Theory of Computation Neso notes

Theory of Computation (Delhi Technological University)

INTRODUCTION

- >> 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

Binary strings - end - 0

11010110 = 0 ✓

= 1 X

Accepts all valid Java codes ✓

↓

binary

↓

?

✓Valid

✓Invalid

Accepts all valid Java codes and never goes into infinite loop

X

X

= X

Finite state machine

Context Free Language

set of string

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}

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

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

Cardinality – number of elements in a set

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

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

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

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

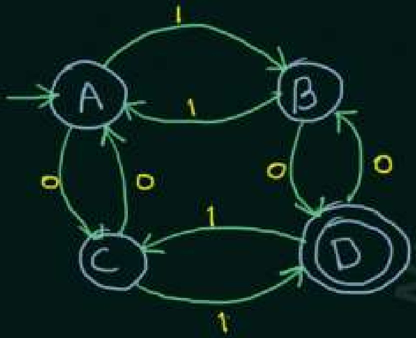
$\hookrightarrow$ infinite.

Finite State Machine

- Finite Automata
  - F A with input
    - Moore Machine
    - Mealy Machine
  - F A without input
    - DFA
    - NFA
    - ϵ-NFA

DFA - Deterministic Finite Automata

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



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

Q = Set of all States

Σ = Inputs

q_0 = Start state / Initial state

F = Set of Final States

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

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

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

$q_0 = A$

$F = \{D\}$

δ =

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

Finite State Machine (Prerequisites)

> Infinite

00:14:00.03
hr min sec

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

11:27 / 15:31

Scroll for details

Σ^3 = set of all strings of length 3 : $\Sigma^3 = \{000, 001, 010, 011, 100, \ldots 1\}$

Σ^n = set of all strings of length n

<u>Cardinality</u> :- number of elements in a set

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

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

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

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

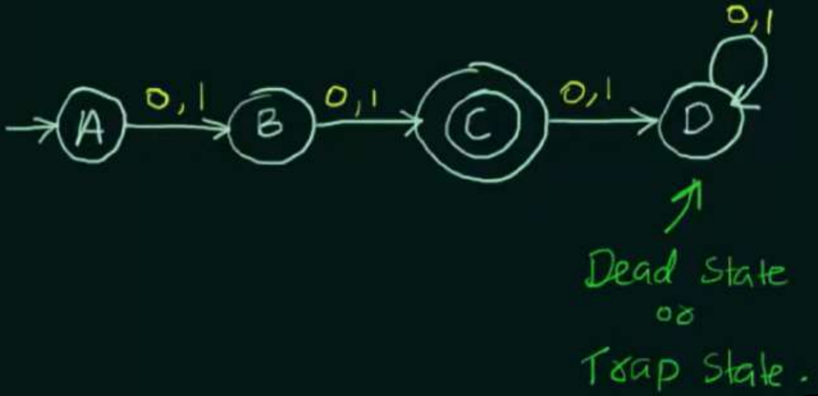
$\hookrightarrow$ infinite

Deterministic Finite Automata (Example-2)

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

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

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



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

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

Dead State
or
Trap State.

Eg. 00 ✓

$A \xrightarrow{0} B \xrightarrow{0} C$ - Final State

Eg. 10 ✓

$A \xrightarrow{1} B \xrightarrow{0} C$ - Final State

Eg. 001 X

$A \xrightarrow{0} B \xrightarrow{0} C \xrightarrow{1} D$ Not final State

Eg. 1 X

$A \xrightarrow{1} B$ - Not final State

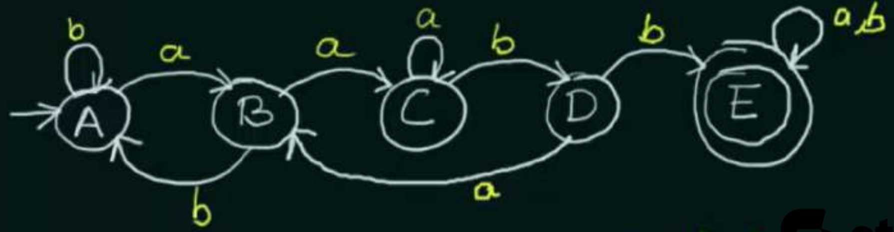
Deterministic Finite Automata (Example-3)

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

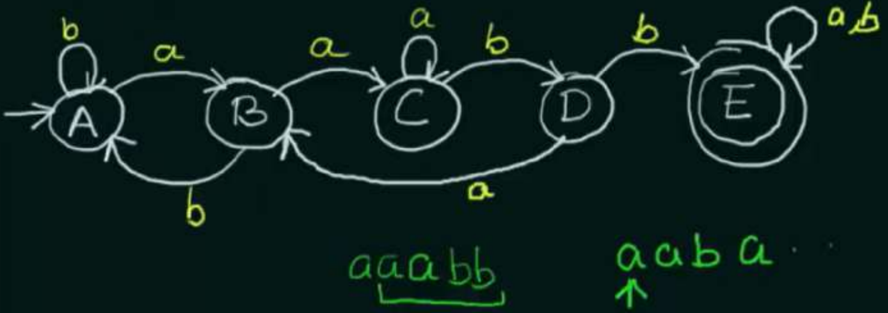
Let us construct a DFA that accepts all strings over {a,b} that contains the string aabb in it

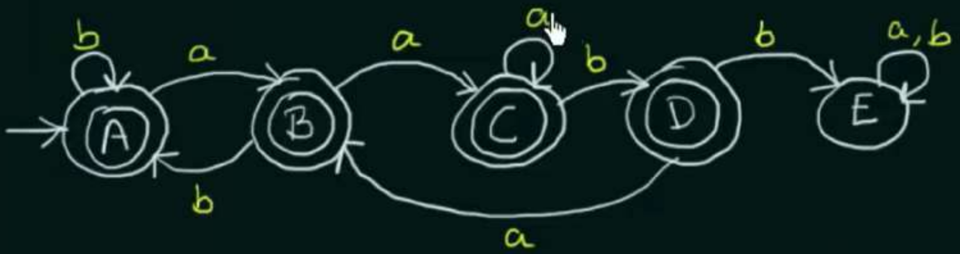


aaabb

aabaa

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

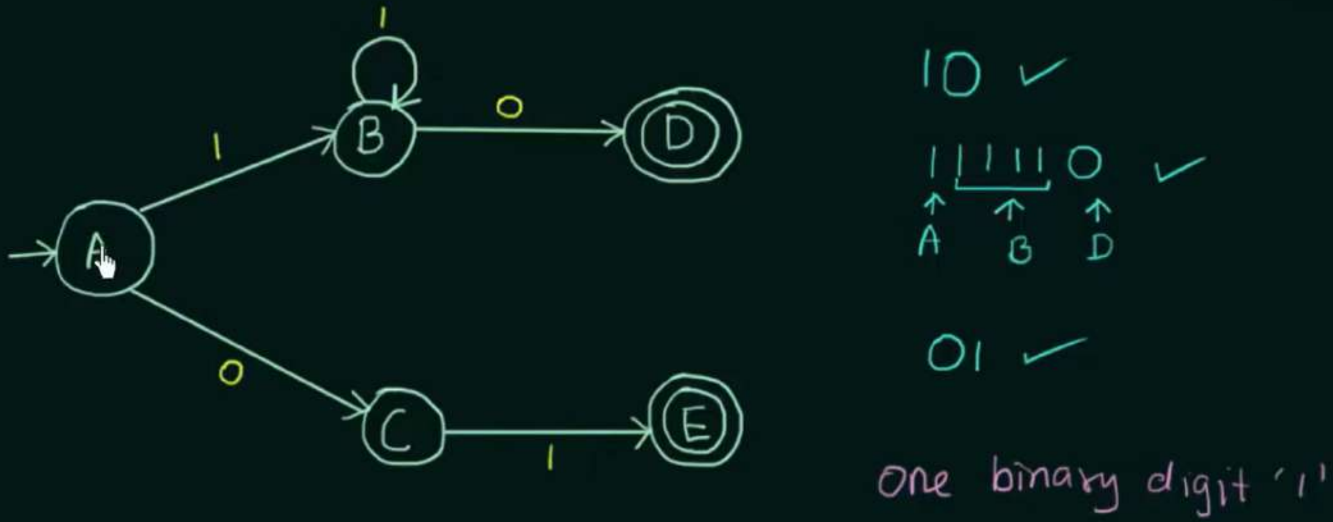




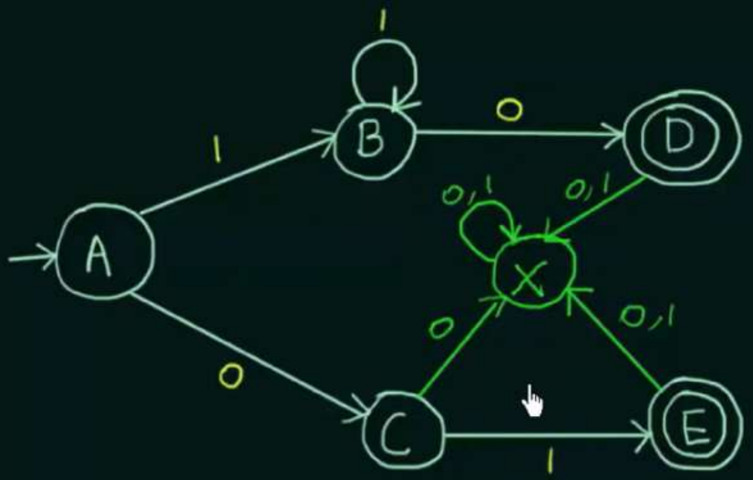
- Flip the states
- Make the Final State into non final state and
- Make the non final states into final states

Deterministic Finite Automata (Example-4)

How to figure out what a DFA recognizes?



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



10 ✓

$\underset{\uparrow A}{1}\,\underset{\uparrow B}{\underbrace{1111}}\,\underset{\uparrow D}{0}$ ✓

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. $\underbrace{ababb}\,\underbrace{ababb}$.

Eg. $a^N b^N$.

aaabbb

$\underbrace{aaaa}\,\underbrace{bbbb}$

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

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

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

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

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

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

<u>Theorem 1</u>: The class of Regular Languages is closed under UNION

<u>Theorem 2</u>: The class of Regular Languages is closed under CONCATENATION

NFA - Non-deterministic Finite Automata

Deterministic Finite Automata

↓

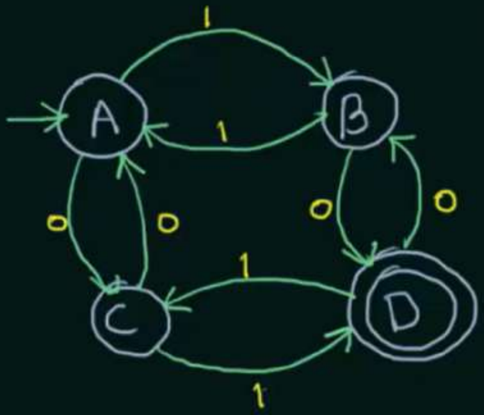
DETERMINISM

>> 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

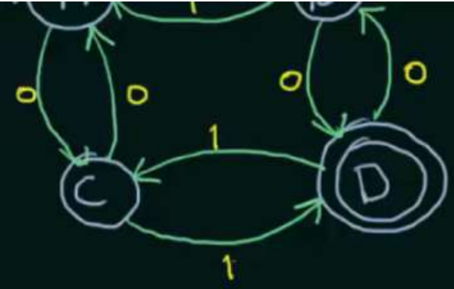


>> In DFA, given the current state
we know what the next state will be

>> It has only one unique next state

>> It has no choices or randomness

>> It is simple and easy to design



Non-deterministic Finite Automata

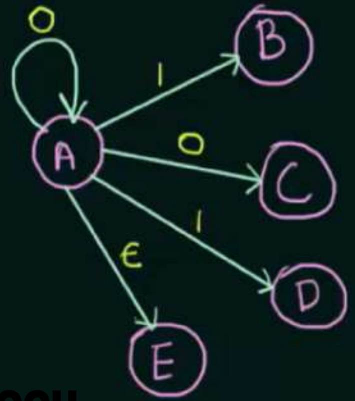
↓

NON-DETERMINISM

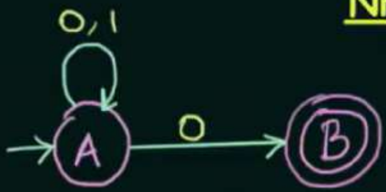
>> In NFA, given the current state
there could be multiple next states

>> 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
$\delta = Q \times \Sigma \rightarrow 2^Q$	–	?

$A \times 0 \rightarrow A$

$A \times 0 \rightarrow B$

$A \times 1 \rightarrow A$

$B \times 0 \rightarrow \phi$

$B \times 1 \rightarrow \phi$

$A \xrightarrow{1} A, B, AB, \phi - 2^2 - 4$

3 States - A, B, C

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

Q = Set of all states — $\{A, B\}$

Σ = inputs — $\{0, 1\}$

q_0 = Start state | initial state — A

F = Set of final states — B

$\delta = Q \times \Sigma \rightarrow \underline{2^Q}$ — ?

$A \times 0 \rightarrow A$

$A \times 0 \rightarrow B$

$A \times 1 \rightarrow A$

$B \times 0 \rightarrow \phi$

$B \times 1 \rightarrow \phi$

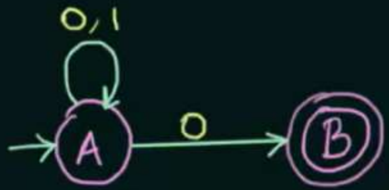
$A \xrightarrow{1} A, B, AB, \phi$ - 2^2 - 4

$\underline{3}$ States - A, B, C

$A \xrightarrow{1} A, B, C, AB, AC, BC, ABC, \phi$

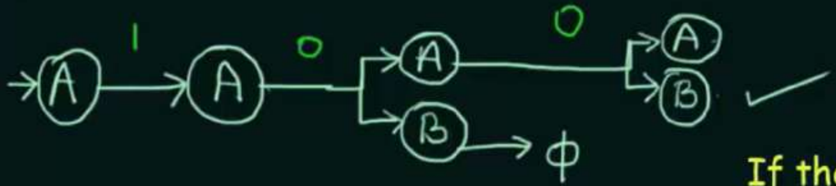
2^3 - 8

NFA - Example-1



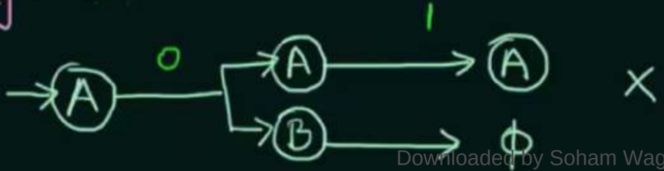
L = { Set of all strings that end with 0 }

Eg. 100



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

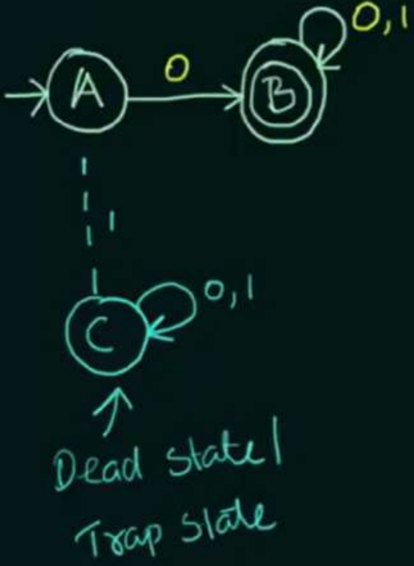
Eg. 01



NFA - Example-2

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

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



Eg. 001 ✓

$$\rightarrow A \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{1} B$$

Eg. 101 X

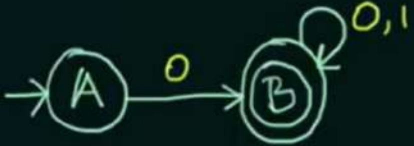
$$\rightarrow A \xrightarrow{1} \phi$$

Dead configuration

NFA - Example-2

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

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



Eg. 001 ✓

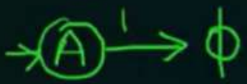
$$\rightarrow A \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{1} B$$

Eg. 101 X

$$\rightarrow A \xrightarrow{1} \phi$$

Dead configuration

Eg. 101 X

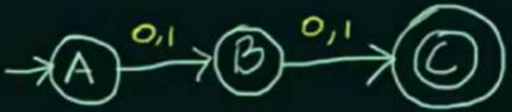


Dead configuration

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

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

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

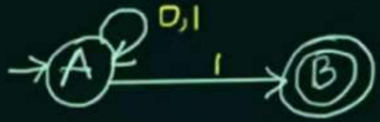


Eg. 00 ✓

Eg. 001 X

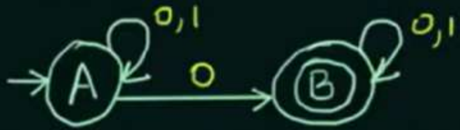
NFA - Example-3

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



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

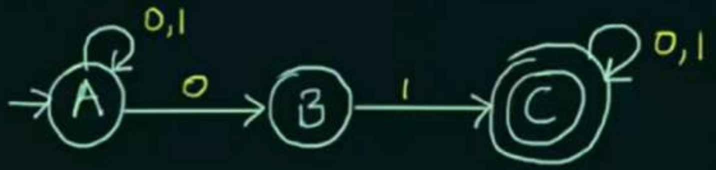
Ex 2) L2 = { Set of all strings that contain '0' }



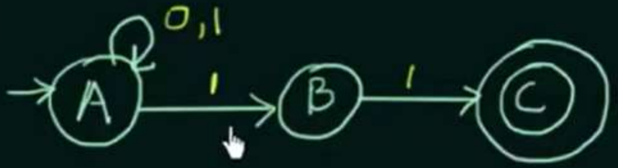
Ex 3) L3 = { Set of all strings that starts with '10' }

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

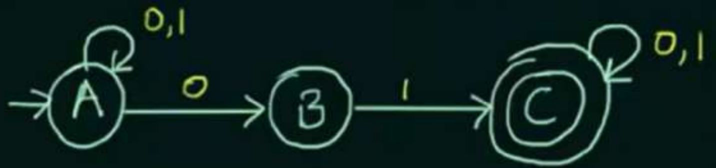
Ex 4) L4 = { Set of all strings that contain '01' }



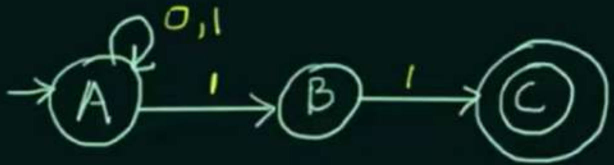
Ex 5) L5 = { Set of all strings that ends with '11' }



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



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



<u>Assignment</u>: 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

$\delta = \underbrace{Q \times \Sigma \rightarrow Q}$

NFA

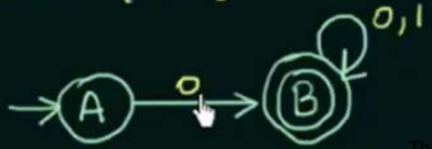
$\delta = \underbrace{Q \times \Sigma \rightarrow 2^Q}$

NFA $\cong$ DFA

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

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

NFA

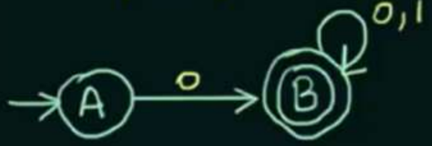


NFA $\cong$ DFA

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

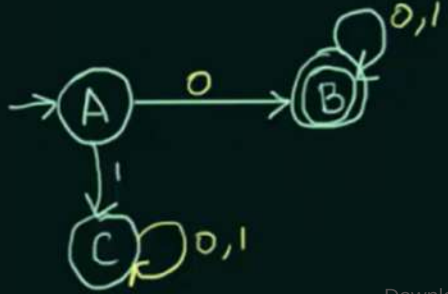
$\Sigma = \{0,1\}$

NFA



	0	1
A	B	ϕ
B	B	B

DFA



	0	1
A	B	C
B	B	B
C	C	C

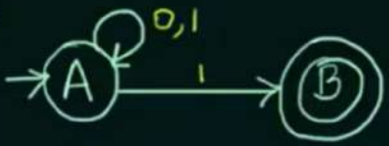
C - Dead State /
Trap State

Conversion of NFA to DFA - Examples (Part 1)

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

$\Sigma = 0, 1$

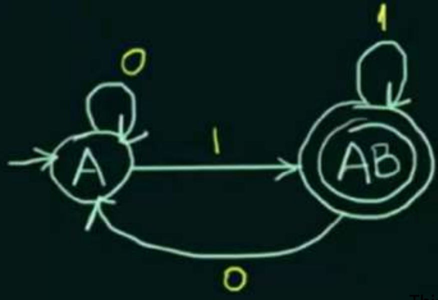
NFA



	0	1
A	{A}	{A, B}
B	ϕ	ϕ

Subset construction method

DFA



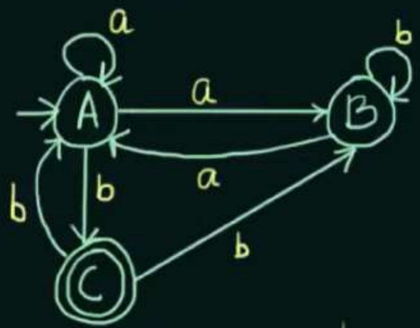
	0	1
A	{A}	{AB}
AB	[illegible]	[illegible]

AB - Single State

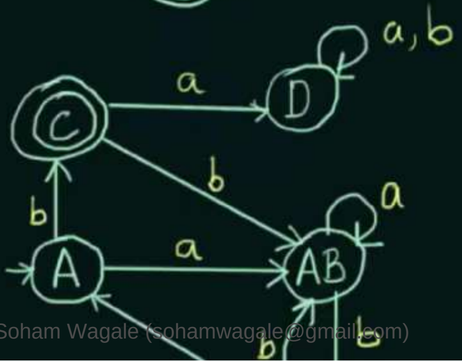
Conversion of NFA to DFA - Examples (Part-2)

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

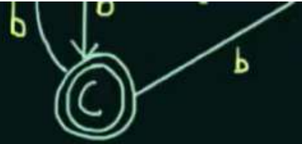
	a	b
→A	A, B	C
B	A	B
C	–	A, B



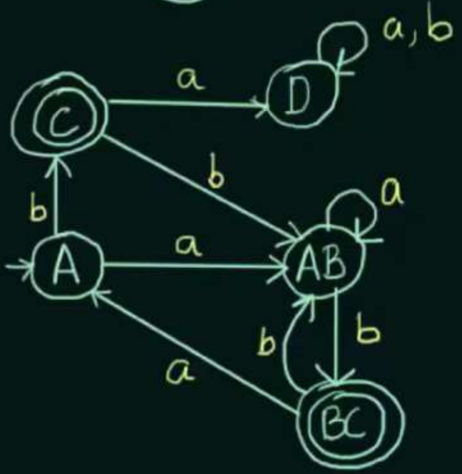
	a	b
→A	AB	C
AB	AB	BC
BC	A	AB
C	D	AB



C	–	A,B



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



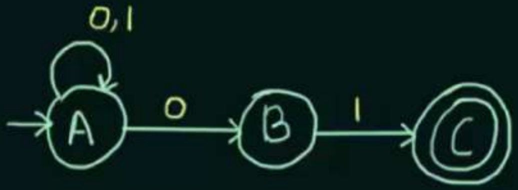
Assignment: Try to find out what does this NFA and its equivalent DFA accept

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

NFA



	0	1
→A	A, B	A
B	ϕ	C
C	ϕ	ϕ

	0	1
→A	AB	A
AB	AB	AC
AC	AB	A

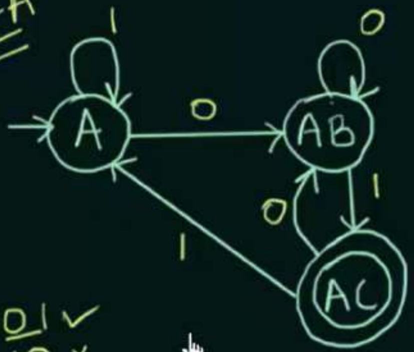
Given below is the NFA for a language

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

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

1101 ✓
1110 ✗

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
1101010

NFA

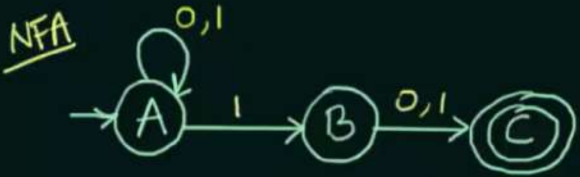
	0	1
→A	A	A, B
B	C	C
C	ϕ	ϕ

DFA

second last symbol is always "1". Then convert it to its equivalent DFA.

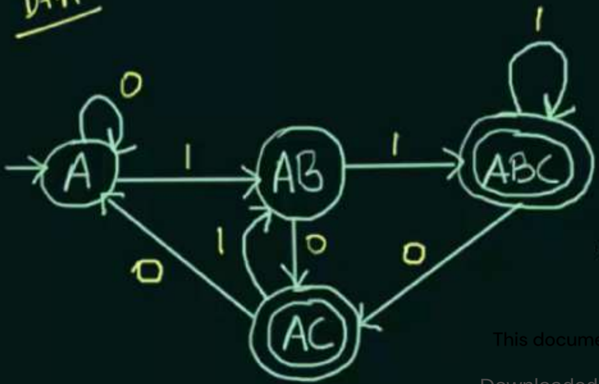
Eg. 1010 ✓
110 ✓
1101010 ✓

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	A[illegible]	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

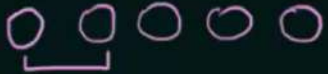
5 States   4 States

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

$$\delta(A, X) \rightarrow F \quad \text{and} \quad \delta(B, X) \rightarrow F$$

OR

$$\delta(A, X) \not\rightarrow F \quad \text{and} \quad \delta(B, X) \not\rightarrow F$$

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

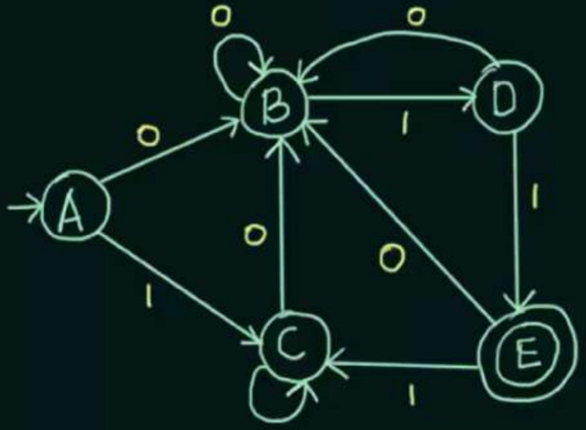
If $|X| = 2$, then A and B are said to be 2 equivalent

⋮

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

00:29:39.54
hr min sec

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}

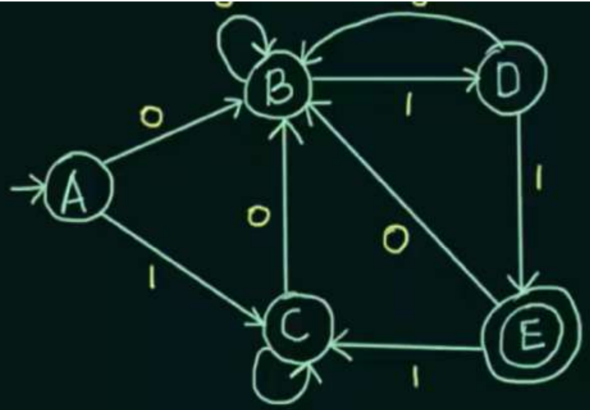
1 Equivalence : {A, B, C} {D} {E}

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

A, B ✓

A, C ✓

C, D ✗



	0	1
→A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

00:37:03.10
hr min sec

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 Equivalence : $\{A, B, C, D\}$ $\{E\}$

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

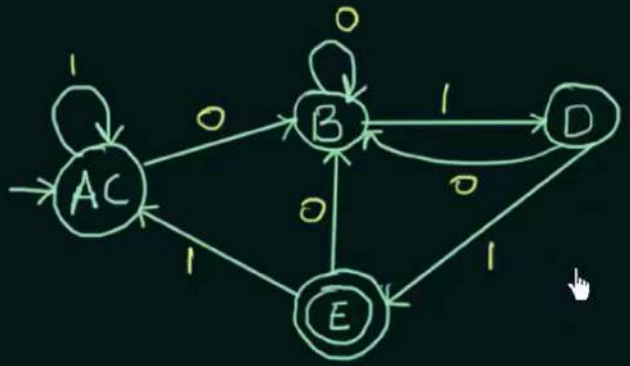
2 Equivalence : $\{A, C\}$ $\{B\}$ $\{D\}$ $\{E\}$

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
$\rightarrow q_0$	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

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

$\rightarrow q_0$	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

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

1- Equivalence

$\{q_0, q_4, q_6\}$

$\{q_1, q_7\}$

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

2- Equivalence

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

3- Equivalence

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

00:44:55.17
hr min sec

q_7	q_6	q_2

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

3- Equivalence

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

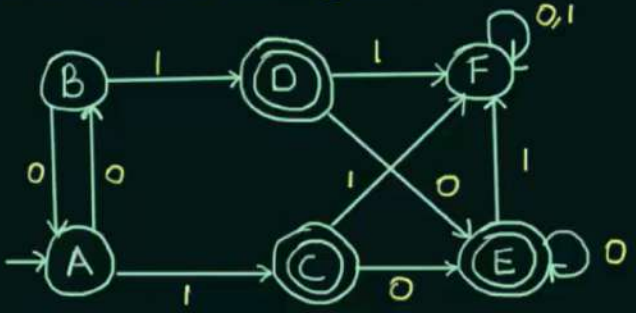
	0	1
$\rightarrow q_0$	q_1	q_5
q_1	q_6	q_2
q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

	0	1
$\rightarrow \{q_0, q_4\}$	$\{q_1, q_7\}$	$\{q_3, q_5\}$
$\{q_6\}$	$\{q_6\}$	$\{q_0, q_4\}$
$\{q_1, q_7\}$	$\{q_6\}$	$\{q_2\}$
$\{q_3, q_5\}$	$\{q_2\}$	$\{q_6\}$
$\{q_2\}$	$\{q_0, q_4\}$	$\{q_2\}$

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}

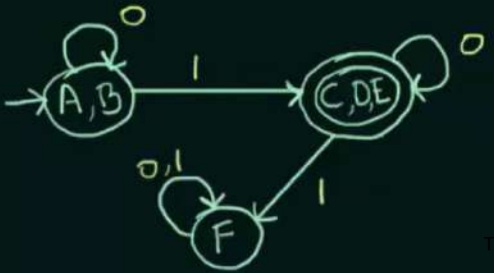
2-Equivalence – {A, B} {F} {C, D, E}

→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}

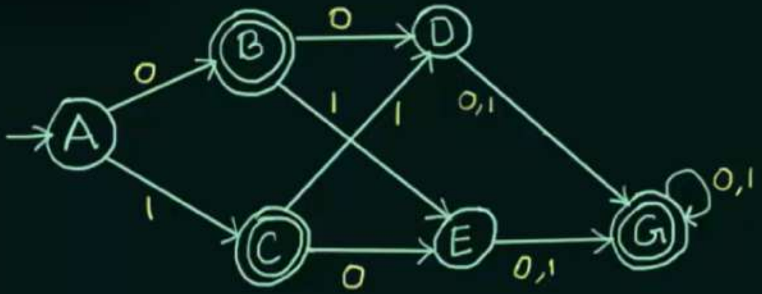


	0	1
→{A, B}	{A, B}	{C, D, E}
{F}	{F}	{F}
{C, D, E}	{C, D, E}	{F}

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}

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

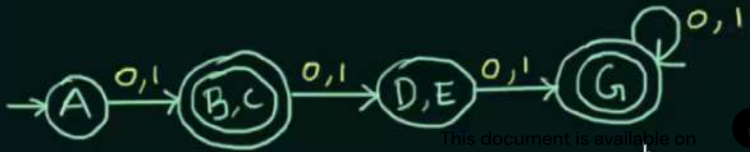
Initial State

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}

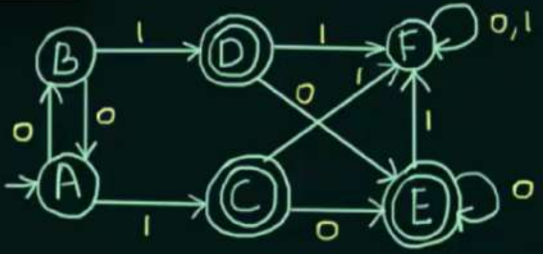


	0	1
→{A}	{B,C}	{B,C}
{D,E}	{G}	{G}
{B,C}	{D,E}	{D,E}
{G}	{G}	{G}

00:39:49.71
hr min sec

Minimization of DFA - Table Filling Method

(Myhill-Nerode Theorem)



	A	B	C	D	E	F
A						
B						
C	✓	✓				
D	✓	✓				
E	✓	✓				
F			✓	✓	✓	

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

Myhill Nerode Theorem - Table Filling Method

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

	A	B	C	D	E
B					
C	✓	✓			
D	✓	✓			
E	✓	✓			
F	✓	✓	✓	✓	✓

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

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

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

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

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

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

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

00:08:05.93
hr min sec

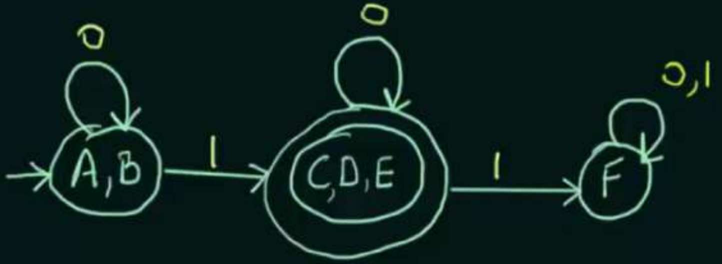
15:49 / 19:39

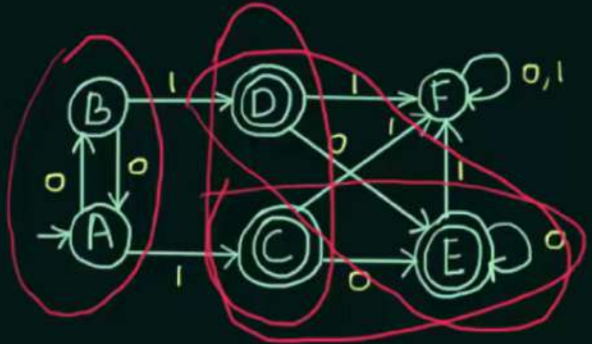
Scroll for details

$\delta(D,0) = E\}$ $\delta(D,1) = F\}$ $\delta(B,0) = A\}$

(B,A) - $\left.\begin{array}{l}\delta(B,0) = A \\ \delta(A,0) = B\end{array}\right\}$ $\left.\begin{array}{l}\delta(B,1) = D \\ \delta(A,1) = C\end{array}\right\}$

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

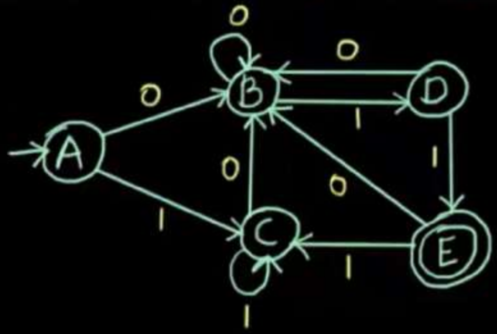




Minimization of DFA - Table Filling Method (Myhill Nerode Theorem)

(Example)

Minimize the following DFA using Table Filling Method



	A	B	C	D	E
A					
B					
C					
D	✓	✓			
E	✓	✓	✓	✓	

(B,A) - $\left.\begin{array}{l}\delta(B,1) = D \\ \delta(A,1) = C\end{array}\right\}$ $\left.\begin{array}{l}\delta(B,0) = B \\ \delta(A,0) = B\end{array}\right\}$

(C,B) - $\left.\begin{array}{l}\delta(C,0) = B \\ \delta(B,0) = B\end{array}\right\}$ $\left.\begin{array}{l}\delta(C,1) = C \\ \delta(B,1) = D\end{array}\right\}$

(D,B) - $\left.\begin{array}{l}\delta(D,0) = B \\ \delta(B,0) = B\end{array}\right\}$

(C,A) - $\left.\begin{array}{l}\delta(C,0) = B \\ \delta(A,0) = B\end{array}\right\}$ $\left.\begin{array}{l}\delta(C,1) = C \\ \delta(A,1) = C\end{array}\right\}$

(D,A) - $\left.\begin{array}{l}\delta(D,0) = B \\ \delta(\text{[illegible]},0) = B\end{array}\right\}$ $\left.\begin{array}{l}\delta(D,1) = E \\ \delta(A,1) = C\end{array}\right\}$ $\left.\begin{array}{l}\delta(D,1) = E \\ \delta(B,1) = D\end{array}\right\}$

$(C,A) - \left.\begin{matrix}\delta(C,0)=B \\ \delta(A,0)=B\end{matrix}\right\} \quad \left.\begin{matrix}\delta(C,1)=C \\ \delta(A,1)=C\end{matrix}\right\}$

$\underline{(D,A)} - \left.\begin{matrix}\delta(D,0)=B \\ \delta(A,0)=B\end{matrix}\right\} \quad \left.\begin{matrix}\delta(D,1)=E \\ \delta(A,1)=C\end{matrix}\right\} \quad \left.\begin{matrix}\delta(D,1)=E \\ \delta(B,1)=D\end{matrix}\right\}$

$\underline{(D,C)} - \left.\begin{matrix}\delta(D,0)=B \\ \delta(C,0)=B\end{matrix}\right\} \quad \left.\begin{matrix}\delta(D,1)=E \\ \delta(C,1)=C\end{matrix}\right\}$

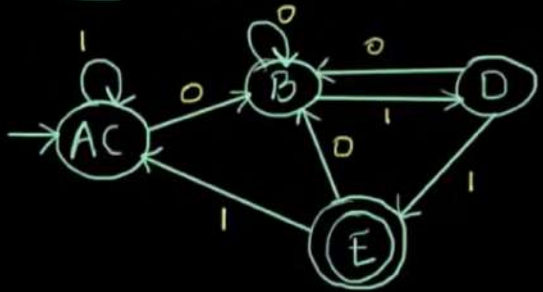
$\underline{(B,A)} - \left.\begin{matrix}\delta(B,0)=B \\ \delta(A,0)=B\end{matrix}\right\} \quad \left.\begin{matrix}\delta(B,1)=D \\ \delta(A,1)=C\end{matrix}\right\}$

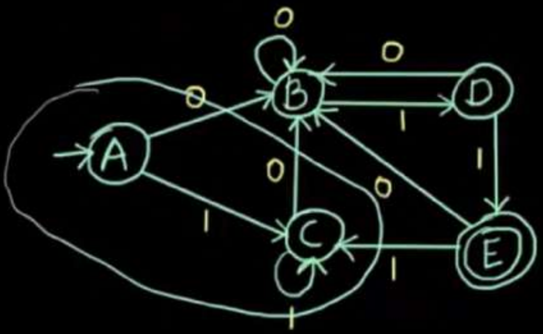
$(C,A) - \left.\begin{matrix}\delta(C,0)=B \\ \delta(A,0)=B\end{matrix}\right\} \quad \left.\begin{matrix}\delta(C,1)=C \\ \delta(A,1)=C\end{matrix}\right\}$

$\underline{(C,B)} - \left.\begin{matrix}\delta(C,0)=B \\ \delta(B,0)=B\end{matrix}\right\} \quad \left.\begin{matrix}\delta(C,1)=C \\ \delta(B,1)=D\end{matrix}\right\}$

$(C,A) - \left.\begin{matrix}\delta(C,0)=B \\ \delta(A,0)=B\end{matrix}\right\} \quad \left.\begin{matrix}\delta(C,1)=C \\ \delta(A,1)=C\end{matrix}\right\}$

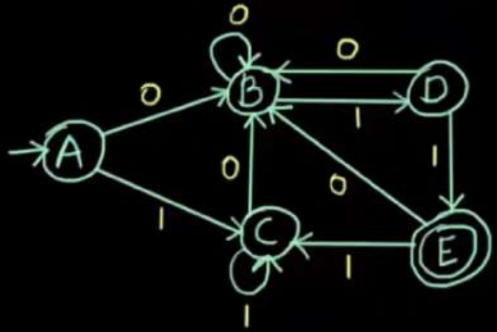
$\underline{A,C} \quad \underline{B} \quad \underline{D} \quad \underline{E}$





(Example)

Minimize the following DFA using Table Filling Method



	A	B	C	D	E
A					
B	✓				
C		✓			
D	✓	✓	✓		
E	✓	✓	✓	✓	

00:09:20.80
hr min sec

(B,A) - $\left.\begin{matrix}\delta(B,1)=D \\ \delta(A,1)=C\end{matrix}\right\}$ $\left.\begin{matrix}\delta(B,0)=B \\ \delta(A,0)=B\end{matrix}\right\}$

(C,B) - $\left.\begin{matrix}\delta(C,0)=B \\ \delta(B,0)=B\end{matrix}\right\}$ $\left.\begin{matrix}\delta(C,1)=C \\ \delta(B,1)=D\end{matrix}\right\}$

(D,B) - $\left.\begin{matrix}\delta(D,0)=B \\ \delta(B,0)=B\end{matrix}\right\}$

(C,A) - $\left.\begin{matrix}\delta(C,0)=B \\ \delta(A,0)=B\end{matrix}\right\}$ $\left.\begin{matrix}\delta(C,1)=C \\ \delta(A,1)=C\end{matrix}\right\}$

(D,A) - $\left.\begin{matrix}\delta(D,0)=B \\ \delta(A,0)=B\end{matrix}\right\}$ $\left.\begin{matrix}\delta(D,1)=E \\ \delta(A,1)=C\end{matrix}\right\}$ $\left.\begin{matrix}\delta(D,1)=E \\ \delta(B,1)=D\end{matrix}\right\}$

(D,C) - $\left.\begin{matrix}\delta(D,0)=B \\ \delta(C,0)=B\end{matrix}\right\}$ $\left.\begin{matrix}\delta(D,1)=E \\ \delta(C,1)=C\end{matrix}\right\}$

(B,A) - $\delta(B,0)=B$ $\left.\begin{matrix}\delta(B,1)=D \\ \delta(A,1)=C\end{matrix}\right\}$

(C,A) - $\left.\begin{matrix}\delta(C,0)=B \\ \delta(A,0)=B\end{matrix}\right\}$

$\delta(C,1)=C$

Finite Automata With Outputs

MEALY MACHINE

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

where

Q = Finite Set of States

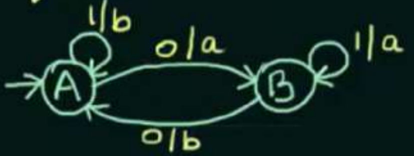
Σ = Finite non-empty set of Input Alphabets

Δ = The set of Output Alphabets

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

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

q_0 = Initial State / Start State



MOORE MACHINE

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

where

Q = Finite Set of States

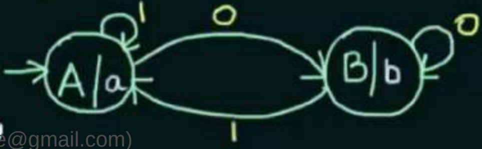
Σ = Finite non-empty set of Input Alphabets

Δ = The set of Output Alphabets

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

λ = Output function: $Q \to \Delta$

q_0 = Initial State / Start State



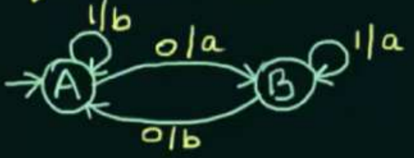
Alphabets

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



Eg. 1010

$\rightarrow A \rightarrow A \rightarrow B \rightarrow B \rightarrow A$

b a a b

n − n

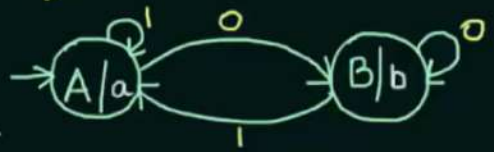
Alphabets

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



Eg. 1010

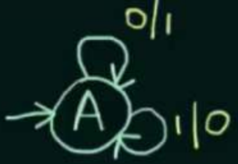
$\rightarrow A \rightarrow A \rightarrow B \rightarrow A \rightarrow B$

a a b a b

$n \rightarrow n+1$

Construction of Mealy Machine

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

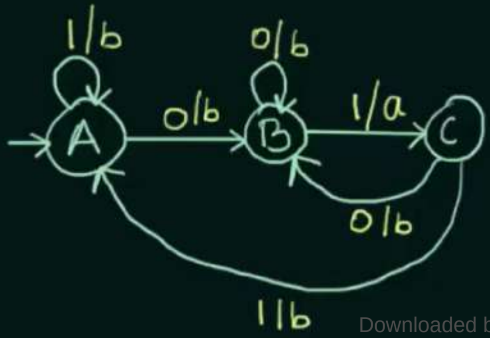


10100

01011

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

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



0110

babb

1000

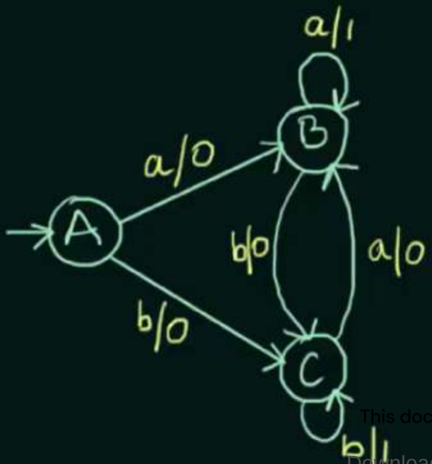
bbbb

Construction of Mealy Machine - Examples (Part-1)

Design a Mealy Machine accepting the language consisting of strings from Σ^*, where $\Sigma = \{a,b\}$ and the strings should end with either aa or bb

aa - 1

bb - 1



abb

baa

ba

00

aa

01

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' complement = 1's complement + 1

MSB ← LSB

Eg. 10100
1's. 01011
+1
2's = 01100

Eg. 11100
1's- 00011
+1
2's = 00100

Eg. 1111
1's 0000
+1
2's 0001