Theory of Computation and Compiler design (8-12 M)

DFA (Deterministic finite Automata)

Theory of Computation (1)

Symbol → a, b, A, B, 0, 1, @, ..

↓

"Σ" ↔ Alphabet – {a, b}, {a, b, c}, {0, 1}. (set of symbols)

↓

String → Sequence of symbols - a, b, aa, ab, abc.

If $|\Sigma| = K$ then no of strings of length n is

$$|\Sigma|^n$$

Ex. $|\Sigma| = 2$ $\Sigma = \{a, b\}$

If n = 2

a	a
a	b
b	a
b	b

$= 2^2 = 4$

Language – Collection of strings.

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

(1) L_1 = set of all strings of length 2

= { aa, ab, ba, bb }

(2) L_2 = set of all strings of length 3

= { aaa, aab, aba, baa, abb, bab, bba, bbb }

(3) L_3 = set of all strings where each string starts with a.

= { a, aa, ab, aaa, aab, aba, abb, .. }

L_1 and L_2 are finite but L_3 is infinite.

Powers of Σ

say $(\Sigma) = \{a, b\}$

$\Sigma^1 = \{a, b\}$ → set of all strings over Σ of length '1'

$\Sigma^2 = \Sigma \cdot \Sigma = \{a,b\} \cdot \{a,b\} = \{aa, ab, ba, bb\}$

$\Sigma^3 = \Sigma \cdot \Sigma \cdot \Sigma = \{aaa, aab \ldots\}$ $|\Sigma^3| = 8$

Σ^n = n length strings $|2^n|$

$\Sigma^0 = \{\varepsilon\}$ → Epsilon is a special symbol used to represent length '0'. ⇒ $|\varepsilon| = 0$

$\Sigma^* \rightarrow \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

$\Sigma^* = \{\varepsilon\} \cup \{a,b\} \cup \{aa, ab, ba, bb\} \cup \ldots$

It is infinite. set of all strings possible over {a,b} of all lengths.

Languages $\subseteq \Sigma^*$ → Each Language is subset of Σ^*

Language possible over Σ^* are infinite.

Ex → for C program Σ can be finite as

$\Sigma = \{a, b \ldots z, A, B, \ldots Z, 0, 1, \ldots 9, +, *, \ldots\}$

Program

```
void main() {
    int a, b;
    :
}
```

for C it is a program but for TOC it is string.

C- programming language → set of all 'valid' programs.

If a Language is containing a finit set of string or say finite language then it is easy to verify that given string(s) belongs to Language(L) or not

L is finite

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

$S = aaa$

We can say (s)is not present

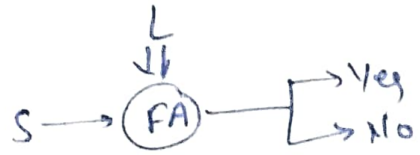
But if L is infinit then,

$L_2 = \{a, aa, aaa, ab, \ldots\}$

$S = baba$

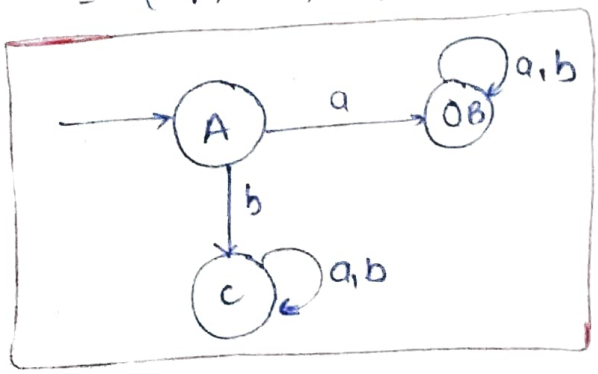
It may leads to infinite comparison.

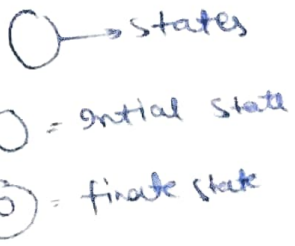
So, to make a language finite,

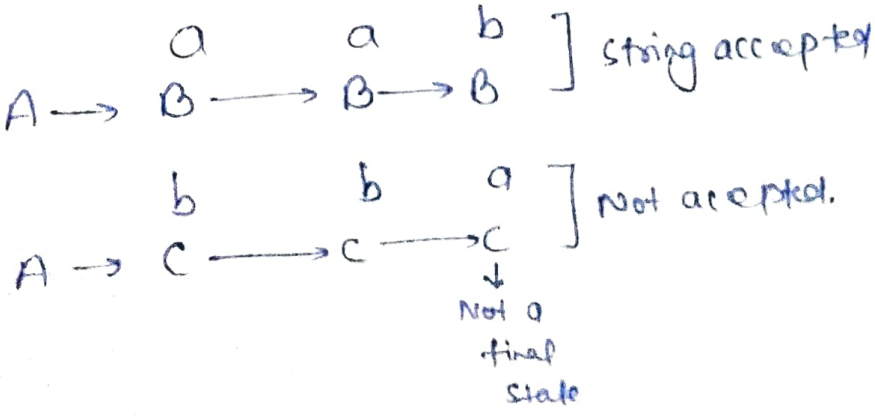


Ex L_1 = set of all strings which starts with 'a'

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





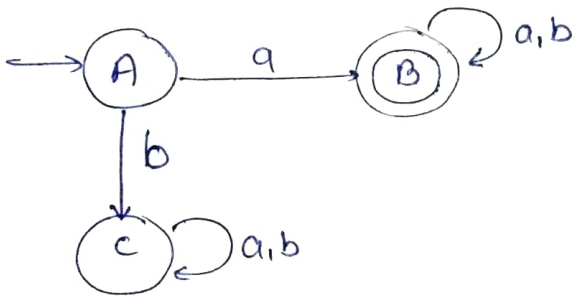


Finite Automata

- FA with Output
  - Moore
  - Mealy
- FA without output
  - DFA
  - NFA
  - ϵ-NFA

DFA → Deterministic finite Automata,

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



$Q = \{A, B, C\}$

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

$\delta =$

$q_0 = \{A\}$

$F = \{B\}$

$Q \rightarrow$ Finite set of states

$\Sigma \rightarrow$ Input alphabet

$q_0 \rightarrow$ start state

$F \rightarrow$ set of final states

$F \subseteq Q$

$\delta \rightarrow$ transition function

$\delta : Q \times \Sigma \rightarrow Q$

$\{A, B, C\} \times \{a\ b\} \rightarrow Q$

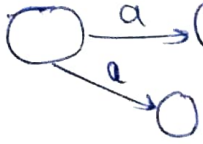
(A, a)
(A, b)
(B, a)
(B, b)
(C, a)
(C, b)

A
B
C

On a single input it will go for only one state q_0

[✓]

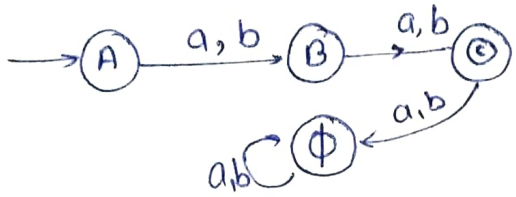
not



[x]

Q Construct a DFA accepts set of all strings over {a, b} of length 2

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



ϕ → Dead state or trap state

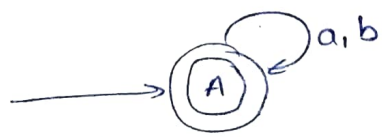
If on scanning the string we travel through starting state and reached at final state at end then string is called as accepted.

Acceptance of string :-

Acceptance of Language :- A FA is said to accept the Language if all the strings in the Language are accepted and all the strings not in Language are rejected by the final automata.

* Same state can be initial and final both.

Ex



↓

This automata will accept all the strings over {a, b} of all the length.

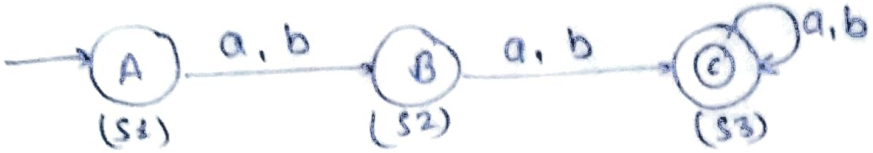
Construction of minimal DFA

Construct a DFA where $\Sigma = \{a, b\}$

(i) L = set of all strings length greather than equals to two.

(i) $w \in \{a, b\}$ $|w| \geq 2$

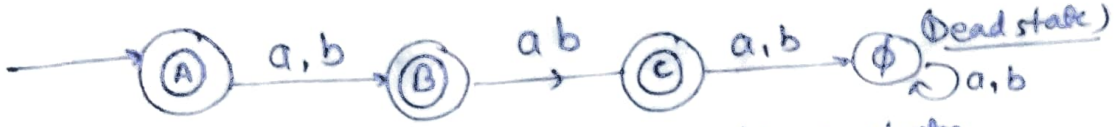
Infinite data language.



(ii)

$w \in \{a, b\}$ $|w| \leq 2$

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



To accept ε make initial state as final state

A language may contain many DFA's but it will have only one minimal DFA.

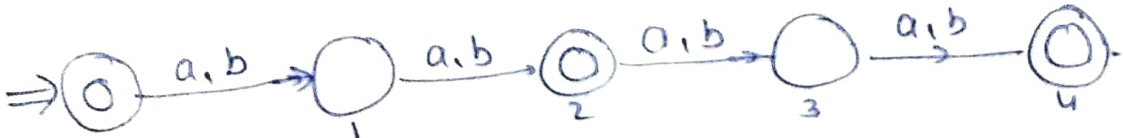
Imp

If $|w| = n$ $(n+2)$ $|w| \geq n$ $(n+1)$ $|w| \leq n$ $(n+2)$ → no of state require

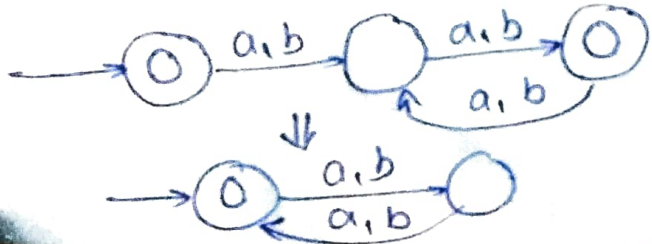
Construct a minimal DFA over $w \in \{a, b\}$ where $|w| \bmod 2 = 0$

$L = \{\varepsilon, aa, ab, ba, bb, aaaa \ldots bbbb \ldots\}$

Language is infinite

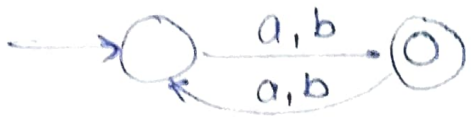


Above is infinte but it can be modified as



only accept the string of even length

$|w| \bmod 2 = 1$ | odd length string



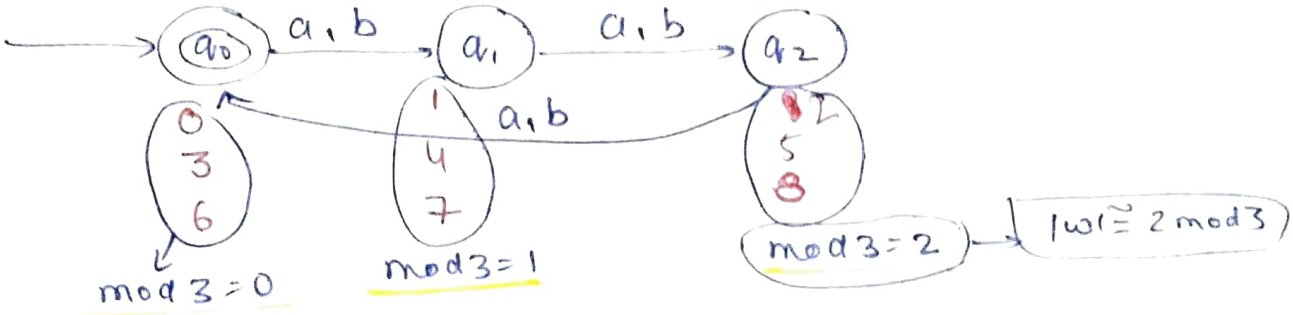
for a question in which $|w| \bmod k$ is ask then no. of states occur will be k. Imp

(iv) $w \in \{a,b\}$ $|w| \bmod 3 = 0$

$L = \{\varepsilon, aaa, aab \ldots bbb \quad aaaaaa \ldots\}$

↓

Infinite Language.



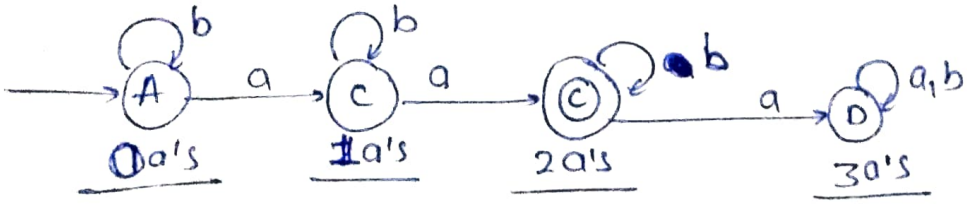
If $|w| \bmod 3 = 1$ then q_1 will be final state and if $|w| \bmod 3 = 2$ then q_2 = final state.

Imp

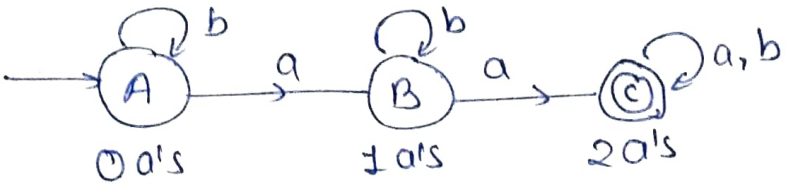
$|w| \bmod 3 = 1$ can be written as $|w| \cong 1 \bmod 3$

Not only for this particular one we can write multiple expression as this same.

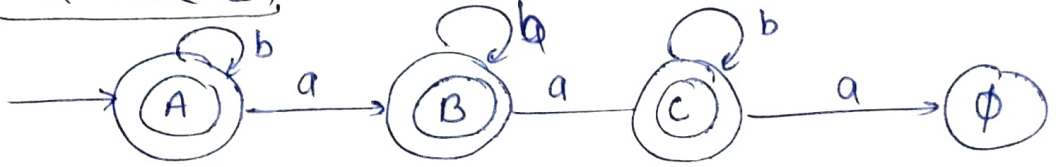
Construct a minimal DFA accepts $w \in \{a,b\}$ where no of 'a' should be two.



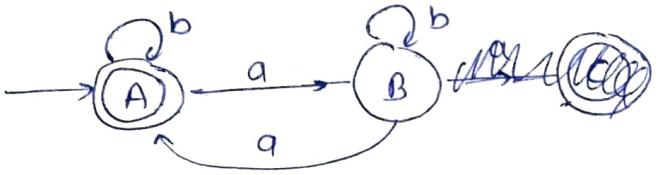
$n_a(w) \geq 2$



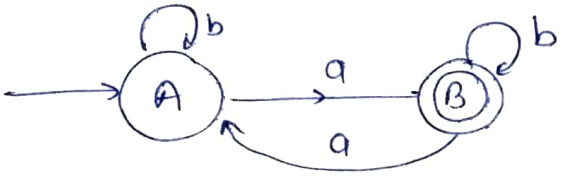
$n_a(w) < 2$



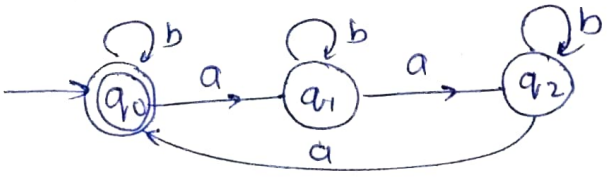
$|n_a| \bmod 2 = 0$ $\quad n_a(w) \cong 0 \bmod 2$



$|n_a| \bmod 2 = 1$



$|n_a| \bmod 3 = 0$



$(n_a)w \cong k \bmod N$ → It will require N state in DFA and k is some constant.

Construct a minimum DFA

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

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

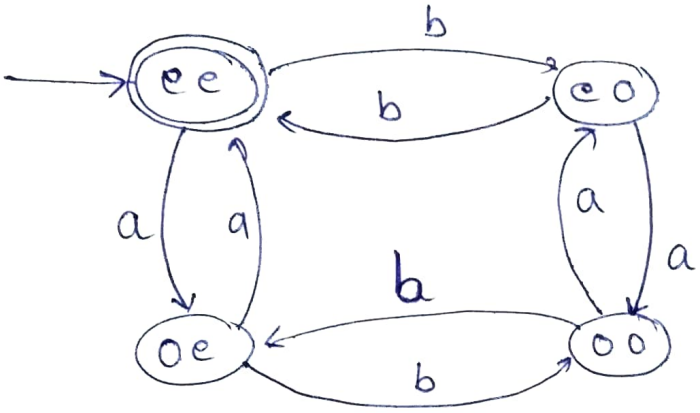
&&

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

$L = \{ \varepsilon, aabb, abab, bbaa, abba, \ldots \}$
aa, bb

State	$n_a(w)$	$n_b(w)$	
S_1	even	even	$\rightarrow \varepsilon, aa, bb$
S_2	even	odd	$\rightarrow aab$
S_3	odd	even	$\rightarrow abb$
S_4	odd	odd	$\rightarrow ab$

→ String like this so more strings are also possible



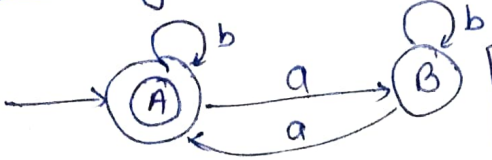
e → even
o → odd

Q Construct a DFA that accepts set of all strings over $\{a,b\}$ in which number of a's and number of b's are even.

Solution using cross product method.

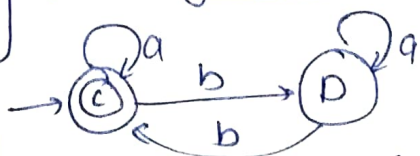
$w \in (a,b)^*$ $n_a(w) \cong 0 \bmod 2$ && $n_b(w) \cong 0 \bmod 2$

Counting a's ignoring b's

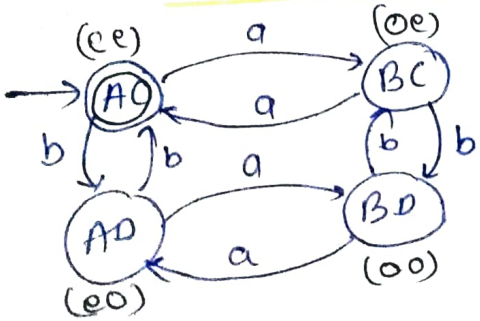


$\{A, B\}$

Counting b's ignoring a's



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



$A_c \xrightarrow{a} B_c$ | $B_c \xrightarrow{a} A_c$

$A_c \xrightarrow{b} A_D$ | $B_c \xrightarrow{b} B_D$

Similarly check for more transitions using DFA's.

If

a's		b's	final state
e	&&	e	(ee)
e	OR	e	{ee, eo, oe}

If one automata have 'n' number of states and second automata have 'm' number of states then no. of states in final automata will be 'nxm'.

Q Construct a minimal DFA which accepts set of all strings over {a, b} in which number of a's are divisible by 3 and number of b's are divisible by 2.

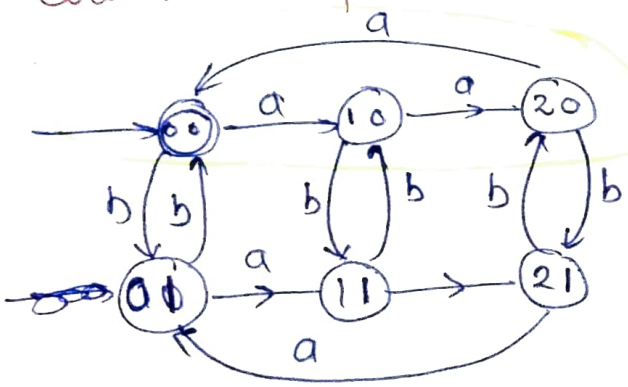
$w \in (a,b)^*$ $\quad n_a(w) \equiv 0 \bmod 3$ (m)

&&

$n_b(w) \equiv 0 \bmod 2$ (n)

$m \times n = 6$

Count no. of a's horizontally and no. of b vertically



x y

$x = n_a \bmod 3$

$y = n_b \bmod 2$

(i) If $n_a(w) \bmod 3 \geq n_b(w) \bmod 2$ then final states are

{00, 20, 10, 21, 11}

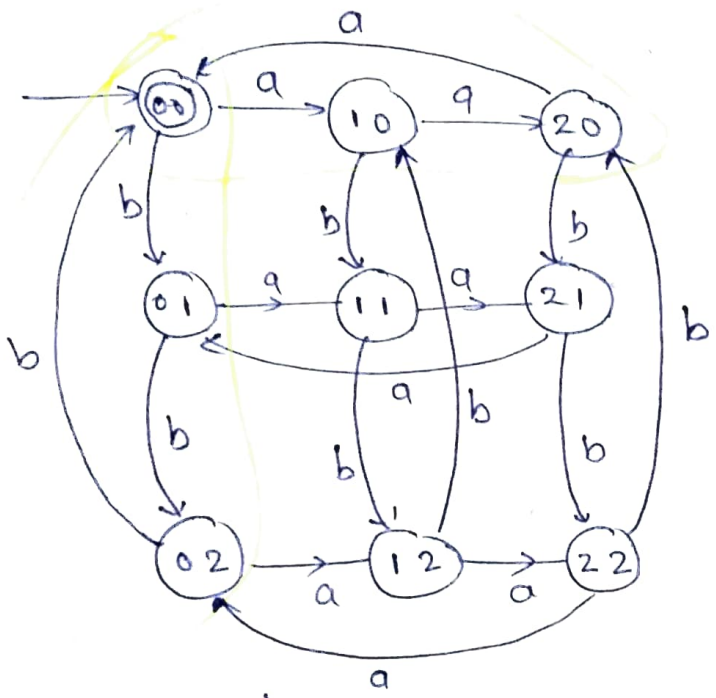
(ii) If $n_a(w) \bmod 3 = n_b(w) \bmod 2$

Final state = {00, 11}

Q Construct a minimal DFA which accepts set of all strings over {a, b} in which number of a's are divisible by 3 and no. of b's are divisible by 3.

$m \times n = 3 \times 3 = 9$

$w \in (a,b)^*$, $n_a(w) \cong 0 \bmod 3$ && $n_b(w) \cong 0 \bmod 3$



x y

$x \rightarrow n_a \bmod 3$

$y \rightarrow n_b \bmod 3$

i) $n_a(w) \bmod 3 = 1$ && $n_b(w) \bmod 3 = 2$

final state {12}

(ii) $n_a(w) \bmod 3 > n_b \bmod 3$

final states = {10, 20, 21}

(iii) $(n_a(w) > n_b(w)) \bmod 3$

Final states = {10, 20, 21}

Both are same but we can write in both ways due to modular arithmetic

$n_a(w) \cong 0 \bmod m$

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

→ $m \times n$ states in final resulting automata.

Construct a minimal DFA which accepts set of all strings over {a, b} in which $n_a \bmod 3 = 0$ and $n_b \bmod 3$ equals to 0.

$w \in (0,1)^*$ $L = \{$ Binary number divisible by 2 $\}$

Points to remember

Unary number system → 0

0 → $(1)_{10}$

00 → $(2)_{10}$

0...0 (n) → $(n)_{10}$

Decimal → $(0\text{-}9)_{10}$

Binary → $(0,1)_2$

Ternary → $(0,1,2)_3$

Hexadecimal → $(0 \cdots F)_{16}$

---

12 → 1x10+2

123 → 12x10+3

So, 1 2 3 4

1x10+2 = 12

12x10+3 = 123

123x10+4 = 1234

(1 0)

1x2+0 = 2

(1 0 1)

1x2+0 = 2

(2 1)

2x2+1 = 5

(1 0 1 1 0 1)

1x2+0 = 2

2x2+1 = 5

5x2+1 = 11

11x2+0 = 22

22x2+1 = 45

Multiply with base and add consecutively next successive digit in number

base 1 → 0 (1)

base 2 → 0, 1 (2)

⋮

base n → (0 ... n-1) (n)

Ex $(101)_3$

$1 \times 3 + 0 = 3$

$3 \times 3 + 1 = 10$

1234 can be evaluated as

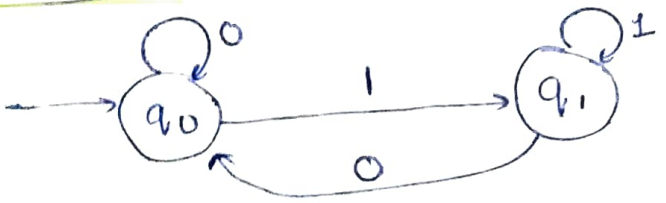
(123) x base + 4

(12 x base) + 3

(1 x base + 2)

It can be done Recursively

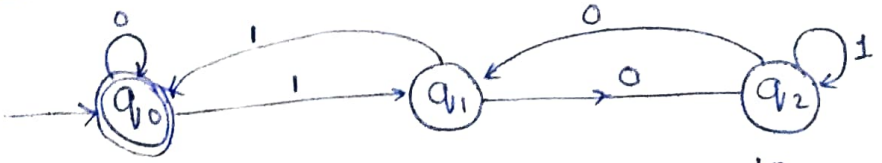
On devide any no by two we get two remainders (0, 1)



10010 → Scan the string from start

Binary Number divisible by 3

$\Sigma = \{0, 1\}$ Possible remainder → 0, 1, 2



0, 11, 110, ...

1, 100, ...

10, 101, ...

पहले देखा तो कौनसी state किस Remainder को denote कर रही है फिर उसके According Binary no. को लिख लो और DFA बनालो.

Transition Table

	0	1
→ *q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

DFA is also known as Transition diagram.

If $n \cong 1 \bmod 3$ then q_1 will be the final state.

* → final state
→ → initial state

In transition table q_0, q_1, q_2 comes successively one after another. You can make even a longer table using this.

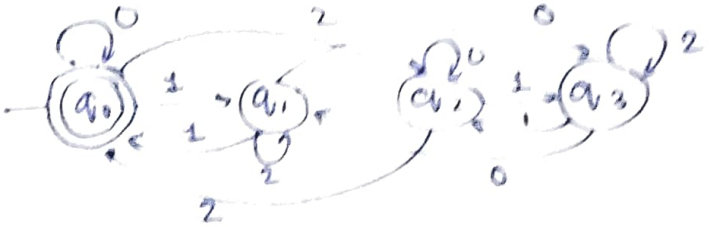
If there is $k \cong a \bmod(n)$ then there should be n states in DFA

DFA of no. divisible by 4

$\Sigma = \{0, 1, 2\}$ → Ternary number

	0	1	2
→ *q_0	q_0	q_1	q_2
q_1	q_3	q_0	q_1
q_2	q_2	q_3	q_0
q_3	q_1	q_2	q_3

q_0, q_1, q_2, q_3 in sequence



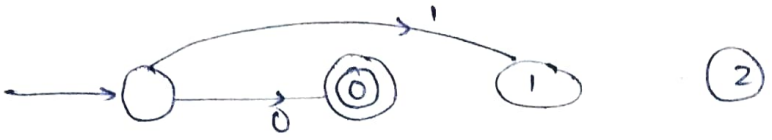
Question may extend upto base k number $\Sigma = \{0, 1, \ldots, k-1\}$ and number is divisible by n states. so have n states.

DFA of binary no divisible by n and epsilon is not in the language.

In question's solution if mod 3 is considered,

Here 0 is both initial and final state so Epsilon (ε) is consider as 0 here but if in

question it is given as ε should not be considered as zero then make a separate state as

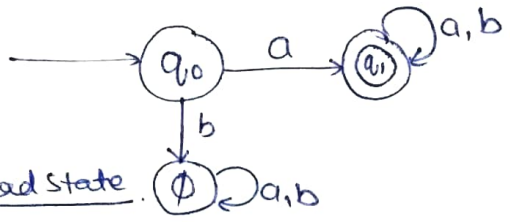


Here initial and final state are not same.

String which starts with 'a'.

$\Sigma = \{a, b\}$ L = starts with a

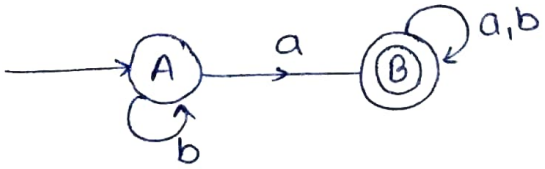
$= \{a, aa, ab, aaa \cdots\}$



Dead state is not a special state. It is a state from which we can never reach upto the final state.

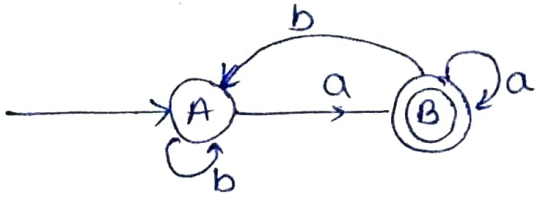
Strings which contains a

$\Sigma = \{a, b\}$ $L = \{a, ab, aa, aaa, aab\ aba, \cdots\}$



Strings ends with a

$\Sigma = \{a, b\}$ $L = \{a, ba, aa, aaa, aba, baa, \cdots\}$



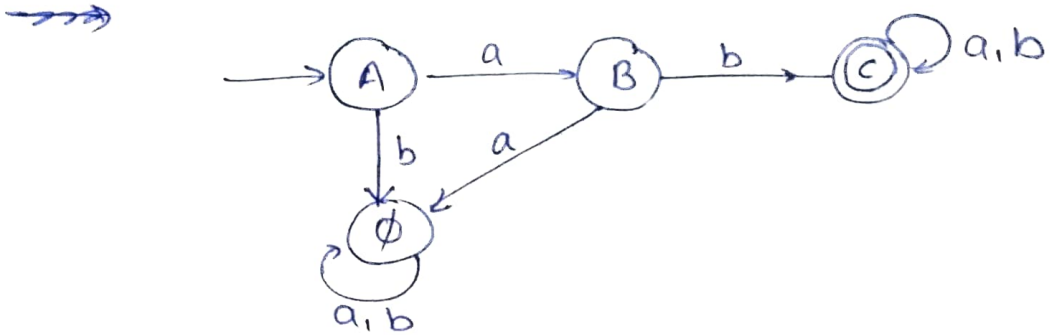
Comparison between different DFA's

PTO

Starts with a	Containing a	ends with a
We got a dead state. Initial state act as a traffic Police that allows strings starts with a	Draw DFA	Draw DFA

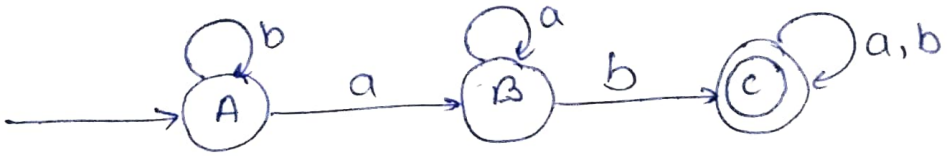
DFA of strings which starts with ab

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

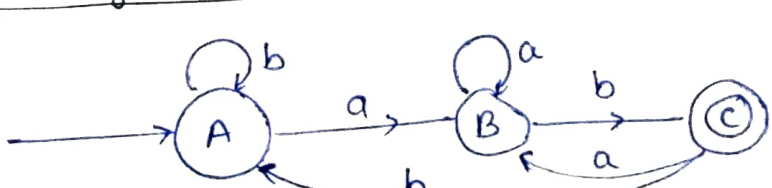


DFA of strings contains ab as substring

$\Sigma = \{a, b\}$ $L = \{ ab, aab, aba, bab, \ldots \}$



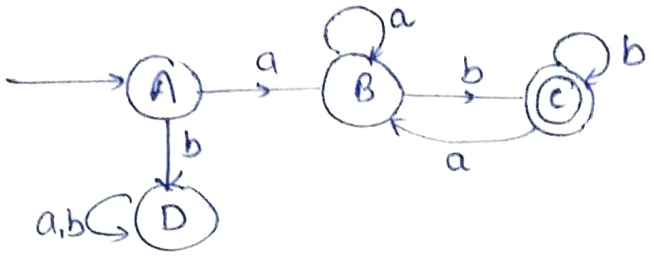
Strings ends with ab



$L = \{ ab, aab, bab, abab, \ldots \}$

DFA problem and concatenation of DFA

String start with a, ends with b.



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

$L_2 = \{ b, ab, bb, aab, \ldots \}$

$L_1 . L_2 = \{ ab, aab, abb, \ldots \}$

↳ Concatination

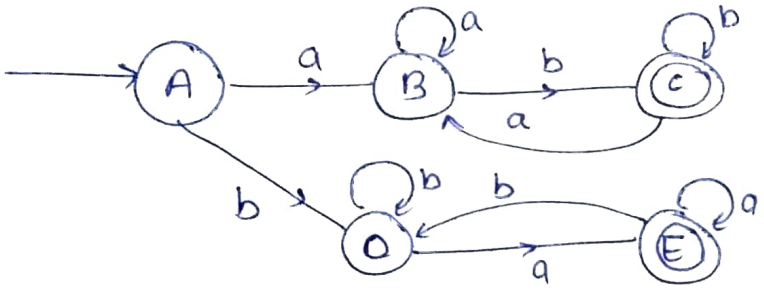
DFA accepts all the strings starts and ends with different symbol.

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

a ____ b

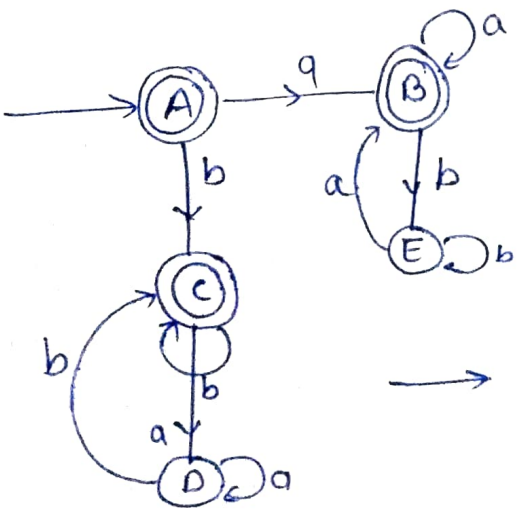
b ____ a

$L_1 = \{ ab, ba, aab, baa, abb, bba \ldots \}$



$L_1 = L_2'$

String starting and ends with same symbol



$L_2 = \{ \varepsilon, a, b, aa, bb, aba, aaa, bab, bbb, \ldots \}$

Should be consider.

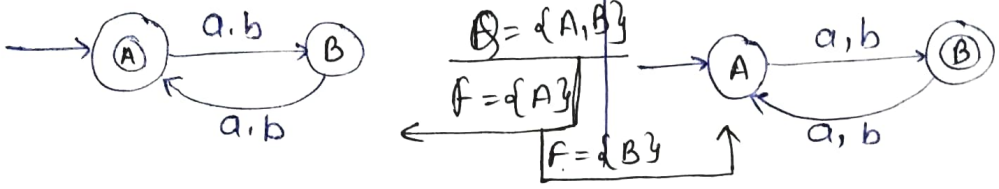
→ Since Language was complement of DFA discussed above so DFA (here) is having final states as complement to DFA is previous question

Two languages L_1 and L_2 are complement if

$$L_1 = \Sigma^* - L_2$$

Complementation of DFA

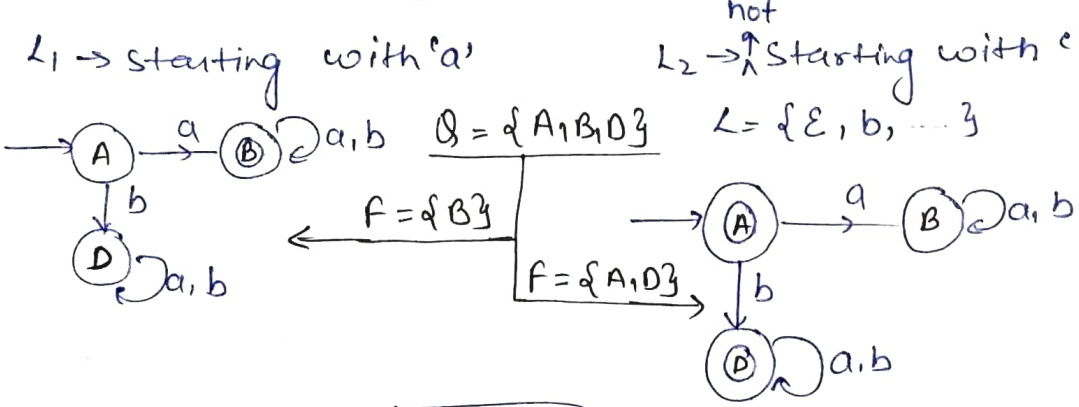
String of even length (L_1) | odd length string (L_2)



Both are complement. ($L_1 = \overline{L_2}$)

L_1 → starting with 'a'

L_2 → not starting with 'a'



$$L_1 = \overline{L_2}$$

If you make all non final states become final and final become non final over the 'same transitions' in a 'DFA' called as complement.

Condition for complements:-

i) Only DFA

ii) $(Q, \Sigma, \delta, q_0, f) \rightarrow$ DFA (L_1)

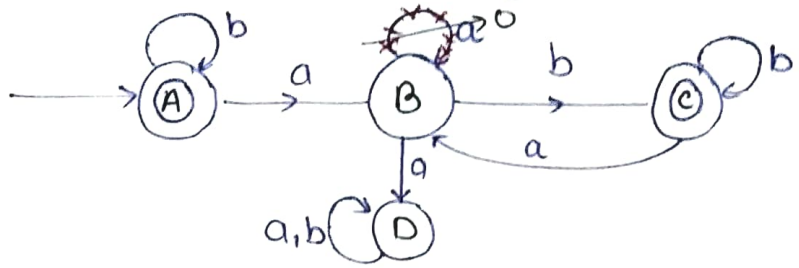
$$\downarrow\uparrow \quad \overline{f}$$

$(Q, \Sigma, \delta, q_0, Q-F) \rightarrow$ DFA (Complement)

DFA accepts strings in which every 'a' is followed by 'b'.

$w \in \{a, b\}^+$ L = Every 'a' should be followed by a 'b'.

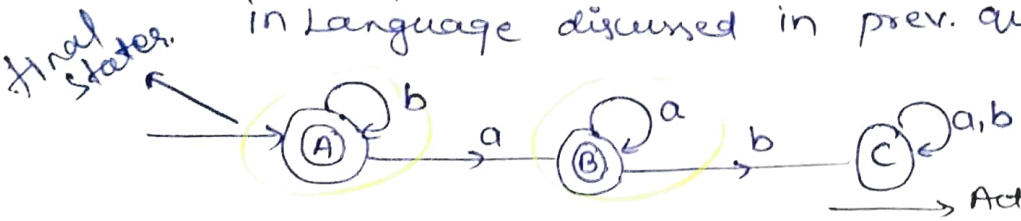
= { ab, abab, abbab, ..., b, bb }



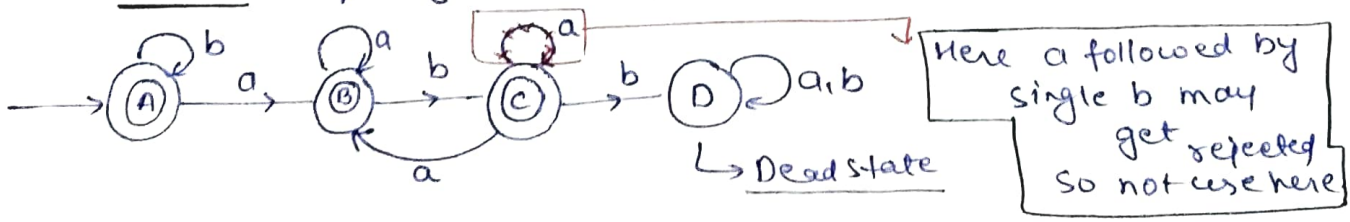
'a' is never followed by 'b'.

L = { ε, a, aa, aaa, ... b, bb, ... ba, bba ... }

Not complement due to presence of some intersections in Language discussed in prev. que.

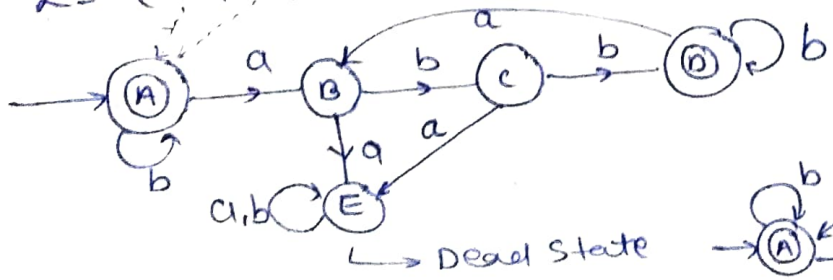


String in which every 'a' is never be followed by 'bb'. L = { ε, b, bb, bab, ab, bab, ... a, a, aa, ... }

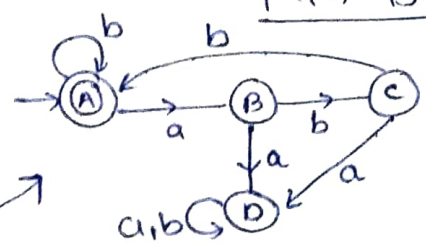


Every 'a' has to be followed by 'bb'.

L = { ε, b, bb, ..., abb, bbabb, ... }



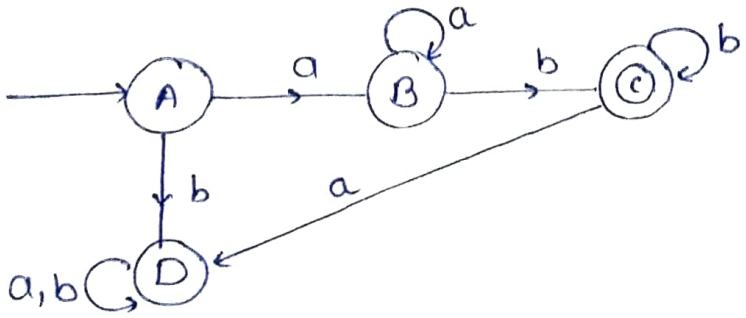
Min DFA



Both are correct but min. DFA is

Ex: $L = \{ w = a^n b^m : n, m \geq 1 \}$

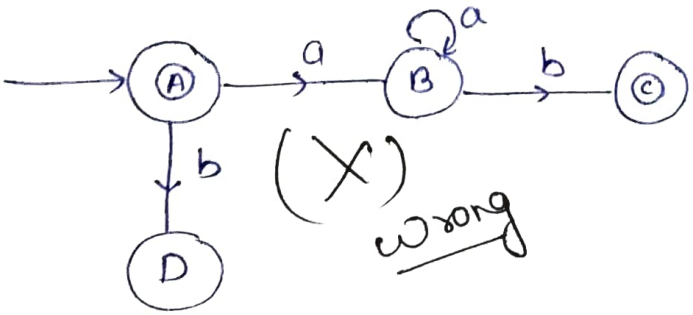
$= \{ ab, aab, abb, aabb, aaab, aabbb, \ldots \}$

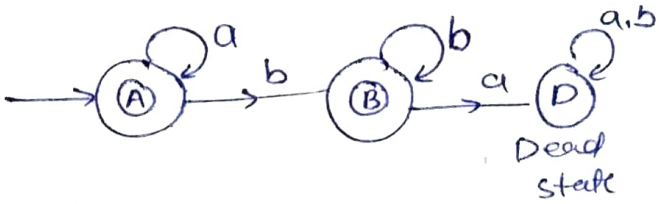


Here, is not allowed 'a' after 'b'.

Ex: $L = \{ w = a^n b^m : n, m \geq 0 \}$

$= \{ \varepsilon, ab, aab, abb, \ldots, aa, bbb \}$

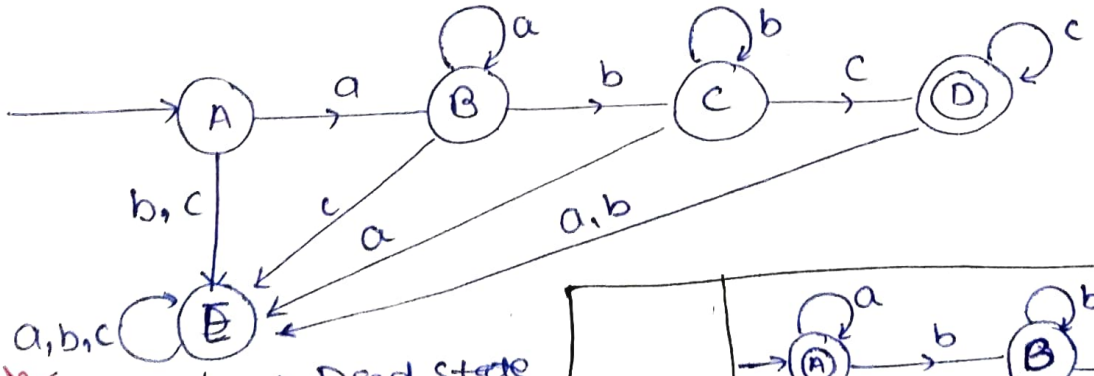




(✓) Right

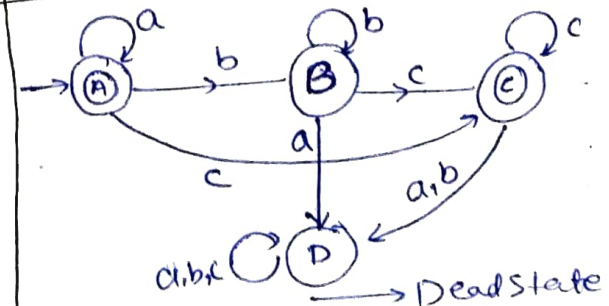
$L = \{ w = a^n b^m c^l : m, n, l \geq 1 \}$

$= \{ abc, aabc, abbc, abcc, \ldots \}$



Ex: $L = \{ w = a^n b^m c^l : m, n, l \geq 0 \}$

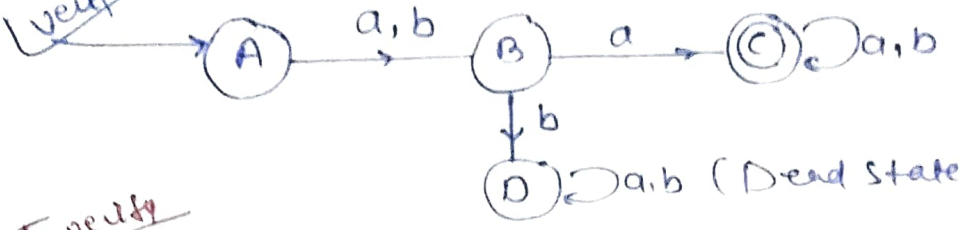
$= \{ \varepsilon, a, aa \ldots b, bb \ldots c, cc \ldots$
$abc, aabc \ldots \}$



L → Strings that contains second symbol from LHS as 'a'.

Towards verified

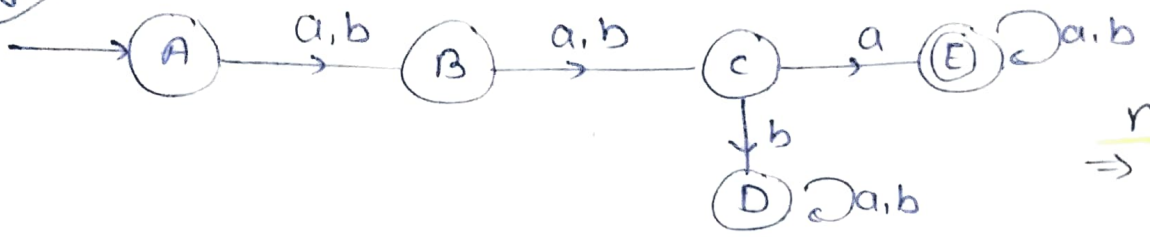
$w \in \{a,b\}^*$, $L = \{aa, ba, aaa, bab, baa$



$(n+2)$ $(n=2)$
$= 4$ states

To verify

verified 2nd symbol from LHS is 'a'.



$n+2$ $(n=3)$
$\Rightarrow$ 5 states

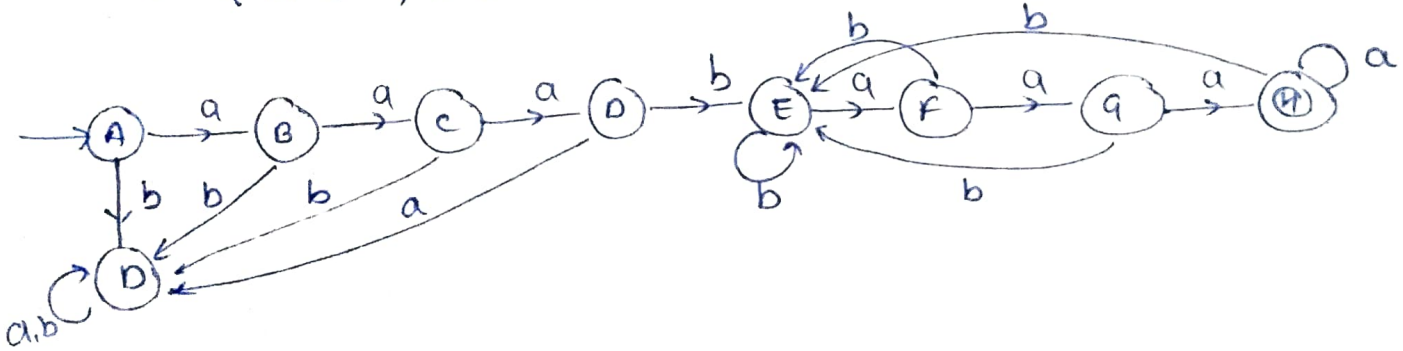
If it is accepting n^{th} symbol from LHS is either a or b then no of states must be

$\boxed{n+2}$.

Strings over $\{a,b\}$ of form a^3bwa^3 where w is any string over $\{a,b\}$.

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

$L = \{a^3ba^3, a^3baa^3, a^3bba^3, a^3baaa^3, \ldots\}$



Operations on DFA

Union → Starts and end with different symbols.

Concatenation → Starts with 'a' ends with 'b'.

Cross Product → even no of 'a' and even no of 'b's'.

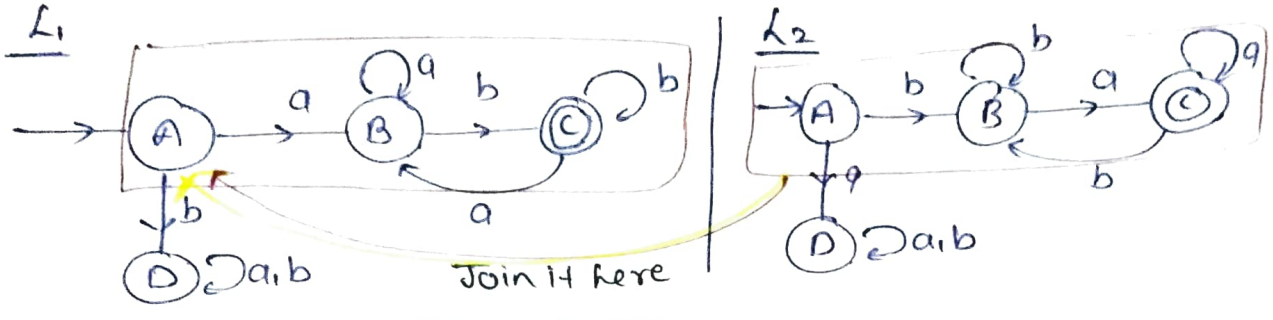
Complement → Does not contain 'a'.

Reversal → awb → bwa

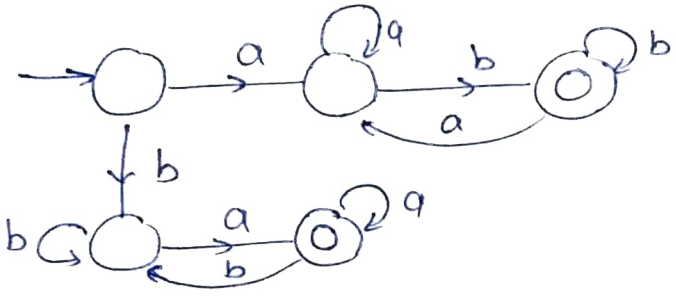
Ex Union $\Sigma = \{a, b\}$

$L_1 = \{ab, aab, abb, \ldots\}$ Case 1

$L_2 = \{ba, baa, bba, \ldots\}$ Case 2



$L_1 \cup L_2$



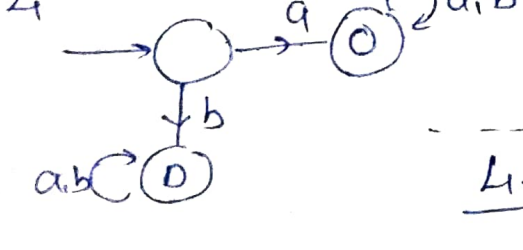
On Union join part of 2nd DFA in initial state of 1st DFA

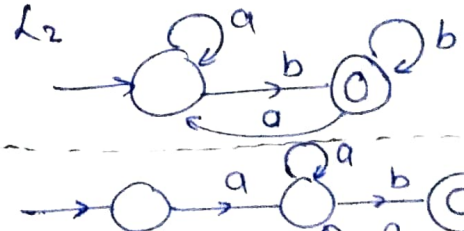
Ex Concatenation

$L_1 = \{a, aa, ab, aaa \ldots\}$, $L_2 = \{b, ab, bb, aab, \ldots\}$

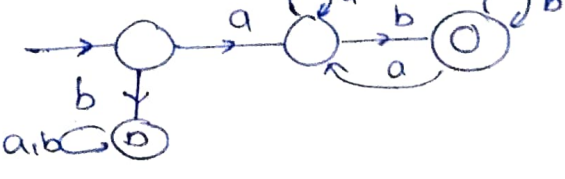
$L_1 \cdot L_2 = \{ab, aab, abb \ldots\}$

Take a string from L_1 and concatenate it with strings of L_2. Do the same with all strings of L_1 and remove common strings from $L_1 \cdot L_2$.





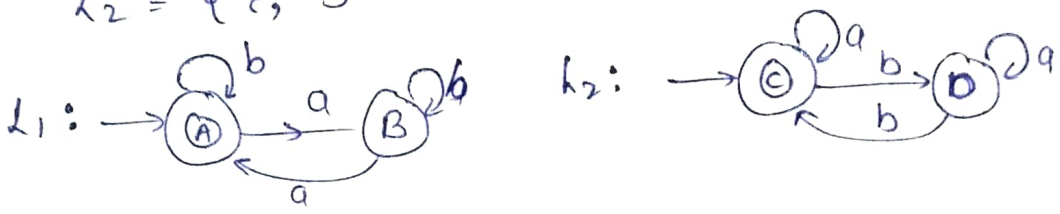
$L_1 \cdot L_2$



Cross Product

$L_1 = \{ \varepsilon, aa, baa, \dots \}$

$L_2 = \{ \varepsilon, bb, bba, abb, bbbb, \dots \}$



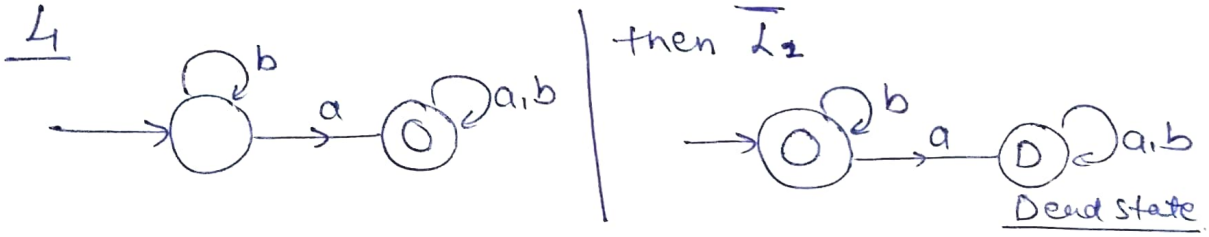
Discussed previously in solving question.

Complement (interchange final and non final states)

$L_4 = \{$ containing a $\}$

$= \{ a, aa, ab, aaa, bab, \dots \}$

$\overline{L_4} = \{ \varepsilon, b, bb, bbb \dots \}$



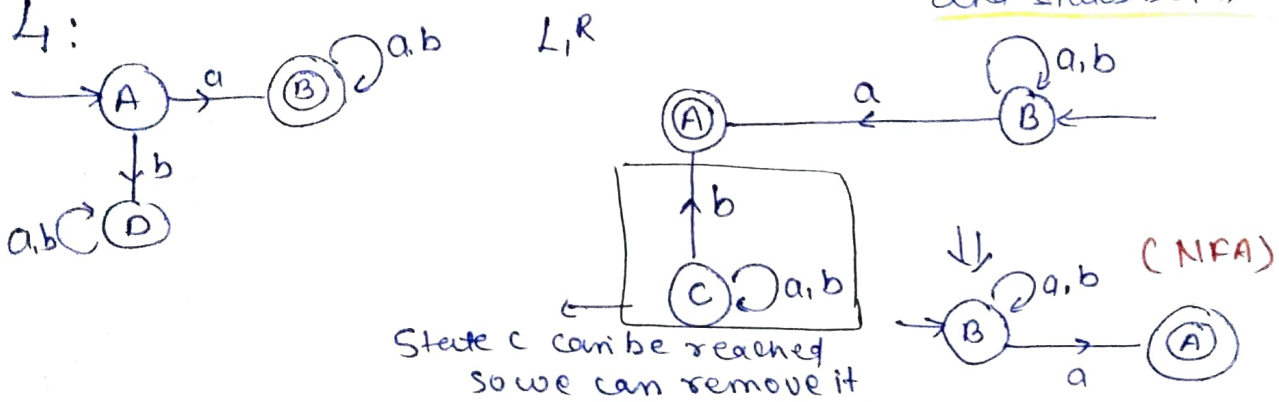
Reversal

$L_4 = \{$ Starts with a $\}$

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

$L_2 = L_4^R = \{ a, aa, ba, aaa, baa, aba, \dots \}$

$L_4^R = \{$ Ending with a $\}$ → Reverse the arrow and and states both.



Reversal of Language may give DFA or NFA in result.

Steps of reversal

i) Interchange initial and final state
ii) Reverse the direction of arrows and leave the loops as it is.
iii) Remove all the non reachable states.

Construct a minimal DFA over {a}

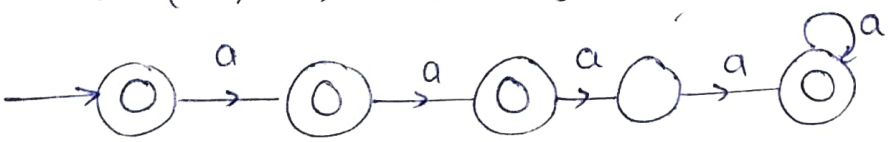
1. For $\{a^n : n \geq 0, n! = 3\}$
2. For $\{a^n : n \geq 0, n! = 2, n! = 4\}$

1. $\Sigma = \{a\}$

Σ^*

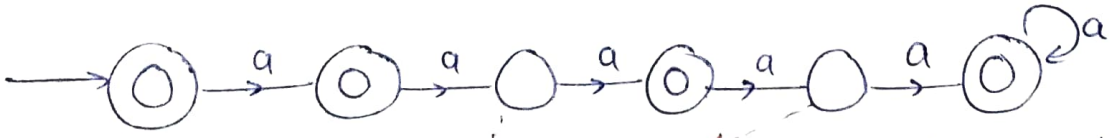
L	$\overline{L}$

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



[a^3 is not accepted]

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



[a^2 and a^4 are not accepted]

No. of 2 state DFA with a designated initial and final state over {a, b}.

	a	b
→ X	X/Y	X/Y
* Y	X/Y	X/Y

Once you make X as intial state and Y as final state then you can't reverse it

Total $2 \times 2 \times 2 \times 2 = 16$ possibilites.

If designated final state is not given.

	a	b
→X	(16)	
Y		

	a	b
→*X	(16)	
*Y		

	a	b
→X	(16)	
*Y		

	a	b
→*X	(16)	
Y		

It may possible that there may be no final state, finale state may be X, final state may be Y, final state may be both X and Y

It may leads to 64 different DFA's.

No. of 2 state DFA's with designated initial state accepting the empty language over {a,b}.

	a	b
→X		
Y		

Empty Language = { } or ϕ

Epsilon (ε) is not a empty language. It is an empty string.

If initial state and final state is same then it will accept ε but we have an empty language. So we can't do this. Hence we have two choices

i) Y as final state

	a	b
→X	X	X
*Y	X/Y	X/Y

We don't want to reach Y to make a transition so total

$2 \times 2 = 4$

Ans = 16 + 4 = 20

ii) Y without final state

	a	b
→X	X/Y	X/Y
Y	X/Y	X/Y

(16) DFA's possible because there is no final state so Language is accepting nothing hence empty Language