ect-33

NFA (Non deterministic finite Automata)

Difference between DFA and NFA

If we are starting from a state, on seeing some input, we will go to some particular state. (DFA)

$$q_1 \xrightarrow{w} q_2$$

If we are starting from a state, on seeing some input, we might end up at one state or many states or may end up at nothing. (NFA)

$$q \xrightarrow{w} q_1$$
$$q \longrightarrow q_2$$
$$\vdots$$
$$q \longrightarrow q_n$$
$$q \longrightarrow \phi$$

NFA is a 5 tuple set which contains

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

where Q = set of ~~sets~~ states (Finite)

Σ = input alphabet (Finite)

q_0 = start state

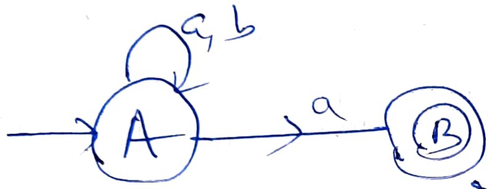
$F \rightarrow$ set of final state

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

Ex Q Construct a NFA which accepts the language in which every string ends with 'a' over the input alphabet $\Sigma = \{a, b\}$

Ans $L = \{\text{ends with } a\}$, $L = \{a, aa, ba, \ldots\}$

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

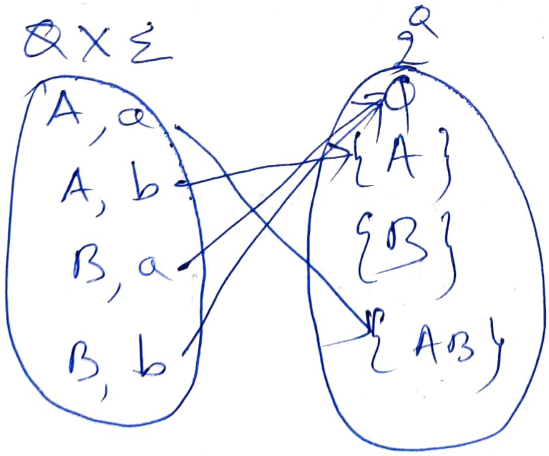


$A \xrightarrow{a} A$

$A \xrightarrow{a} B$

In NFA, on one state, on seeing one input we are not going to exactly one state here, we might go to more than one state.

$Q = \{A, B\}$ $\Sigma = \{a, b\}$



In NFA

$\boxed{\delta : Q \times \Sigma \rightarrow 2^Q}$

$\boxed{\text{In DFA} \quad \delta : Q \times \Sigma \rightarrow Q}$

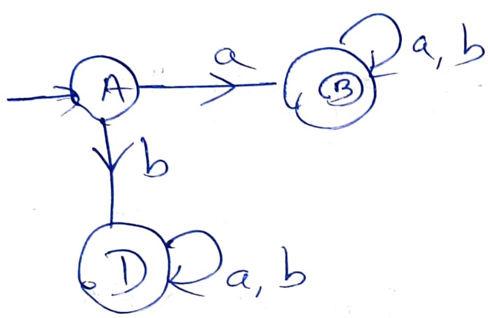
Every DFA is NFA, not viceversa

Lect-34 Construct a NFA which accepts the language in which every string starts with 'a' over the input alphabet {a,b}

L_1 = { every string starts with 'a' }

L = {a, aa, ab, aaa, abb - - - - }

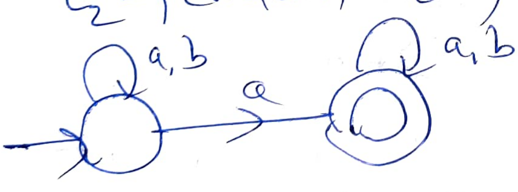
In DFA,



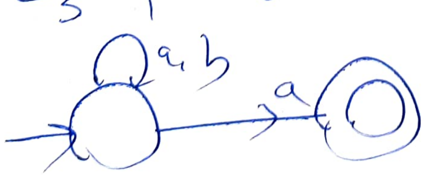
In DFA we have dead state. In NFA we do not have dead state. we have dead configuration.

Dead configuration

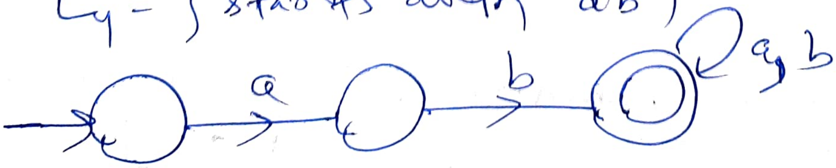
Q L_2 = { contain 'a' }



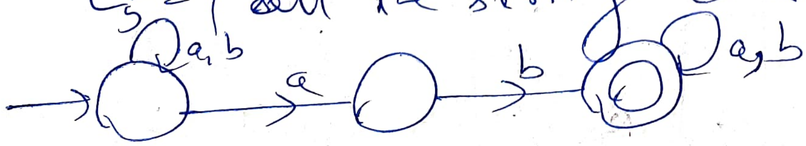
Q L_3 = { ends with 'a' }



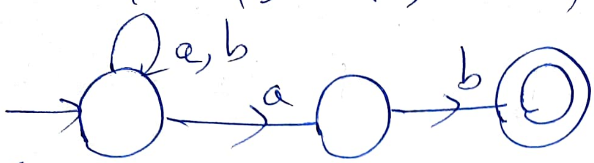
Q L_4 = { starts with 'ab' }



Q L_5 = { all the string contain 'ab' }



Q- $L = \{$ ends with 'ab' $\}$



Lect-35

Conversion of NFA to DFA

Both NFA and DFA are equally powerful.

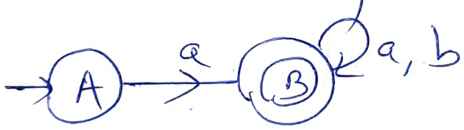
NFA $\rightarrow$ DFA (NFA can be converted to DFA)

DFA $\rightarrow$ NFA (Every DFA is NFA)

NFA $\cong$ DFA

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

$L_1 = \{$ starts with 'a' $\}$

NFA : 

→ state transition diagram

we will use subset construction method to convert NFA to DFA.

State transition table for NFA

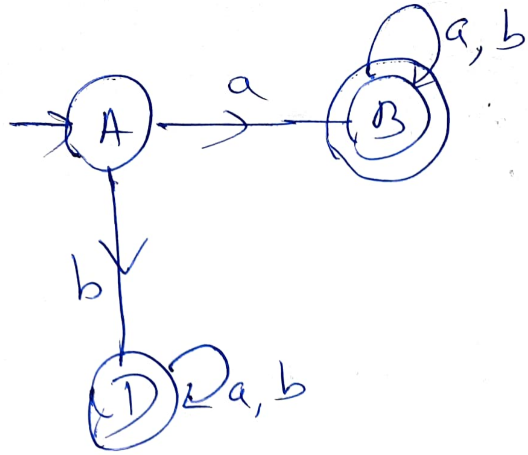
	a	b
→A	B	ϕ
*B	B	B

rows as states
column as inputs

State transition table for DFA

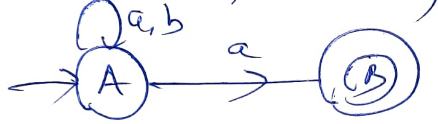
	~~a~~	~~b~~
~~A~~	~~B~~	~~D~~
~~B~~	~~B~~	~~B~~

	a	b
→A	B	D
B*	B	B
D	D	D



Lect-36 Conversion of NFA to DFA for all strings ends with 'a'.

L_1 = {ends with an 'a'}

NFA:  → state transition diagram

state transition table for NFA

	a	b
→A	{A, B}	{A}
*B	{ }	{ }

{ } = ϕ same

state transition table for DFA

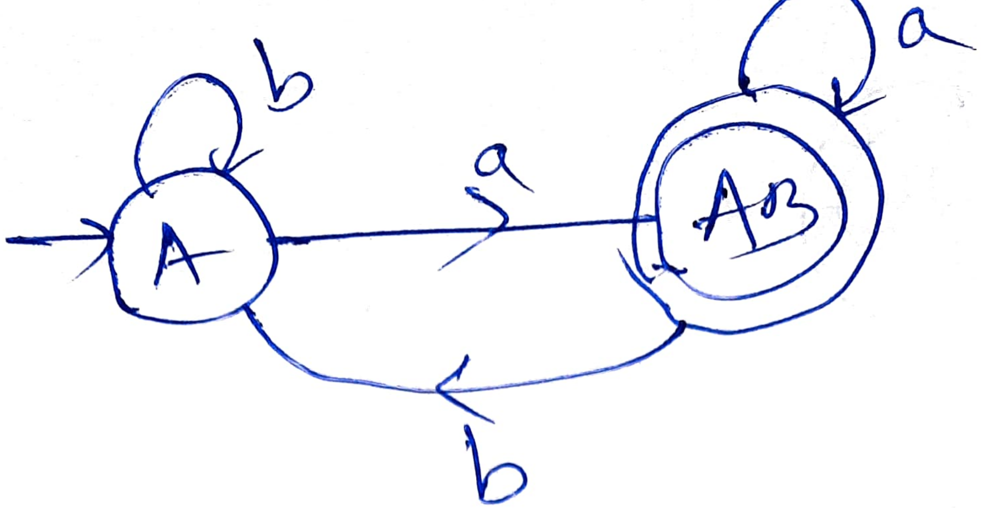
	a	b
→[A]	[AB]	[A]
*[AB]	[AB]	[A]

[AB] → It's a single state bcoz in DFA we can reach at one state only

steps are

1 → Start with initial state

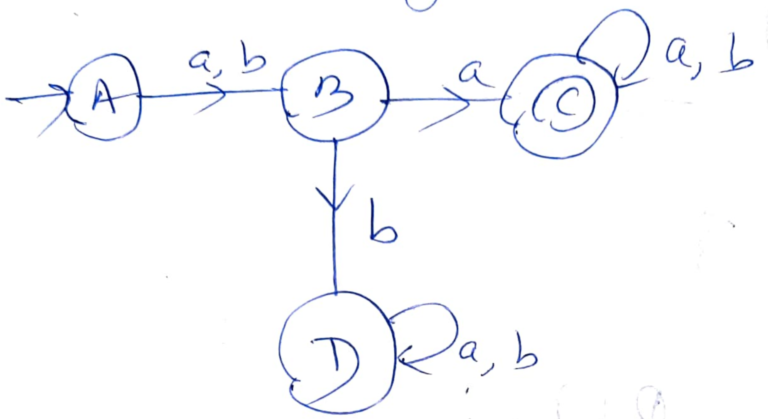
2 → Draw all the states which are reachable from initial states on inputs


A
AB
a
b
a
b

Lecture-37. Conversion of NFA to DFA

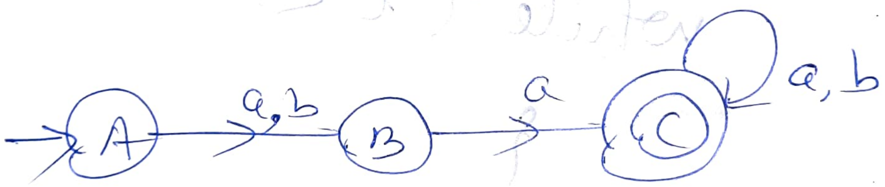
Q Conversion of NFA to DFA for the example all strings in which second symbol from RHS is 'a'.

DFA For second symbol from LHS is 'a'.



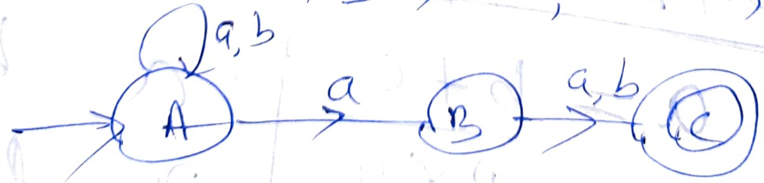
_ a _ _

NFA For second symbol from LHS is 'a'.



NFA For second symbol from RHS is 'a'.

L = { aa, ab, aaa, aab, - - - }



$\Sigma^* a(a+b)$

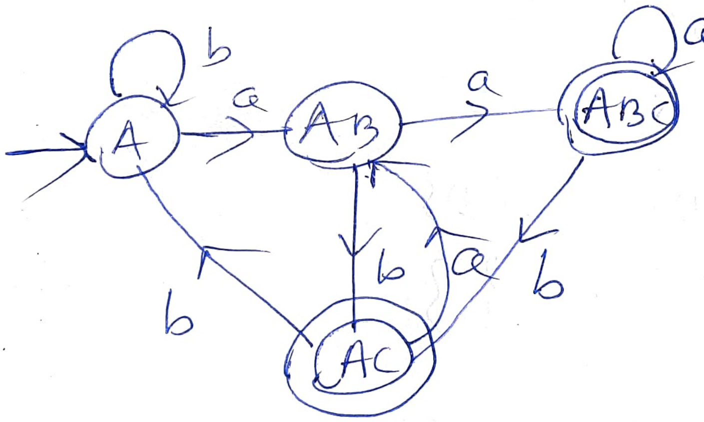
$\Sigma^* aa \quad \Sigma^* ab$

State transition table for NFA

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

State transition table For DFA

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



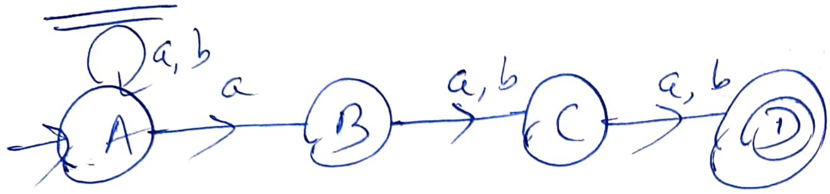
Any state which is containing C is final state.

Lecture - 38

Conversion of NFA to DFA for the example
all strings in which third symbol from RHS is 'a'.

Σ^* aaa
ab
ba
bb

NFA



Transition table For NFA

	a	b
→A	{A, B}	{A}
B	{C}	{C}
C	{D}	{D}
*D	{ }	{ }

Transition table For DFA

	a	b
A	[AB]	[A]
[AB]	[ABC]	[AC]
[AC]	[ABD]	[AD]
~~[ABC]~~	~~[ABCD]~~	~~[ACD]~~
*[AD]	[AB]	[A]
[ABC]	[ABCD]	[ACD]

	a	b
*[ABD]	[ABC]	[AC]
*[ACD]	[ABD]	[AD]
*[ABCD]	[ABCD]	[ACD]

Lecture-39

NFA For strings of length

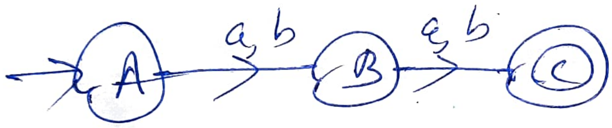
a. exactly 2

b. atmost 2

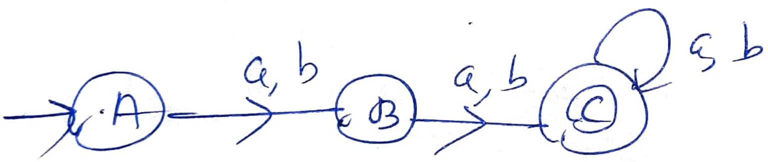
c. atleast 2

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

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



Strings of length exactly 2



atleast ~~atmost~~ 2

atmost 2