Finite state machine:

symbol – a, b, c, 0, 1, 2 . . .

Alphabet – Σ

collection of symbols : {a, b}, {0, 1, 2} . .

string – sequence of symbols : a, b, aa, 01, 0 . .

Language – set of strings : If $\Sigma = \{0, 1\}$

Ex: L_1 = set of all strings of length 3

$= \{000, 001, 010, \ldots\ldots 111\}$

↓ Finite

L_2 = set of all strings that begin with 0

$= \{0, 00, 01 \ldots\ldots\}$ → infinite

Powers of Σ : suppose, $\Sigma = \{0, 1\}$

Σ^0 = set of all strings of length 0 $= \{\varepsilon\}$

Σ^2 = " " " length 2 $= \{00, 01, 10, 11\}$

Σ^n = " " length n

Cardinality: number of elements in a set

$$\Sigma^n = 2^n$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \dots$$

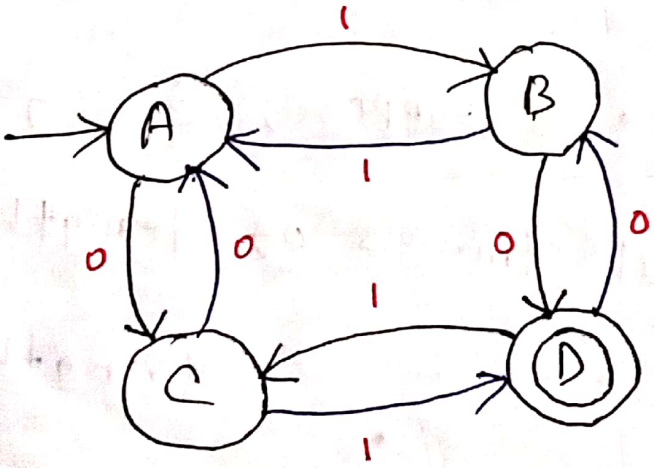
$$= \{\Sigma\} \cup \{0, 1\} \cup \{ \dots \}$$

= set of all possible strings of all lengths over $\{0, 1\}$ → infinite

Deterministic finite Automata (DFA)

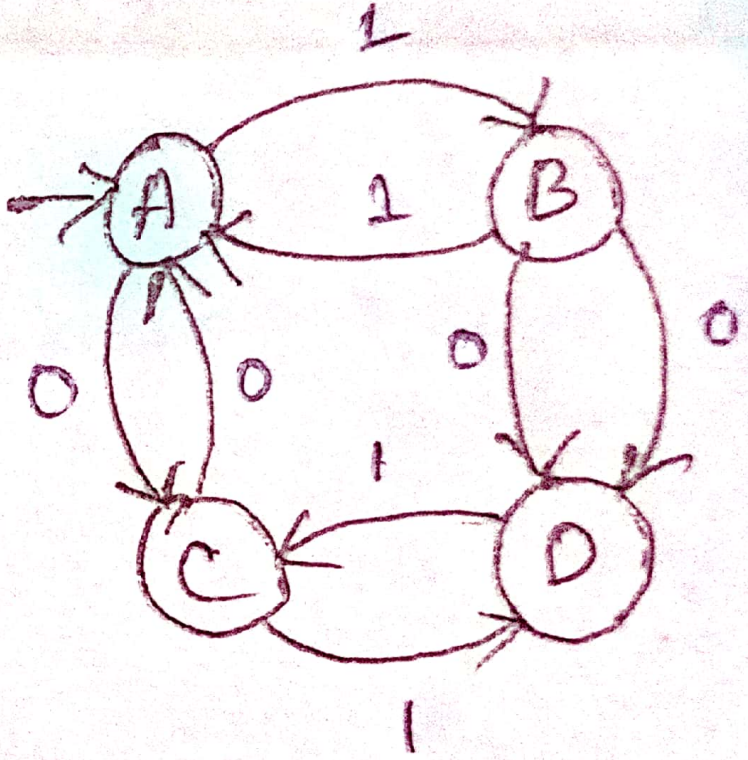
- simplest model of computation
- has very limited memory

Ex:



DFA:

- given current state
- we know what the next state is
- no choice / randomness
- simple / easy to design
- only one unique next state



Q - set of all states Eg: - $\{A, B, C, D\}$

Σ - inputs e.g: $\{0, 1\}$

q_0 - start state e.g: A

F - set of final state e.g: $\{D\}$

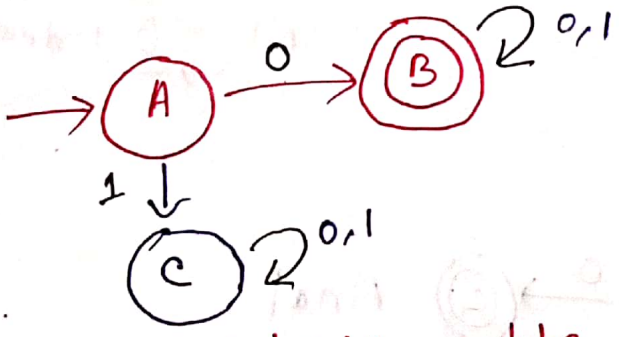
δ - transition function from $Q \times \Sigma \rightarrow Q$

e.g:

	0	1
A	C	B
B	D	A
C	A	D
D	B	C

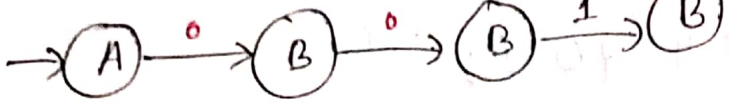
Example: (1) L_1 = set of all strings that start with '0'

$= \{0, 00, 01, 010, \ldots\}$

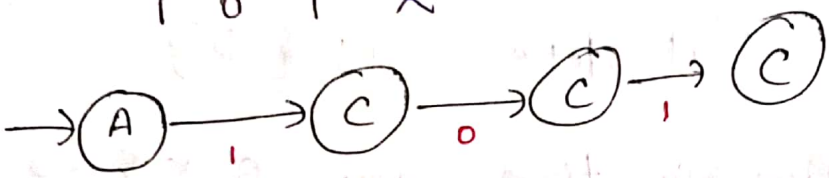


dead state / trap state

Eg: 0 0 1 ✓



1 0 1 ✗

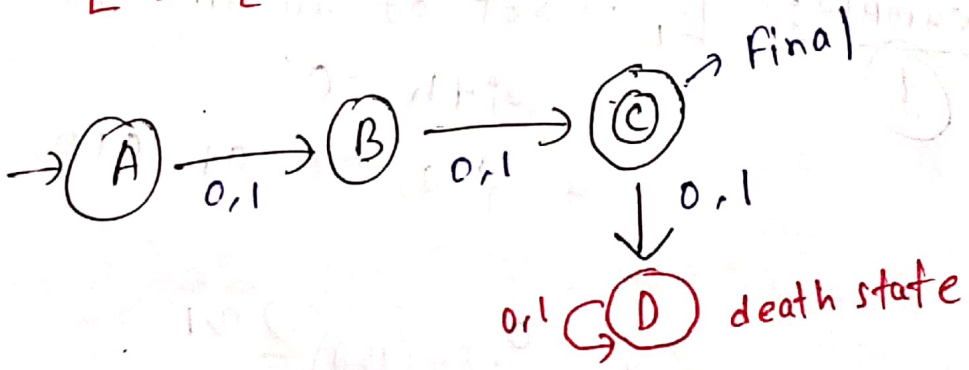


Example: (2) Construct a DFA that accepts sets of all strings over {0, 1} of length 2.

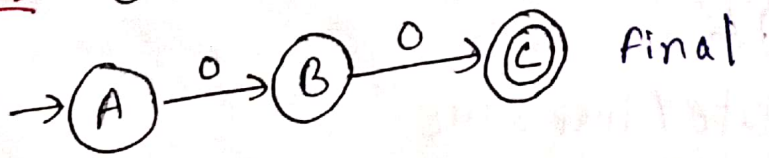
$\Sigma = \{0, 1\}$

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

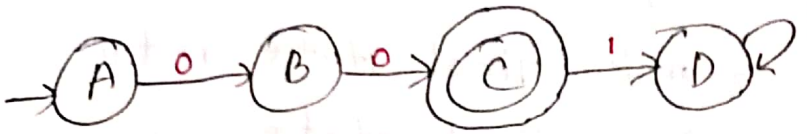
Ans:



Eg: 0 0 ✓

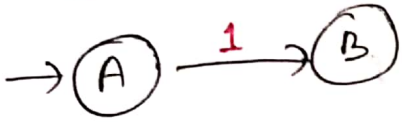


00 11 X



not final

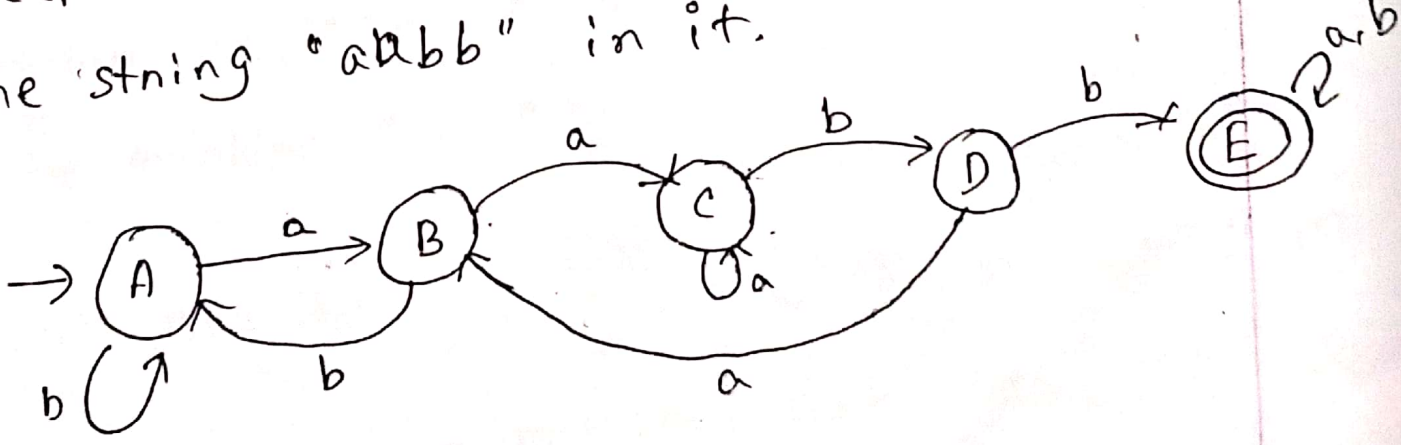
1 X



not final

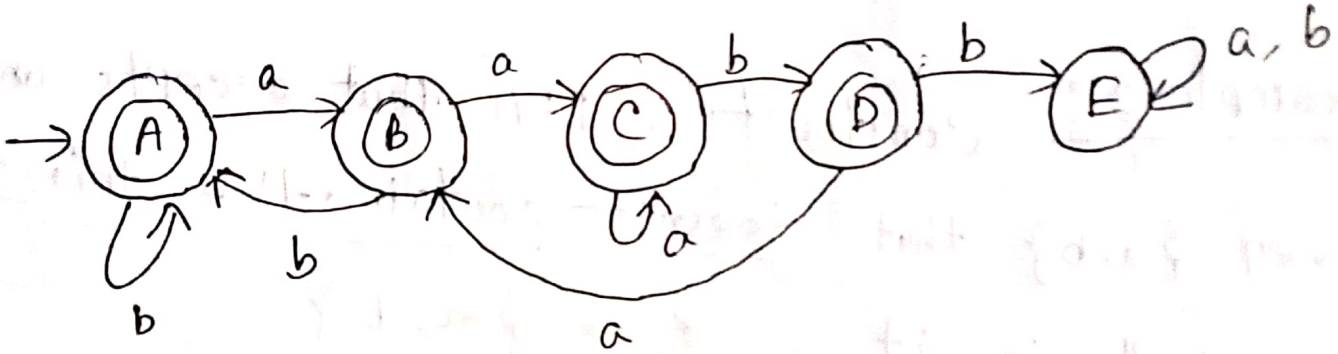
Example 3: Construct a DFA that accepts any strings over {a, b} that doesn't contain the string "aabb" in it. $\Sigma = \{a, b\}$

Solution: At first, we'll construct a DFA that accepts all strings over {a, b} that contains the string "aabb" in it.

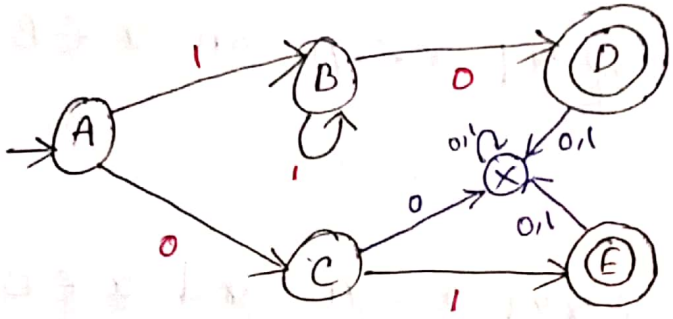


2nd step: now - flip the states
- make final state into non-final state
- make non-final states into final states

So, the answer is:



figuring out what a DFA recognizes?



accepts : 1) 0 1

2) 1 1 1 ··· 1 0
A B D

L = { accepts the string 01 on a string of at least one '1' followed by a '0' }

* Regular Language:

A language is said to be a regular language if and only if some finite state machine recognizes it.

not regular :
- not recognized by any FSM
- requires memory

* operations on regular languages:

Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

concatenation:

$A \circ B = \{xy \mid x \in A \text{ and } x \in B\}$

Star: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and } x_i \in A\}$

Eg: $A = \{pq, r\}$, $B = \{t, uv\}$

$A \cup B = \{pq, r, t, uv\}$

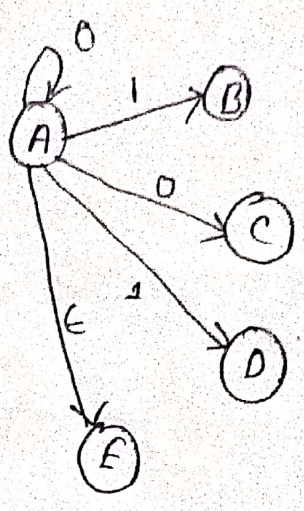
$A \circ B = \{pqt, pquv, rt, ruv\}$

$A^* = \{\epsilon, pq, r, pqr, rpq, rr, pqpq, rrr \ldots\}$

N.B: The class of regular languages is closed under union and concatenation.

Non-deterministic finite Automata (NFA)

- given the current state
- multiple next states
- next state may be random
- all next states may be chosen in parallel



* Here,
  ϵ – empty string
  (not found in DFA)

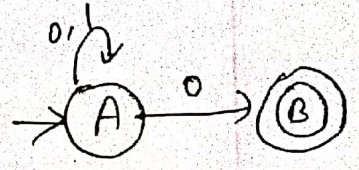
Q – set of all states – {A, B}

Σ – inputs – {0, 1}

q_0 – initial state – A

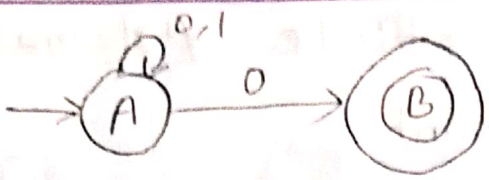
f – set of final states – {B}

δ – $Q \times \Sigma \rightarrow 2^Q$



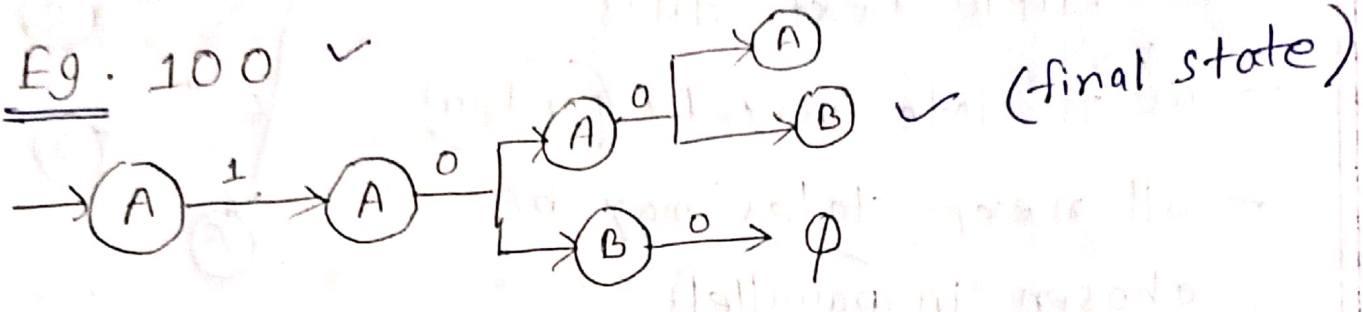
Explanation: $A \times 0 \rightarrow A$, $A \times 0 \rightarrow B$
$A \times 1 \rightarrow A$, $B \times 0,1 \rightarrow \emptyset$

Ex 1:



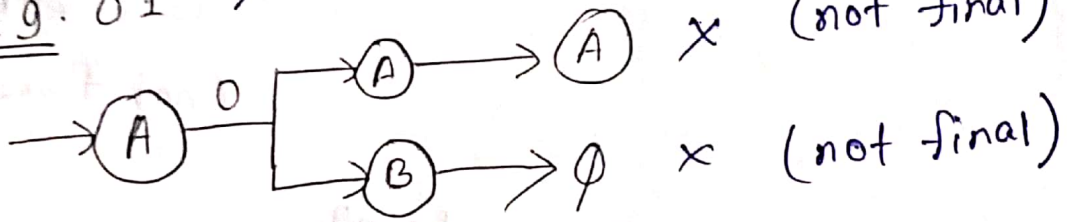
L = { set of all strings that end with 0 }

Eg. 100 ✓



✓ (final state)

Eg. 01 ✗



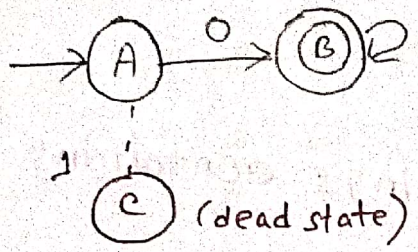
✗ (not final)

✗ (not final)

N.B: If there is any way to run the machine that ends in any set of states out of which at least one state is a final state, then the NFA accepts the string.

Ex 2: L = {set of all strings that stat with 0}

= {0, 00, 01, 010, 0000 . . .}

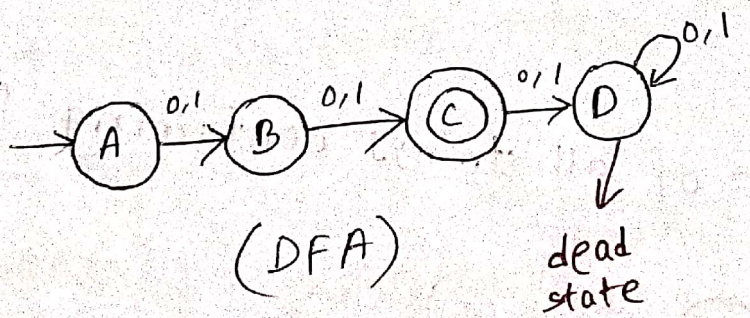


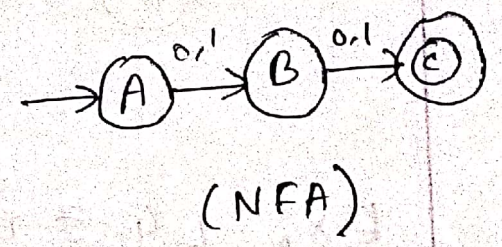
* In case of NFA, dead state doesn't need to be mentioned.

Ex 3: Construct a NFA that accepts sets of all strings over {0, 1} of length 2.

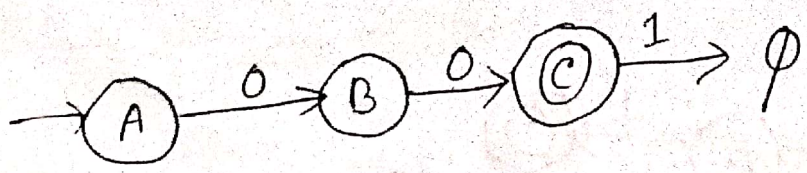
$\Sigma = \{0, 1\}$

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



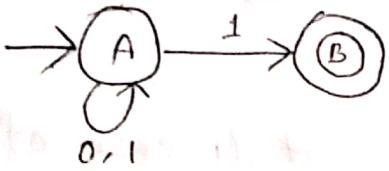


Eg. 001

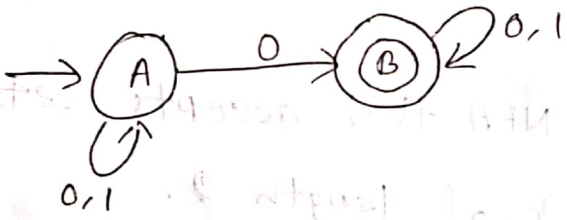


(not final, not accepted)

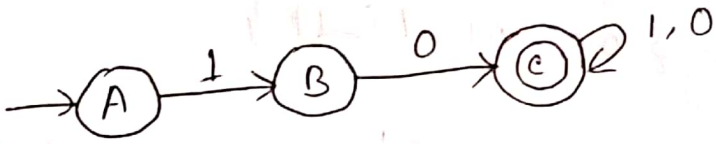
Ex: L1 = {set of all strings ending with '1'}



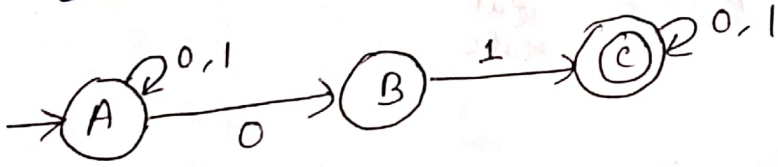
Ex: L2 = {set of all strings containing '0'}



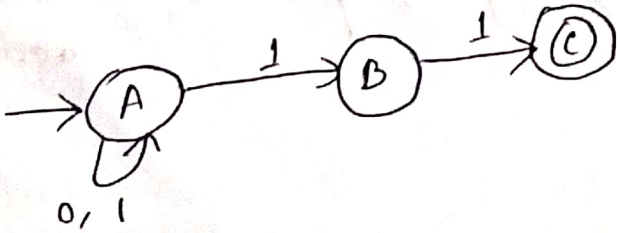
Ex: L3 = {set of all strings starting with '10'}



Ex: L4 = {set of all strings containing '01'}



Ex: L5 = {set of all strings ending with '11'}



Convension of NFA to DFA

N.B: Every DFA is an NFA, but not every NFA is a DFA. But there is an equivalent DFA for every NFA.

Explaination: DFA : $\delta = Q \times \Sigma \rightarrow Q$

NFA : $\delta = Q \times \Sigma \rightarrow 2^Q$

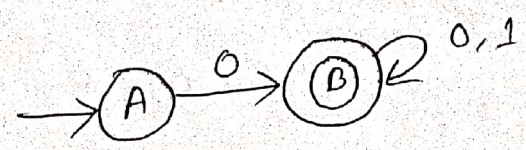
Q also included in 2^Q

So, DFA is included in NFA

** L = {set of strings over (0,1) starting with 0}

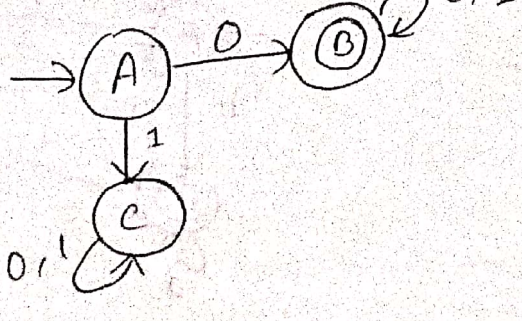
$\Sigma = \{0, 1\}$

NFA:



	0	1
A	B	ϕ
B	B	B

DFA:



	0	1
A	B	C
B	B	B
C	C	C

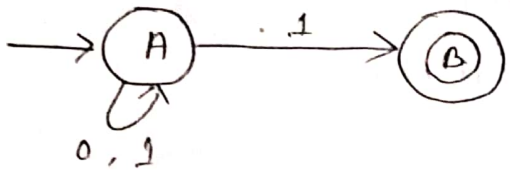
↓ Dead state

(subset construction method)

Ex : 1

L = { set of strings ending with 1 }

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



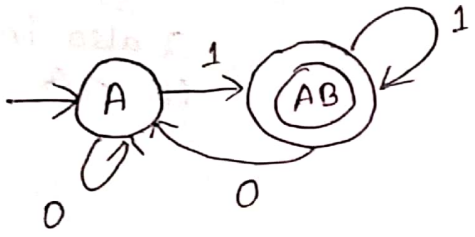
(NFA) →

	0	1
A	{A}	{A, B}
B	ϕ	ϕ

↓

	0	1
A	{A}	{AB}
AB	{A}	{AB}

AB – single state

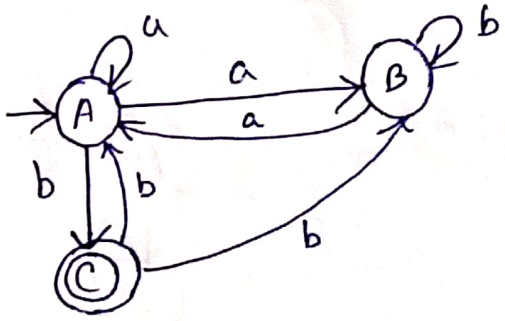


← (DFA)

Ex 2: Find the equivalent DFA for the NFA given by $M = [\{A, B, C\}, (a, b), \delta, A, \{C\}]$

where δ is given by :

	a	b
→ A	A, B	C
B	A	B
(C)	ϕ	A, B

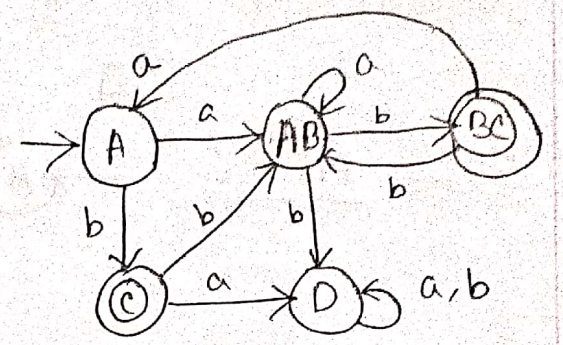


DFA

	a	b
→ A	AB	C
AB	AB	BC
BC	A	AB
C	D	AB
D	D	D

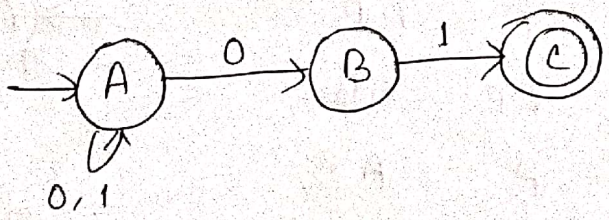
Final: BC, C

* NFA তে C final state ছিল: DFA তে যেই state গুলোতে C আছে, এগুলোই final state হবে।



Ex 3: Given below is the NFA for a language. L = {set of all strings over (0,1) that ends with '01'} Construct its eq. DFA.

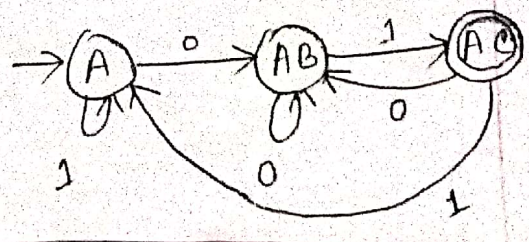
NFA:



	0	1
→ A	A,B	A
B	ϕ	C
C	ϕ	ϕ

DFA:

	0	1
→ A	AB	A
AB	AB	AC
AC	AB	A

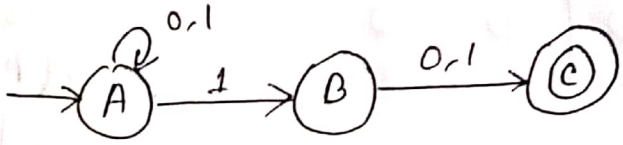


* In NFA → 2 states are possible
* In DFA → (2) (More than 1 state) states are not possible

Ex 4: Design an NFA for a language that accepts all strings over {0,1} in which the second last symbol is always '1'. then convert it to its eq. DFA.

Ans:

NFA:



(state diagram)

	0	1
A	A	A,B
B	C	C
(C)	∅	∅

(transition table)

DFA:

	0	1
A	A	AB
AB	AC	ABC
(AC)	A	AB
(ABC)	AC	ABC