state to reach then it is called DFA

Eg. 

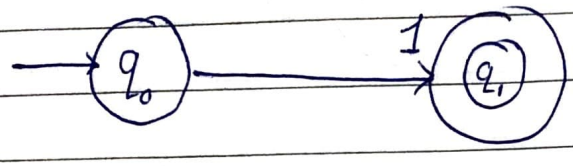
For every input we need to have ~~state~~ transition to one ~~state~~.

→ NFA mean that for transition fun^n we have no clearly possible state to reach then its NFA.

Eg. 

Q Construct a DFA which accept all strings starts with 1 $\Sigma = \{0, 1\}$

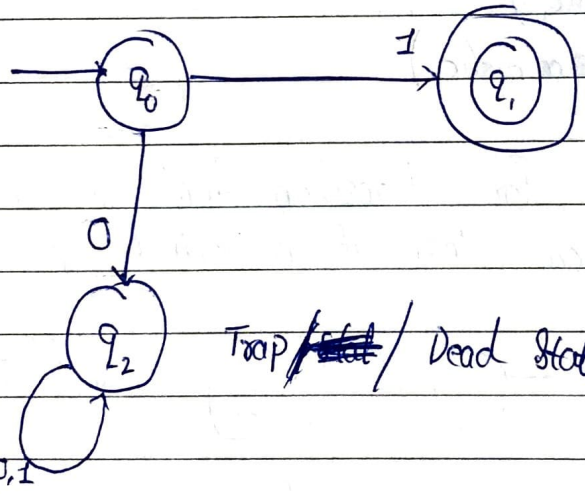
Soln $R.E = 1 \cdot (0+1)^*$, $L = \{1, 11, 10, 1011, \ldots\}$



→ If we make a self loop for input 0 in q_0 then it will accept string 01 too. So NP.

→ If we make $q_0 \xrightarrow{0} q_1$ then also for input 0 it will reach final state. So NP.

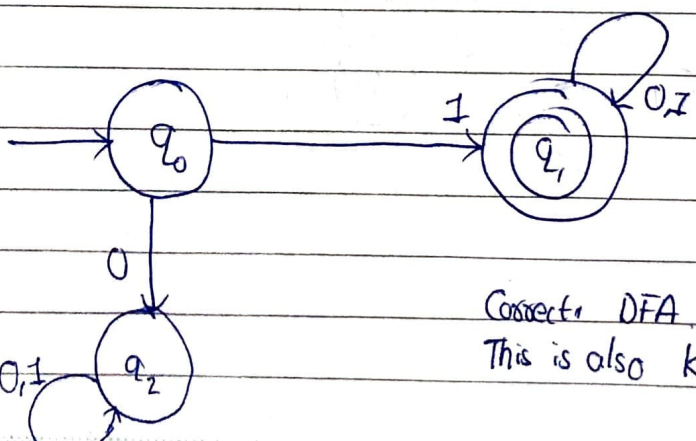
→ So make another state



Trap/Dead State (It indicates that if input is 0 then there is no chance to reach final state)

Did because in DFA it is necessary.

→ Self loop for input 0, 1 to q_1 self loop



Correct DFA

This is also known as transition diagram.

→ Transition State

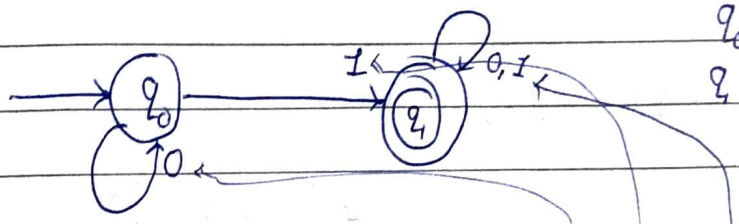
State	Input 0	Input 1
q_0	q_2	q_1
q_1	q_1	q_1
q_2	q_2	q_2

Q $\Sigma = \{0, 1\}$, Strings contain 1

$L = \{1, 01, 100, \ldots\}$

R.E. = $(0+1)^* \cdot 1^+$ or $(0+1)^* \; 1 \; (0+1)^*$

DFA

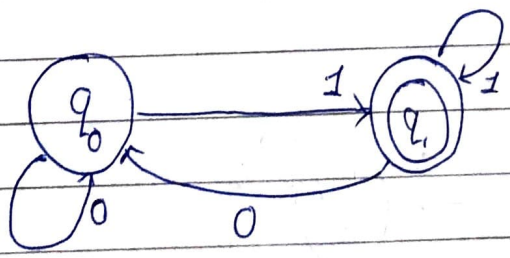


	0	1
q_0	q_0	q_1
q_1	q_1	q_1

By this diagram we can make R.E. = $0^* \; 1 \; (0+1)^*$

Q Strings end with 1

R.E. = $(0+1)^* \cdot 1$, $L = \{0, 001, 101, \ldots\}$



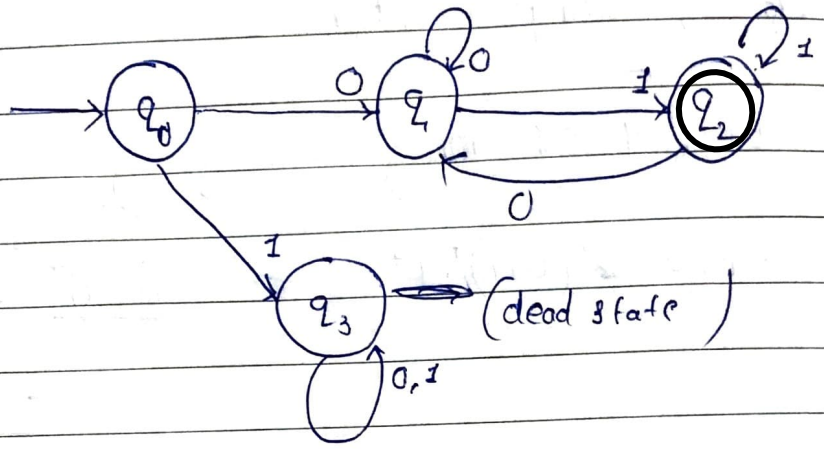
	0	1
q_0	q_0	q_1
q_1	q_0	q_1

→ From q_1 for input 0 ~~then~~ if we do self loop then 10 string will also be accepted.

Q DFA strings start with 0 end with 1

01, 001, 0111

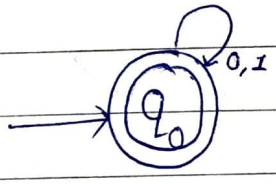
~~DFA~~ RE = $0(0+1)^* 1$



Q DFA which accept language consist of all string 0 & 1

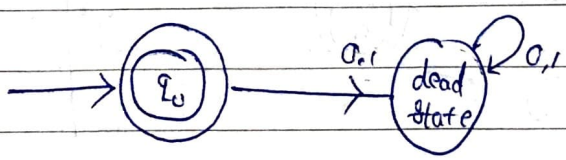
Soln $\Sigma^* = \{0, 1\}^*$

$L = \{\wedge, 0, 1, 00, 01, \ldots\}$

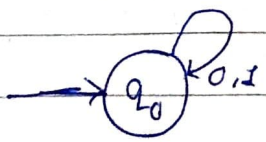


Q DFA $\Sigma = \{0, 1\}$, $L = \{\wedge\}$

Soln



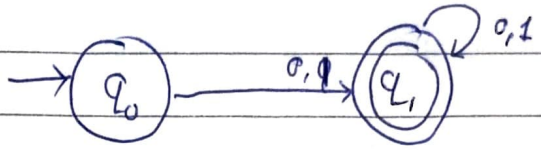
Q DFA, $L = \{\}$, $\Sigma = \{0, 1\}$



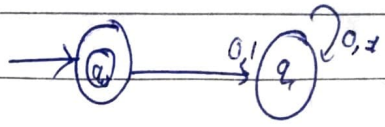
No final state as we don't want to accept anything.

Page No.: YOUVA
Date:

Q DFA $\Sigma = \{0, 1\}$, $\Sigma^* - \{\wedge\}$ (The ~~DFA~~ RE which accepts all string except null)



Another approach

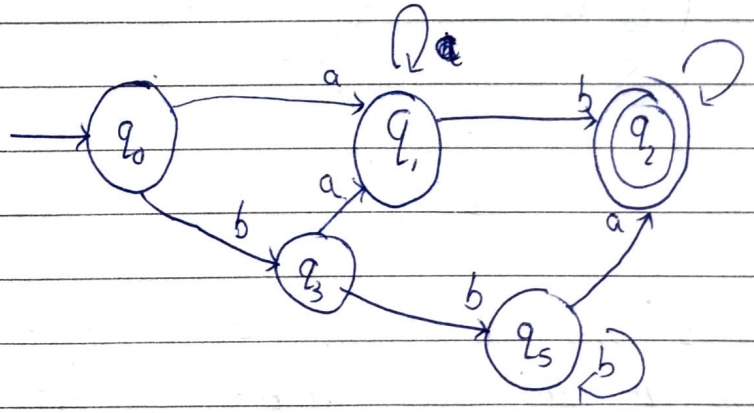


do for null $L = \{\wedge\}$
then do conversion
~~interchange final & no~~
Final state → Non-final
Non final → final.

Q DFA contains either strings ab or bba

$L = \{ab, bba, aaab, bbaab, \overset{abbba}{\ldots}\}$

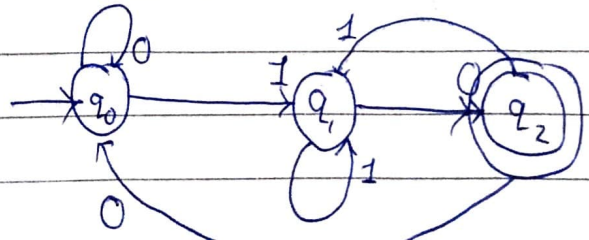
~~RE~~ $(a+b)^* (ab + bba)^+ (a+b)^*$



Made initially with ab & bba & then did connect all the states.

Q Strings end with 10 , $\Sigma = \{0, 1\}$

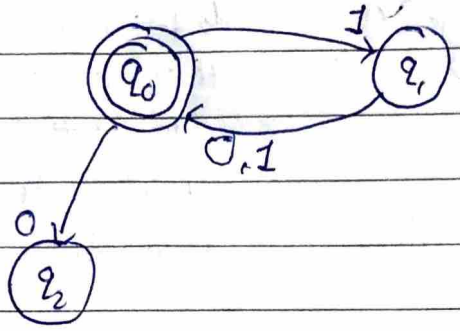
$L = \{10, 010, 1110, \ldots\}$ Test case: 01010101110



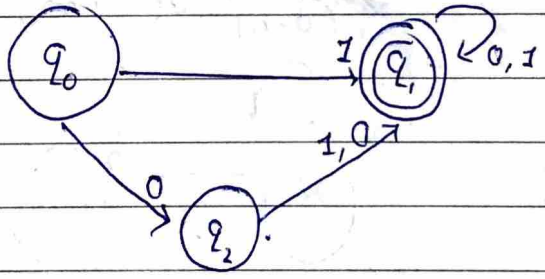
Q DFA

1) $RE = (11+10)^*$
2) $RE = (0+1)^*(1+00)(0+1)^*$

Soln 1) $L = \{\wedge, 11, 10, 1110, 1010, 111111, \ldots\}$



2) $L = \{1, 00, 110, 0000001, \ldots\}$



R.L = A language L over an alphabet Σ is regular iff there is an FA with input alphabet Σ that accepts L.

Eg. $L_1 = \{\wedge\}$, this is R.L.

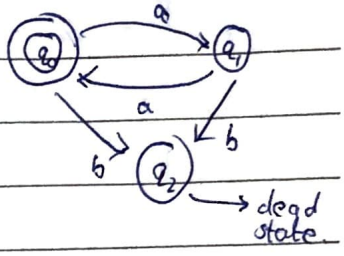
→ If DFA/NFA or RE can be constructed then the language can be known as R.L.

→ Finite Automata does not have any memory

Eg. a^{2n} is this a R.L.

$L = \{aa, aaaa, aaaaaa, \ldots\}$

→ Here we can create DFA so it is R.L.



Eg. $a^n b^n$, Regular language or not.

⇒ Distinguishing one string from another.

Let L be a language in Σ^* & x any string in Σ^*
The set L/x is defined as

$$L/x = \{ z \in \Sigma^* \mid xz \in L \}$$

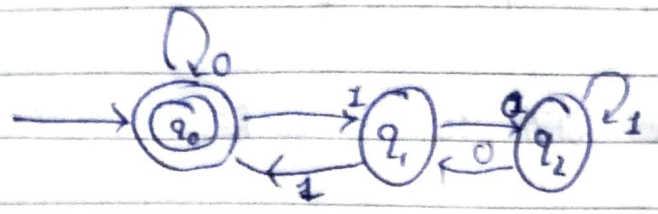
→ Two strings x & y are said to be distinguishable w.r.t. L if $L/x \neq L/y$. Any string Z that is in one of the two sets but not the other is distinguish x & y wrt. L. if $L/x = L/y$, x & y are indistinguishable w.r.t. L.

Eg. DFA whose number is divisible by 3

0000 ⟶ accepted
0011 ⟶ accepted (∵ Decimal value 3)
0110 ⟶ "

Sol^n 3 values of remainder = 0, 1, 2 , L =

→ Construct states = Total number of values of remainders



0 1 1 0
1 1 0
0 1 0 0

0 1 0 1

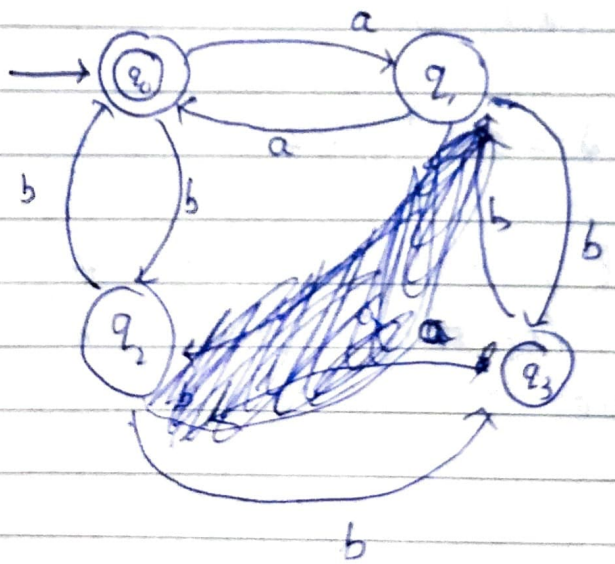
Just see values & put it in state as per values.

Q ~~DF~~ DFA , no. of a's & no. of b's are even

$\Sigma = (a, b)$

$L = \{ \wedge, aa, bb, abab, bababa, \ldots \}$

Sol^n



Just changing the final state we can get all the combinations

1) A odd, B odd
2) A even, B odd
3) A odd, B even