INTRODUCTION (Theory Of Computation)

>> One of the most fundamental courses of Computer Science

>> Will help you understand how people have thought about Computer Science as a Science in the past 50 years

>> It is mainly about what kind of things can you really compute mechanically, how fast and how much space does it take to do so

how much space does it take to do so

Binary strings - end - 0

11010110 = 0 ✓
↑ = 1 ✗

Accepts all valid Java codes

✓

Eg. Compilers

↓ binary

↓ valid ? invalid

Accepts all valid Java codes and never goes into infinite loop.

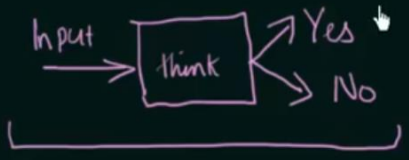
✗

✗

✗

Eg. Compilers

valid ? invalid



FSM - Finite State Machine

CFL - Context Free Language

↓ set of strings

Finite State Machine (Prerequisites)

Symbol – a, b, c, 0, 1, 2, 3, ...

Alphabet – Σ – collection of symbols – Eg. {a, b}, {d, e, f, g} {0, 1, 2} ....

String – sequence of symbols. Eg. a, b, 0, 1, aa, bb, ab, 01, ..

Language – set of strings

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

L_1 = set of all strings of length 2

= {00, 01, 10, 11}

String – sequence of symbols. Eg. a, b, 0, 1, aa, bb, ab, 01, ..

Language – set of strings

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

L_1 = set of all strings of length 2.
= {00, 01, 10, 11}

L_2 = set of all strings of length 3
= {000, 001, 010, 011, 100, 101, 110, 111}

finite

L_3 = set of all strings that begin with 0
= {0, 00, 01, 000, 001, 010, 011, 0000, ...}

infinite

finite

infinite

Powers of Σ : $\Sigma = \{0, 1\}$

Σ^0 = set of all strings of length 0 : $\Sigma^0 = \{\epsilon\}$

Σ^1 = set of all strings of length 1 : $\Sigma^1 = \{0, 1\}$

Σ^2 = set of all strings of length 2 : $\Sigma^2 = \{00, 01, 10, 11\}$

Σ^3 = set of all strings of length 3 : $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Σ^n = set of all strings of length n

Σ^3 = Set of all strings of length 3 : $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Σ^n = Set of all strings of length n

Cardinality - number of elements in a set

↳ $\Sigma^n = 2^n$

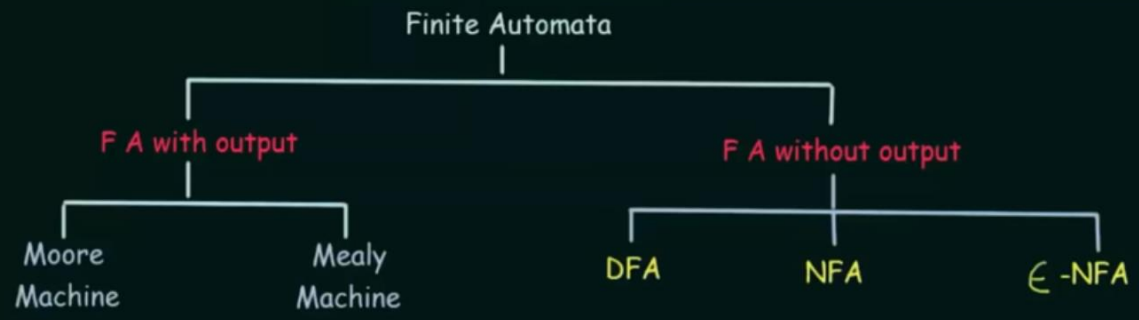
$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cdots$

$= \{\epsilon\} \cup \{0,1\} \cup \{00, 01, 10, 11\} \cup \cdots$

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

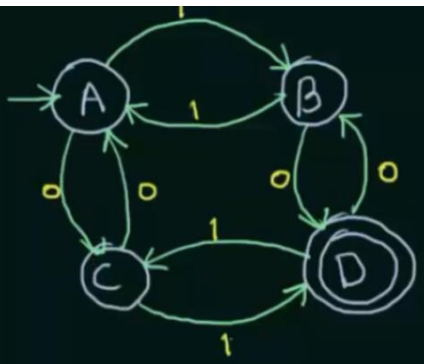
↳ infinite.

Finite State Machine



DFA - Deterministic Finite Automata

- It is the simplest model of computation
- It has a very limited memory



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

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

$q_0 = A$

$F = \{D\}$

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

Q = set of all states

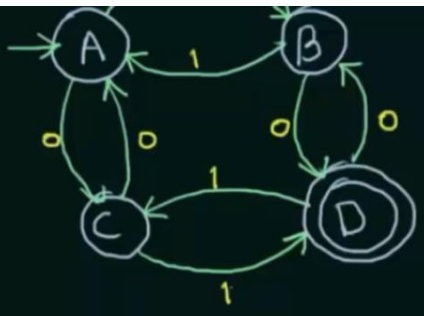
Σ = inputs

q_0 = start state / initial state

F = set of final states.

δ = transition function from $Q \times \Sigma \to Q$

	0	1
A		
B		
C		



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

Q = set of all states

Σ = inputs

q_0 = start state / initial state

F = set of final states.

δ = transition function from $Q \times \Sigma \to Q$

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

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

$q_0 = A$

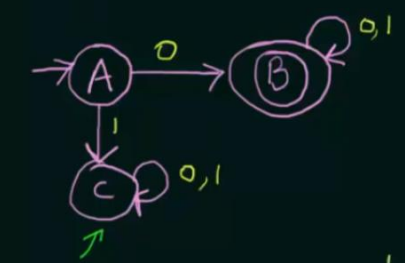
$F = \{D\}$

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

Deterministic Finite Automata (Example-1)

L1 = Set of all strings that start with '0'

$= \{0, 00, 01, 000, 010, 011, 0000, \ldots\}$



Dead State or trap State

Eg. 001 ✓

Initial state A → B → B → B – Final State

Eg. 101 ✗

Initial State A → C → C → C – Not final State

Deterministic Finite Automata (Example-2)

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

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

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

Dead State or Trap State.

Eg. 00 ✓

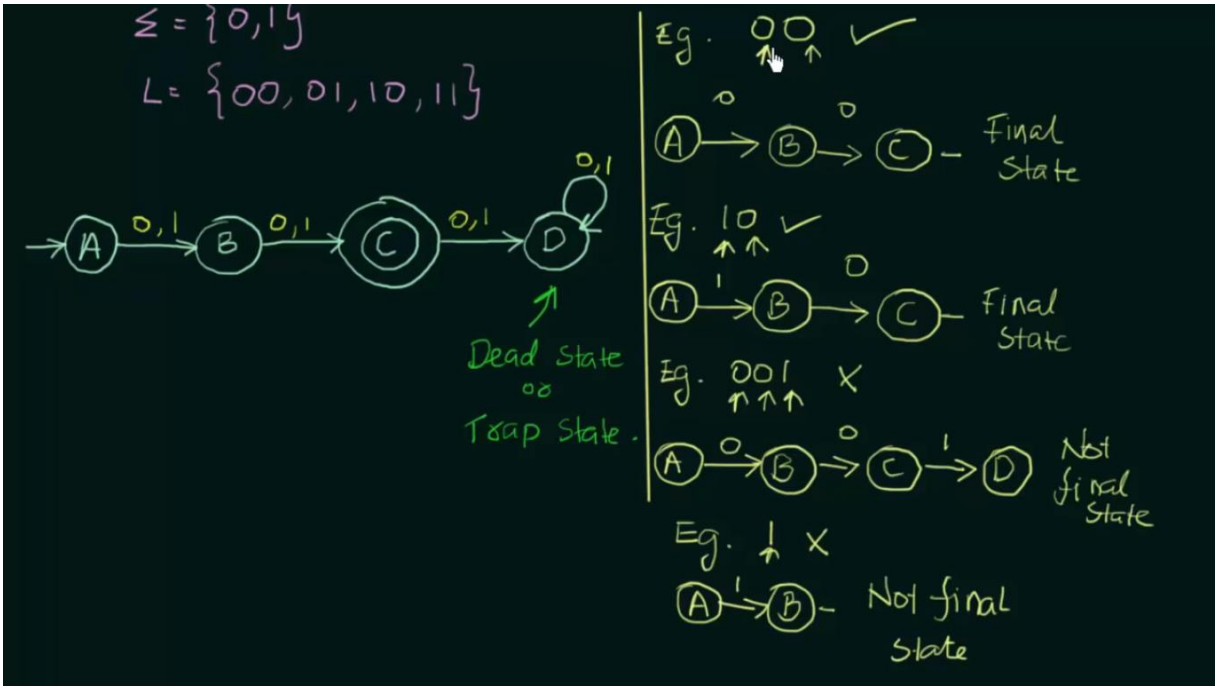
A → B → C – Final State

Eg. 10 ✓

A → B → C – Final State

Eg. 001 ✗

A → B → C → D – Not final State



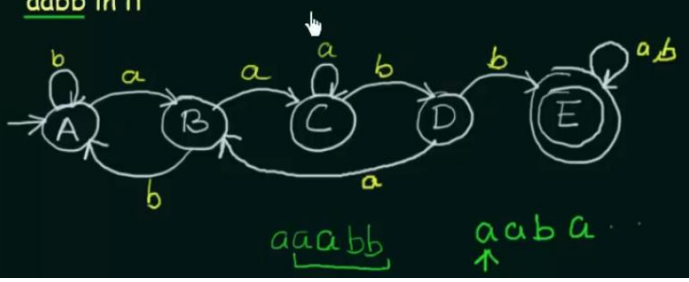
Deterministic Finite Automata (Example-3)

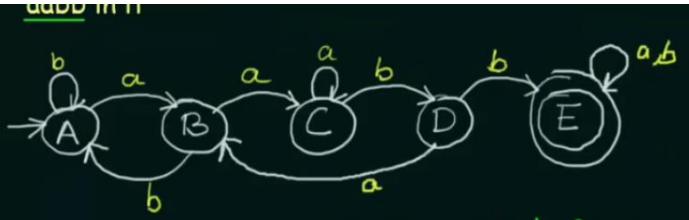
Construct a DFA that accepts any strings over {a,b} that does not contain the string aabb in it.

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

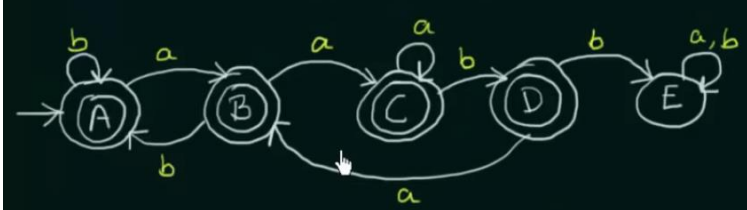
Try to design a simpler problem

Let us construct a DFA that accepts all strings over {a,b} that contains the string aabb in it



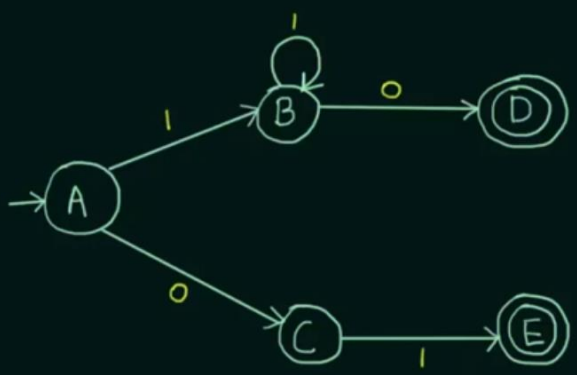


- Flip the States
- Make the Final State into non final state and
- Make the non final states into final states

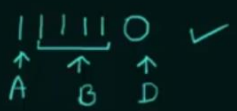


Deterministic Finite Automata (Example-4)

How to figure out what a DFA recognizes?



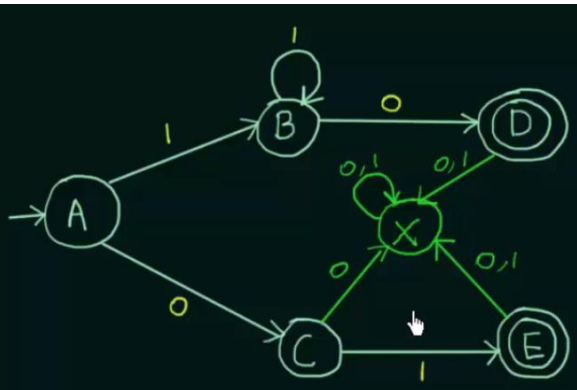
10 ✓



01 ✓

one binary digit '1'

L = {Accepts the string 01 or a string of atleast one '1' followed by a '0'}



10 ✓

01 ✓

one binary digit '1'

L = {Accepts the string 01 or a string of atleast one '1' followed by a '0'}

Eg. 001, 010, 011, 1101, 1100

X - Dead State

Regular Languages

- A language is said to be a REGULAR LANGUAGE if and only if some Finite State Machine recognizes it

So what languages are NOT REGULAR ?

The languages
>> Which are not recognized by any FSM
>> Which require memory

- Memory of FSM is very limited
-It cannot store or count strings

Eg. ababbababb

Eg. $a^N b^N$

aaabbb

aaaa, bbbb

Operations on Regular Languages

UNION – $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$

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

STAR – $A^* = \{ x_1 x_2 x_3 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \}$

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

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

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

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

STAR

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

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

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

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

Theorem 1: The class of Regular Languages is closed under UNION

Theorem 2: The class of Regular Languages is closed under CONCATENATION

NFA - Non-deterministic Finite Automata

Deterministic Finite Automata

↓

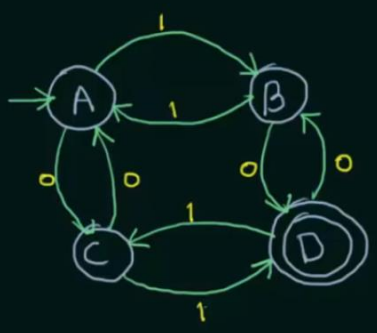
DETERMINISM

>> In DFA, given the current state we know what the next state will be

>> It has only one unique next state

>> It has no choices or randomness

>> It is simple and easy to design

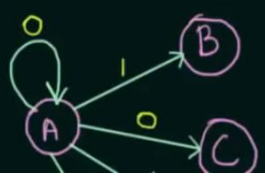


Non-deterministic Finite Automata

↓

NON-DETERMINISM

>> In NFA, given the current state there could be multiple next states



>> In DFA, given the current state we know what the next state will be

>> It has only one unique next state

>> It has no choices or randomness

>> It is simple and easy to design

Non-deterministic Finite Automata

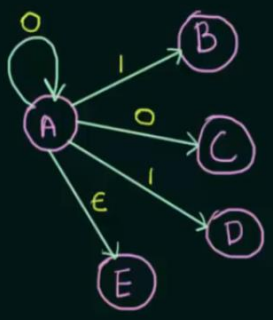
↓

NON-DETERMINISM

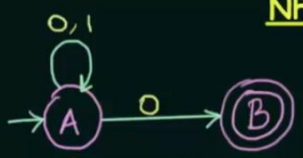
>> In NFA, given the current state there could be multiple next states

>> The next state may be chosen at random

>> All the next states may be chosen in parallel



NFA - Formal Definition



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

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

Q = Set of all states - {A, B}

Σ = inputs - {0, 1}

q_0 = Start State / initial State - A

F = Set of final States - B

δ = Q × Σ → __ - ?

A × 0 → A

A × 0 → B

A × 1 → A

B × 0 → ϕ

B × 1 → ϕ

A → A, B, AB, ϕ

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

Q = Set of all states - {A, B}

Σ = inputs - {0, 1}

q_0 = Start State / initial State - A

F = Set of final States - B

δ = Q × Σ → 2^Q - ?

A × 0 → A

A × 0 → B

A × 1 → A

B × 0 → ϕ

B × 1 → ϕ

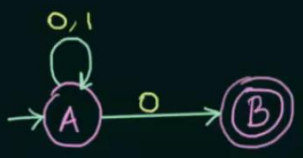
A → A, B, AB, ϕ - 2^{2-4}

3 States - A, B, C

A → A, B, C, AB, AC, BC, ABC, ϕ

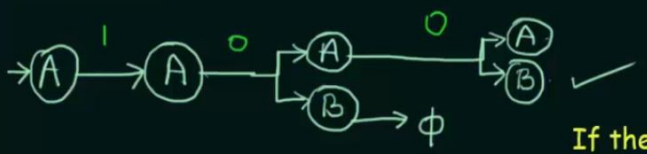
2^{3-8}

NFA - Example-1

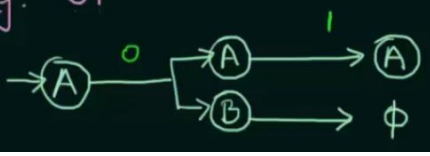


L = { Set of all strings that end with 0 }

Eg. 100



Eg. 01



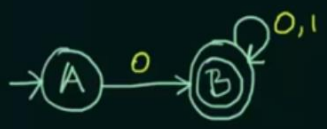
X

If there is any way to run the machine that ends in any set of states out of which atleast one state is a final state, then the NFA accepts

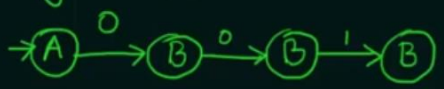
NFA - Example-2

L = { Set of all strings that start with 0 }

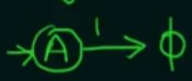
= { 0, 00, 01, 000, . . . }



Eg. 001 ✓



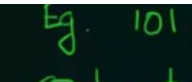
Eg. 101 X



Dead configuration

>> Construct a NFA that accepts sets of all strings over {0,1} of length 2

Eg. 101 X

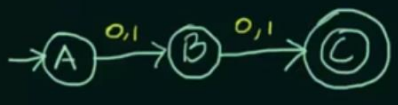


Dead configuration

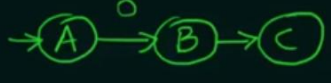
>> Construct a NFA that accepts sets of all strings over {0,1} of length 2

Σ = {0,1}

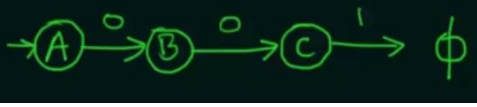
L = {00, 01, 10, 11}



Eg. 00 ✓

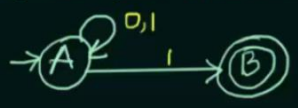


Eg. 001 X



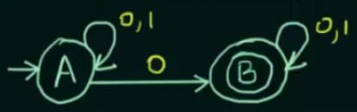
NFA - Example-3

Ex 1) L1 = { Set of all strings that ends with '1' }

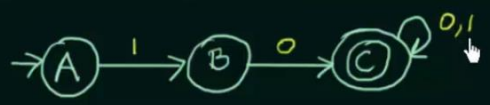


01, 001, 0001, 0*1, 1, 101, 1101,

Ex 2) L2 = { Set of all strings that contain '0' }

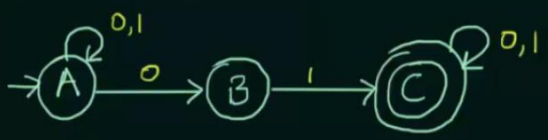


Ex 3) L3 = { Set of all strings that starts with '10' }

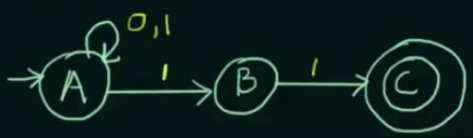


Ex 4) L4 = { Set of all strings that contain '01' }

Ex 4) L4 = { Set of all strings that contain '01' }



Ex 5) L5 = { Set of all strings that ends with '11' }



Assignment: If you were to construct the equivalent DFAs for the above NFAs, then tell me how many minimum number of states would you use for the construction of each of the DFAs

Conversion of NFA to DFA

Every DFA is an NFA, but not vice versa

But there is an equivalent DFA for every NFA

DFA

$\delta = Q \times \Sigma \rightarrow Q$

NFA

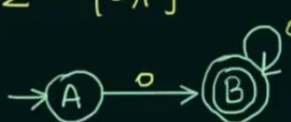
$\delta = Q \times \Sigma \rightarrow 2^Q$

NFA $\cong$ DFA

L = { Set of all strings over (0,1) that starts with '0' }

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

NFA

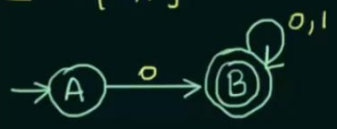


NFA ≅ DFA

L = { Set of all strings over (0,1) that starts with '0' }

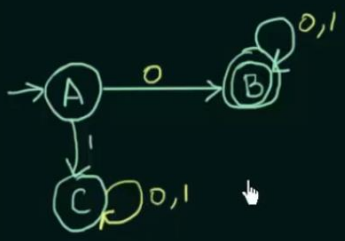
$\Sigma = \{0,1\}$

NFA



	0	1
A	B	ϕ
B	B	B

DFA



	0	1
A	B	C
B	B	B
C	C	C

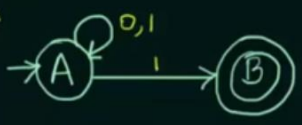
C - Dead state / Trap state

Conversion of NFA to DFA - Examples (Part 1)

L = { Set of all strings over (0,1) that ends with '1' }

$\Sigma = 0, 1$

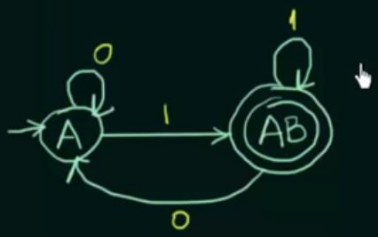
NFA



	0	1
A	{A}	{A, B}
B	ϕ	ϕ

Subset construction method

DFA



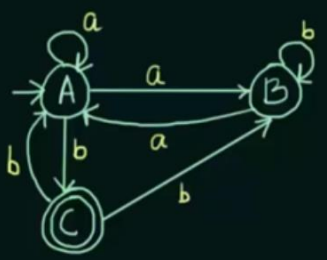
	0	1
A	{A}	{AB}
AB	{A}	{AB}

AB - Single State

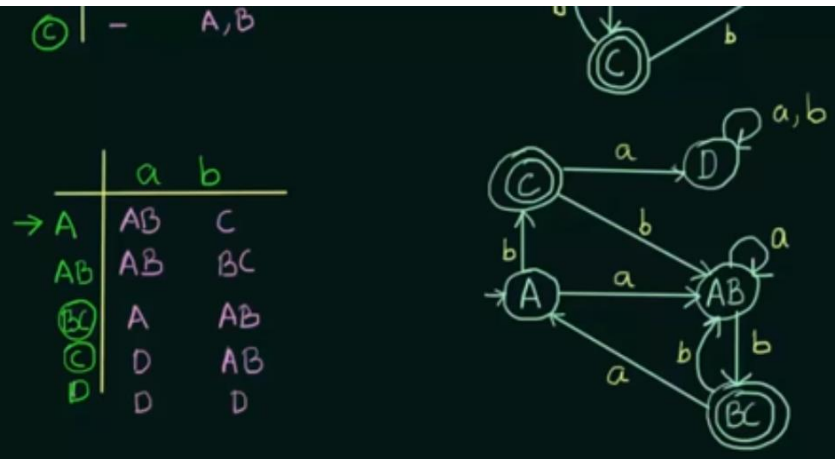
Conversion of NFA to DFA - Examples (Part-2)

Find the equivalent DFA for the NFA given by M= [{A,B,C}, (a,b), δ, A, {C}] where δ is given by:

	a	b
→A	A,B	C
B	A	B
©C	–	A,B



	a	b
→A	AB	C
AB	AB	BC
BC	A	AB
©C		

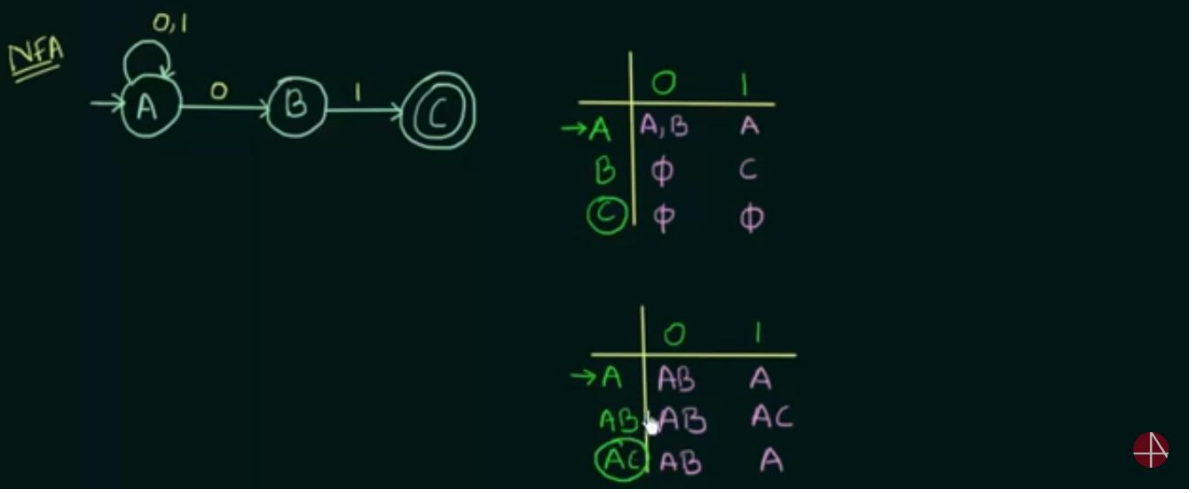


Assignment: Try to find out what does this NFA and its equivalent DFA accept

Conversion of NFA to DFA - Examples (Part-3)

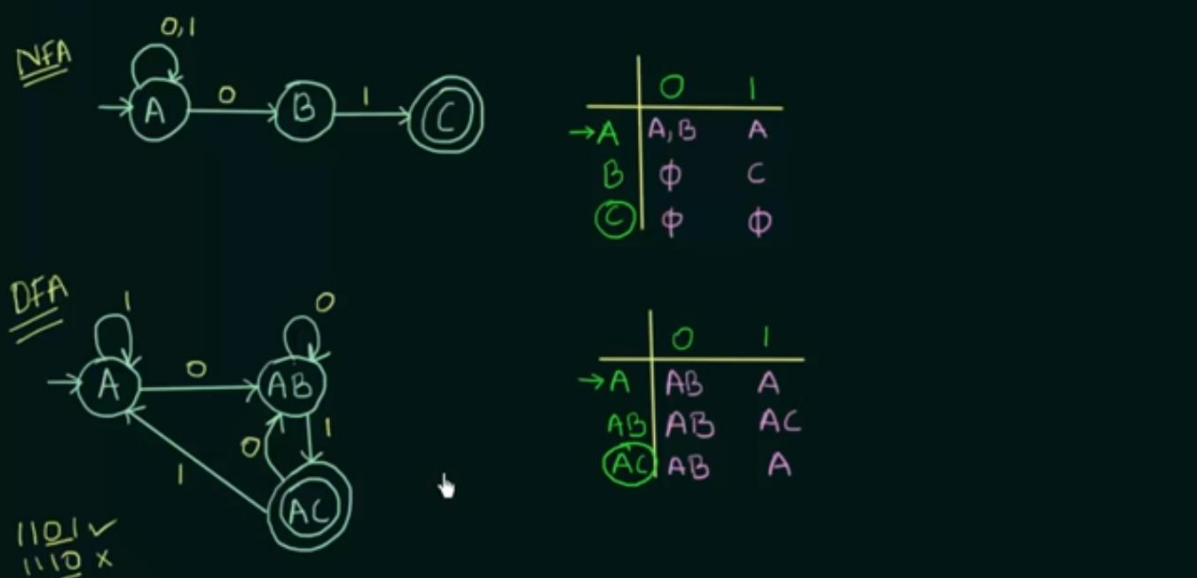
Given below is the NFA for a language

L = { Set of all strings over (0,1) that ends with '01' }. Construct its equivalent DFA



Given below is the NFA for a language

L = { Set of all strings over (0,1) that ends with '01' }. Construct its equivalent DFA

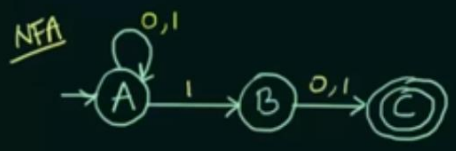


Conversion of NFA to DFA - Examples (Part-4)

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 equivalent DFA.

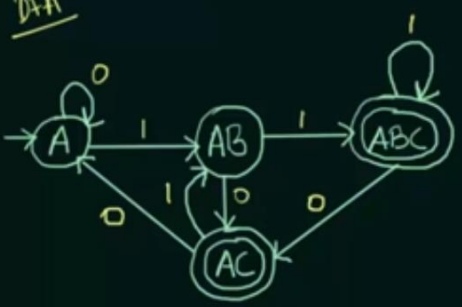
Eg. 1010
110
11010

NFA



	0	1
→A	A	A,B
B	C	C
(C)	ϕ	ϕ

DFA



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

Minimization of DFA

Minimization of DFA is required to obtain the minimal version of any DFA which consists of the minimum number of states possible

DFA 5 states 4 states

Equivalent

Two states 'A' and 'B' are said to be equivalent if

$\delta(A, X) \rightarrow F$ and $\delta(B, X) \rightarrow F$ OR $\delta(A, X) \not\rightarrow F$ and $\delta(B, X) \not\rightarrow F$

where 'X' is any input String

Equivalent

Two states 'A' and 'B' are said to be equivalent if

$\delta(A, X) \rightarrow F$ and $\delta(B, X) \rightarrow F$ OR $\delta(A, X) \not\rightarrow F$ and $\delta(B, X) \not\rightarrow F$

where 'X' is any input String

If |X| = 0, then A and B are said to be 0 equivalent

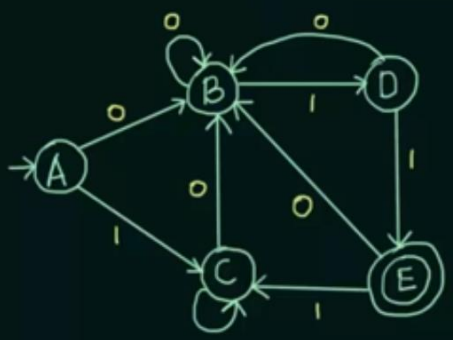
If |X| = 1, then A and B are said to be 1 equivalent

If |X| = 2, then A and B are said to be 2 equivalent

⋮

If |X| = n, then A and B are said to be n equivalent

Minimization of DFA - Examples (Part-1)



	0	1
→ A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

0 Equivalence : {A, B, C, D} {E}

1 Equivalence : {A, B, C} {D} {E}

2 Equivalence : {A, C} {B} {D} {E}

A, B ✓
A, C ✓
C, D ✗

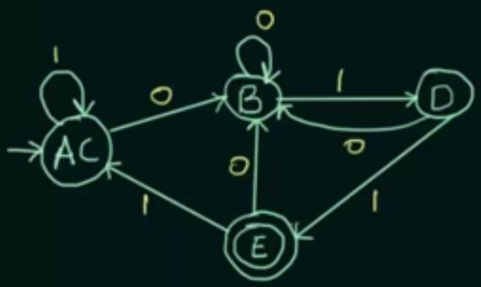
0 Equivalence : {A, B, C, D} {E}

1 Equivalence : {A, B, C} {D} {E}

2 Equivalence : {A, C} {B} {D} {E}

3 Equivalance : {A, C} {B} {D} {E}

A, B ✓
A, C ✓
C, D ✗



	0	1
→ A	B	C
B	B	D
C	B	C
D	B	E
(E)	B	C

Minimization of DFA - Examples (Part-2)

Construct a minimum DFA equivalent to the DFA described by

	0	1
→q_0	q_1	q_5
q_1	q_6	q_2
(*q_2)	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

0 Equivalence

$\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$ $\{q_2\}$

1- Equivalence

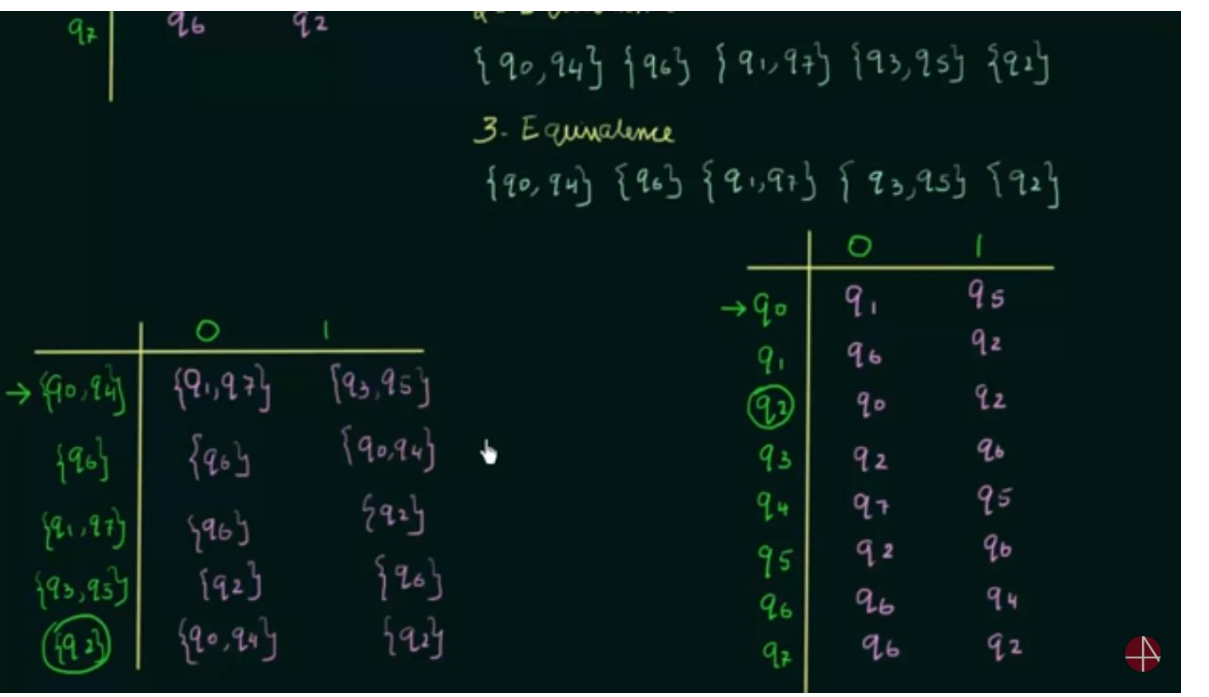
$\{q_0, q_4, q_6\}$

$\{q_1, q_7\}$

$\{q_3, q_5\}$ $\{q_2\}$

2- Equivalence

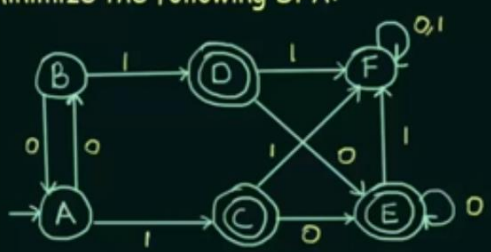
$\{q_0, q_4\}$ $\{q_6\}$ $\{q_1, q_7\}$ $\{q_3, q_5\}$ $\{q_2\}$



Minimization of DFA - Examples (Part-3)

When there are more than one Final States involved

Minimize the following DFA:



	0	1
→A	B	C
B	A	D
(*C)	E	F
(*D)	E	F
(*E)	E	F
F	F	F

0-Equivalence - $\{A, B, F\}$ $\{C, D, E\}$

1-Equivalence - $\{A, B\}$ $\{F\}$ $\{C, D, E\}$

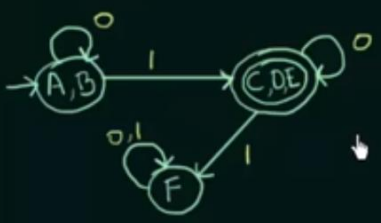
2-Equivalence - $\{A, B\}$ $\{F\}$ $\{C, D, E\}$

	0	1
→A	B	C
B	A	D
C	E	F
D	E	F
E	E	F
F	F	F

0-Equivalence - {A,B,F} {C,D,E}

1-Equivalence - {A,B} {F} {C,D,E}

2-Equivalence - {A,B} {F} {C,D,E}

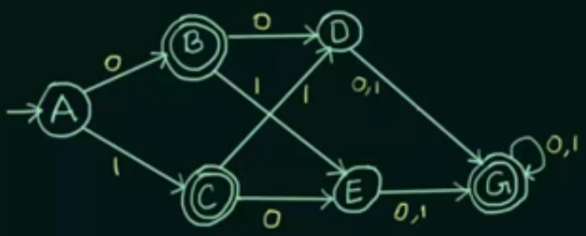


	0	1
→{A,B}	{A,B}	{C,D,E}
{F}	{F}	{F}
{C,D,E}	{C,D,E}	{F}

Minimization of DFA - Examples (Part-4)

When there are Unreachable States involved

A state is said to be Unreachable if there is no way it can be reached from the Initial State



	0	1
→A	B	C
B	D	E
C	E	D
D	G	G
E	G	G
G	G	G

0-Equivalence : {A,D,E} {B,C,G}

1-Equivalence : {A,D,E} {B,C} {G}

2-Equivalence : {A} {D,E} {B,C} {G}

3-Equivalence : {A} {D,E} {B,C} {G}

Initial State

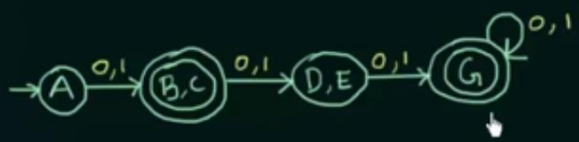
	0	1
→A	B	C
B	D	E
C	E	D
D	G	G
E	G	G
G	G	G

0-Equivalence : {A,D,E} {B,C,G}

1-Equivalence : {A,D,E} {B,C} {G}

2-Equivalence : {A} {D,E} {B,C} {G}

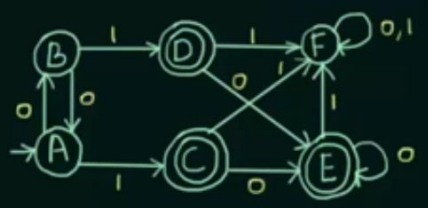
3-Equivalence : {A} {D,E} {B,C} {G}



	0	1
→{A}	{B,C}	{B,C}
{D,E}	{G}	{G}
{B,C}	{D,E}	{D,E}
{G}	{G}	{G}

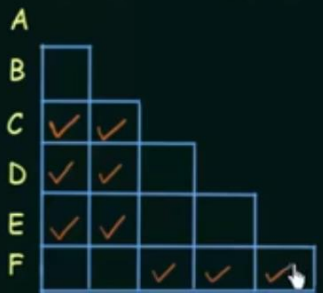
Minimization of DFA - Table Filling Method

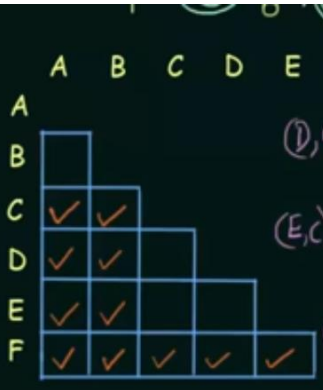
(Myhill-Nerode Theorem)



Steps:

1) Draw a table for all pairs of states (P,Q)
2) Mark all pairs where $P \in F$ and $Q \notin F$
3) If there are any Unmarked pairs (P,Q) such that $[\delta(P,x), \delta(Q,x)]$ is marked, then mark [P,Q] where 'x' is an input symbol
   REPEAT THIS UNTIL NO MORE MARKINGS CAN BE MADE
4) Combine all the Unmarked Pairs and make them a single state in the minimized DFA





(D,C) - $\delta(D,0) = E$, $\delta(D,1) = F$; $\delta(C,0) = E$, $\delta(C,1) = F$

(E,C) - $\delta(E,0) = E$, $\delta(E,1) = F$; $\delta(C,0) = E$, $\delta(C,1) = F$

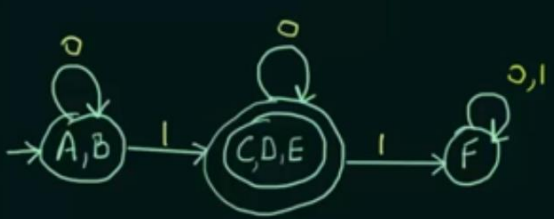
(E,D) - $\delta(E,0) = E$, $\delta(E,1) = F$; $\delta(D,0) = E$, $\delta(D,1) = F$

(F,A) - $\delta(F,0) = F$, $\delta(F,1) = F$; $\delta(A,0) = B$, $\delta(A,1) = C$

(F,B) - $\delta(F,0) = F$; $\delta(B,0) = A$

(B,A) - $\delta(B,0) = A$, $\delta(B,1) = D$; $\delta(A,0) = B$, $\delta(A,1) = C$

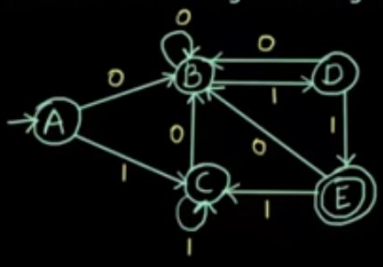
(A,B) (D,C) (E,C) (E,D)

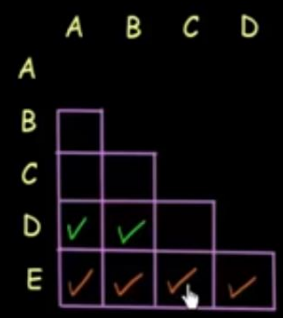


Minimization of DFA - Table Filling Method (Myhill Nerode Theorem)

(Example)

Minimize the following DFA using Table Filling Method





(B,A) - $\delta(B,1)=D$, $\delta(A,1)=C$; $\delta(B,0)=B$, $\delta(A,0)=B$

(C,B) - $\delta(C,0)=B$, $\delta(B,0)=B$; $\delta(C,1)=C$, $\delta(B,1)=D$

(D,B) - $\delta(D,0)=B$, $\delta(B,0)=B$

(C,A) - $\delta(C,0)=B$, $\delta(A,0)=B$; $\delta(C,1)=C$, $\delta(A,1)=C$

(D,A) - $\delta(D,0)=B$, $\delta(A,0)=B$; $\delta(D,1)=E$, $\delta(A,1)=C$

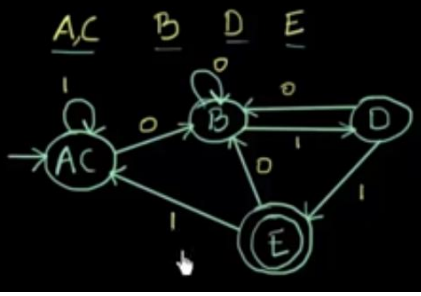
(D,C) - $\delta(D,0)=B$, $\delta(C,0)=B$; $\delta(D,1)=E$, $\delta(C,1)=C$

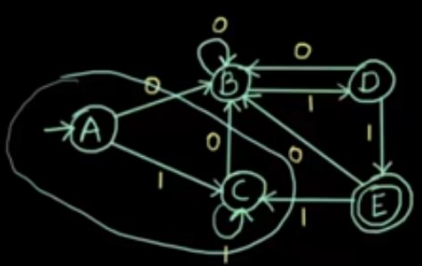
(B,A) - $\delta(B,0)=B$, $\delta(A,0)=B$; $\delta(B,1)=D$, $\delta(A,1)=C$

(C,A) - $\delta(C,0)=B$, $\delta(A,0)=B$; $\delta(C,1)=C$, $\delta(A,1)=C$

(C,B) - $\delta(C,0)=B$, $\delta(B,0)=B$; $\delta(C,1)=C$, $\delta(B,1)=D$

(C,A) - $\delta(C,0)=B$, $\delta(A,0)=B$; $\delta(C,1)=C$, $\delta(A,1)=C$





Finite Automata With Outputs

MEALY MACHINE

$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where

Q = Finite Set of States

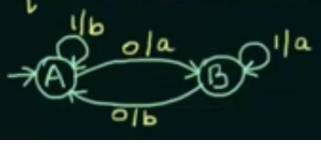
Σ = Finite non-empty set of Input Alphabets

Δ = The set of Output Alphabets

δ = Transition function: $Q \times \Sigma \rightarrow Q$

λ = Output function: $\Sigma \times Q \rightarrow \Delta$

q_0 = Initial State / Start State



MOORE MACHINE

$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where

Q = Finite Set of States

Σ = Finite non-empty set of Input Alphabets

Δ = The set of Output Alphabets

δ = Transition function: $Q \times \Sigma \rightarrow Q$

λ = Output function: $Q \rightarrow \Delta$

q_0 = Initial State / Start State

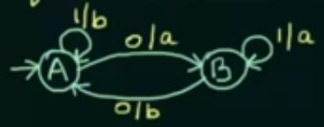
Alphabets

Δ = The set of Output Alphabets

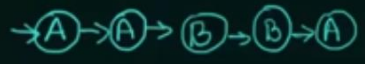
δ = Transition function: $Q \times \Sigma \to Q$

λ = Output function: $\Sigma \times Q \to \Delta$

q_0 = Initial State / Start State



Eg. 1 0 1 0



b a a b

n − n

Alphabets

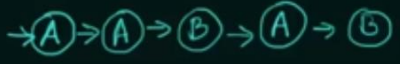
Δ = The set of Output Alphabets

δ = Transition function: $Q \times \Sigma \to Q$

λ = Output function: $Q \to \Delta$

q_0 = Initial State / Start State

Eg. 1 0 1 0

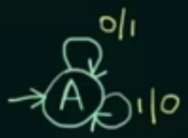


a a b a b

n → n+1

Construction of Mealy Machine

Ex-1) Construct a Mealy Machine that produces the 1's Complement of any binary input string.

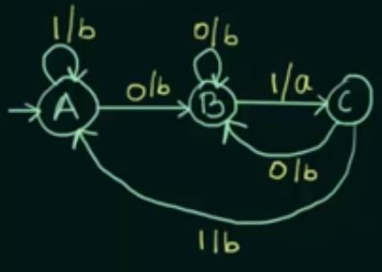


1 0 1 0 0

0 1 0 1 1

Ex-2) Construct a Mealy Machine that prints 'a' whenever the sequence '01' is encountered in any input binary string.

$\Sigma = \{0,1\}$ $\Delta = \{a,b\}$



0 1 1 0

b a b b

1 0 0 0

b b b b

Construction of Mealy Machine -Examples (Part-2)

Construct a Mealy Machine that gives 2's Complement of any binary input. (Assume that the last carry bit is neglected)

2's complement = 1's complement + 1

MSB ← LSB

Eg. 1 0 1 0 0

1's. 0 1 0 1 1

+ 1

2's = 0 1 1 0 0

Eg. 1 1 0 0 0

1's. 0 0 0 1 1

+ 1

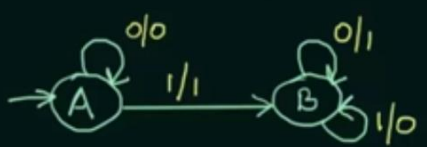
2's = 0 0 1 0 0

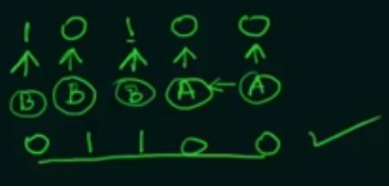
Eg. 1 1 1 0 1

1's 0 0 0 1 0

2's + 1

0 0 0 1 1



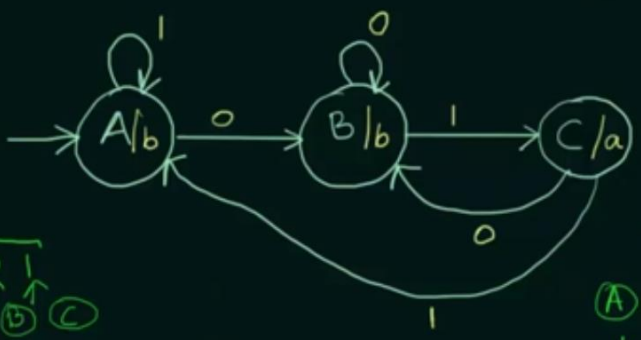


Construction of Moore Machine

Construct a Moore Machine that prints 'a' whenever the sequence '01' is encountered in any input binary string

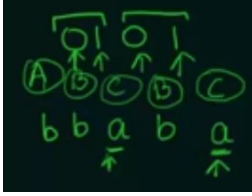
$\Sigma = \{0,1\}$

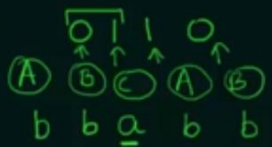
$\Delta = \{a,b\}$



0101

01101

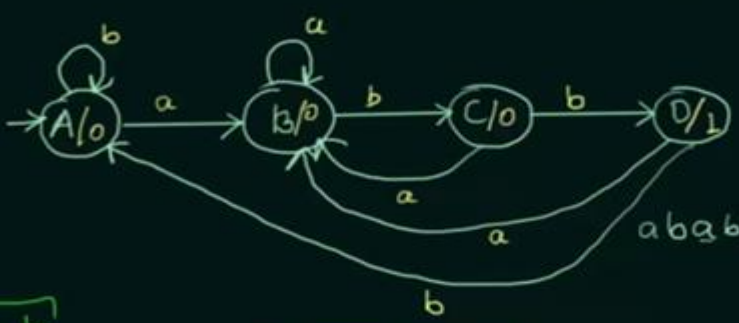




Construction of Moore Machine - Examples (Part-1)

Construct a Moore Machine that counts the occurences of the sequence 'abb' in any input strings over {a,b}

$\Sigma = \{a,b\}$ $\Delta = \{0,1\}$



abbabb

abbabb

abbabb

= 2

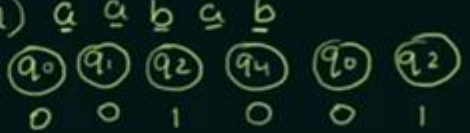
Construction of Moore Machine- Examples (Part-2)

For the following Moore Machine the input alphabet is $\Sigma = \{a,b\}$ and the output alphabet is $\Delta = \{0,1\}$. Run the following input sequences and find the repective outputs:

(i) aabab (ii) abbb (iii) ababb

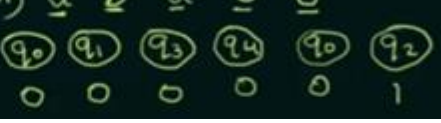
States	a	b	Outputs
$\rightarrow q_0$	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

i) a a b a b



ii) a b b b

iii) a b a b b

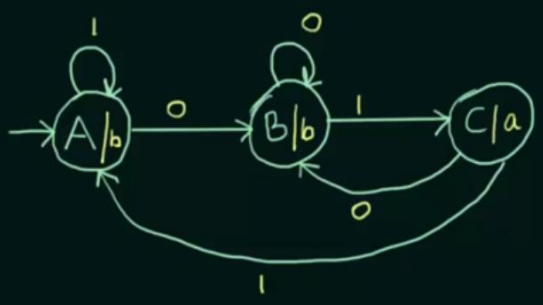


Conversion of Moore Machine to Mealy Machine

Construct a Moore Machine that prints 'a' whenever the sequence '01' is encountered in any input binary string and then CONVERT IT TO ITS EQUIVALENT MEALY MACHINE

$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

Moore Machine $\Longleftrightarrow$ Mealy Machine



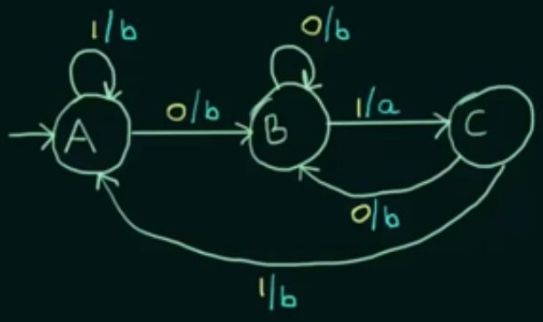
State	0	1	Output
→ A	B	A	b
B	B	C	b
C	B	A	a

Conversion of Moore Machine to Mealy Machine

Construct a Moore Machine that prints 'a' whenever the sequence '01' is encountered in any input binary string and then CONVERT IT TO ITS EQUIVALENT MEALY MACHINE

$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

Moore Machine $\Longleftrightarrow$ Mealy Machine

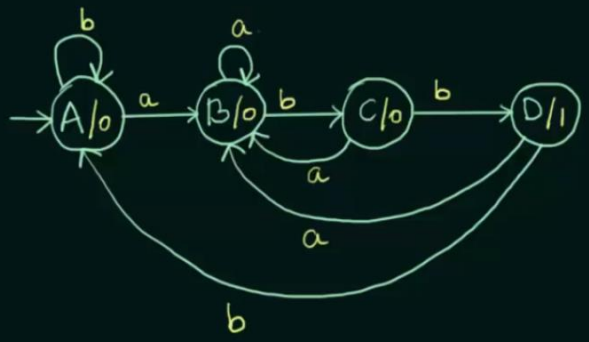


State	0	1	Output
→ A	B, b	A, b	b
B	B, b	C, a	b
C	B, b	A, b	a

Conversion of Moore Machine to Mealy Machine - Examples (Part-1)

The given Moore Machine counts the occurences of the sequence 'abb' in any input binary strings over {a,b}. CONVERT IT TO ITS EQUIVALENT MEALY MACHINE

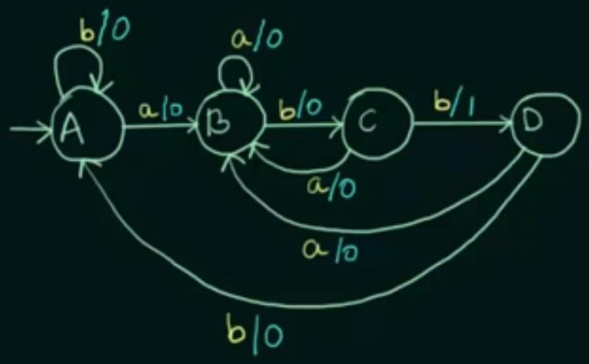
$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



Conversion of Moore Machine to Mealy Machine - Examples (Part-1)

The given Moore Machine counts the occurences of the sequence 'abb' in any input binary strings over {a,b}. CONVERT IT TO ITS EQUIVALENT MEALY MACHINE

$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



State	a	b
→A	B, 0	A, 0
B	B, 0	C, 0
C	B, 0	D, 1
D	B, 0	A, 0

Conversion of Moore Machine to Mealy Machine- Examples (Part-2)

Convert the given Moore Machine to its equivalent Mealy Machine.

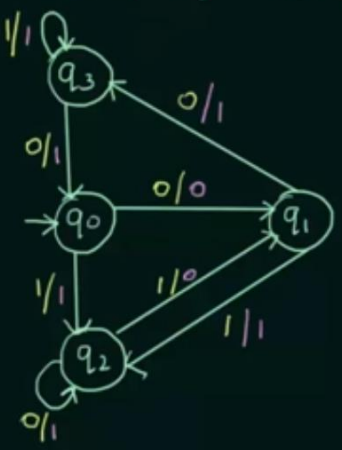
Moore

State	0	1	Output
→q_0	q_1	q_2	1
q_1	q_3	q_2	0
q_2	q_2	q_1	1
q_3	q_0	q_3	1

Mealy

State	0	1
→q_0	q_1, 0	q_2, 1
q_1	q_3, 1	q_2, 1
q_2	q_2, 1	q_1, 0
q_3	q_0, 1	q_3, 1

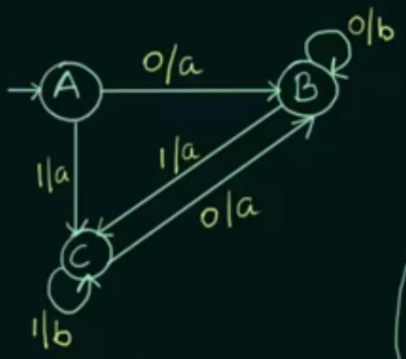
$\Sigma = \{0, 1\}$ $\Delta = \{0, 1\}$



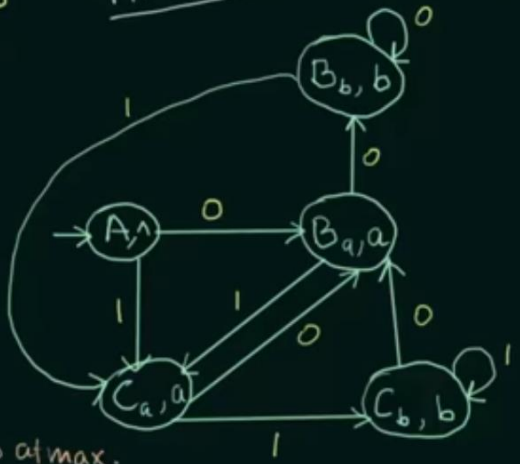
Conversion of Mealy Machine to Moore Machine

Convert the following Mealy Machine to its equivalent Moore Machine

$\Sigma = \{0, 1\}$
$\Delta = \{a, b\}$



Moore machine



Moore -> Mealy => No. of states were same

Mealy -> Moore => No. of states increased

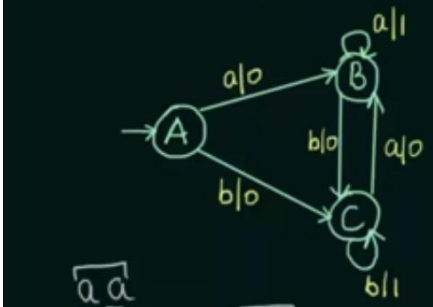
n and y ⇒ (nxy) no of states at max.

n → states, y → outputs

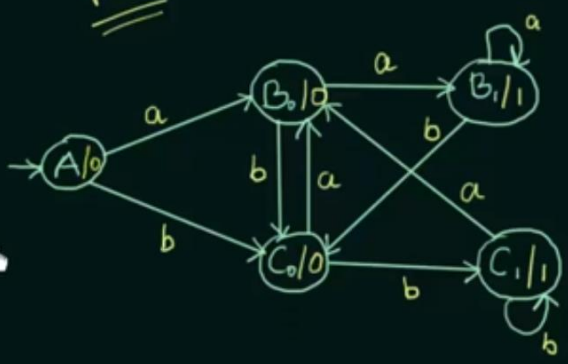
Conversion of Mealy Machine to Moore Machine - Examples (Part-1)

Given below is a Mealy Machine that prints '1' whenever the sequence 'aa' or 'bb' is encountered in any input binary string from Σ^* where $\Sigma = \{a,b\}$.
DESIGN THE EQUIVALENT MOORE MACHINE FOR IT.

$\Sigma = \{a,b\}$
$\Delta = \{0,1\}$



Moore



a a
0 0 1 ✓

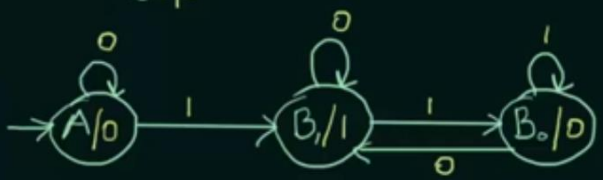
b b
0 0 1 ✓

a b
0 0 0

Conversion of Mealy Machine to Moore Machine- Examples (Part-2)

Convert the given Mealy Machine that give the 2's complement of any binary input to its equivalent MOORE MACHINE.

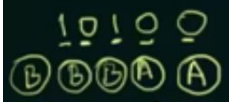
$\Sigma = \{0,1\}$ $\Delta = \{0,1\}$



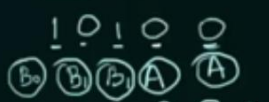
Moore

Eg. 10100
1's. 01011
+ 1
01100

Mealy:



Moore:



Conversion of Mealy Machine to Moore Machine -Examples (Part-3)

Using Transition Table

Convert the given Mealy Machine to its equivalent Moore Machine

State	a	b
→q_0	q_3, 0	q_1, 1
q_1	q_0, 1	q_3, 0
q_2	q_2, 1	q_2, 0
q_3	q_1, 0	q_0, 1

q_1 q_2
q_{10} q_{11} q_{20} q_{21}
output 0 output 1 output 0 output 1

State	a	b	Output
→q_0	q_3, 0	q_{11}, 1	
q_{10}	q_0, 1	q_3, 0	
q_{11}	q_0, 1	q_3, 0	
q_{20}	q_{21}, 1	q_{20}, 0	
q_{21}	q_{21}, 1	q_{20}, 0	
q_3	q_{10}, 0	q_0, 1	

Conversion of Mealy Machine to Moore Machine -Examples (Part-3)

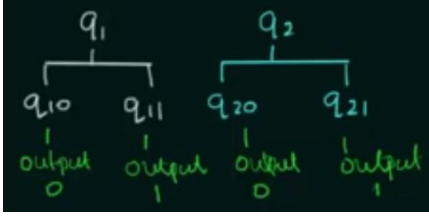
Using Transition Table

Convert the given Mealy Machine to its equivalent Moore Machine

State	a	b
→q_0	q_3, 0	q_1, 1
q_1	q_0, 1	q_3, 0
q_2	q_2, 1	q_2, 0
q_3	q_1, 0	q_0, 1

Moore

State	a	b	Output
→q_0	q_3	q_{11}	1
q_{10}	q_0	q_3	0
q_{11}	q_0	q_3	1
q_{20}	q_{21}	q_{20}	0
q_{21}	q_{21}	q_{20}	1
q_3	q_{10}	q_0	0

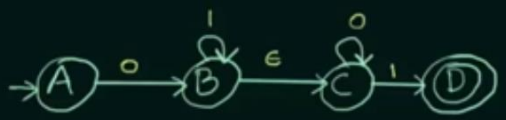


Epsilon (ϵ) - NFA

ϵ- NFA
↳ empty symbols

$\{Q, \Sigma, q_0, \delta, F\}$

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

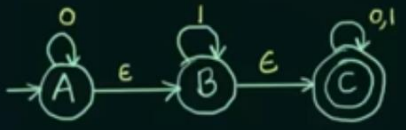


Every state on ϵ goes to itself.

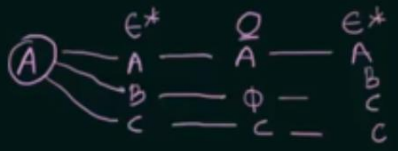
Conversion of ϵ -NFA to NFA

Convert the following ϵ-NFA to its equivalent NFA

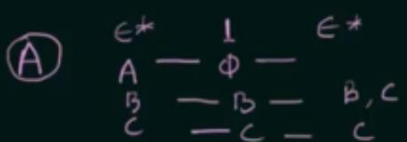
State ϵ* input ϵ*



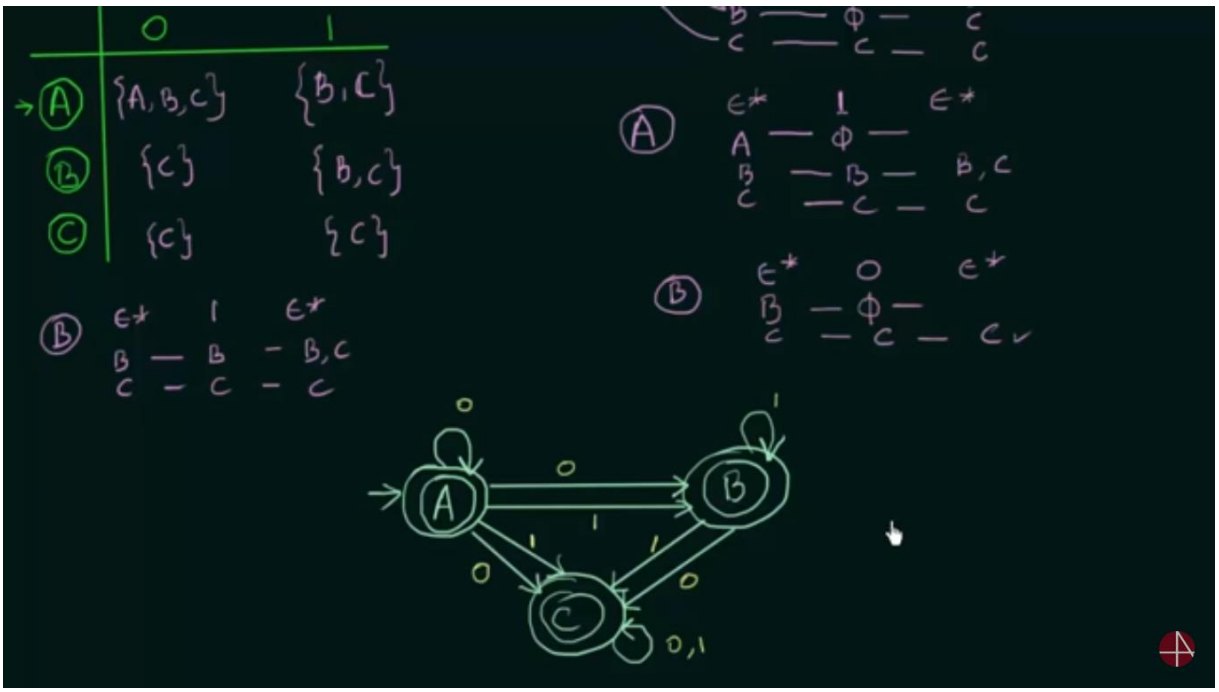
ϵ-Closure (ϵ*) - All the states that can be reached from a particular state only by seeing the ϵ symbol



	0	1
→A	{A, B, C}	{B, C}
B		
C		

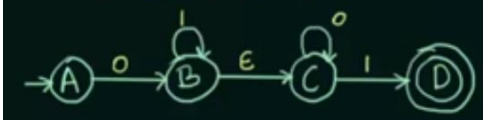


	0	1
→ A	{A, B, C}	{B, C}
B	{C}	{B, C}
C	{C}	{C}



Conversion of ϵ -NFA to NFA -Examples (Part-1)

Convert the following ϵ-NFA to its equivalent NFA



NFA

	0	1
→ A	B, C	ϕ
B	C	B, C, D
C	C	D
D	ϕ	ϕ

	ϵ*	0	ϵ*
A	A	B	B, C
B	B	ϕ	ϕ
	C	C	C
C	C	C	C
D	D	ϕ	ϕ

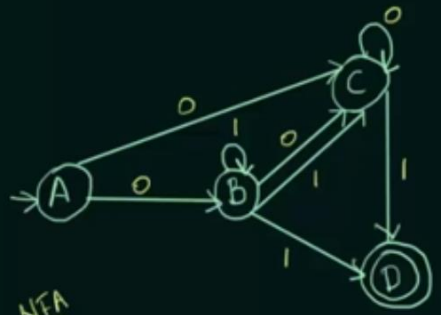
	ϵ*	1	ϵ*
A	A	ϕ	ϕ
B	B	B	B, C
	C	D	D
C	C	D	D
D	D	ϕ	ϕ

NFA

	0	1
→ A	B, C	ϕ
B	C	B, C, D
C	C	D
D	ϕ	ϕ

B	B	ϕ	ϕ
	C	C	C
C	C	C	C
D	D	ϕ	ϕ

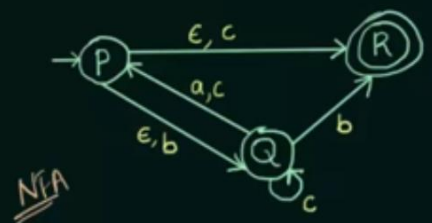
B	B	B	B, C
	C	D	D
C	C	D	D
D	D	ϕ	ϕ



NFA

Conversion of ϵ -NFA to NFA -Examples (Part-2)

Convert the following ϵ-NFA to its equivalent NFA



NFA

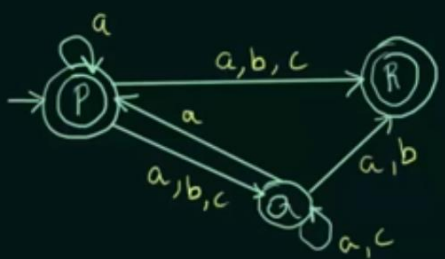
	a	b	c
→ (P)	{P, Q, R}	{Q, R}	{Q, R}
Q	{P, Q, R}	{R}	{Q}
(R)	ϕ	ϕ	ϕ

	ϵ^*	a	ϵ^*
P	P	ϕ	ϕ
	Q	P	P
	R	ϕ	Q
			R
			-
Q	Q	P	P
			Q
			R
R	R	ϕ	-

	ϵ^*	b	ϵ^*
P	P	Q	Q
	Q	R	R
	R	ϕ	-
Q	Q	R	R
R	R	ϕ	-

	ϵ^*	c	ϵ^*
P	P	R	R
	Q	Q	a
	R	ϕ	-
Q	Q	Q	Q
R	R	ϕ	-

	a	b	c
→ (P)	{P, Q, R}	{Q, R}	{Q, R}
Q	{P, Q, R}	{R}	{Q}
(R)	ϕ	ϕ	ϕ



NFA