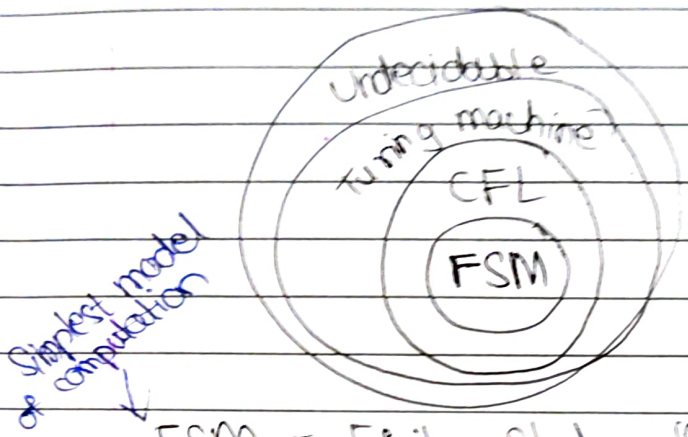
Theory of Computations



FSM - Finite State Machine.

CFL - Context Free Language
↳ This actually is a set of strings and not a language

- 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}

L_3 = set of all strings that begin with 0

infinite set ↓

= {0, 00, 01, 000, 001, 010, 011
, 0000, ... }

Powers of Σ

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

- Σ^0 = set of all strings of length zero
  $\therefore \Sigma^0 = \{\epsilon\}$
- 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

No. of elements in a set

Cardinality of $\Sigma^n = 2^n$

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

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

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

↳ infinite set.

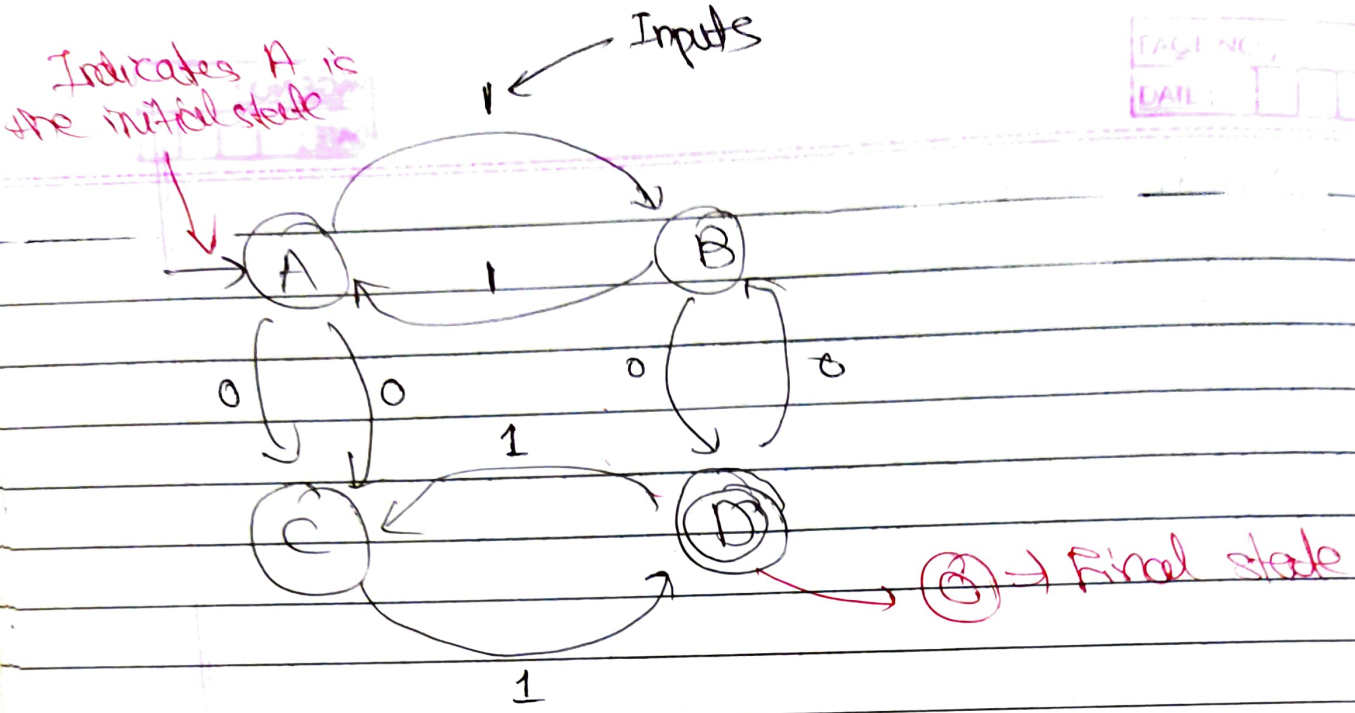
FSM / Finite Automata

Finite Automata

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

DFA (Deterministic Finite Automata)

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



Every DFA can be defined using these 5 tuples:
$(Q, \Sigma, q_0, F, \delta)$

Q = set of all states
Σ = inputs
q_0 = start state / initial state
F = set of final states
δ = Transition function that maps from $Q \times \Sigma \to Q$

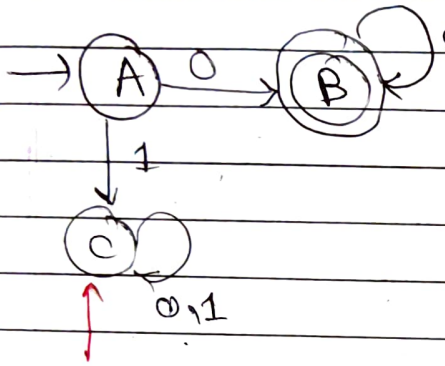
$Q = \{A, B, C, D\}$
$\Sigma = \{0, 1\}$
$q_0 = A$
$F = \{D\}$
$\delta =$

δ	0	1
A	C	B
B	D	A
C	A	D
D	B	C

Accepting State = Final State

DFA (Example - 1)

L1 = Set of all strings that start with "0".
= { 0, 00, 01, 000, 010, 011, 0000, ... }



Dead state / Trap State

Eg. 001 ✓

Initial state A → B → B → B

Final state

Eg. 101

A → C → C → C

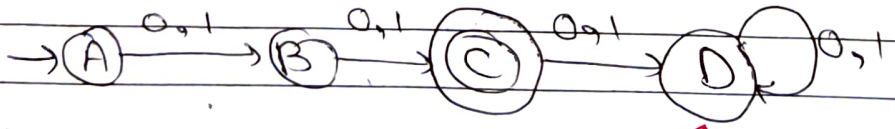
Not the final state

DFA (Example - 2)

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

Σ = {0, 1}

L = { 00, 01, 10, 11 }



Dead / 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

Eg. 1

A → B — Not final State

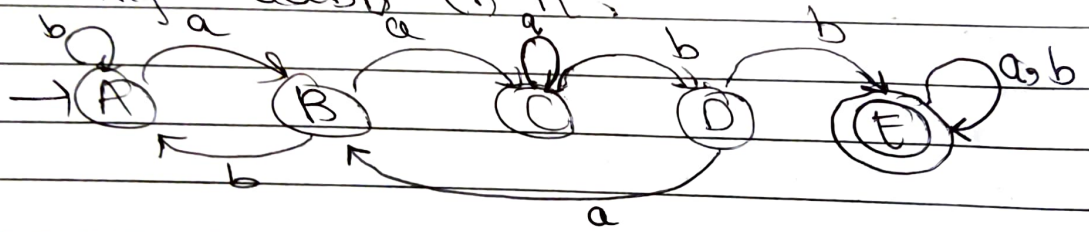
DFA (Example - 3)

Construct a DFA that accepts any string 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 any string over {a, b} that contains the string aabb in it.

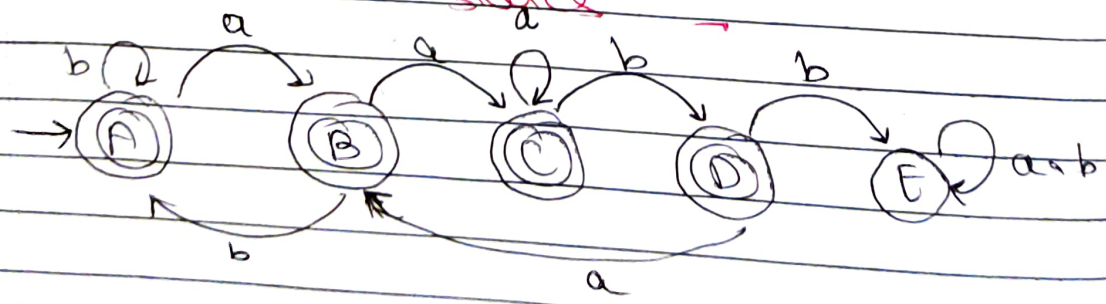


Send it to the state where you get ur 1st a

Now flip the states

i.e. make the final state into non-final state and make the non-final states into final states.

⇒

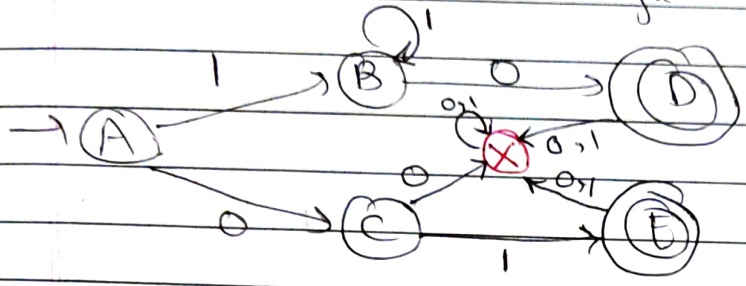


Memory of FSM is very limited.
It cannot store or count strings.

DFA (Example - 2)

X = Dead State

Given a DFA, how to figure out what a DFA recognizes?



10 ✓

111110 ✓
A B D

01 ✓

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

Eg. 001 , 010 , 011 , 1101 , 1100

Regular languages

- A language is said to be a REGULAR LANGUAGE iff some FSM recognizes it.

NOT REGULAR languages
>> Which are not recognized by any FSM
>> Which require memory.

Eg. abababababb ← Cannot be recognized by FSM (Hence not regular).

Eg. $a^N b^N$

eg. aaa bbb eg aaaa bbbb ← cannot be recognized by FSM

Because we have to count how many "a"s were read and then MEMORY comes into picture!!!

Operations on Reg. lang.

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

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

→ $A \cup B = \{pq, r, t, uv\}$

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

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

(ϵ ↓ empty string)

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)

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

Neso

Lecture 8 : Regular Languages

- A language is said to be regular iff some FSM recognizes it.

- The languages
  - which are not recognized by any FSM
  - which require memory

  are NOT regular.

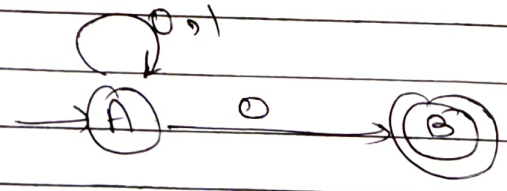
eg. of non-regular languages :-

eg.1. $a^n b^n$ $\quad$ aaaabbbb

eg.2. $a^N b^N$ $\quad$ aaaabbbb

NFA - Formal Definition

eg.



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

Q = Set of all states
Σ = Inputs
q_0 = start state / initial state
F = Set of Final states
δ = $Q \times \Sigma \rightarrow 2^Q$

Here, $Q = \{A, B\}$
$\Sigma = \{0, 1\}$
$q_0 = A$
$F = \{B\}$

A × 0 → A
A × 0 → B
A × 1 → A
B × 0 → ϕ
B × 1 → ϕ

A → A, B, AB, ϕ 2 − 4

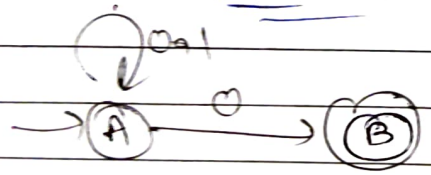
3 states A, B, C

A → A, B, C, AB, AC, BC, ABC, ϕ 3 − 8

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

NFA - Example - 1

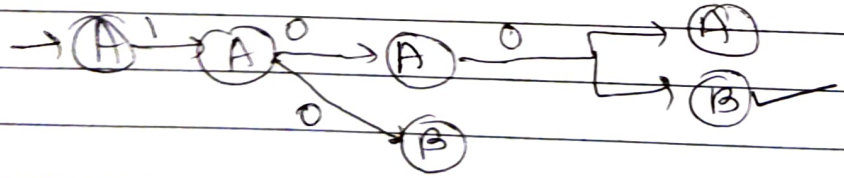
Q.1.



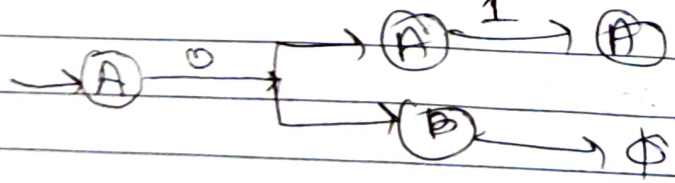
L = { Set of all strings that end with 0 }

→

eg. 100

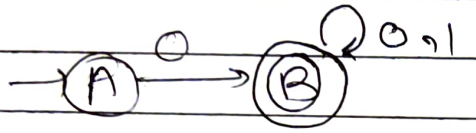


eg. 01



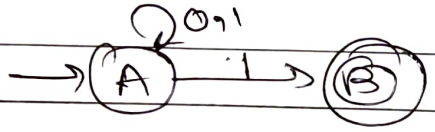
NFA - Eg 2

L = { set of all strings that start with 0 }
= { 0, 00, 01, 000, ... }



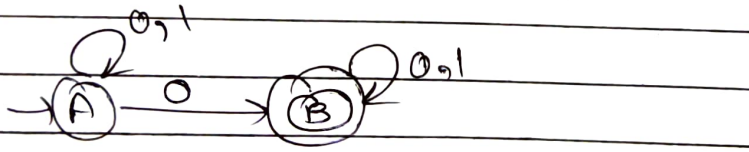
NFA - Example - 3

1) L1 = { set of all strings that ends with "1" }



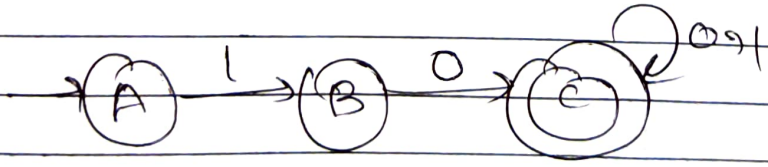
2) L2 = { set of all strings that contain "0" }

→

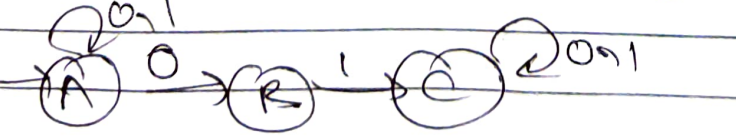


3) L3 = { set of all strings that starts with "10" }

→

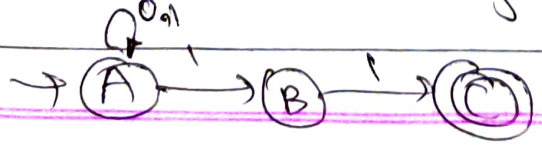


4) L4 = { set of all strings that contain "01" }



5) L5 = { set of all strings w that end with "11" }

→



PAGE NO.
DATE

Conversion of NFA to DFA

Every DFA is an NFA, but not vice versa.

But there is an equivalent DFA for every NFA.

DFA $\delta = Q \times \Sigma \to Q$

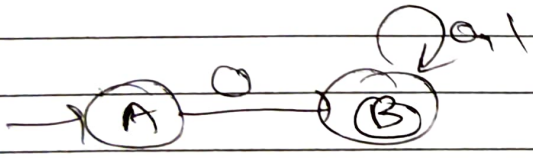
NFA $\delta = Q \times \Sigma \to 2^Q$

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

→

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

NFA



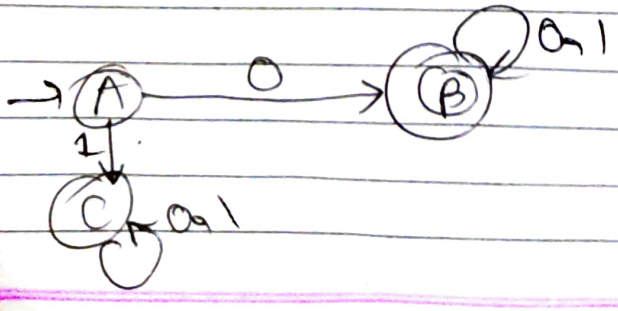
	0	1
A	B	ϕ
B	B	B

State Transition Diagram

DFA

	0	1
A	B	C
B	B	B
C	C	C

C - Dead State / Trap State



Minimization of DFA

2 states can be combined when they are equivalent.

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

$\delta(A,x) \to F$ and $\delta(B,x) \to F$ OR $\delta(A,x) \not\to F$ and $\delta(B,x) \not\to F$

where 'x' is any input string

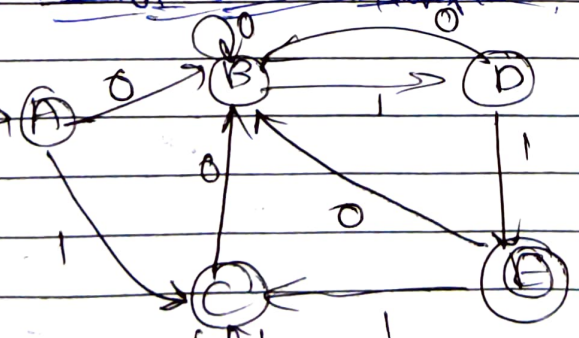
If $|x| = 0$, then A and B are said to be 0 equivalent.

:

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

Min. of DFA - Examples

Q.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}

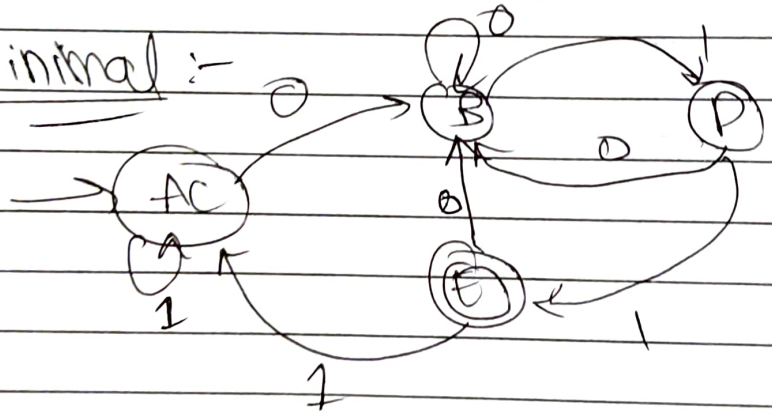
1 Equivalence : {A, B, C} {D} {E}

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

A, B X
A, C
C, D X

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

minimal :-



Min. of DFA (Part-2)

0/ Equivalence : {q0, q1, q2, q3, q4, q5, q6, q7} {q8}

1 Equival. : {q0, q4} {q1, q7} {q2, q5} {q3, q6} {q8}

Min. of DFA (Unreachable states involved)

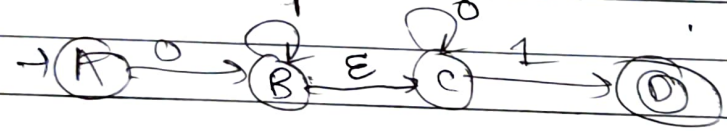
Just remove the unreachable state from DFA

ε-NFA

ε-NFA
↳ empty symbols

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

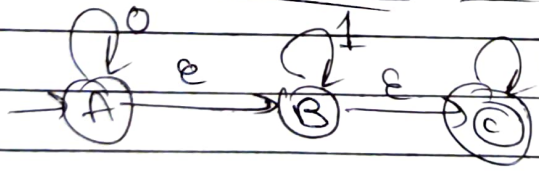
$\delta : Q \times \Sigma \cup \epsilon \rightarrow 2^Q$



eg.

★ Every state on ε goes to itself.

Conversion of ε-NFA to NFA



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

	ε^*	0	ε^*
★A	εA	A	A, B, C
	B	ϕ	
	C	C	C
A	A	ϕ	B, C
	B	B	
	C	C	C

similarly do for B & C.

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

All are final states.