- Chapter : Sampling Methods.

9/9/24/Mon

Formal Language & Automata - 22AIE302

Language

→ Natural Language ⟶ Alphabet ⟶ Word $\xrightarrow{\text{Grammar}}$ Sentences.

→ Formal Language ⟶ Symbols ⟶ String $\xrightarrow{\text{Syntax (G)}}$ Programmes.
(Pgmming Language) (Σ) (W)

Computing (CPU)
Memory (temporary memory)

Automata

Finite (FA) Automata — comp + no memory
Push Down (PDA) Automata — comp + memory is a stack
Turing Machine (TM) — comp + memory = RAM

All the above 3 are different based on the memory. The computer part remains same for all 3.

10/9/24 | Tue

Membership Problem:

- Containment problem.
- Whether a set 'A' has element 'x' in it.
- set of all subsets of set A = power set of A
  if $n \rightarrow$ no: of elements in A
  then $2^n \rightarrow$ no: of elements in P(A)

Formal Language: (L)

- They are the abstract representation of pgmming languages.
- It is basically a set of strings.

Elements in L:

1. Alphabet: (Σ)
- finite, non-empty set of symbols.
- eg: binary alphabet : $\Sigma\{0, 1\}$
  $L = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

  English alphabet : $\Sigma = \{a, b, c, \ldots, z\}$
  $L = \{a, b, \ldots, z, ab, ac, \ldots, \}$

2. Word / String (w):

- It is a collection of symbols or alphabets.
- eg: $\Sigma = \{0, 1\}$
  $w_1 = 0110$,
  $w_2 = 100$, etc.

Properties of strings:

a) Length of a string: – no: of characters in the string $|w|$
   – eg: $w_1 = 0110$; $|w_1| = 4$

b) Reverse of a string : – reverse the string w^R
   – $w_1 = 011$; $w_1^R = 110$

c) Concatenation : – Joining two strings $w_1 \cdot w_2$
   – $w_1 = 01$ $w_2 = 110$ $w_1 \cdot w_2 = 01110$
   $w_2 \cdot w_1 = 11001$

epsilon or lambda

d) Empty/Null String: - string with no character ε (λ)
- $|\varepsilon| = 0$
- $w\varepsilon = \varepsilon w = w$

12/9/24 Thursday.

e) Substring: - w = abcde
- substrings: a, b, c, d, e
ab, bc, cd, de,
abc, bcd, cde
abcd, bcde
abcde
- any portion of a string.

f) Prefix & suffix: - $w = uv$
prefix ← u, v → suffix
- eg: w = abcde

prefix	suffix
a, ab,	e, de,
abc, abcd,	cde, bcde,
abcde	abcde
ε	ε

proper prefix ← (a, ab, abc, abcd) ; (e, de, cde, bcde) → proper suffix

We can add epsilon (empty string) anywhere so it comes as both prefix ~~as well as~~ and suffix

All prefixes or suffixes except the original string is called proper prefix or suffix respectively. This includes ε.

Power of an alphabet: (Σ^k)

eg: binary alphabet : $\Sigma = \{0, 1\}$

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

The power here stands for the length of the string.

ie; if length of string = k, then, Σ^k is the set of all such strings with length k.

★ $\Sigma^0 = \{\varepsilon\}$

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

Star Closure (Σ^*)

- Set of all strings from alphabet Σ including ε.
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$ ($\cup$ → union)

Σ^+

- Set of all strings except epsilon (ε)

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

Complement of a Language:

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

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$

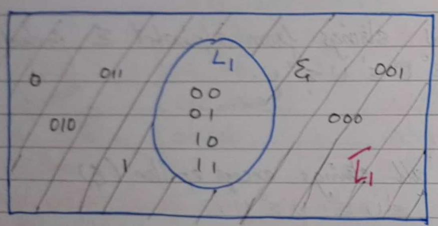
Let L_1 be a language with alphabet having 2 chars only. Then,

$L_1 = \{00, 01, 10, 11\}$

Now $\bar{L}_1$ (L_1 complement) is:

$\bar{L}_1 = \{\varepsilon, 0, 1, 000, 001, 010, 011, 100, \dots\}$

$\bar{L}_1 = \Sigma^* - L_1$



Types of Automata:

	Automata	Architecture	Memory	
1.	Finite Automata FA	$\boxed{\text{computer}}$	No memory	Only solves Membership problems
2.	Push Down Automata PDA	$\boxed{\text{comp}} \leftrightarrow \boxed{\text{stack}}$	Stack	
3.	Turing Machine TM	$\boxed{\text{comp}} \leftrightarrow \boxed{\text{RAM}}$	RAM: → solves Membership problems of some computations. (addition, substraction, mult etc)	

What do membership problems mean?

A machine ~~is~~ represents ~~by~~ a language.

input : w → $\boxed{\text{FA for language } L}$ → output : $w \in L$ / Accept

$w \notin L$ / Reject

The languge / machine checks whether the string belongs to the language and thereby chooses whether or not to accept the string.

18|9|24| Wed.

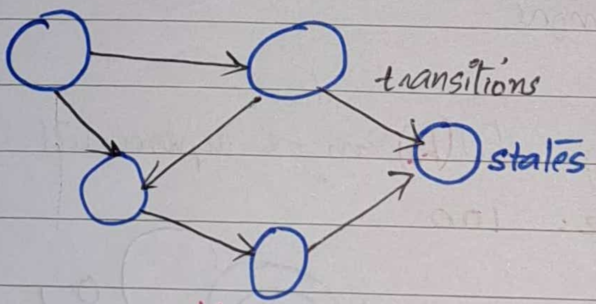
Formal Language & Automata

1. Finite Automata (FA)

- Every finite automata represent a language.

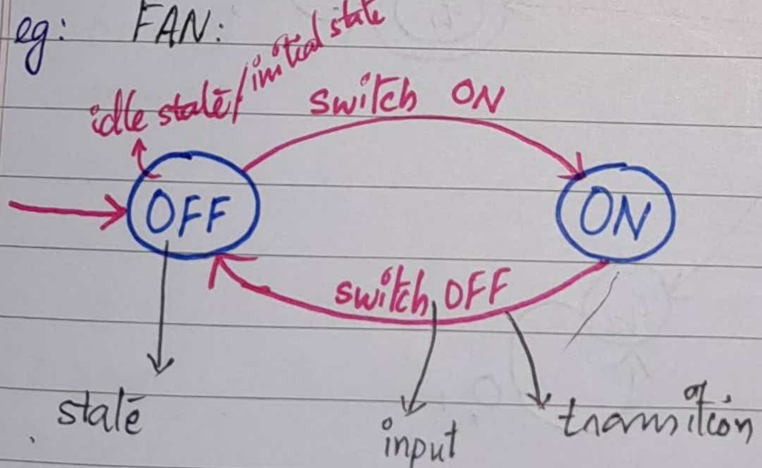
Q. How to represent or design FA?

↳ • We use directed graph to represent FA.
- This representation is called transition diagram.
- eg:

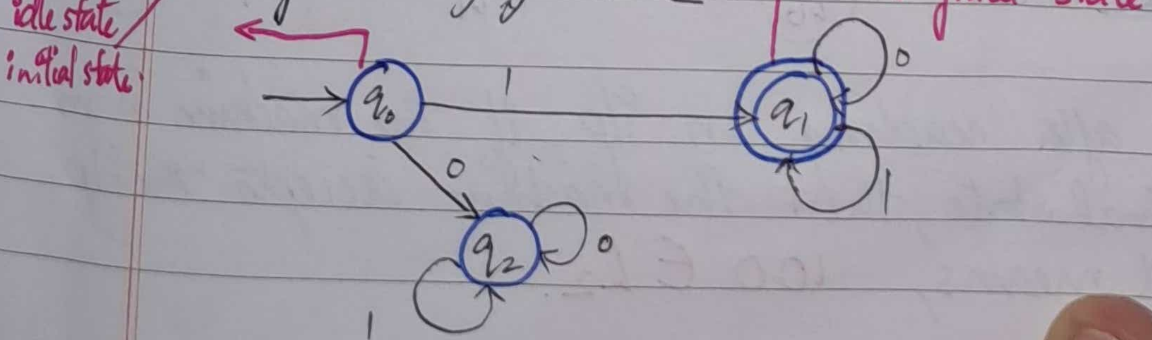


nodes: states
~~arrow~~ edges: transitions.

- eg: FAN:



FA for language L_2

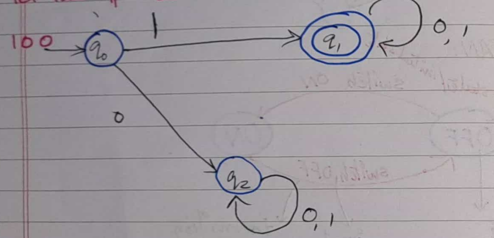


Types of states

1. Initial state
   - single & unique. → q_0
2. Final state
   ~~to last~~ q_n
   - one or more
3. other states.
   - intermediate state q_n
   - 0 or more

The language FA(L_2) can also be represented as

let the i/p be: 100



~~initial~~ state: $100 \rightarrow q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1$

If after reading an i/p, if the machine is in final state, then the machine accepts the i/p. That means, $100 \in L_2$.

i/p = 001

$001 \rightarrow q_0 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_2$ ↳ not final state

So $001 \notin L_2$.

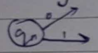
i/p	Accept / Reject
0	R
1	A
00	R
01	R
10	A
11	A
000	R
~~001~~ 101	~~R~~ A
010	R
011	R
100	A
0000	R
0101	R
1010	A
1100	A
1001	A

$L_2 = \{ w \mid w \text{ starts with } 1, \Sigma = \{0,1\} \}$

FA

DFA
Deterministic FA

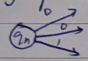
exactly one transition for every i/p symbol.



NFA
Nondeterministic FA

zero or more transitions for every i/p symbol

eg:



formal definition

DFA:

It is defined as a 5 tuple

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

Q - finite set of states

eg: $Q = \{q_0, q_1, q_2, q_3\}$

Σ - finite set of i/p symbols / alphabet

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

δ - transition function which maps

current state $\leftarrow \underbrace{Q \times \Sigma}_{\text{current i/p}} \longrightarrow Q$ next state

q_0 - initial state

F - finite set of final states.

$F \subseteq Q$

eg.. $\delta(q_0, 1) = q_1$ (for the L_2 FA)

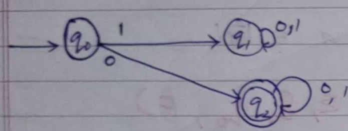
$\delta(q_1, 1) = q_1$

$\delta(q_0, 0) = q_2$

23|09|24| Mon.

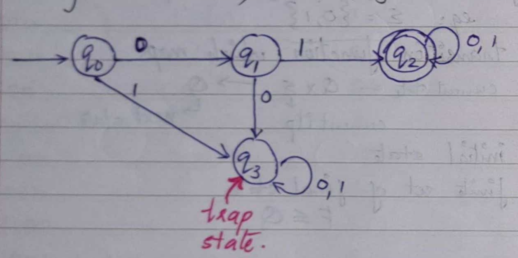
Q. Design a DFA that accepts all strings starting with zero, $\Sigma = \{0, 1\}$?

A:



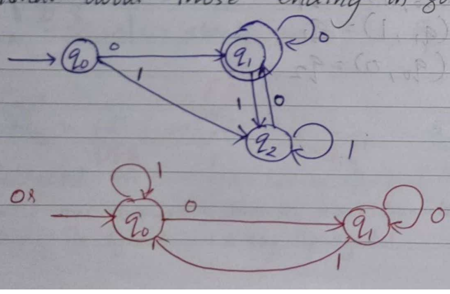
Q. Design a DFA that accepts all strings starting with '01', $\Sigma = \{0, 1\}$.

A:



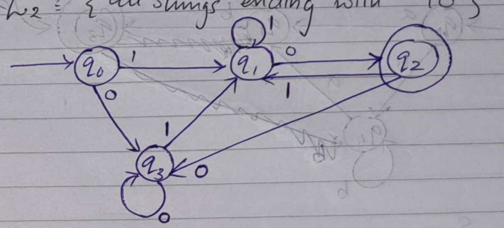
Q. What about those ending in zero?

A:



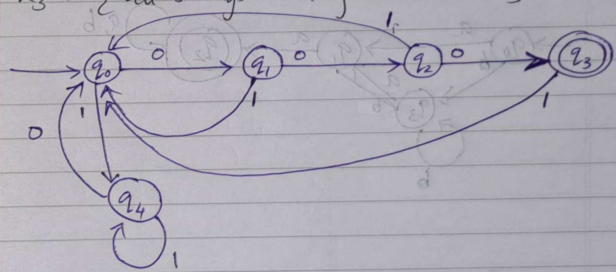
Q. $L_2 = \{$ all strings ending with '10' $\}$

A:



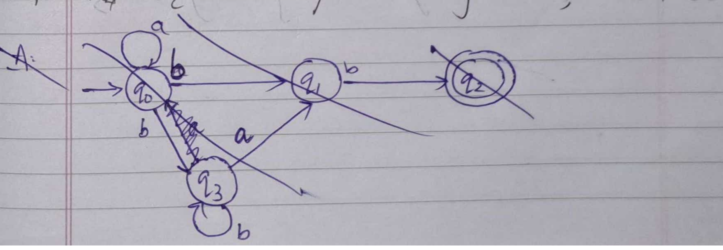
Q. $L_3 = \{$ all strings ending with '000' $\}$

A:

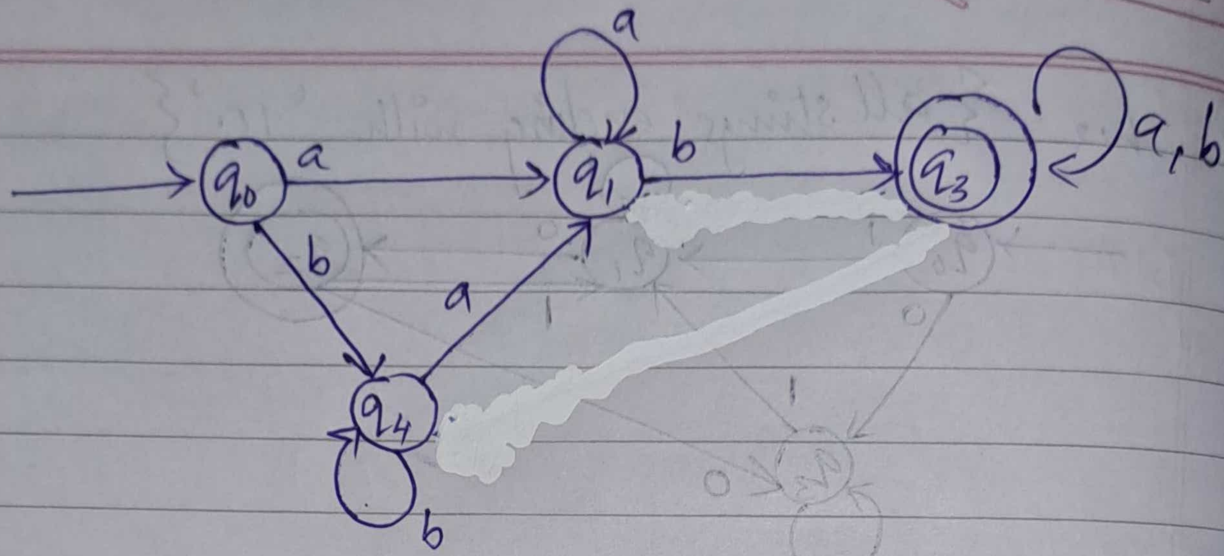


Q. $L_4 = \{$ all strings containing 'ab', $\Sigma = \{a, b\}\}$

~~A:~~

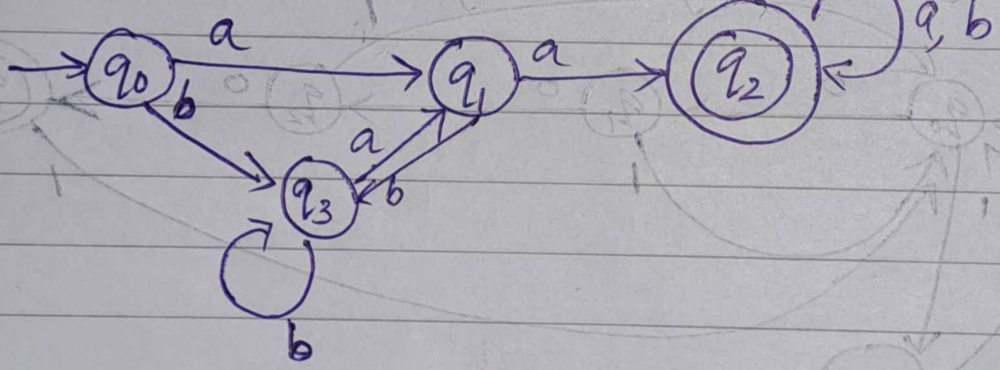


A:



Q. L_5 = {all strings containing atleast 2 a's}.

A:



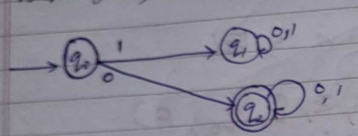
Q. L_6 = { all strings containing atmost 2 a's}

Q. L_7 = { all strings containing exactly 2 a's}

23/09/24/Mon.

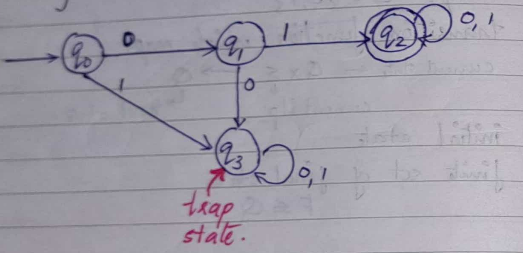
Q. Design a DFA that accepts all strings starting with zero, $\Sigma = \{0, 1\}$?

A:



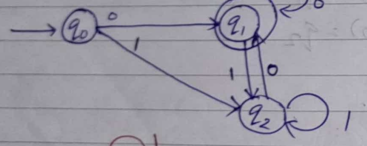
Q. Design a DFA that accepts all strings starting with '01', $\Sigma = \{0, 1\}$.

A:

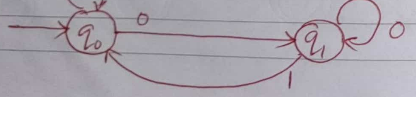


Q. What about those ending in zero?

A:

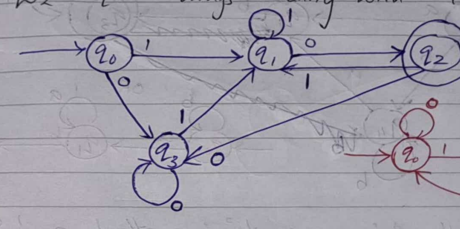


OR

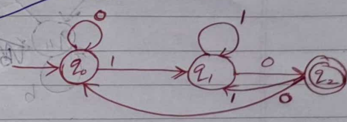


Q: L_2 = {all strings ending with '10'}

A:

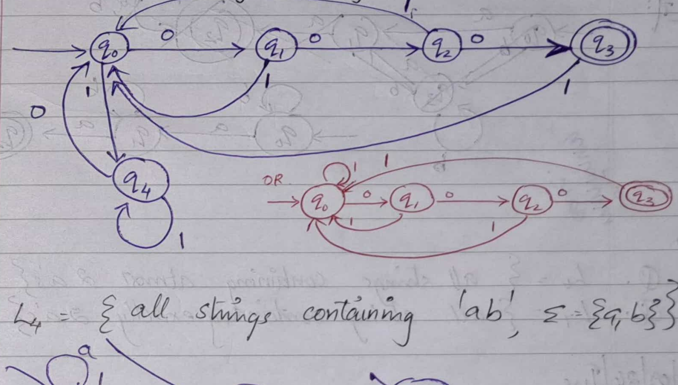


OR



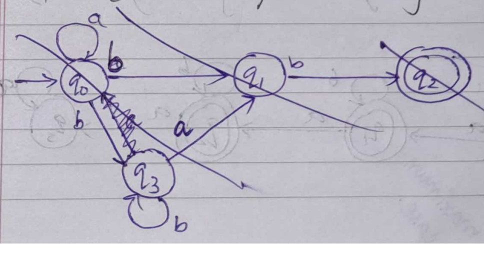
Q. L_3 = {all strings ending with '000'}

A:

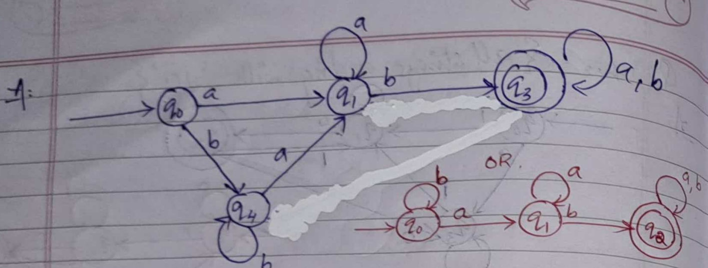


Q. L_4 = {all strings containing 'ab', $\Sigma = \{a, b\}$}

A:

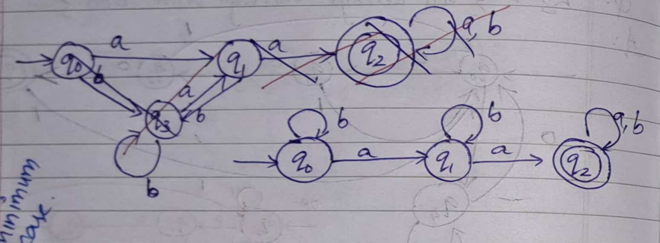


A:



Q. L_5 = {all strings containing atleast 2 a's}.

A:



minimum case

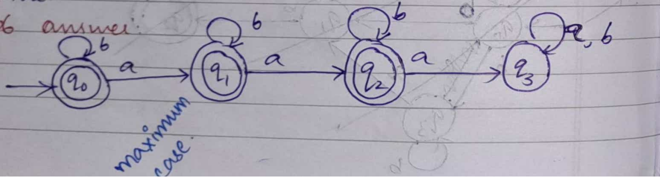
~~24/09~~

Q. L_6 = { all strings containing atmost 2 a's}

Q. L_7 = { all strings containing exactly 2 a's}

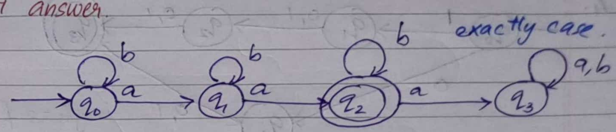
24/09/24 | Tue.

L6 answer:



maximum case

L7 answer

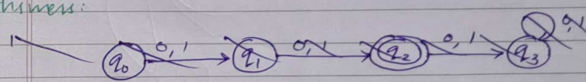


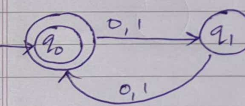
exactly case.

Q. Design DFA for the following:

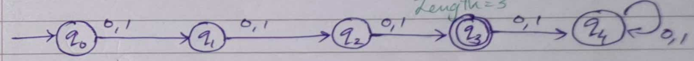
1. L_1 = { all strings of even length } $\Sigma = \{0,1\}$
2. L_2 = { all strings with length == 3 } $\Sigma = \{0,1\}$
3. L_3 = { all strings with length: the multiple of 3 } $\Sigma = \{0,1\}$
4. L_4 = { all strings with length 3 and starts with 1 } $\Sigma = \{0,1\}$

Answers:

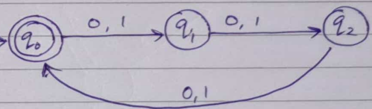


1. 

multiple of 2

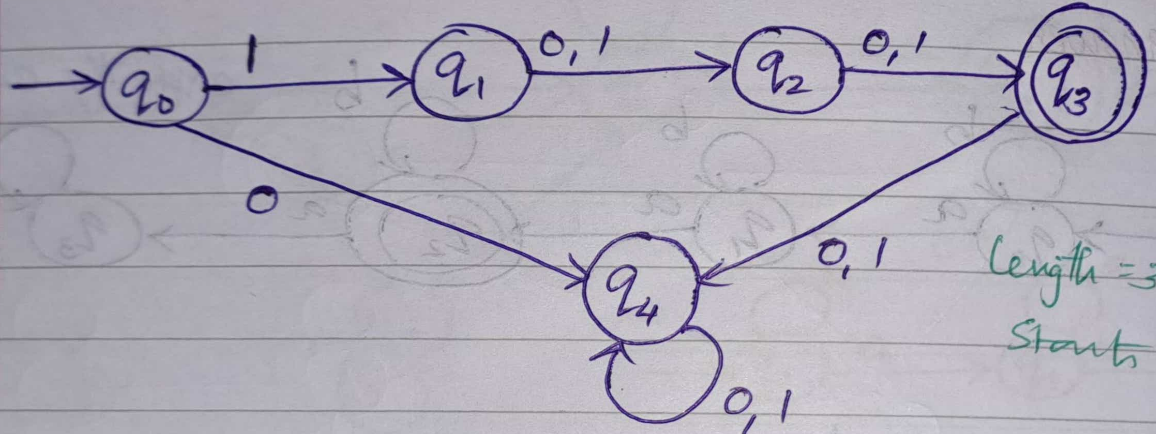
2. 

Length = 3

3. 

Multiple of 3

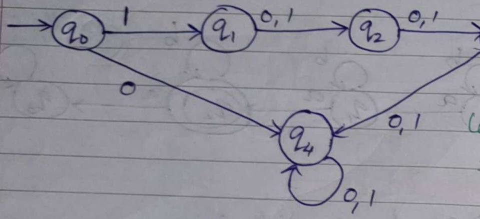
4.



Length = 3 &

Starts with 1

4.



Length = 3 & Starts with 1

03/10/24 | Thursday.

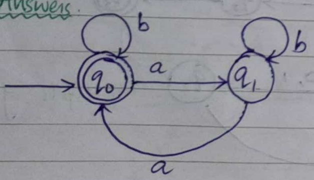
Tutorial - 1 :

Q. Design DFA for the following languages

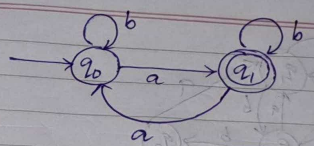
1. L_1 = {all strings with even no: of a's}, $\Sigma = \{a, b\}$
2. L_2 = {all strings with odd no: of a's}, $\Sigma = \{a, b\}$
3. L_3 = {all strings with even no: of a's & odd no: of b's}, $\Sigma = \{a, b\}$
4. ~~L_4 = {all strings with~~
4. L_4 = {all binary strings divisible by 4}, $\Sigma = \{0, 1\}$
5. L_5 = {all decimal no:s divisible by 5}, $\Sigma = \{0, 1, 2, ..., 9\}$
6. $L_6 = \{ n_a(w) \% 3 = 0$ and $n_b(w) \% 2 = 0 \}$, $\Sigma = \{a, b\}$
7. L_7 = {all binary strings not divisible by 4}, $\Sigma = \{0, 1\}$.

Answers.

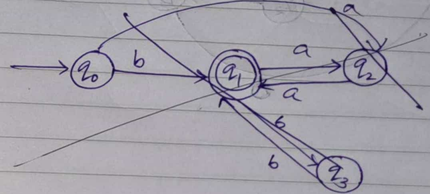
1.



2.

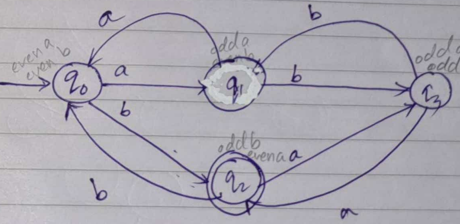


~~3.~~



~~4~~

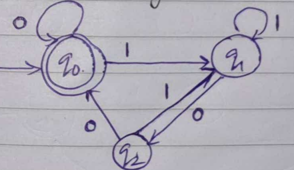
3.



0	0
1	1
10	2
11	3
100	0
101	1
110	2
111	3
1000	0
1100	0

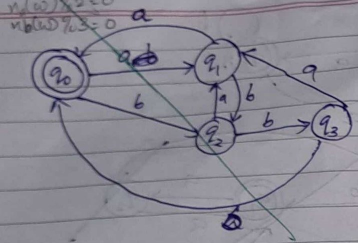
10101100

4. Last two digits have to be zero.

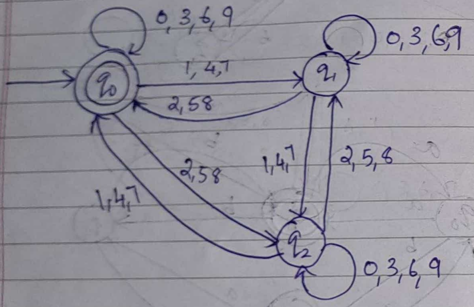


6.

$n_a(w) \% 2 = 0$
$n_b(w) \% 3 = 0$



5.

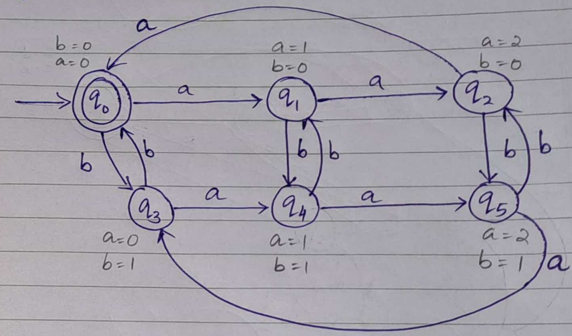


$$\begin{array}{r} 1,4,7 \\ +\ 1,4,7 \\ \hline 2,8,14 \\ 1,4,7 \\ \hline 3,12,21 \end{array}$$

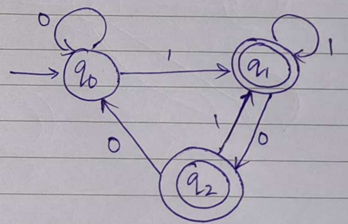
3 groups : $\begin{cases} 0, 3, 6, 9 \\ 1, 4, 7 \\ 2, 5, 8 \end{cases}$

~~6.~~

6. $n_a(w) \% 3 = 0$ States : a=0 b=0 ; a=1 b=0 ; a=2 b=0
$n_b(w) \% 2 = 0$ a=0 b=1 ; a=1 b=1 ; a=2 b=1



7.



7/10/24 Monday.

Non-deterministic finite Automata:

Definition: NFA is defined as a 5-tuple

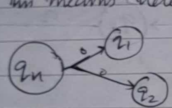
$M = (Q, \Sigma, \delta, q_0, F)$

Q: ~~finite~~ finite set of states

Σ: finite set of alphabet / i/p symbol.

δ: it maps $Q \times \Sigma \rightarrow 2^Q$ set of subsets.

This means there can be multiple transitions

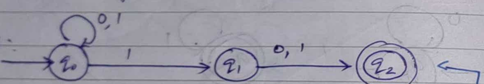


$\delta(q_n, 0) = \{q_1, q_2\}$

q_0: initial state

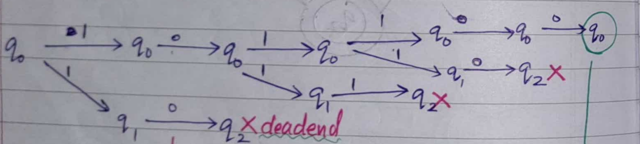
F: final state.

eg.



How to check if a string is accepted in NFA?

$w = 101100$



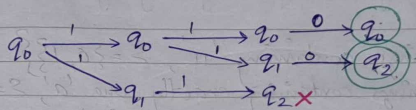
deadend

because no more transitions possible at q_2

final possible state.

This is not final state and so this word ~~does~~ is not accepted. ~~belong to this aut~~

$w = 110$.

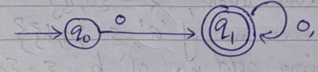


This is a final state so it is accepted.

Design NFA for the following languages.

1. $L_1 = \{$ all strings start with 0, $\Sigma = \{0, 1\}\}$

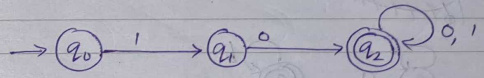
A:



Here we need not bother with transitions that we do not prefer (such as what to do if 1 came as first digit)

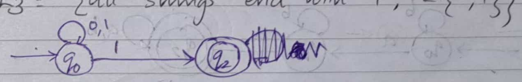
2. $L_2 = \{$ all strings starts with '10', $\Sigma = \{0, 1\}\}$

A:



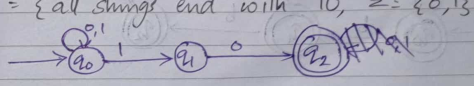
3. $L_3 = \{$ all strings end with 1, $\Sigma = \{0, 1\}\}$

A:



4. $L_4 = \{$ all strings end with '10', $\Sigma = \{0, 1\}\}$

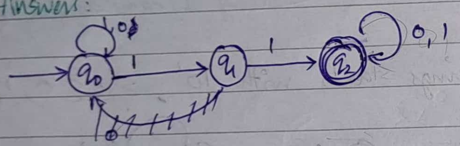
A:

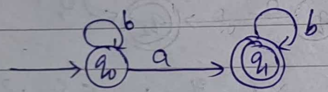


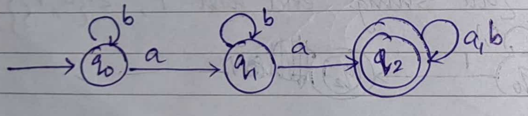
Design NFA for the following languages:

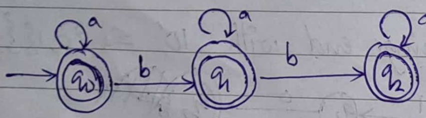
1. L_1 = {all strings containing '11', Σ = {0, 1}}
2. L_2 = {all strings containing exactly one 'a', Σ = {a,b}}
3. L_3 = {all strings containing atleast 2 'a', Σ = {a,b}}
4. L_4 = {all strings containing atmost two 'b', Σ = {a,b}}
5. L_5 = {all strings with length multiple of three, Σ = {0,1}}.
6. L_6 = {all strings with even length, Σ = {0,1}}.
7. L_7 = {all strings with even no: of 'a', Σ = {a,b}}
8. L_8 = {all strings containing odd no: of 'b', Σ = {a,b}}.

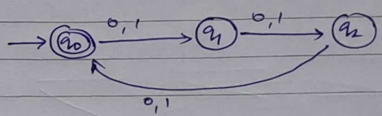
Answers:

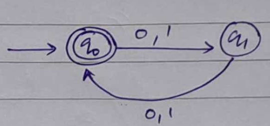
1. 

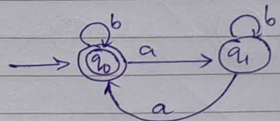
2. 

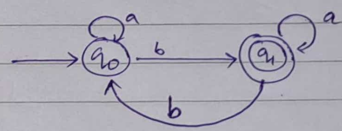
3. 

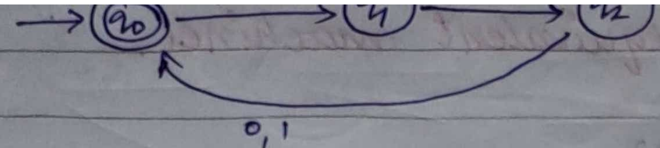
4. 

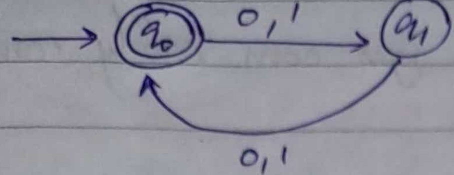
5. 

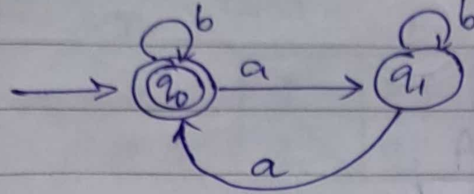
6. 

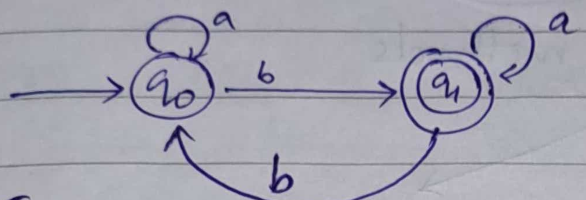
7. 

8. 



6. 

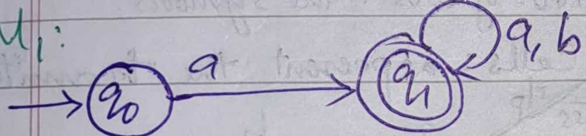
7. 

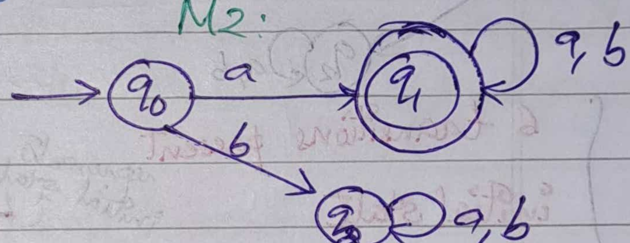
8. 

8/10/24/Tue.

Equivalence of NFA & DFA:

★ DFA is a special case of NFA.

M_1: 

M_2: 

M_1 : NFA M_2 : DFA

both M_1 & M_2 accepts all strings starting with 'a'.

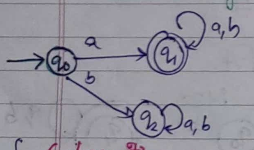
When two machines accept the same language, they are called equivalent machines.

ie, $M_1 \equiv M_2$
(NFA) (DFA)

or in other words,
if you have an NFA, you can easily convert it to a DFA.

FA
→ DFA
→ NFA

representation methods
→ Transition diagram
→ Transition table



Transition table
No: of rows: the no: of states present
No: of cols: the symbols.
Cells: represent the transition.

6 transitions present

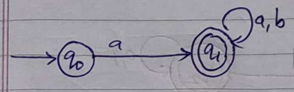
Info: we get from the diagram:
- initial state
- final state
- the other states
- i/p symbols

states \ i/p	a	b
→ q_0	q_1	q_2
* q_1	q_1	q_1
q_2	q_2	q_2

represents initial state
represents final state

→ : initial state
* : final state.

For NFA.



states \ input	a	b
→ q_0	q_1	–
* q_1	q_1	q_1

or.

$\{q_1\}$	$\{\}$
$\{q_1\}$	$\{q_1\}$

we can also write in set representation

Q. How to convert NFA to DFA?

Step 1: Draw the transition table for NFA.

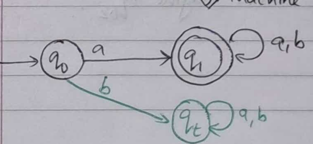
states \ input	a	b
→ q_0	q_1	$\{\}$
* q_1	q_1	q_1

Step 2: Write the transition table for DFA required.
if blank: introduce a trap state
AND always start from initial state.

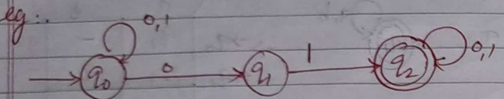
states \ input	a	b
→ q_0	q_1	q_t
* q_1	q_1	q_1
q_t	q_t	q_t

Add the trap state onto the transition & make itself the transition for every i/p

⇓ Machine form (Diagram)



eg:



Convert to DFA:

A: Step 1:

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{\}$	$\{q_2\}$
$* q_2$	$\{q_2\}$	$\{q_2\}$

Step2:

	0	1
$\rightarrow q_0$	q_{01}	q_0
q_{01}	q_{01}	q_{02}
$* q_2$	/ / /	/ / /

There are two transitions. So we write a new state combining the both. Now we go to the new state

	0	1
q_0	q_0, q_1	q_0
q_1	–	q_2
	$\{q_0, q_1\}$	$\{q_0, q_2\}$
	q_{01}	q_{02}

Take union.

Now define q_{02}

	0	1
$\rightarrow q_0$	q_{01}	q_0
q_{01}	q_{01}	q_{02}
$* q_{02}$	q_{012}	q_{02}
$* q_{012}$	q_{012}	q_{02}

	0	1
q_0	$\{q_0\, q_1\}$	$\{q_0\}$
q_2	$\{q_2\}$	$\{q_2\}$
	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
	q_{012}	q_{02}

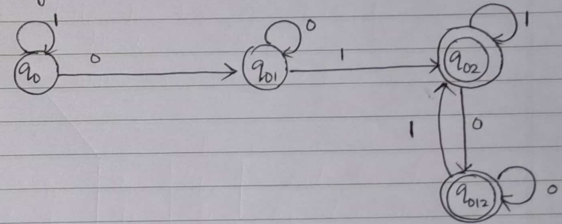
	0	1
q_0	$q_0\, q_1$	q_0
q_1	–	q_2
q_2	q_2	q_2
	q_{012}	q_{02}

All states have been defined.

Now which is the final state?

Originally q_2 is the final state. So all states with q_2 is the final state.

The diagram will be:

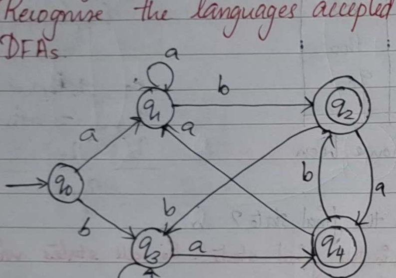


10/10/24 | Thu.

Tutorial - 2 (DFA)

1. Recognise the languages accepted by the following DFAs.

a)

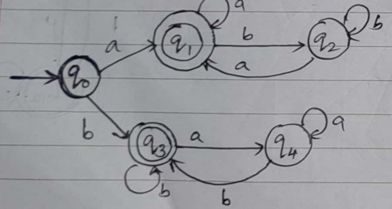


A: {All strings ending with "ab" or "ba"}

or

$L_1 = \{ w \mid w$ ends with "ab" or "ba", $\Sigma = \{a,b\} \}$

b)

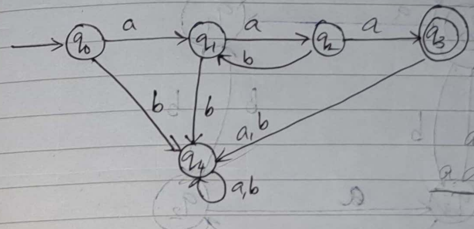


A:

$L_2 = \{ w \mid w$ ends with the starting letter, $\Sigma = \{a,b\} \}$

aa bababaa

c)



aaa

aabaa

aa(ba)a

- aaa together
- starts with aa. ends with exactly one baa
- no 2 bs come together

So $aa(ba)^n a$; $n \geq 0$

aaa $\quad aa(ba)^n a$

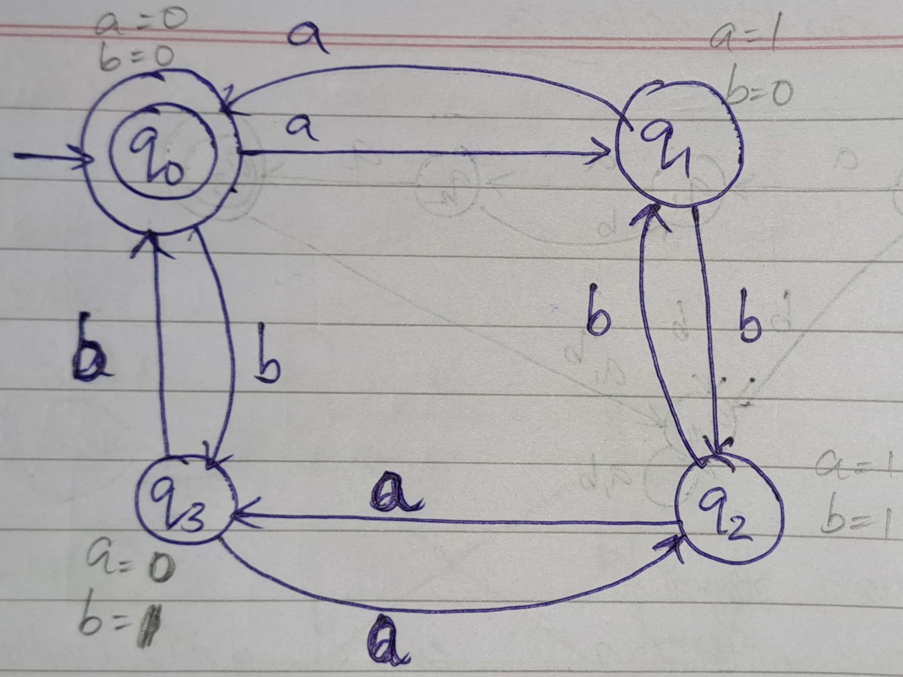
A: ~~$L_3 = \{ w \mid w$ is either 'aaa' or of the form $aa(ba)^n a$, $n \geq 0$ $n \in W$, $\Sigma = \{a,b\} \}$~~

A: $L_3 = \{ w \mid w$ is of the form $aa(ba)^n a$, $n \geq 0$ $n \in W$; $\Sigma = \{a,b\} \}$

II. Design DFA to recognise strings in the following languages.

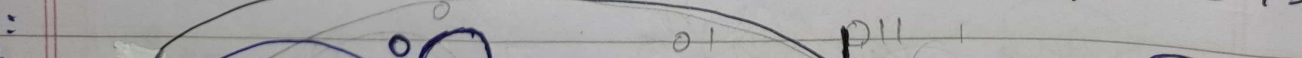
a) $L_1 = \{ w \mid w$ contains even no. of a's and even no. of b's, $\Sigma = (a,b) \}$

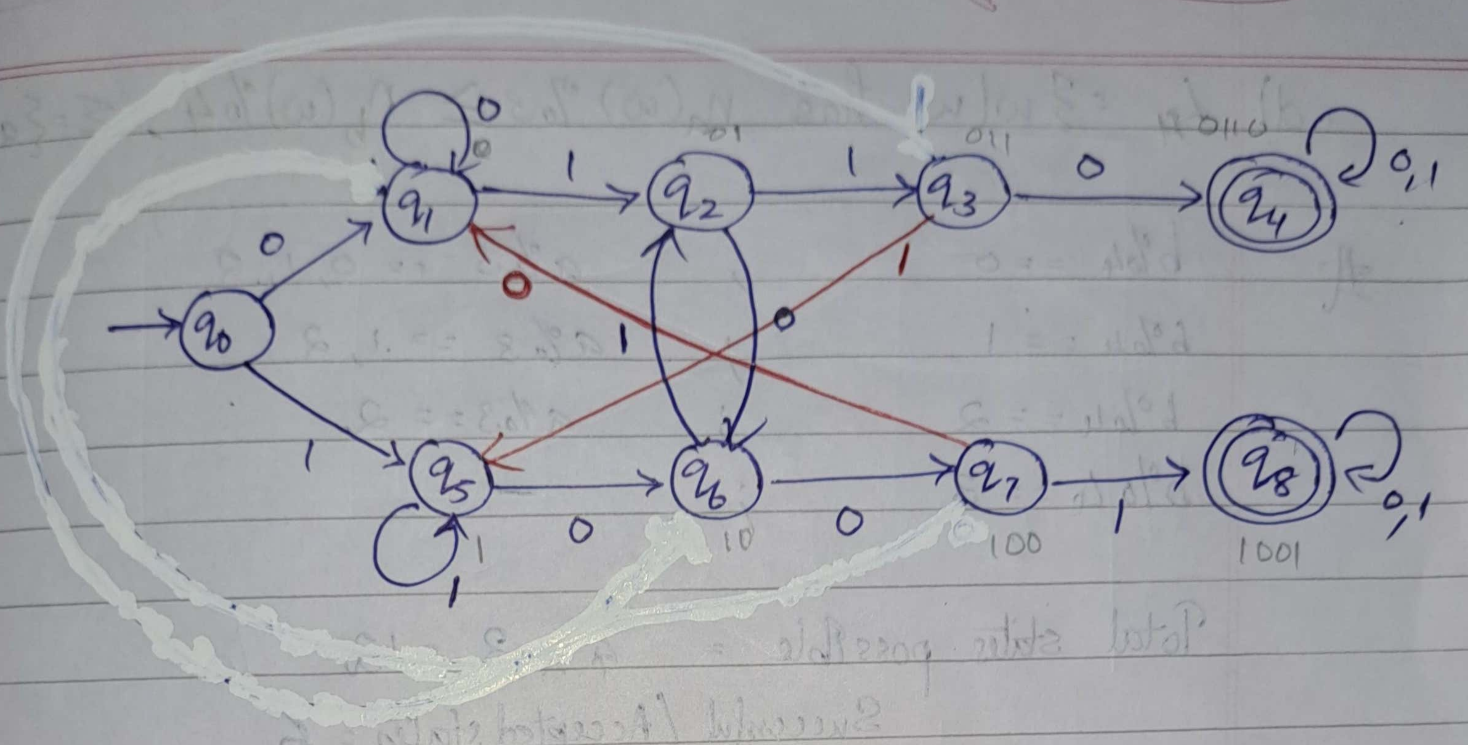
A:



b) $L_2 = \{ w \mid w$ contains 0110 or 1001, $\Sigma = \{0,1\}\}$

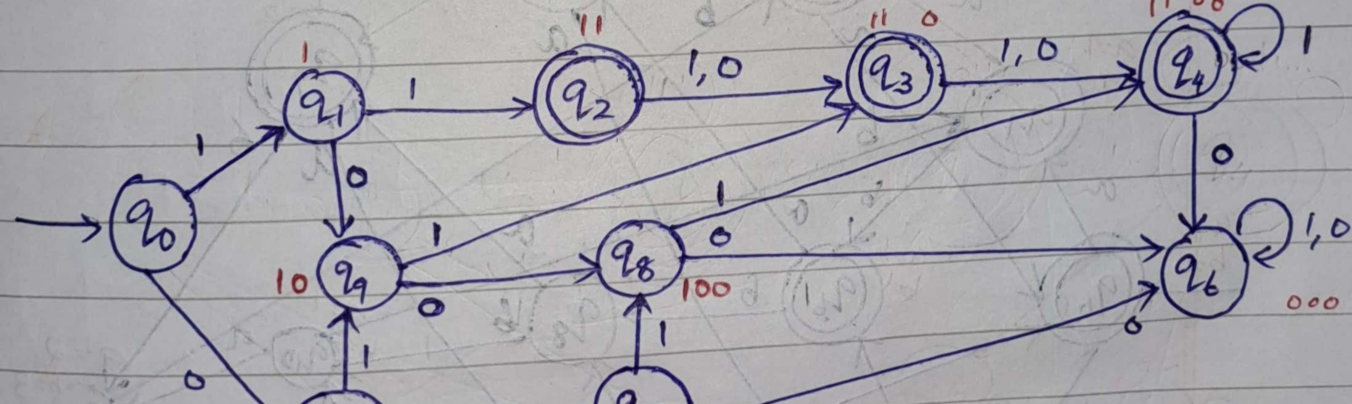
A:





c) $L_3 = \{w | w$ is a binary string with atleast two ones and almost two zeroes, $\Sigma = \{0,1\}\}$

A:



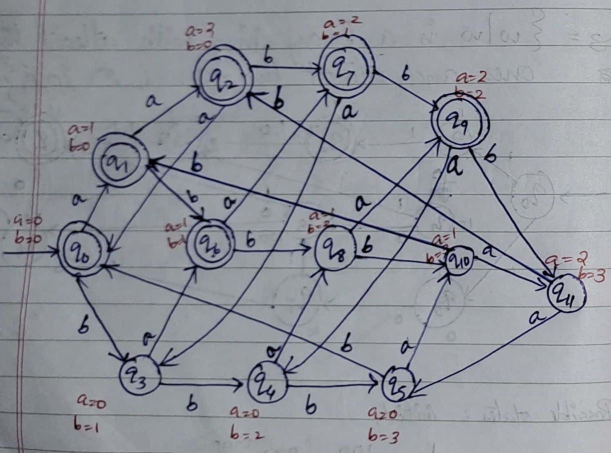
Possible states : initial,

1	100	000 → trap.
0	00	
10	110	
11	1100	

d) $L_4 = \{ w \mid w$ has $n_a(w) \% 3 \geq n_b(w) \% 4$, $\Sigma = \{a, b\}\}$

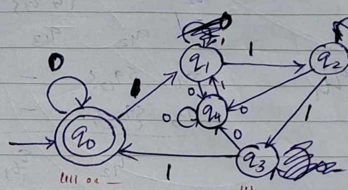
A: b%4 == 0 ; a%3 == 0, 1, 2
b%4 == 1 ; a%3 == 1, 2
b%4 == 2 ; a%3 == 2
b%4 == 3 ; –

Total states possible = 4 × 3 = 12 etc.
Succesful / Accepted states = 6



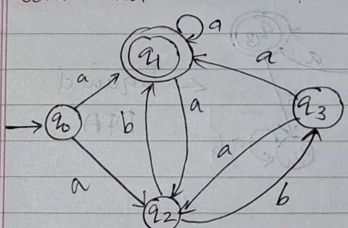
e) $L_5 = \{ w \mid$ No: of consecutive 1's in w is 0 or multiple of 4, $\Sigma = \{0, 1\}\}$.

A:



III. Convert the following NFA to DFA using Subset Construction method.

a)



A:

	a	b
→ q_0	$\{q_1, q_2\}$	–
* q_1	$\{q_1, q_2\}$	–
q_2	–	$\{q_1, q_3\}$
q_3	$\{q_1, q_2\}$	–

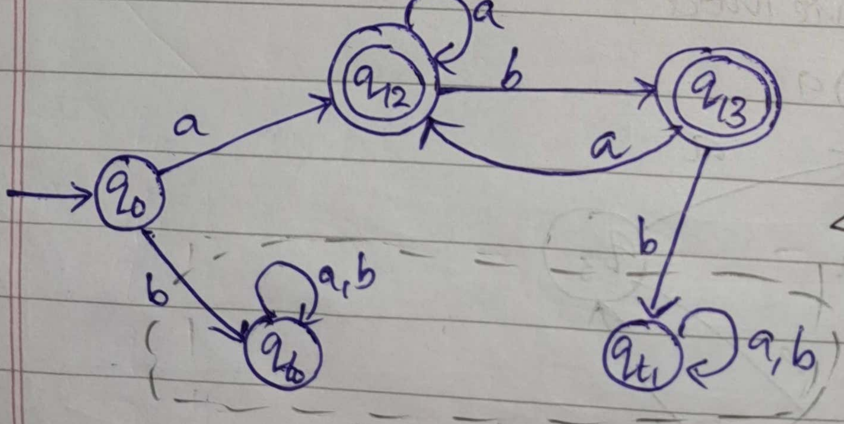
→ transition table of NFA given.

	a	b
$\rightarrow q_0$	q_{12}	q_{t_0}
$* q_{12}$	q_{12}	q_{13}
q_{t_0}	q_{t_0}	q_{t_0}
$* q_{13}$	q_{12}	q_{t_1}
q_{t_1}	q_{t_1}	q_{t_1}

q_1	$\{q_1, q_2\}$	$-$
q_2	$-$	$\{q_1, q_3\}$
	q_{12}	q_{13}

q_1	$\{q_2\}$	$-$
q_3	$\{q_2\}$	$-$
	q_{12}	q_{t_1}

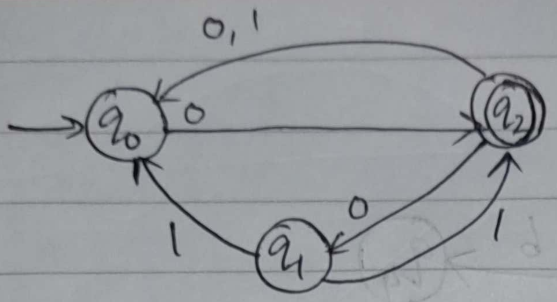
$\downarrow$ diagram.



$\leftarrow$ Required DFA.

Can be written as single trap state.

b)



A: Transition table of NFA:

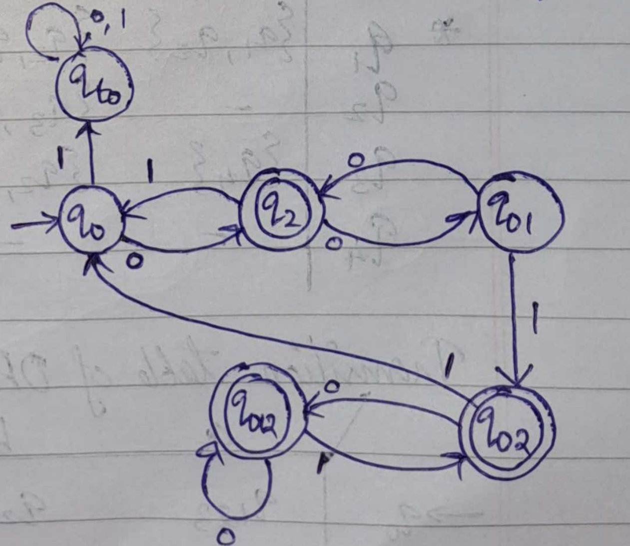
	0	1
$\rightarrow q_0$	$\{q_2\}$	–
q_1	–	$\{q_0, q_2\}$
$* q_2$	$\{q_0, q_1\}$	$\{q_0\}$

DFA transition table:

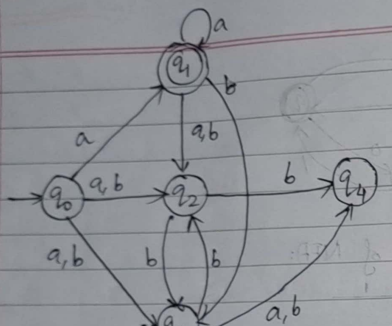
	0	1
$\rightarrow q_0$	q_2	q_{t0}
q_{t0}	q_{t0}	q_{t0}
$* q_2$	q_{01}	q_0
q_{01}	q_2	q_{02}
$* q_{02}$	q_{012}	q_0
$* q_{012}$	q_{012}	q_{02}

$\Rightarrow$

DFA transition diagram.



c)



A: Transition table of NFA:

	a	b
$\rightarrow q_0$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$
$* q_1$	$\{q_1, q_2\}$	$\{q_2, q_3\}$
q_2	–	$\{q_3, q_4\}$
q_3	$\{q_4\}$	$\{q_2, q_4\}$
q_4	–	–

Transition table of DFA:

	a	b
$\rightarrow q_0$	q_{123}	q_{23}
$* q_{123}$	q_{124}	q_{234}
q_{23}	q_4	q_{234}
$* q_{124}$	q_{12}	q_{234}
q_{234}	q_4	q_{234}
q_4	q_{t0}	q_{t0}
$* q_{12}$	q_{12}	q_{234}

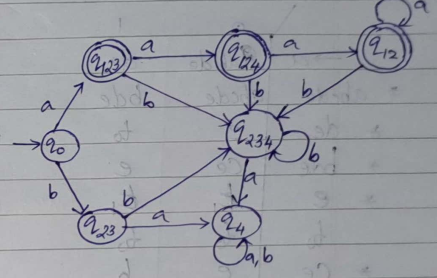
	a	b
q_{t0}	q_{t0}	q_{t0}

Table:

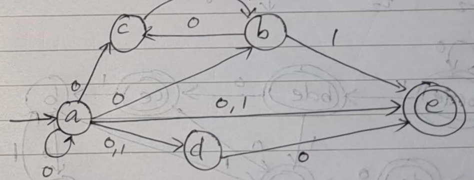
	a	b
$\rightarrow q_0$	q_{123}	q_{23}
$* q_{123}$	q_{124}	q_{234}
q_{23}	q_4	q_{234}
$* q_{124}$	q_{12}	q_{234}
q_{234}	q_4	q_{234}
q_4	q_{t0}	q_{t0}
$* q_{12}$	q_{12}	q_{234}
q_{t0}	q_{t0}	q_{t0}.

written as single trap state

$\Rightarrow$



d)



A: Transition table for NFA:

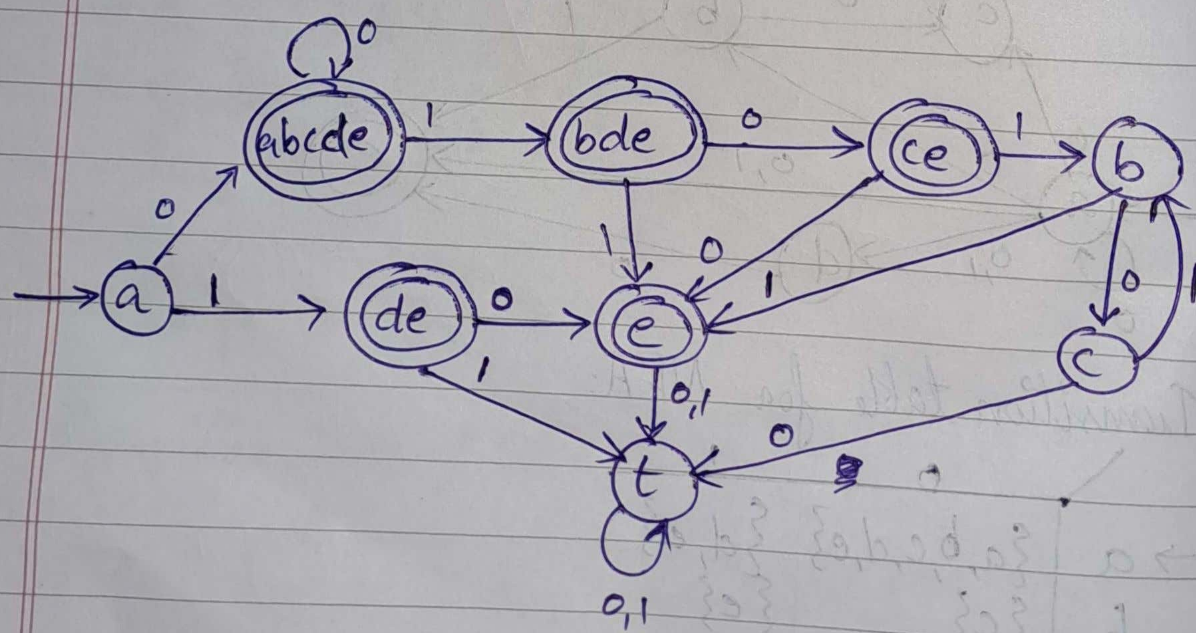
	0	1
$\rightarrow a$	$\{a, b, c, d, e\}$	$\{d, e\}$
b	$\{c\}$	$\{e\}$
c	–	$\{b\}$
d	$\{e\}$	–
$* e$	–	–

Transition table DFA:

	0	1
→ a	abcde	de
* abcde	abcde	bde
* de	e	t_0
* bde	ce	e
* e	t_1	t_1
t_0	t_0	t_0
* ce	e	b
b	c	e
c	t_2	b
t_2	t_2	t_2

All trap states made into One
that is:: $t_0, t_1, t_2 = t$.

Diagram.

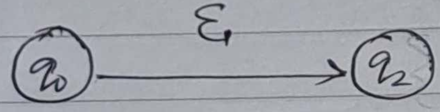


Gayathri B Nair
AM.EN.U4AIE22117

14/10/24 | Monday.

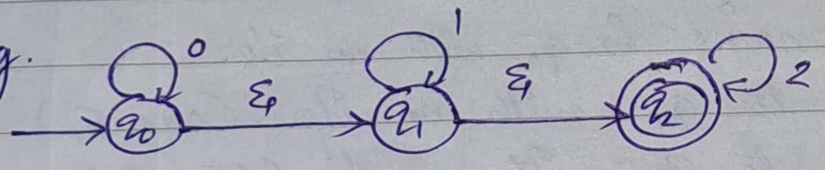
ε - NFA

- It is the NFA with ε transitions — ε → empty string.
- Not possible in DFA.
- without i/p transition takes place.



- ε transition.

eg.



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

1. $w = 0$

   $w = 0 = = w = 0\,\varepsilon\,\varepsilon$ → so accepted

   So 0 is accepted.

2. $w = 00$ → $00\,\varepsilon\,\varepsilon$ → accepted.
3. $w = 1$ → $\varepsilon\,1\,\varepsilon$ → accepted.
4. $w = 2$ → $\varepsilon\,\varepsilon\,2$ → accepted.

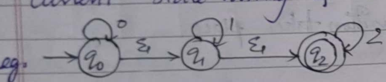
ie; $L_1 = \{0, 00, 000, \ldots, 1, 11, 111, \ldots, 2, 22, 222, \ldots,$
$01, 02, 12, 0011, 001122, \ldots\}$

$L_1 = \{w \mid w$ is of the form $0^i 1^j 2^k$; $i, j, k \geq 0$;
$\Sigma = \{0, 1, 2\}\}$

ϵ-closure of a state (ϵ^*) ϵ-NFA $\longrightarrow$ NFA $\longrightarrow$ DFA

The set of states that are reachable from the current state using ϵ-transitions

eg.



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

$\epsilon^*(q_0) = \{q_0, q_1, q_2\}$

epsilon closure of q_0.

q_0 remains in q_0 with no i/p.
q_0 goes to q_1 with no i/p (one ϵ)
q_0 goes to q_2 with no i/p (two ϵ).

$\epsilon^*(q_1) = \{q_1, q_2\}$

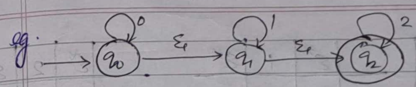
$\epsilon(q_2) = \{q_2\}$

Convert ϵ-NFA to NFA:

Step 1. Find ϵ^* of all states in ϵ-NFA.
Step 2. Write the transition table of ϵ-NFA
Step 3. Find the transition of NFA using the following tabular method

States	ϵ^*	i/p	ϵ^*

eg.



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

Step 1:

$\epsilon^*(q_0) = \{q_0, q_1, q_2\}$
$\epsilon^*(q_1) = \{q_1, q_2\}$
$\epsilon^*(q_2) = \{q_2\}$

Step 2. transition table

	0	1	2	ϵ
$\rightarrow q_0$	$\{q_0\}$	-	-	$\{q_1, q_2\}$
q_1	-	$\{q_1\}$	-	$\{q_2\}$
$* q_2$	-	-	$\{q_2\}$	-

Step 3. transition for i/p 0 — ϵ closure of those states

	ϵ^*	0	ϵ^*
q_0	q_0	q_0	q_0
	q_1	-	q_1
	q_2	-	q_2

	ϵ^*	1	ϵ^*
q_0	q_0	-	
	q_1	q_1	q_1
	q_2	-	q_2

	ϵ^*	2	ϵ^*
q_0	q_0	-	-
	q_1	-	-
	q_2	q_2	q_2

This means that for our NFA, for i/p 0 @ q_0, the transitions are the last column.

ie; $q_0 \xrightarrow{0} q_0$, $q_0 \xrightarrow{0} q_1$, $q_0 \xrightarrow{0} q_2$

NFA	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
⋮			

for q_1:

	ε^*	0	ε^*
q_1	q_1	-	-
	q_2	-	-

	ε^*	1	ε^*
q_1	q_1	q_1	q_1
	q_2	-	q_2

	ε^*	2	ε^*
q_1	q_1	~~q_2~~ -	
	q_2	q_2	q_2

for q_2

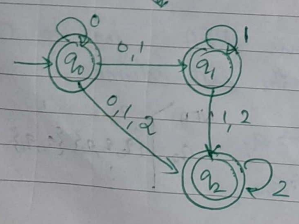
	ε^*	0	ε^*
q_2	q_2	-	-

	ε^*	1	ε^*
q_2	q_2	-	-

	ε^*	2	ε^*
q_2	q_2	q_2	q_2

Thus NFA becomes:

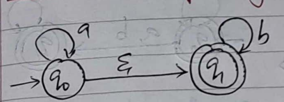
	0	1	2
$\rightarrow * q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$* q_1$	-	$\{q_1, q_2\}$	$\{q_2\}$
$* q_2$	-	-	$\{q_2\}$



q_0 & q_1 are also final states because the ε^* of q_0 & q_1 has q_2 (the original final state) in it.

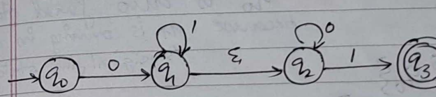
Convert the following ε-NFA to NFA:

1.



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

2.



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

Answers:

1. $\varepsilon^*(q_0) = \{q_0, q_1\}$

$\varepsilon^*(q_1) = \{q_1\}$

	a	b	ε
$\rightarrow q_0$	$\{q_0\}$	-	$\{q_1\}$
$* q_1$	-	$\{q_1\}$	-

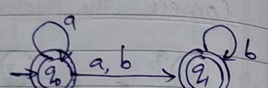
	ε^*	a	ε^*
q_0	q_0	q_0	q_0
	q_1	-	q_1
q_1	q_1	-	-

	ε^*	b	ε^*
q_0	q_0	-	-
	q_1	q_1	q_1
q_1	q_1	q_1	q_1

NFA transition table:

	a	b
$\rightarrow * q_0$	$\{q_0, q_1\}$	$\{q_1\}$
$* q_1$	–	$\{q_1\}$

$\Rightarrow$



q_0 is also final state
because q_1 is coming in ε^* of q_0
→ original final state.

2. $\varepsilon^*(q_0) = \{q_0\}$
$\varepsilon^*(q_1) = \{q_1, q_2\}$
$\varepsilon^*(q_2) = \{q_2\}$
$\varepsilon^*(q_3) = \{q_3\}$

	0	1	ε
$\rightarrow q_0$	$\{q_1\}$	–	–
q_1	–	$\{q_1\}$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_3\}$	–
$* q_3$	–	–	–

	ε^*	0	ε^*
q_0	q_0	q_1	q_1, q_2
q_1	q_1, q_2	–, q_2	–, q_2

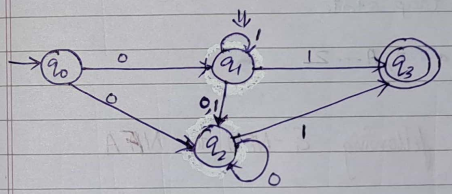
	ε^*	1	ε^*
q_0	q_0	–	–
q_1	q_1, q_2	q_1, q_3	q_1, q_2, q_3

	ε^*	0	ε^*
q_2	q_2	q_2	q_2
q_3	q_3	–	–

	ε^*	1	ε^*
q_2	q_2	q_3	q_3
q_3	q_3	–	–

Thus NFA transition table is:

	0	1
$\rightarrow q_0$	$\{q_1, q_2\}$	–
q_1	$\{q_2\}$	$\{q_1, q_2, q_3\}$
q_2	$\{q_2\}$	$\{q_2\}$
$* q_3$	–	–



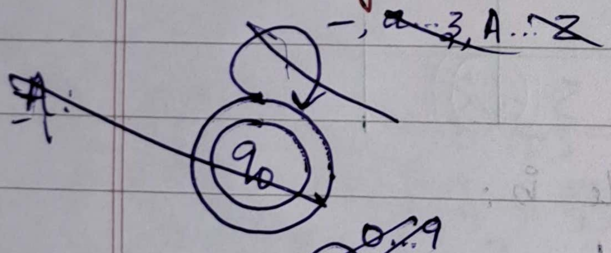
Only q_3 is final state because ε^* of q_0, q_1 & q_2 do not have q_3.

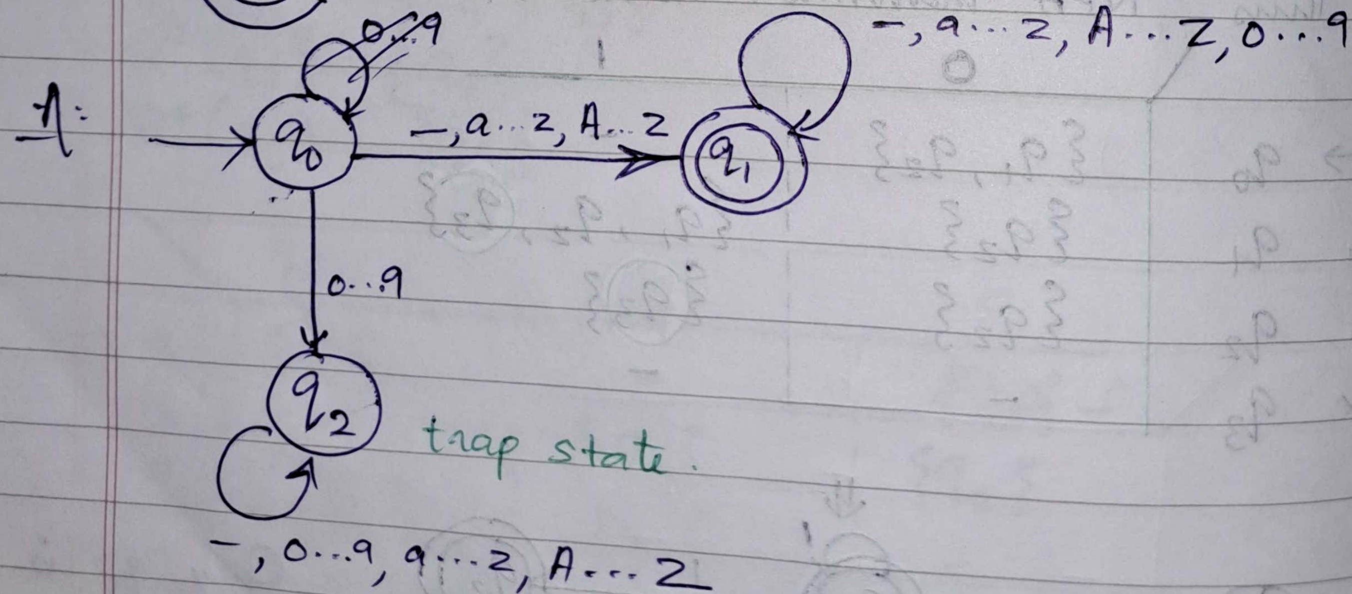
15|10|24|Tuesday.

Q. Design a FA to accept valid identifier

$\Sigma = \{ \_, a \ldots z, A \ldots Z, 0 \ldots 9 \}$

rule: if it starts with a digit then reject it.





17/10/24 | Thursday.

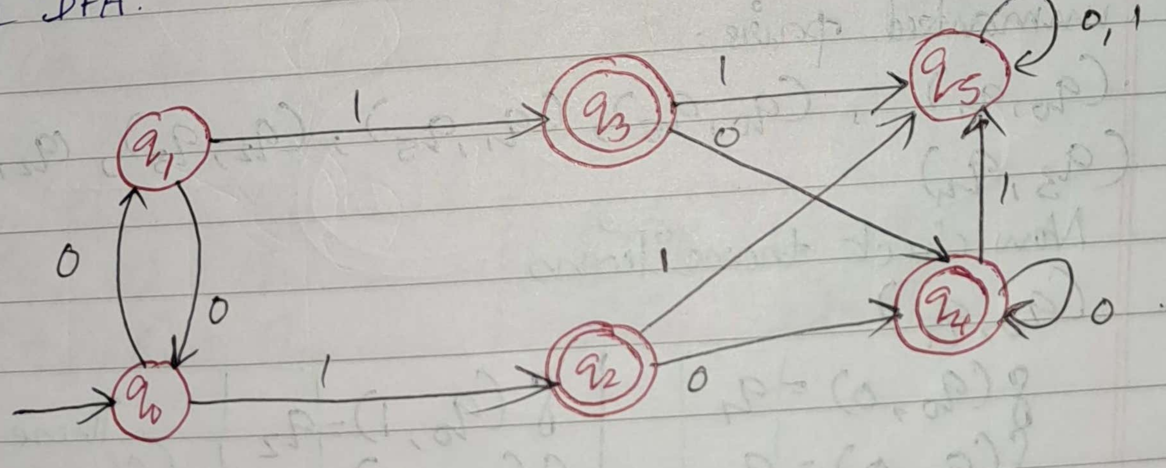
Minimization of DFA using MyHill-Nerode theorem:

- To reduce the no: of states
- Table filling method.

Steps:

1. Write a table of all pairs of states (P, Q).
2. Mark all pairs, where $P \in F$ & $Q \notin F$ [F is the set of final states) : meaning, pairs where one is a final state & other is not.
3. If there are any unmarked pair (P, Q) such that $[\delta(P, x), \delta(Q, x)]$ is marked then mark (P, Q), where x is the input symbol.
4. Repeat step 3 until no more marking is possible.
5. Combine the unmarked pairs to single unit to reduce the DFA.

eg.



Answer in next page,

~~Step 1~~ Step 1 & 2:

	q_0	q_1	* q_2	* q_3	* q_4
q_0					
q_1					
* q_2	✓	✓			
* q_3	✓	✓			
* q_4	✓	✓			
q_5	✓	✓	✓	✓	✓

Now mark pairs where one is final state ↑ (q_0, q_3) is also considered even though no direct path is present (green ticks)

Step 3:

unmarked pairs:

(q_0,q_1), (q_0,q_5), (q_1,q_5), (q_2,q_3), (q_2,q_4), (q_3,q_4)

Now check transitions

(q_0,q_1)

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

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

These are (q_0,q_1) & (q_2,q_3) are unmarked so we cannot do anything.

(q_0,q_5)

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

$\delta(q_5,0) = q_5$ | $\delta(q_5,1) = q_5$

(q_1,q_5) & (q_2,q_5)

(q_1,q_5) ↓ unmarked

(q_2,q_5) ↳ marked → since this is marked we can mark the corresponding pair (q_0,q_5) as well (Red ticks)

(q_1,q_5) → mark it

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

$\delta(q_5,0) = q_5$ | $\delta(q_5,1) = q_5$

↑ marked just now

→ marked

(q_2,q_3)

$\delta(q_2,0) = q_4$ | $\delta(q_2,1) = q_5$

$\delta(q_3,0) = q_4$ | $\delta(q_3,1) = q_5$

(q_2,q_4)

$\delta(q_2,0) = q_4$ | $\delta(q_2,1) = q_5$

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_5$

(q_3,q_4)

$\delta(q_3,0) = q_4$ | $\delta(q_3,1) = q_5$

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_5$

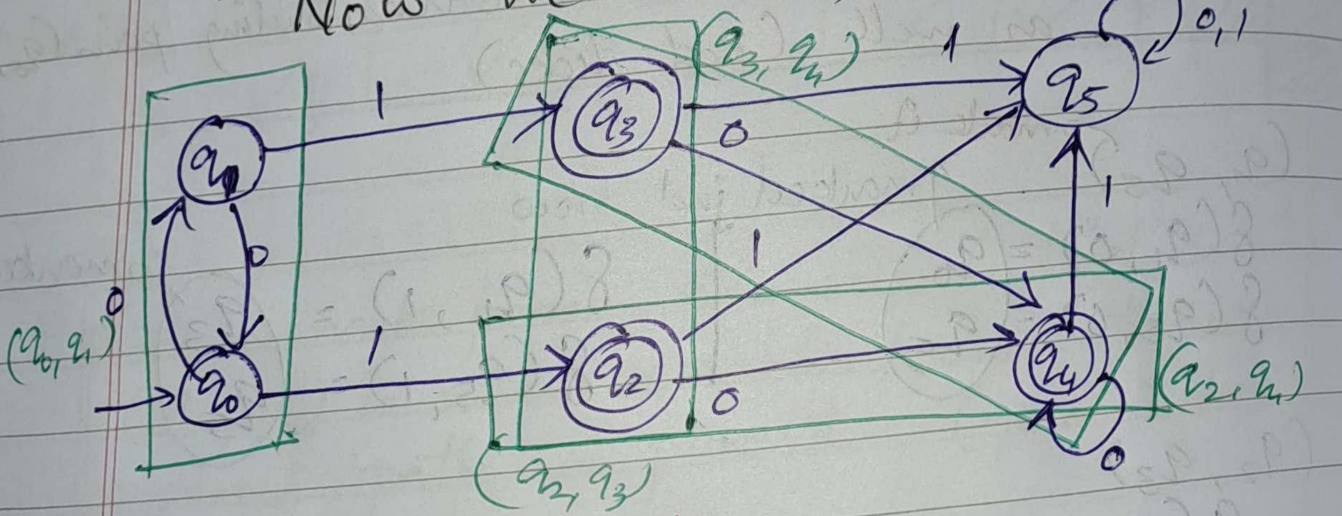
Step 4:

Repeating it again we see 4 unmarked pairs:

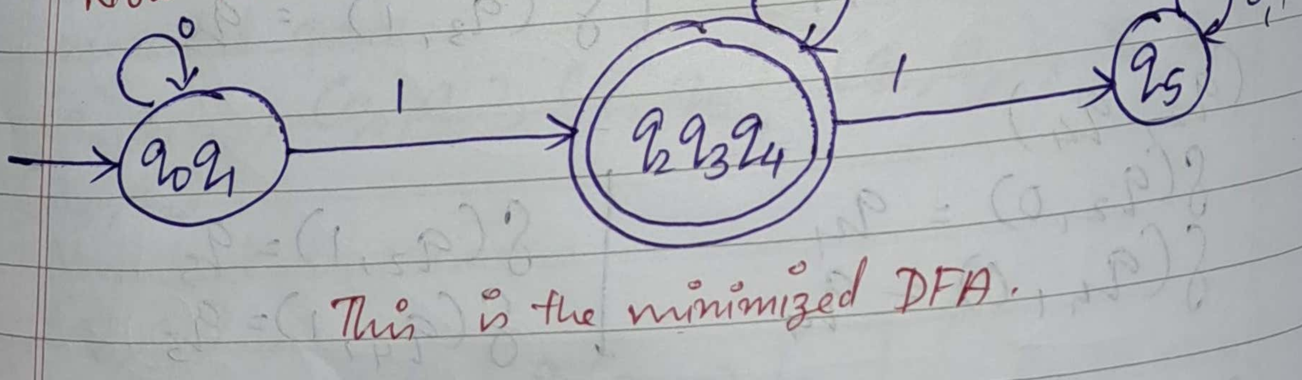
(q_0, q_1), (q_2, q_3), (q_2, q_4), (q_3, q_4)

Step 5:

Now we combine,



Now draw transitions



This is the minimized DFA.

21/10/24 | Monday.

Regular Expression (RE)

It is an expression that represents a pattern.

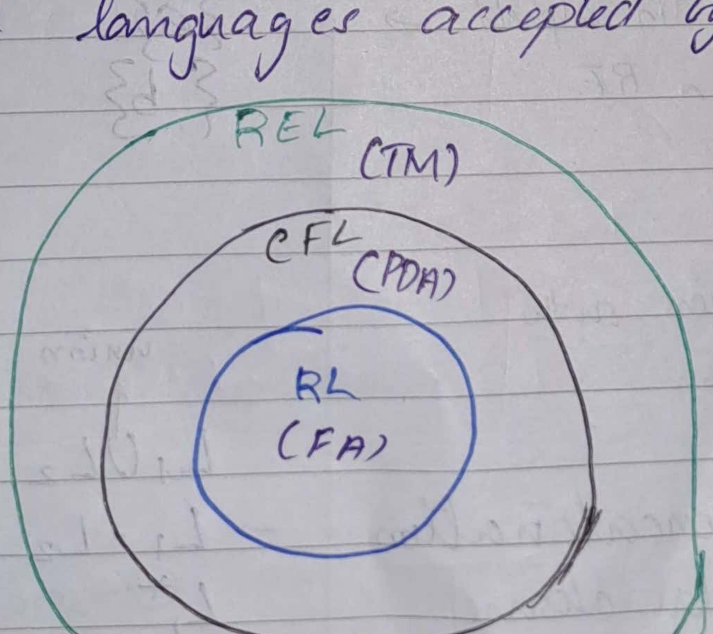
Regular Languages: (RL)
All languages accepted by finite automata.

Context free Languages: (CFL)
All ~~st~~ languages accepted by push down automata

~~Turing Machine:~~

Recursively Enumerable Languages: (REL)

All languages accepted by Turing Machine.



ie; Turing machine can accept RL, CFL & REL.

Regular Language

- language accepted by / represented with FA.
- can be represented with regular expressions.

$L_1 \longrightarrow \boxed{FA_1}$ OR RE_1

$L_2 \longrightarrow \boxed{FA_2}$ or RE_2

Definition

RE	Language
1. ϕ	$\{\}$
2. ε	$\{\varepsilon\}$
3. if $\Sigma = \{a, b\}$	
a → every symbol in alphabet	$\{a\}$
b is representing a RE.	$\{b\}$
4. if r for L_1	
s for L_2	
we can do operations on the two RE's r / s:	
1. $r + s$	$L_1 \cup L_2$ (union)
2. $r \cdot s$ → concatenation	→ $L_1 \cdot L_2$
3. r^* → star closure	L_1^* → zero or more.

priority: $r^* > r \cdot s > r + s$

same symbol - then direction: lt to rt.

say $L_1 = \{a, b, c\}$

$L_2 = \{x, y, z\}$

$r + s \quad L_1 \cup L_2 = \{a, b, c, x, y, z\}$

$r \cdot s \quad L_1 \cdot L_2 = \{ax, ay, az, bx, by, bz, cx, cy, cz\}$

r^* if $L_1 = \{0\}$

$L_1^* = \{0\}^* \rightarrow 0^n \; ; n \geq 0$

↳ zero can occur zero or more times in the string.

$L_1^* = 0^* = \{\varepsilon, 0, 00, 000, \ldots\}$

RE	Language
1. 0	$\{0\}$
2. 0^*	$\{\varepsilon, 0, 00, 000, \ldots\}$
3. 1^*	$\{\varepsilon, 1, 11, 111, \ldots\}$
4. $(0+1)$	$\{0, 1\}$
$\{0\} + \{1\}$ → union	
5. 01	$\{01\}$
6. $0 \cdot 1^*$	$\{0 \cdot \varepsilon, 0 \cdot 1, 0 \cdot 11, 0 \cdot 111, \ldots\}$
$0 \cdot \{\varepsilon, 1, 11, 111, \ldots\}$	$\{0, 01, 011, 0111, \ldots\}$
7. $1 \cdot 0^* \cdot 1$	$\{11, 101, 1001, \ldots\}$
$\{1\} \; \{\varepsilon, 0, 00, \ldots\} \; \{1\}$	

8.	$(1+0)^*$ → $\{1,0\}^*$	$\{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$
9.	$(0+1)^* \cdot 11$	$\{11, 011, 111, 0011, 0111, \ldots\}$ all strings ending with '11'

Q. Find the regular expression for the following (binary) languages?

1. All strings start with 0.

A: ~~$0 \cdot \{1,0\}$~~ $0 \cdot (1+0)^*$ → $0 \cdot (1+0)^*$

2. All strings end with 00

A: $(1+0)^* \cdot 0 \cdot 0$ → $(1+0)^* \cdot 00$

3. All strings containing 101

A: $(0+1)^* \cdot 1 \cdot 0 \cdot 1 \cdot (0+1)^*$ → $(0+1)^* \cdot 101 \cdot (0+1)^*$

4. all strings containing three 1s

A: ~~$(0+1)^*$~~ ~~$0^* \cdot 111 \cdot 0^*$~~

~~$0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^*$~~

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

5. All strings containing odd no. of 0s

A: ~~$1^* \cdot 0 \cdot 1^* \cdot (1^* 0 1^* \cdot 1 \cdot 0)^* \cdot 1^*$~~

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

6. All strings containing exactly ~~3 ones~~? three 1s?

A: ~~$(0+1)^* \cdot 111 \cdot (0+1)^*$~~

$0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^*$

6. All strings containing exactly ~~3 ones~~? three 1s?

A: ~~$(0+1)^* \cdot 111 \cdot (0+1)^*$~~

$$0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^* \cdot 1 \cdot 0^*$$

22/10/24 | Tuesday.

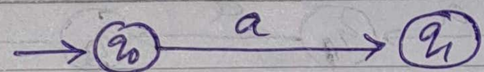
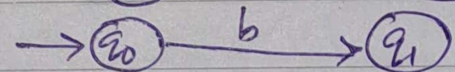
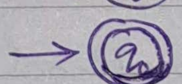
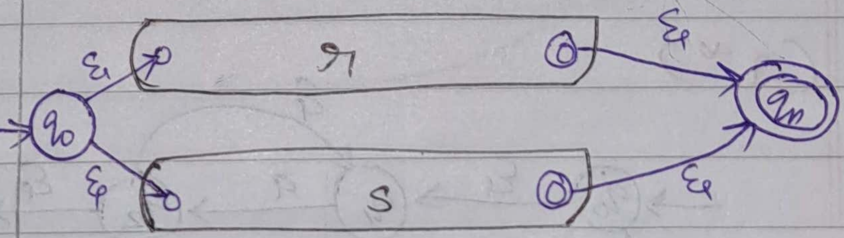
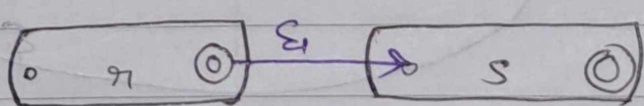
7. {All strings start with 1 and end with 00} $\Sigma = \{0, 1\}$

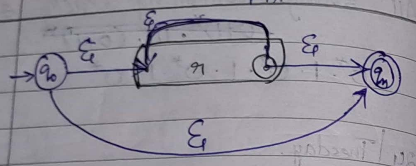
A:

$$1 \cdot (0+1)^* \cdot 00$$

RE to FA : (Thompson's Model)

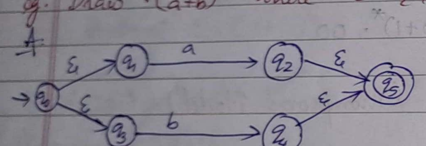
RE $\longleftrightarrow$ FA

RE	FA
a	
b	
ε	
$r+s$	
$r \cdot s$	

RE	FA
r^*	

eg. Draw FA of $(a+b)$ where $\Sigma = \{a, b\}$

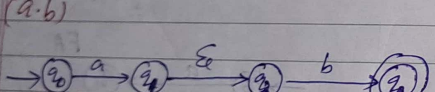
A



$\{a, b\}$

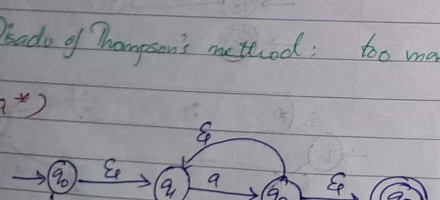
RE ↔ FA

$(a \cdot b)$



★ Disadv of Thompson's method: too many ϵ

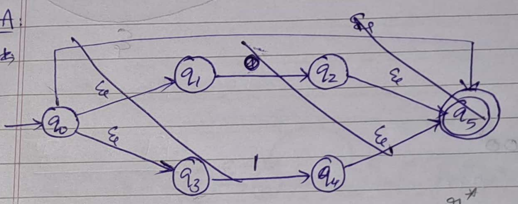
(a^*)



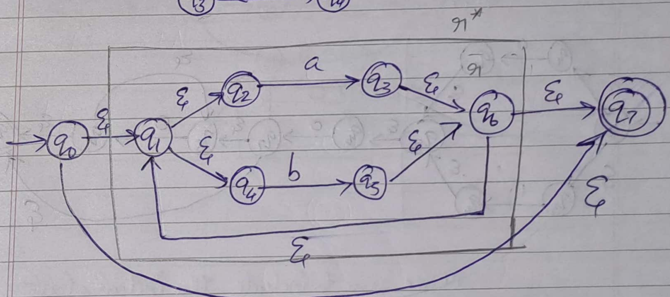
Q. Draw ~~an~~ FAs for following REs?

1. $(1+0)^*$
2. 001^*
3. $(0+1)00^*$
4. $01(0+1)01$
5. 01^*010^*

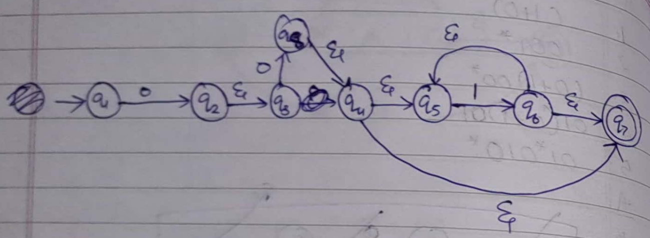
A:



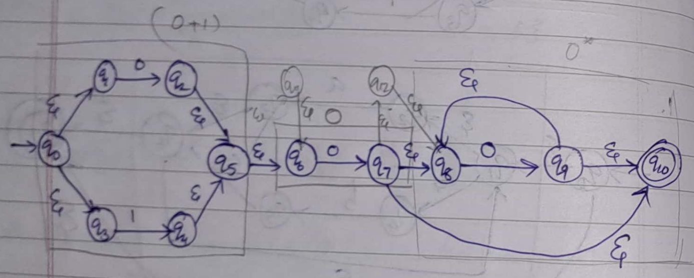
1.



2. 001^*

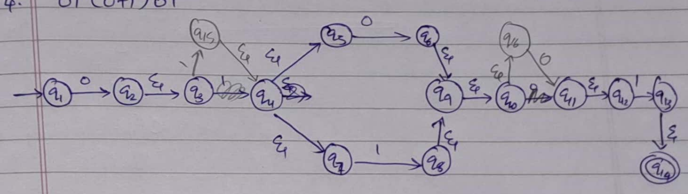


3. $(0+1)\,00^*$



Make sure to include the initial states of all individual parts.

4. $01\,(0+1)\,01$



5. 01^*010^*