UNIT 1 - AUTOMATA

AUTOMATA-

- Finite automata and formal languages (grammar) are used in design of compilers and softwares.
- Turing machines - what we can expect from software?

interactability — NP-hard, NP-complete

decidability (confident answer) — yes, NO

undecidability
↳ halting problem of a prog.

eg-
```
int main(){
    while(1){
        =
    }
    return 0;
}
```

It is undecidable whether this prog. will halt or not. (we are not confident about the answer)

Symbol - basic building block of any machine; it can be any alphabet, number, picture.

eg- a, b, 0, 1, &, $ etc.

Alphabet - An alphabet is any finite set (countable) of symbols.

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

String - A string is a finite sequence of symbols taken from Σ.

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

$S = ab, ba, aa, bb, aba, \ldots$

→ A string has a finite length.

→ No. of strings can be infinite.

Language - A language is a subset of Σ^* for some alphabet Σ. It can be finite or infinite.

eg- Finite $\Sigma = \{a, b\}$

here, length lies a criteria

L = all strings starting with a of length atmost 2

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

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

here, length is not a criteria

L = all strings starting with 'a'

$= \{a, aa, ab, aaa, abb, aab, \ldots\ldots\}$

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

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

→ Σ^* has all the possible string with the symbols present in the particular set, including ϵ. ('ϵ' means NULL)

Length of string - It is the number of symbols present in a string denoted by $|S|$. (S is string)

eg- $S = abcd$

$|S| = 4$

$|\epsilon| = 0$ or λ

(no length)

<u>Kleene star</u> - The Kleene star 'Σ^*' is an unary operation on a set of symbols or strings: Σ, that gives the infinite set of all possible strings of all possible lengths over Σ <u>including 'ϵ'</u>.

<u>eg</u> - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

where Σ^p is the set of all possible strings of length p.

<u>example</u> - $\Sigma = \{a, b\}$

$$\Sigma^* = \{\underbrace{\epsilon}_{\Sigma^0}, \underbrace{a, b}_{\Sigma^1}, \underbrace{aa, ab, ba, bb}_{\Sigma^2}, \underbrace{aaa, aab, \ldots}_{\Sigma^3}\}$$

(language is a subset of Kleene star)

<u>Kleene closure/Plus</u> - The set of 'Σ^+' is the infinite set of all possible strings of all possible lengths over Σ <u>excluding 'Λ or ϵ'</u>

$$\boxed{\begin{aligned} \Sigma^+ &= \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots \\ \Sigma^+ &= \Sigma^* - \{\epsilon \text{ or } \Lambda\} \end{aligned}}$$

<u>Power of Σ</u>

<u>eg</u> - $\Sigma = \{a, b\}$ $\qquad \Sigma^0 = \{\epsilon\} \rightarrow 2^0 = 1$

$\Sigma^1 = \{a, b\} \quad = 2^1 = 2$

$\Sigma^2 = \Sigma\Sigma = \{a, b\}\{a, b\} = 2^2 = 4.$

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

$\rightarrow \Sigma^n = 2^n =$ all n length strings

- Our goal is to find whether the given string 'S' belongs to given lang. 'L'.

and if string doesn't belong to the lang then it gives an error otherwise it does

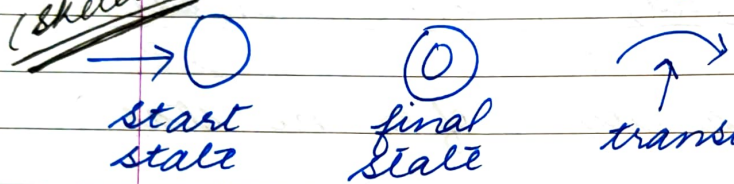
Finite automata

L_1 = set of all strings which start with 'a'.

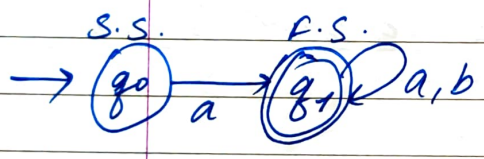
$\Sigma = \{a, b\}$

$L_1 = \{a, aa, ab, aaa, aab, aba, \ldots\}$

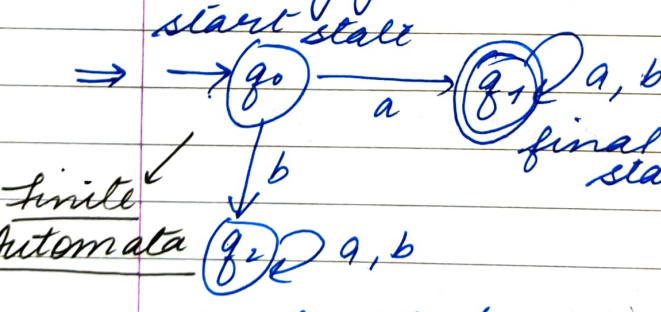
(skeleton)



(By default we are at the start state)



(If we reach to final state then our string gets accepted)



Finite Automata

Dead state (we can't go from this state to final state)

① Identify alphabet

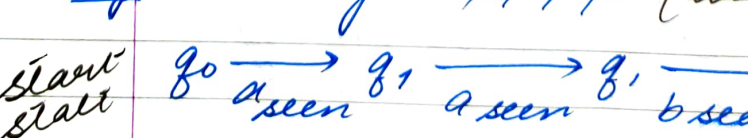
② Identify the lang.

③ Take the smallest possible string and draw the skeleton if set of all strings are infinite

④ Take largest possible string and draw the skeleton if set of all strings are finite

NOTE (If ϵ is part of string then make initial state as final state always)

eg- String → aaba (acceptable)



eg- String → b a a (not acceptable)

~~Start~~ ~~State~~ $q_0 \xrightarrow{b \text{ seen}} q_2 \xrightarrow{a \text{ seen}} q_2 \xrightarrow{a \text{ seen}} \underline{q_2}$ ~~final~~ dead state

Types of Finite Automata

Finite Automata

- Finite Automata without Output
  - DFA
  - NFA
  - ε-NFA
- Finite Automata with output
  - Mealy
  - Moore.

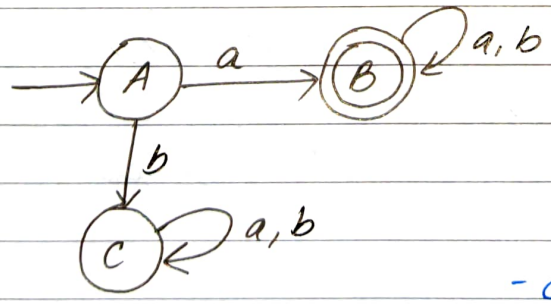
DFA - Deterministic Finite Automata.

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

- Q → finite set of states
- Σ → i/p symbols
- δ → transition function
- q_0 → initial state
- F → set of final states

- In DFA, it is possible to have more than one final state.
- Initial state can be the final state.

eg-

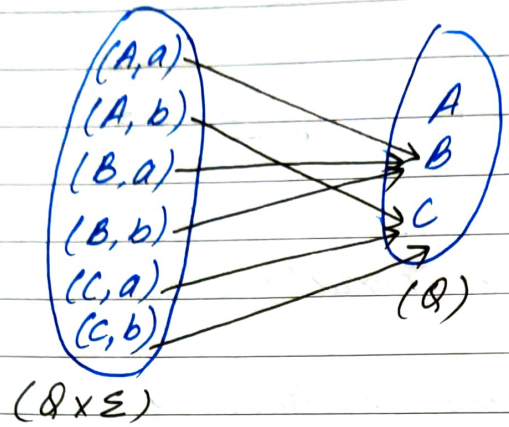


- $q_0 = A$
- $Q = \{A, B, C\}$
- $\Sigma = \{a, b\}$
- $F = \{B\}$
- $\delta : Q \times \Sigma \rightarrow Q$

eg- $(A \times (a, b) \rightarrow \{B, C\})$

↓ transition function.

$$\boxed{\delta : Q \times \Sigma \to Q}$$



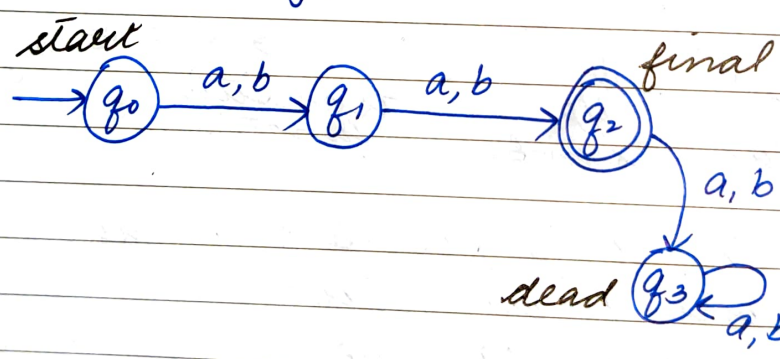
Q construct a DFA, that accepts set of all string over $\Sigma = \{a, b\}$ of length 2.

Sol^n

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

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

(string is finite so we will take longest possible string)



<u>string acceptance</u>- a string is said to be accepted by finite automata, if we are able to reach the final state starting from initial state upon scanning the entire string.

<u>language acceptance</u>- a finite automata is said to accept a lang; if all the

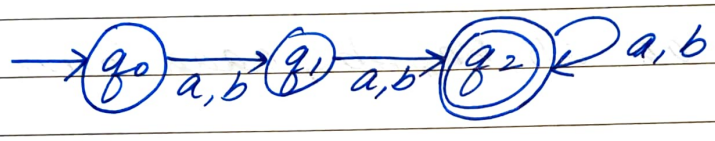
strings in lang. are accepted by the finite automata as well as the string which are not in the lang. should be rejected by finite automata.

Q construct a DFA, that accepts a set of all strings over $\Sigma = \{a, b\}$ where length is atleast 2.

sol^n $\Sigma = \{a, b\}$

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

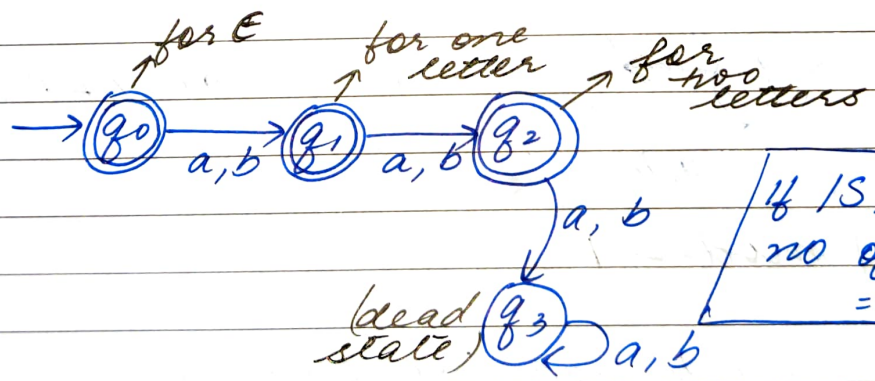
(lang. - infinite, take smallest string)



Q $\Sigma = \{a, b\}$, $|S| \leq 2$

sol^n $L = \{\epsilon, a, b, aa, ab, ba, bb\}$

(finite, take largest string)

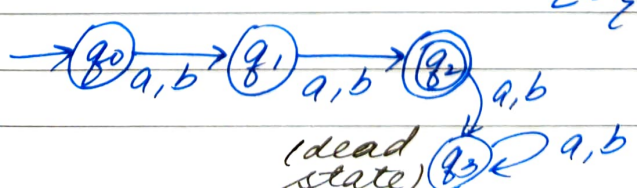


If $|S| \leq n$
no of states
$= n+2$

Q $\Sigma = \{a, b\}$, $|S| = 2$

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

sol^n



If $|S| = n$
no. of states
$= n+2$

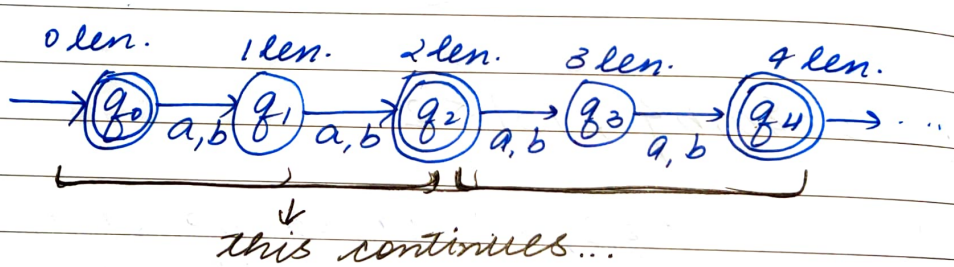
Q $\Sigma = \{a, b\}$, $|S| \geq 2$ $L = \{aa, ab, ba, bb, aaa \ldots\}$

$\boxed{\text{If } |S| \geq n \text{ no of states } = n+1}$

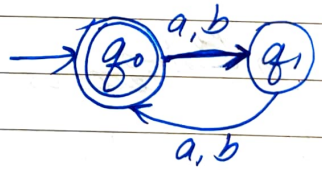
→ We always find minimal DFA.

Q construct a minimal DFA, $\Sigma = \{a, b\}$
$|w| \bmod 2 = 0$

Sol^n $L = \{\epsilon, aa, ab, ba, bb, aaaa, aaab, baaa \ldots\}$



for even length

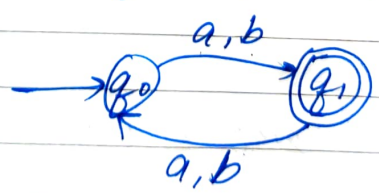


— Minimal DFA.

Q $|w| \bmod 2 = 1$

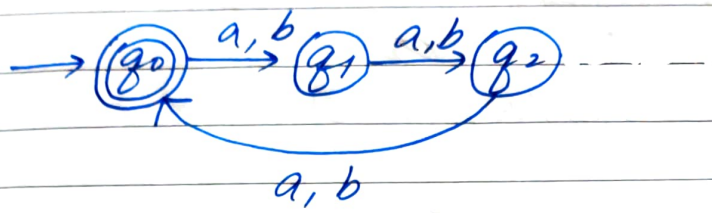
Sol^n $L = \{a, b, aaa, aab, baa \ldots\}$

for odd length



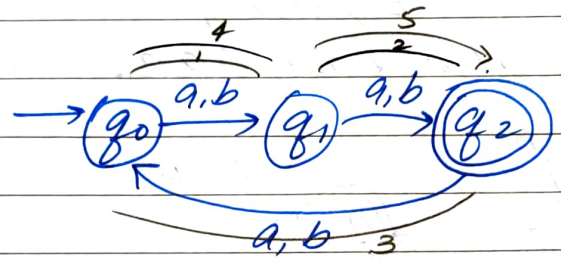
Q $|w| \bmod 3 = 0$ OR $|w| \cong 0 \bmod 3$.
$w \in \{a, b\}$

Sol^n $L = \{\varepsilon, aaa, aab, baa, \ldots\}$



Q $|w| \bmod 3 = 2$

Sol^n



If we take ababa.
(String of 5)
$5 \% 3 = 2$.

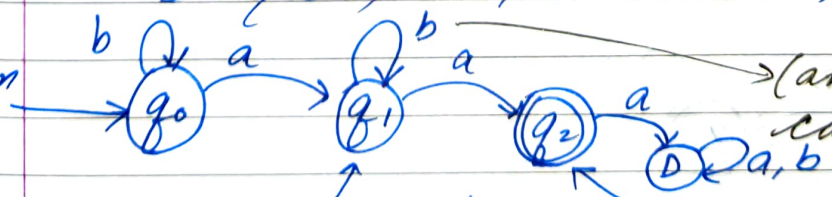
Q $|w| \bmod x = x$
Here, we will make x states and will make transition and then the state 'x' will be made as final.

Homework -

(1) Minimal DFA, $w \in \{a, b\}^*$, $n_a(w) = 2$
no. of 'a' should be equal to 2.

$L = \{aa, aab, aba, baa, aabb, aabbb, abab, \ldots\}$

Sol^n



(any no. of b's can come)

here, 1 a is seen

here, 2 a's are seen

② <u>DFA</u> $w \in \{a,b\}^*$ $n_a(w) \geq 2$

③ $w \in \{a,b\}^*$ $n_a(w) \leq 2$

④ $w \in \{a,b\}^*$ $n_a(w) \bmod 2 = 0$

or

$n_a(w) \equiv 0 \bmod 2$

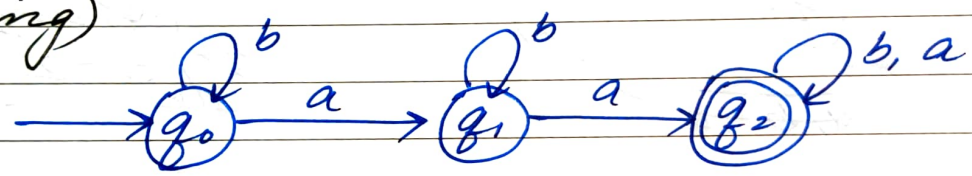
$w \in \{a,b\}^*$

⑤ $n_a(w) \bmod 3 = 0$ or $n_a(w) \equiv 0 \bmod 3$

Sol^n 2. $n_a(w) \geq 2$ $w \in \{a,b\}$

$L = \{aa, aab, aba, baa, aaa, \ldots\}$

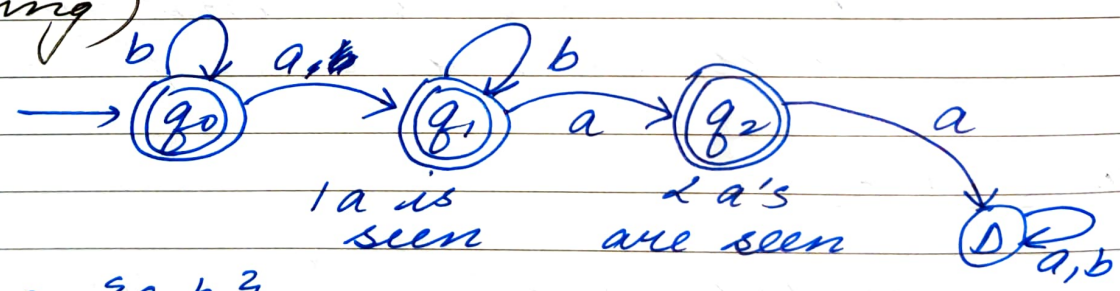
(∞ string)



Sol^n 3. $n_a(w) \leq 2$ $w \in \{a,b\}$

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

(∞ string)

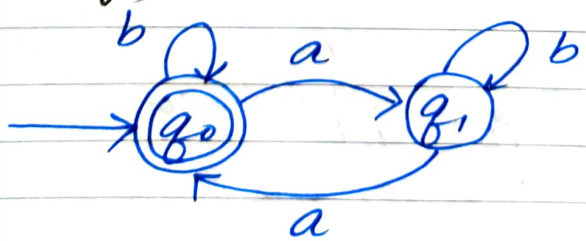


$w = \{a,b\}$

Sol^n 4. $n_a(w) \bmod 2 = 0$ or $n_a(w) \equiv 0 \bmod 2$

$L = \{\epsilon, aa, aab, aba, aaaab, \ldots\}$

(∞ string)

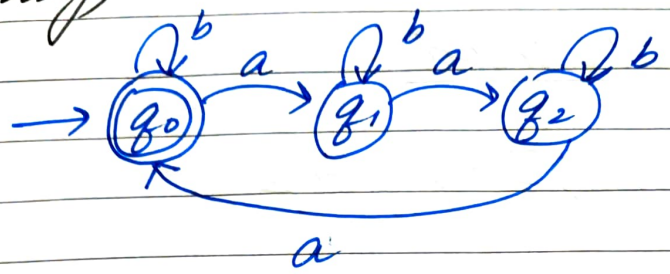


Date:

Sol^n 5. $w \in \{a, b\}$; $n_a(w) \bmod 3 = 0$ or $n_a(w) \cong 0 \bmod 3$

$L = \{\varepsilon, aaa, aaab, aaba, aaaaaa, \ldots\}$

(a string)



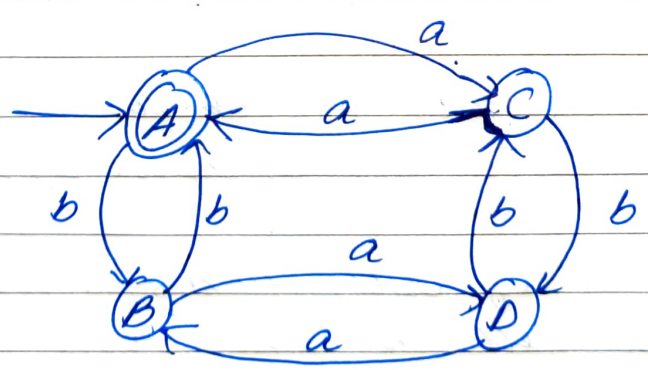
If, $n_a(w) \bmod 3 = 2$, there would've been total 3 transition states and ~~a~~ 2nd state would be made as final.

If $n_a(w) \bmod 2 = 1$, there would've been total 2 transition states and 1st state would be made as final.

Q $w \in \{a, b\}$ $n_a(w) \cong 0 \bmod 2$ && $n_b(w) \cong 0 \bmod 2$

Sol^n $L = \{\varepsilon, aa, bb, aabb, abab, baab, \ldots\}$

A → even a's even b's
B → even a's odd b's
C → odd a's even b's
D → odd a's odd b's



(a and b can be replaced as 0 or 1)

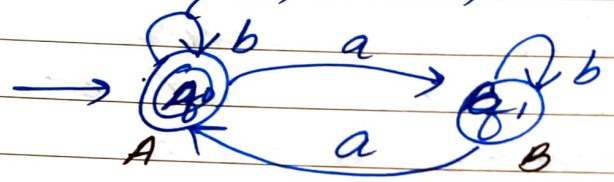
A is made as final state because acc^n to question there should be even a's and even b's)

If for even a's and odd b's we will make transition state 'B' as final.

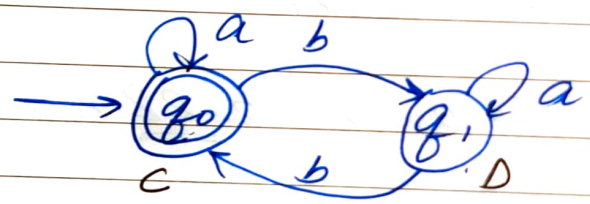
Cross product Method

Q $w \in \{a, b\}$ $n_a(w) \cong 0 \bmod 2$

Sol^n $L = \{\varepsilon, aa, b, aba, aab, baa, \ldots\}$



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



$n_a(w) \cong 0 \bmod 2$ && $n_b(w) \cong 0 \bmod 2$

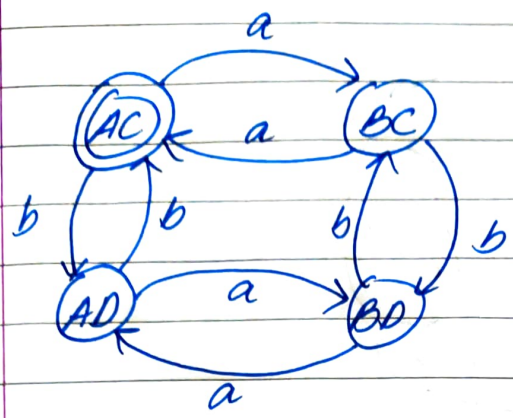
By cross product,

$L_1 \times L_2 \rightarrow \{A, B\} \times \{C, D\}$

$= \{AC, AD, BC, BD\}$

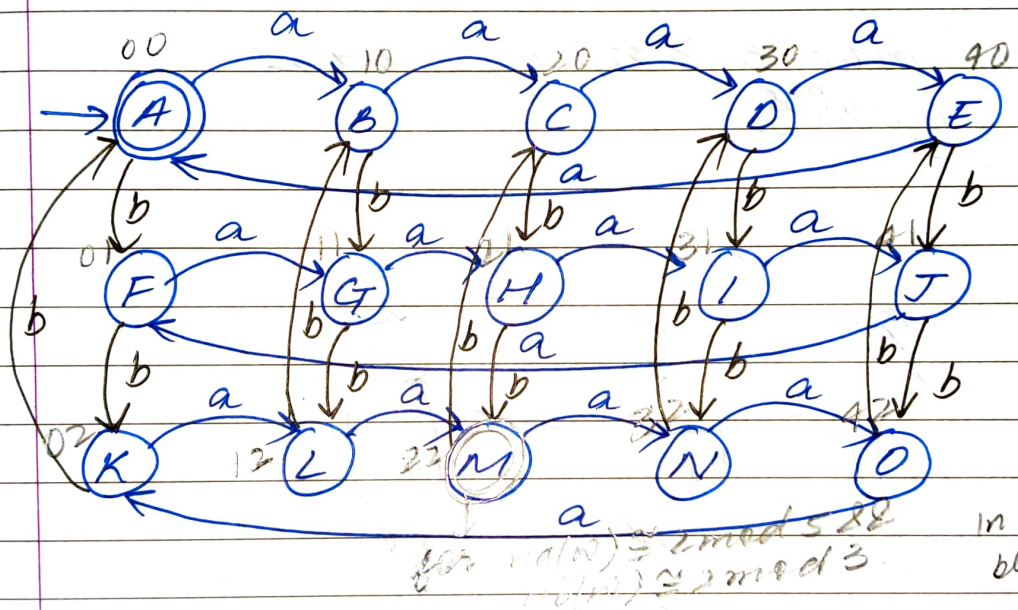
→ AC is the final states as A and C are final states in L_1 and L_2.

→ Combination of both DFA's initial and final state will be the new initial and final states.



<u>Example</u> - DFA of no. of a's divisible by 5 and no. of b's divisible by 3.

<u>Sol^n</u> Make $5 \times 3 = 15$ states.



this represents the no. of a's and b's entered.

for eg: $n_a(w) \cong 2 \bmod 5$, $n_b(w) \cong 2 \bmod 3$

In this 22 will be made as the final state.

(There is one final state, and here a's divisibility is 5 as it has 5 transition state and b's divisibility is 3 as it has 3 transition state).

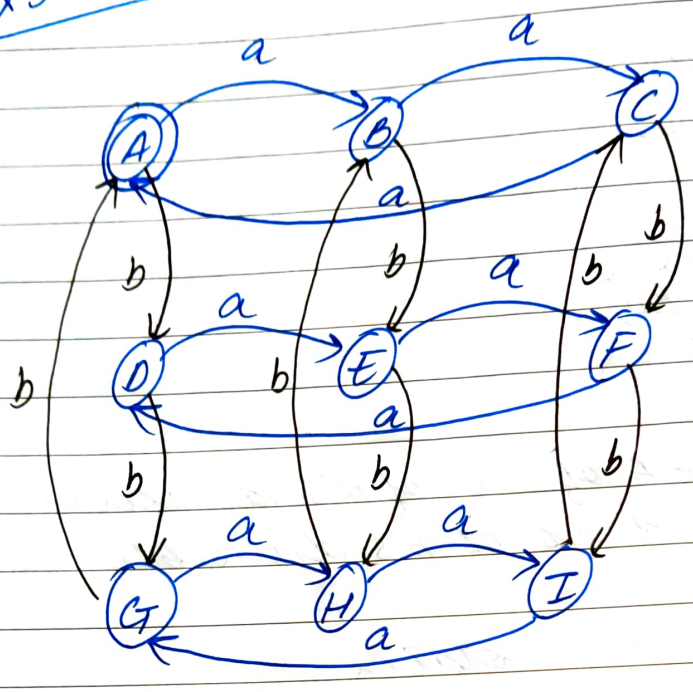
<u>Homework</u> -

Q DFA $w \in \{a, b\}^*$ $\quad n_a(w) \cong 0 \bmod 3$ && $n_a(w) \cong 0 \bmod 3$

Page No.:
Date:

$3 \times 3 = 9$ states

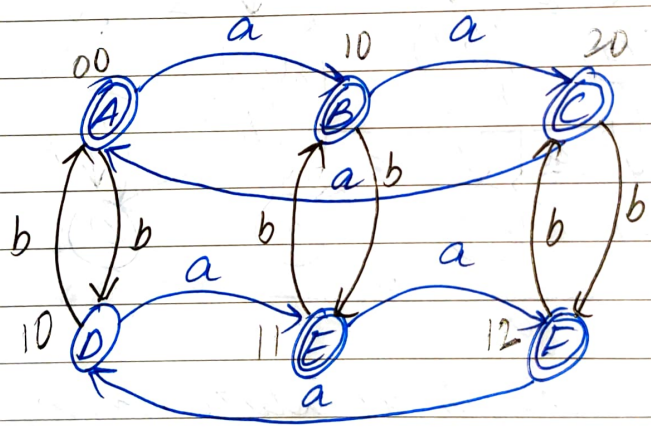
Sol^n



Q $n_a(w) \bmod 3 \geq n_b(w) \bmod 2$

Sol^n $3 \times 2 = 6$ states

00
10
20
11
12



Q construct a minimal DFA, which accepts set of all strings over {0, 1} which when interpreted as binary no's is divisible by 2.

Sol^n eg. $(111)_2 \rightarrow (7)_{10} \rightarrow$ should'nt be accepted

$(100)_2 \rightarrow (4)_{10} \rightarrow$ 4 is divisible by 2, so acceptable.

$(101)_2 \rightarrow (5)_{10} \rightarrow$ not acceptable.

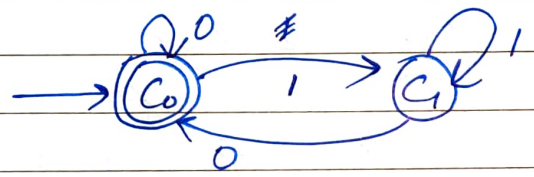
eg- $(11)_2 \rightarrow 1\times2+1 = (3)_{10}$
$(10)_2 \rightarrow 1\times2+0 = (2)_{10}$
$(101)_2 \rightarrow (10)_2\times2+1 = 4+1 = (5)_{10}$
$(1010)_2 \rightarrow (101)_2\times2+0 = 5\times2+0 = (10)_{10}$

when $x\%2$

0 1 → Remainders

$(2n) \leftarrow C_0$ $C_1 \rightarrow (2n+1)$

(Total no. of states = 2) (because we are dividing it by 2 and we are getting 2 remainders).

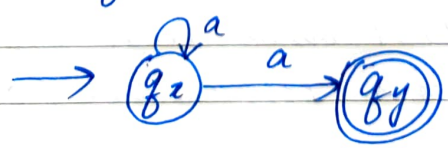


Non-deterministic finite automata.

(By taking any one i/p from one state, it can traverse through several states)

→ Has ability to be in several states at once.
→ Transition from a state on an i/p symbol can be to any set of states.

How it works?
→ Start in one start state.
→ Accept if any sequence of choices leads to a final state.



(two choices to move) (by taking one i/p we can move to multiple states)

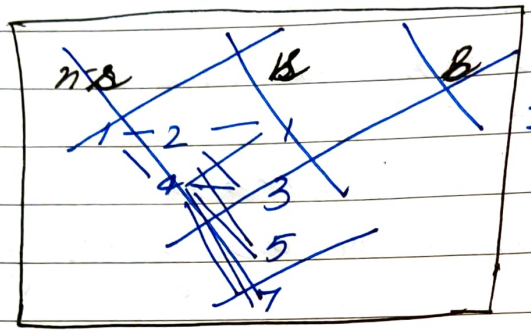
Date:

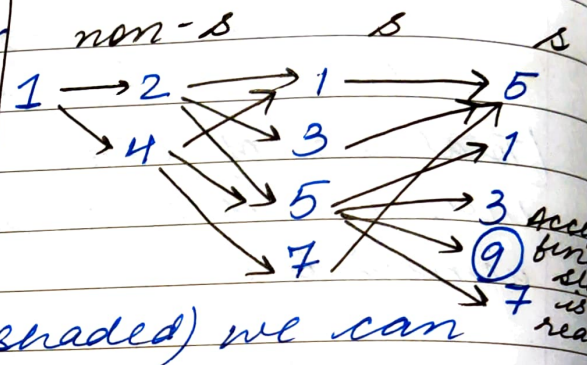
Example- (either move to shaded or ~~start~~ non-shaded)

initial state

1	2	3
4	5	6
7	8	9

→ final state.





(from 1 (for non-shaded) we can go to 2 and 4, from 2 (for shaded) we can go to 1, 3, 5 and from 4 we can go to 1, 5, 7. (we write the states only once))

→ Therefore the possible paths to follow non-s, s, s are 1-2-5-9 and 1-4-5-9. (chosing these paths we will reach the final state).

Define NFA (5 tuple F-A)

- A finite set of states, typically Q.
- an i/p alphabet, typically Σ.
- A transition function, typically δ.
- A start state in Q, typically q_0
- A typical set of final states $F \subseteq Q$.

→ The difference b/w DFA and NFA only comes from the transition function (δ).

→ In NFA, we are not supposed to show all the examples.

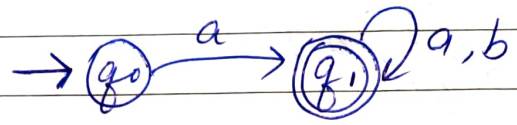
DFA - $\delta(Q, \Sigma) \rightarrow Q$ (one state). (moves to)

NFA - $\delta(Q, \Sigma) \rightarrow 2^Q$ (moves to power set - combination of states)

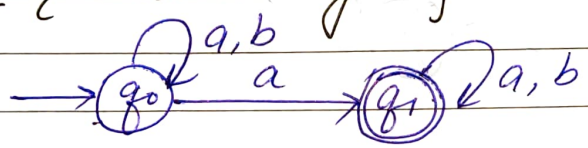
→ every NFA can be converted into DFA.

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

1) NFA starts with 'a'



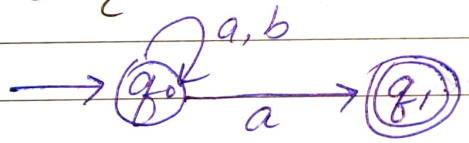
2) $L_2 = \{$containing a$\}$



(contains atleast 1 'a')

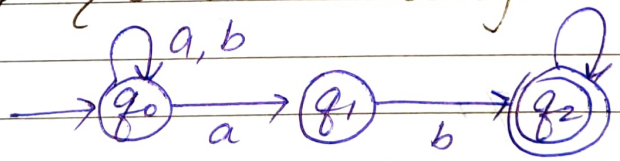
(a, ab, ba, aab, bbab)

3) $L_3 = \{$ends with a$\}$



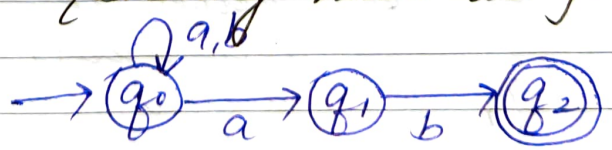
(we don't care whatever comes first but should end with 'a')

4) $L_4 = \{$contains 'ab'$\}$



(contains atleast 1 'ab')

5) $L_5 = \{$ending with 'ab'$\}$ (ab, aab, bab ...$\}$



For NFA.

$\delta : Q \times \Sigma \to 2^Q$ (transition state)

For DFA.

$\delta : Q \times \Sigma \to Q$

$(Q \subset 2^Q)$

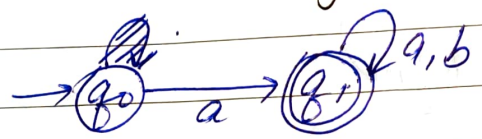
→ Every DFA is a NFA

Conversion of NFA to DFA

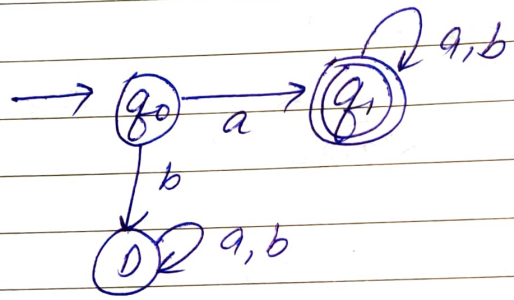
→ Every NFA can be converted to DFA and every DFA can be converted to NFA, so they are equivalent in power.

eg- $L_1 = \{$ starts with a $\}$ $\Sigma = \{a, b\}$

NFA ⇒



DFA ⇒



⇒ Subset construction method.

State transition table

For NFA.

	a	b
→ q_0	q_1	ϕ → dead configuration
* q_1	q_1	q_1

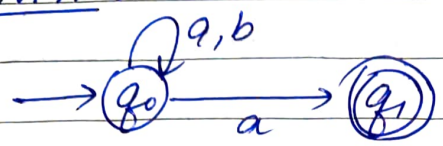
For DFA.

	a	b
→ q_0	q_1	D
q_1	q_1	q_1
D	D	D

from the state transition table of NFA, we draw the DFA table and with the help of the table we draw the following DFA

eg- $L_1 = \{$ strings end with a $\}$ $\quad \Sigma = \{a, b\}$

NFA ⇒



state transition table

NFA

	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
* q_1	$\{\ \}$	$\{\ \}$

we will take their union for DFA table.

(In DFA we have only one state to move so we will combine both states)

reachable states are taken here.

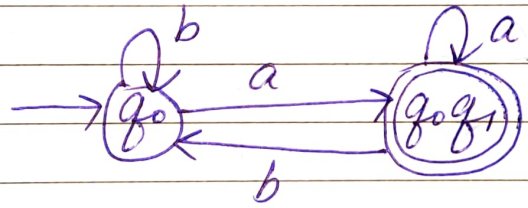
DFA

	a	b
→ q_0	$[q_0 q_1]$	$[q_0]$
$q_0 q_1$	$[q_0 q_1]$	$[q_0]$

$(\{q_0, q_1\} \cup \{\ \})$

($\{q_0, q_1\}$ & ϕ is taken from the NFA table)

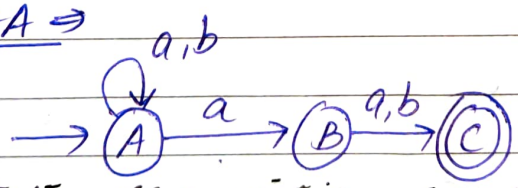
DFA ⇒



(DFA ending with 'a')

eg- all strings in which second symbol from RHS is 'a'. $L = \{\ldots ab, \ldots aa, \ldots aab, \ldots bab\}$

NFA ⇒



(for [AB], union states A & B from NFA table and for [AC], union states A & C from NFA table)

state transition table

FOR NFA

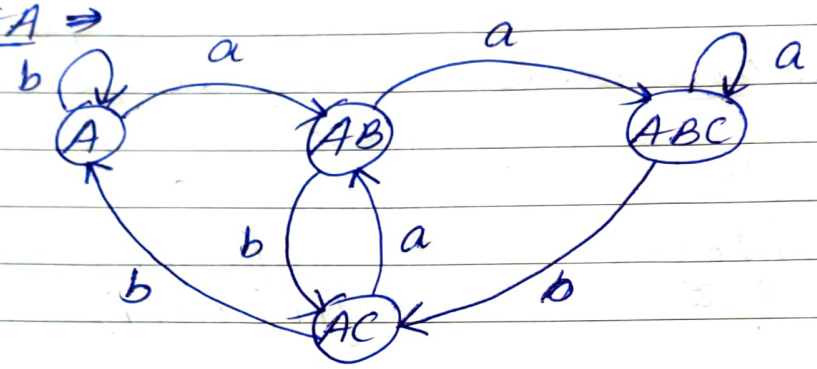
	a	b
→ A	$\{A, B\}$	$\{A\}$
B	$\{C\}$	$\{C\}$
* C	$\{\ \}$	$\{\ \}$

FOR DFA

	a	b
→ [A]	[AB]	[A]
[AB]	[ABC]	[AC]
* [AC]	[AB]	[A]
* [ABC]	[ABC]	[AC]

→ As in NFA, 'C' was final states, so in DFA table all the states containing 'C' will be final states.
(AC and ABC will be final states)

DFA ⇒

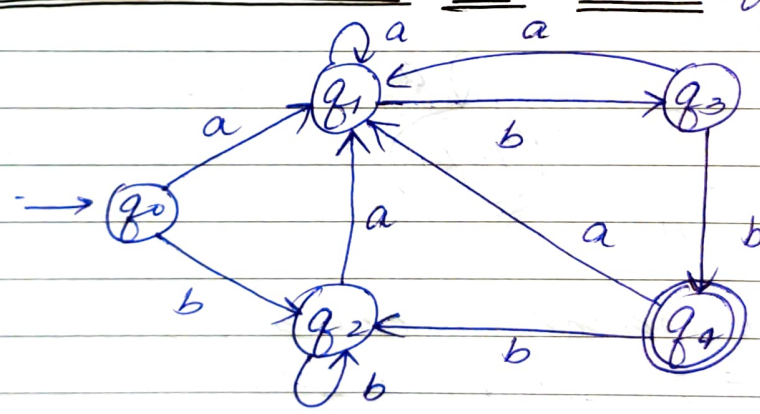


$\boxed{NFA \cong DFA}$

MINIMIZATION OF DFA

Rule 1- find the state which is unreachable from initial state and then delete it.

eg-



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

→ (p, q) equivalent
(p and q are states that are equivalent)

$\delta(p, \underset{\text{string}}{w}) \in F \Rightarrow \delta(q, w) \in F$
$\delta(p, w) \notin F \Rightarrow \delta(q, w) \notin F$.
(p and q should show similar properties)

Page No.: Date: YOUVA

state transition table

	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	* q_4
* q_4	q_1	q_2

reaching same state so they will be in one group.

$(q_0\ q_1)$
$(q_0\ q_2)$
$(q_2\ q_3)$

q_3 is reaching final state so it will be in separate group.

condition for (p, q) equivalent

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

→ if $|w| = 0$ → 0 equivalent
if $|w| = 1$ → 1 equivalent
if $|w| = 2$ → 2 equivalent

set of all non-final states — set of all final states

0 equi. → $[q_0\ q_1\ q_2\ q_3]\ [q_4]$

1 equi → $[q_0\ q_1\ q_2]\ [q_3]\ [q_4]$

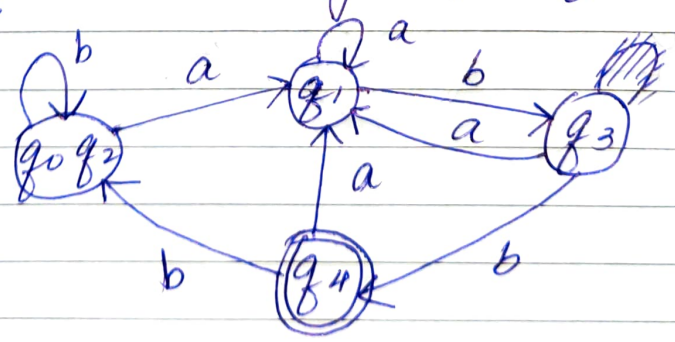
(groups are decided by the table when the state reaches to the same state after passing string)

2 equi → $[q_0\ q_2]\ [q_1]\ [q_3]\ [q_4]$

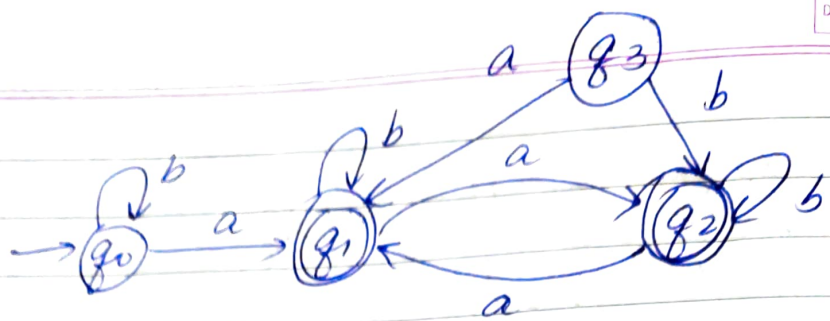
3 equi → $[q_0\ q_2]\ [q_1]\ [q_3]\ [q_4]$

(we continue to find equivalents until we get two same equivalents)

Therefore, these groups forms the ~~final~~ states

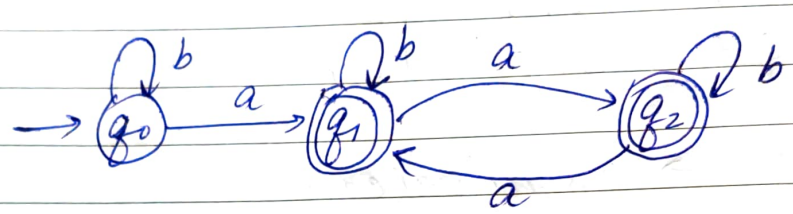


eg-



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

→ 'q3' is not reachable from 'q0', ie initial states, so we will delete 'q3' state and all its transitions.



<u>state transition table</u>

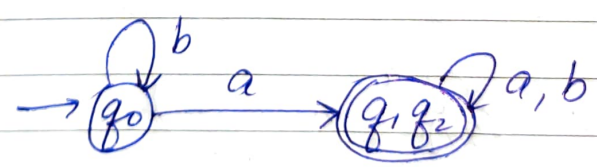
	a	b
→ q_0	* q_1	q_0
* q_1	* q_2]	* q_1]
* q_2	* q_1	* q_2]

<u>0 equi</u> → $[q_0]$ $[q_1\ q_2]$

<u>1 equi</u> → $[q_0]$ $[q_1\ q_2]$

} same.

Two states in DFA. q_0, $q_1 q_2$



ϵ - NFA

From one state on receiving i/ps or ϵ, we can go to multiple states.

ϵ- NFA

(epsilion)

$$\boxed{Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q}$$

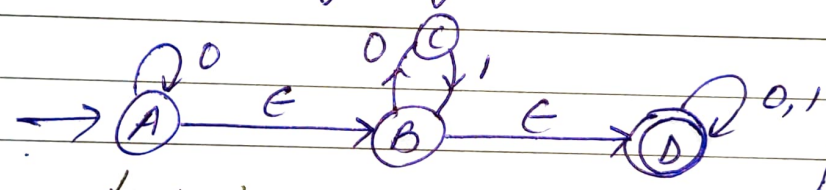
$T = \{ Q, \Sigma, q_0, f, \delta \}$

- Q → set of all states
- Σ → set of allowed symbols
- q_0 → initial state
- f → set of final state
- δ → Transition function

- We can also convert any ϵ-NFA to DFA, such as we can convert any NFA to DFA. (powers are same)

⇒ DFA ≡ NFA ≡ ϵ-NFA (we can receive similar language (L) for all three).

eg-



$\boxed{\Sigma = \{0, 1\} \cup \epsilon}$

⇒ from 'A' we can move from one state to another state ie 'B' without receiving any i/p symbol as we have 'ε symbol'. Similarly, we can go from B to D.

ϵ - closure (A) → A (can remain in same state without recieving i/p)
→ B
→ D

⇒ ϵ - closure (A) = {A, B, D}

ϵ - closure (B) = {B, D}

ϵ - closure (C) = {C}

ϵ - closure (D) = {D}