FLA : Formal Language and Automata

↓

Foundation of Theory of Computation (ToC)

solve problem using computer

↳ What can be computed?

↳ What are the limitations of computation?

↳ How difficult to compute it?

To address these, we have different computational models

↓

abstract computer.

ex of computational models :

Finite Automata (FA) ⟶

Pushdown " (PDA)

Turing machine (TM)

Abstract & mathematical model that shows & solves mathematical proofs & problems that come up during execution of program.

Formal Languages

Problems are described as formal languages

Problem ⟶ Computational Model ⟶ accepted (solved)

⟶ not accepted (not solved)

⊛ Each computational model is associated with a language.

Classification of Problems

1.) Solvabale / Computable Problem :

∃ computational model for that problem.

2) Unsolvable / Uncomputable Problems :

No computing model exists to accept/compute this type of problem.

① Alphabets, strings, languages, finite rep^n

language (problem) — finite (good)
language (problem) — infinite – rep^n using finite rep^n

⊛ Not all infinite languages can be represented using finite rep^n.

infinite $\xrightarrow{\text{Tools}}$ finite rep^n
ex: Regular Expression

② Grammar : Set of rules to generate the language

Ex: Context Free Grammar (CFG)

Regular Grammar

⊛ Class of langs. generated by Regular Grammar is equivalent to class of langs. represented by Regular Expression which are accepted by FA.

⊛ Class of langs. generated by CFG are accepted by Pushdown Automata.

Simplest form of Automata is FA.

Automata : computational models / abstract computing device that accept diff. kind of languages (problem)

⊛ **Finite Automata** (has no output)

- → NFA (Non-deterministic)
- → DFA (Deterministic)

} Both are equivalent

↳ more flexible to cover wide range of problems

⊛ **Finite Automata** (with output)

- → Moore Machine
- → Mealy Machine

Turing Machine is powerful automata.

Minimisation of FA

language — Construct FA that accepts lang.

→ more than one FA those accept the lang.

↓

we choose FA with min^m no. of states

↓

minimised FA

⇒ FA ⟷ RL ⟷ RE

↑

RG

Properties of RL

Given a language → RL or not

not RL → Pumping Lemma (decides)

⊛ set related properties:
Union, Concatenation, Complement etc.

↳ Simplification CFG (Context-Free Grammar)
↓
rules/productions.

↓
eliminate some unnecessary rules

↳ We can eliminate using some 'Normal Forms'

CNF (Chomsky's NF), GNF (Greibach NF)

Properties of CFL

Given a language: CFL or not

not CFL → Pumping Lemma

⊛ Set related properties

Pushdown Automata (PDA)

↳ more powerful than FA (accepts more no. of problems)

↳ RL < CFL
(FA) (PDA)

↳ Accepts class of CFL.

PDA ⟷ CFL ⟷ CFG.

↳ Uses Stack data structure for storage.

Turing Machine (TMs)

- more powerful computing model / automata.
- can perform various functions (simple)
- combine simple TMs ⟶ Complex functions
- uses tape (infinite) for storage.
- structured Grammar (SG)

Decidability & Undecidability

- Decidability → For a problem there exists an automata
- Undecidability → No automata exits.

Complexity

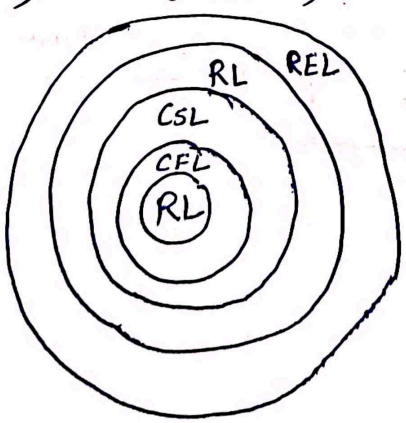
Basic classes : P, NP, NP-Complete

⊛ Using NP-Complete prob. & Reduction you can prove whether a prob. is NP-Complete or not.

Chomsky Hierarchy

RL ⊂ CFL ⊂ CSL ⊂ RL ⊂ REL

(Regular) (Context-Free) (Context-Sensitive) (Recursive) (Recursive enumerable)



Ex: <man, wolf, goat and cabbage problem> using FA.

initial configuration <M, W, G, C> Left | ▭ | Right ϕ

final configuration ϕ | ▭ | <M, W, G, C>

(*) Capacity of boat : can carry man and only one of other three <W, G, C>

(*) If wolf & goat are left unattended : Wolf can eat goat.

(*) If goat and cabbage are left unattended. : goat can eat cabbage.

State MG – WC → ( )

Input

M : man cross river alone

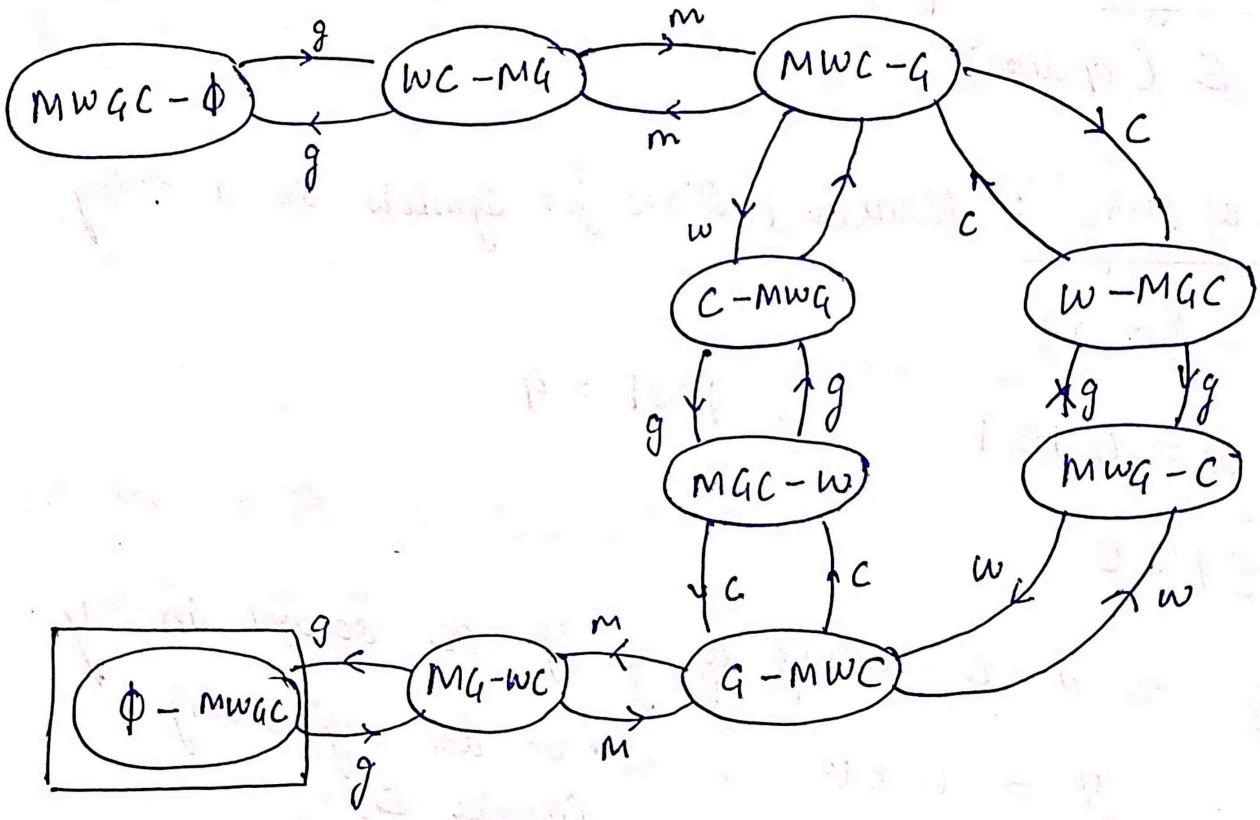
W : man + wolf cross river

g : man + goat " "

C : man + cabbage " "

Initial State MWGC – ϕ $\xrightarrow{\text{Final}}$ ϕ – MWGG

Transition Diagram → FA



Alphabets

finite, non-empty set of symbols

Σ eg: $\Sigma = \{0,1\}$, $\Sigma = \{a,b,c\}$

⊛ Language is collection of strings
↳ collection of alphabets

String : Finite sequence of symbols chosen from some alphabet (Σ)

eg: $\Sigma = \{0,1\}$ $\Sigma = \{a,b,c\}$

$w = 010$ $w = abc, bca, cab$
↑ String

Empty String (string with zero occurence of symbols)

$\hookrightarrow \varepsilon$ (epsilon)

Length of string : Number positions for symbols in a string

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

$w = 0101$, $|w| = 4$

$\Rightarrow |\varepsilon| = 0$

Substring x is a substring of y if x occurs in y

$y = uxv$, u & v are any string (maybe ε)

prefix : prefix of a string is any number of leading symbols of that string.

Suffix : suffix of a string is any number of tailing symbols of that string

eg: $\Sigma = \{a, b, c\}$

$w = abc$ → prefixes $= \{a, ab, abc, \varepsilon\}$

$w = abc$ → suffixes $= \{c, bc, abc, \varepsilon\}$

power of alphabet

alphabet Σ

Σ^k = set of all strings of length k over alphabet Σ

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

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^0 = \{\varepsilon\}$

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

$\Sigma^2 = \{ab, bc, aa, bb, cc, ac, ca, \ldots\}$

$\downarrow$

9

$3 \cdot 3 = 9$

$\Sigma^2 = \Sigma \cdot \Sigma$

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

$\Sigma^2 = \Sigma \cdot \Sigma$

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

= Infinity

Any binary string can be generated from Σ^*

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

$\boxed{\Sigma^* = \Sigma^+ \cup \{\varepsilon\}}$

Concatenation of strings

x & y are two strings. = xy

eg: $x = abc$, $y = ab$ | $xy = abcab$

If w is any string.

$\boxed{\varepsilon \cdot w = w \cdot \varepsilon = w}$

ε is called identity of concatenation.

Language : A language over an alphabet Σ is subset of Σ^*.

$$\Rightarrow \boxed{L \subseteq \Sigma^*}$$

ϕ = empty set = $\{\ \}$

$$\boxed{\phi \neq \{\varepsilon\}}$$

$\hookrightarrow |\phi| = 0$, $|\{\varepsilon\}| = 1$

Example of Languages

↳ Set of all strings over $\Sigma = \{0, 1\}$ that starts with 1.

eg: $L = \{1, 10, 11, 101, 111, ---\}$

↳ Set of all strings over $\Sigma = \{a, b, c\}$ having ac as substring.

$L = \{ac, aac, bac, abacab, --\}$

$w = aab$ $\quad w \notin L$

↳ Set of all strings over $\Sigma = \{0, 1\}$ whose value is prime

$L = \{10, 11, 101, 111, ---\}$

↳ Set of all strings over $\Sigma = \{a, b, \cdots x\}$ that have 'a' at 5th position from the right.

$L = \{abcde, dabcee, abcabcde, ---\}$

Language is a set so all set operations are possible.

Concatenation of languages

L_1 & L_2

$$L_1L_2 = \{ xy \mid x \in L_1 \ \& \ y \in L_2 \}$$

$L_1 = \{0, 1\}$, $L_2 = \{00, 11\}$

$$L_1L_2 = \{000, 011, 100, 111\}$$

⊛ Concatenation of languages is <u>associative</u>.

for 3 langs. L_1, L_2 & L_3.

$$(L_1L_2)L_3 = L_1(L_2L_3)$$

<u>Ex:</u> $L_1 = \{a, ab\}$ $L_2 = \{\varepsilon, bb\}$

$$L_1L_2 = \{a, abb, ab, abbb\}$$

$$\boxed{L_1 \subseteq L_1L_2 \text{ iff } \varepsilon \in L_2}$$

↳ $\varepsilon \in L_2$ iff $L_1 \subseteq L_1L_2$.

L^* (Kleene Star or Kleene Closure of language L)

$$L^* = \bigcup_{i \geq 0} L^i$$

$$= L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$$

$L = \{0, 1\}$

$L^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

↳ Set of all strings over $L = \{0,1\}$

$L^1 = \{0, 1\}$

$L^2 = \{00, 01, 10, 11\}$

$L^3 = L^2 \cdot L = \{000, 001, 010, 011, 100, 101, 110, 111\}$

L^+ (Positive Closure of language L)

$$L^+ = \bigcup_{i \geq 1} L^i = L^1 \cup L^2 \cup L^3 \cup \ldots$$

$L = \{0, 1\}$

$L^+ = \{0, 1, 00, 01, 10, 11, \ldots\}$

$L^* = \{\varepsilon\} \cup L^+$

Ex: Set of all strings over $\{0,1\}$ that start with 0

$L = \{0x \mid x \in \{0,1\}^*\}$ ← Set Representation

Ex: Set of all strings over the alphabet $\Sigma = \{-, -, \ldots\}$ that have ab as substring.

$L = \{xaby \mid x, y \in \Sigma^*\}$

Deterministic Finite Automata

(DFA)

↳ A DFA is denoted by 5 tuples

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

where Q = finite set of states

Σ = finite set of input symbols

δ = transition function

$\delta : Q \times \Sigma \rightarrow Q$ → Mapping

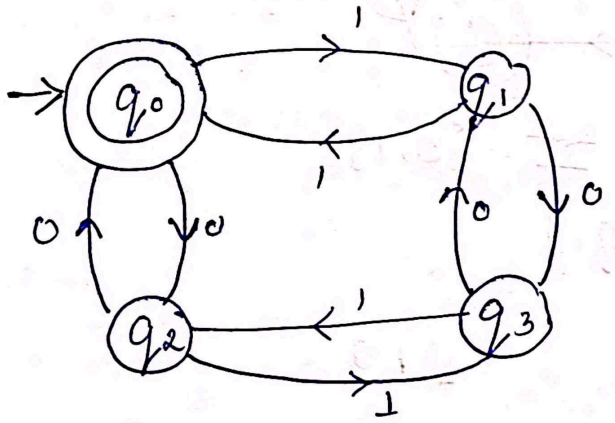
< δ takes arguments as a state & i/p symbol & returns a state >

↳ single

q_0 = Start state

$F \subseteq Q$ = Set of final or accepting states.

Ex:



Transition Diagram of DFA.

↳ $Q = \{q_0, q_1, q_2, q_3\}$

↳ $\Sigma = \{0, 1\}$

↳ $q_0 = q_0$

↳ $F = \{q_0\}$

Final state

Initial state

δ:	0	1
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

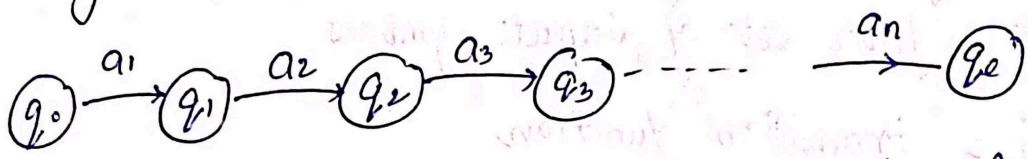
Set of all strings accepted by DFA

= Set of all strings having even no. of 0's & even no. of 1's

DFA is processing string

DFA , $A = (Q, \Sigma, \delta, q_0, F)$

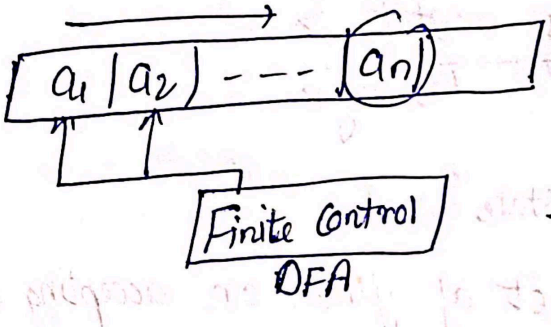
String $w = \overrightarrow{a_1 a_2, \ldots a_n}$, $a_i \in \Sigma$



$q_e \in F$, w is accepted by A.

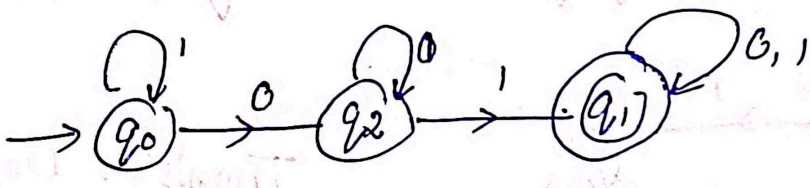
$q_e \notin F$, w is not accepted by A.

Diagram: tape



$\rightarrow \delta(q, a_i) = =$

Ex:



$Q = \{q_0, q_1, q_2\}$ $\Sigma = \{0, 1\}$

$q_0 = q_0$ $F = \{q_1\}$

$L = \{\underline{01}, \underline{101}, 0\underline{01}, 0\underline{01}0 \ldots\}$

= Set of all strings having $\underline{01}$ as substring.

$L = \{x\underline{01}y \mid x, y \in \{0,1\}^*\}$

Non-deterministic Finite Automate (NFA)

↳ We denote NFA by 5 tuples $(Q, \Sigma, \delta, q_0, F)$

where, Q = Finite set of states

Σ = Finite set of i/p symbols

$q_0 \in Q$ = Start state

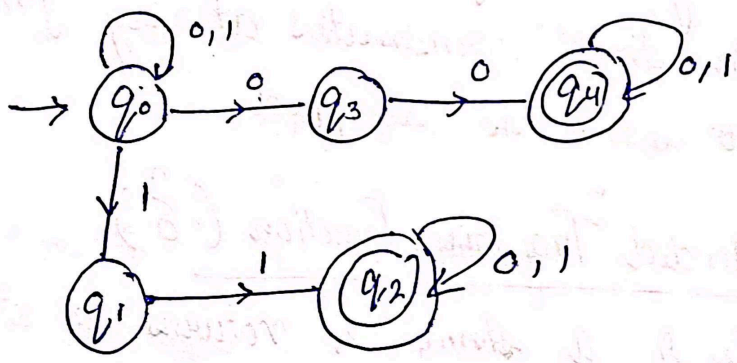
$F \in Q$ = Set of final state

δ = transition function

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

< It has allowed zero, one or more transitions from a state on same i/p symbol >

Ex:



L = Set of all strings having 00 or 11 as substring

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$q_0 = \{q_0\}$

$F = \{q_2, q_4\}$

$L = \{x00y \text{ or } x11y \mid x, y \in \Sigma^*\}$

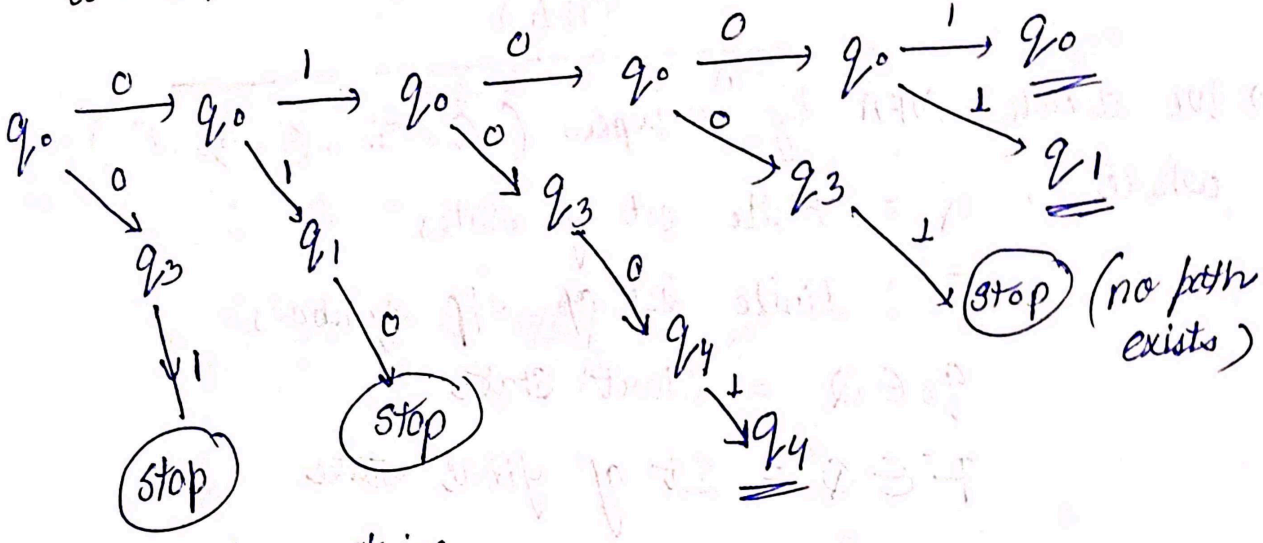
$\delta(q_0, 0) = \{q_0, q_3\}$

$\delta(q_3, 1) = \{\ \}$

Not allowed in DFA but allowed in NFA

δ:	0	1
→ q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	ϕ	$\{q_2\}$
* q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	ϕ
* q_4	$\{q_4\}$	$\{q_4\}$

$w = 01001$



{whole string not processed}

↳ Final states reached $= \{q_0, q_1, q_4\}$

w is accepted by NFA if $\exists$ atleast one path that terminates at any final state than that string is accepted.

Extended Transition Function ($\hat{\delta}$)

$\hat{\delta}$ takes a state & a string & returns a state.

$$\hat{\delta}(q, w) = p$$

$$\left\{ \begin{array}{l} q, p \in Q \\ w \in \Sigma^* \end{array} \right\}.$$

(*) $\hat{\delta}(q, \varepsilon) = q$

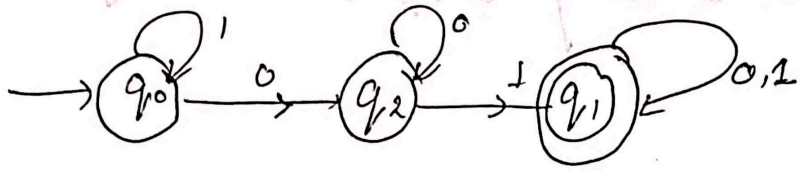
(*) If $w = xa$, $w, x \in \Sigma^*$, $a \in \Sigma$

$$\hat{\delta}(q, w) = \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$$

DFA $\Rightarrow$ If $\hat{\delta}(q, w) \in F$, then w is accepted

If $\hat{\delta}(q, w) \notin F$, then w is not accepted

Ex of DFA



$w = 11101$

$\hat{\delta}(q_0, 11101) = \delta(\hat{\delta}(q_0, 1110), 1) = q_1$

$\hat{\delta}(q_0, 1110) = \delta(\hat{\delta}(q_0, 111), 0) = q_2$

$\hat{\delta}(q_0, 111) = \delta(\hat{\delta}(q_0, 11), 1) = q_0$

$\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = q_0$

$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \varepsilon), 1) = q_0$

$\hat{\delta}(q_0, \varepsilon) = \underline{\underline{q_0}}$

$q_1 \in F$

$\Rightarrow$ w is accepted.

$L = \{ x\underline{01}y \mid x, y \in \{0,1\}^* \}$

Language accepted by DFA / Language of DFA

The language accepted by DFA, $A = (Q, \Sigma, \delta, q_0, F)$ is defined as,

$$L(A) = \{ \omega \in \Sigma^* \mid \hat{\delta}(q_0, \omega) \in F \}$$

Language accepted by DFA / Language of DFA

The language accepted by DFA, $A = (Q, \Sigma, \delta, q_0, F)$ is defined as,

$$L(A) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \}$$

Extended transition function ($\hat{\delta}$) for NFA

$\hat{\delta}(q, w)$

① $\hat{\delta}(q, \varepsilon) = q$

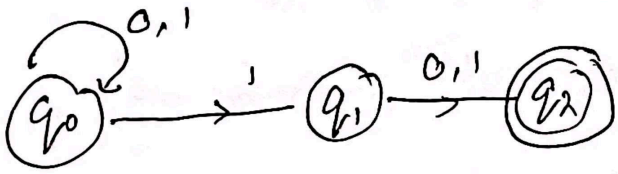
② $w = xa$; $w, x \in \Sigma^*$, $a \in \Sigma$

$$\hat{\delta}(q, w) = \hat{\delta}(q, xa) = \delta(\underline{\hat{\delta}(q, x)}, a)$$

$$= \bigcup_{i=1}^{K} \delta(p_i, a)$$

where $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_K\}$

Ex:



$L = \{ x1y : x \in \{0,1\}^*, y \in \{0,1\} \}$

1 is second last digit.

$w = 0101 0$

$\hat{\delta}(q_0, \varepsilon) = \{q_0\}$

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \varepsilon), 0) = \delta(q_0, 0) = \{q_0\}$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_0, 1) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 010) = \delta(\hat{\delta}(q_0, 01), 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0, q_2\}$

$\hat{\delta}(q_0, 0101) = \delta(\hat{\delta}(q_0, 010), 1) = \delta(q_0, 1) \cup \delta(q_2, 1)$

$= \{q_0, q_1\}$

$\hat{\delta}(q_0, 01010) = \delta(\hat{\delta}(q_0, 0101), 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_2\}$

Language accepted by NFA

Suppose $A = (Q, \Sigma, \delta, q_0, F)$ be a NFA, the language accepted by NFA is defined as $L(A)$

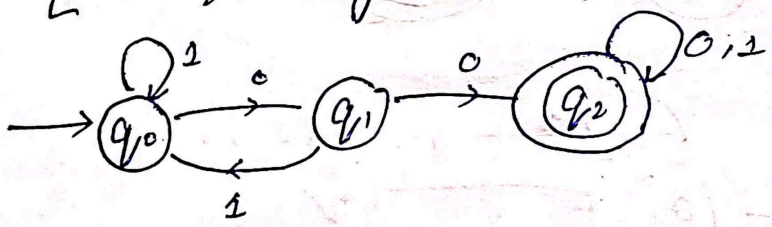
$$L(A) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$$

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

$L = \{ w \mid w$ is a binary string that contains 00 as substring $\}$

Construct corresponding DFA.

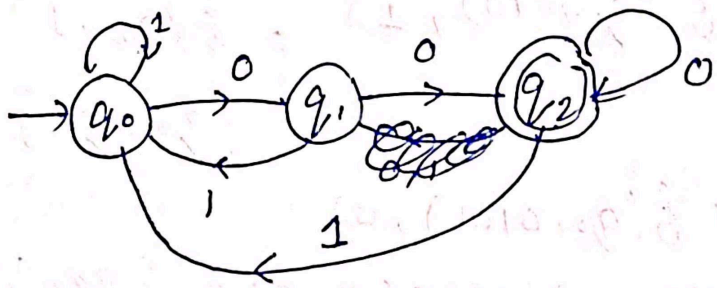
$\Rightarrow$ $L = \{ x00y \mid x, y \in \Sigma^* \}$



	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_2

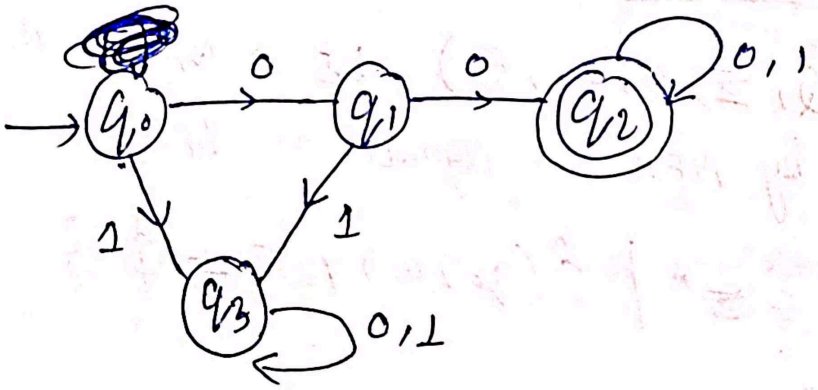
Ex: $\Sigma = \{0, 1\}$

$L = \{ w \mid w \in \Sigma^*$ ending with $00 \}$



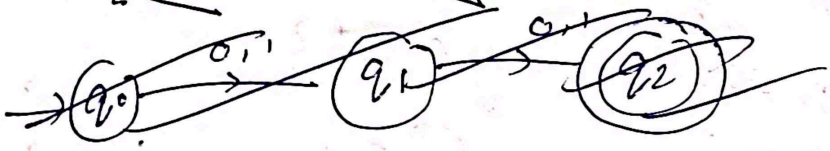
	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

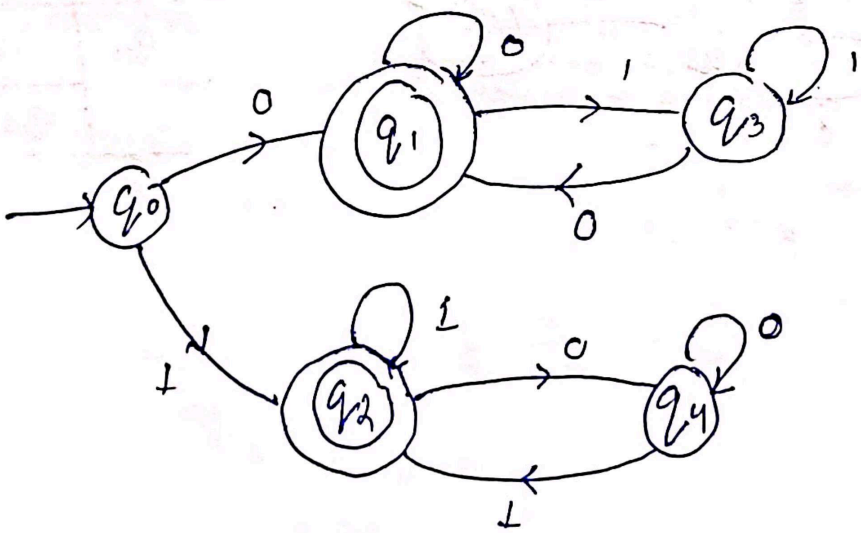
Ex: $L = \{ w \mid w \in \Sigma^*$ starting with $00 \}$



Ex: Construct a DFA, that accepts all binary strings that begin & end with same symbol.

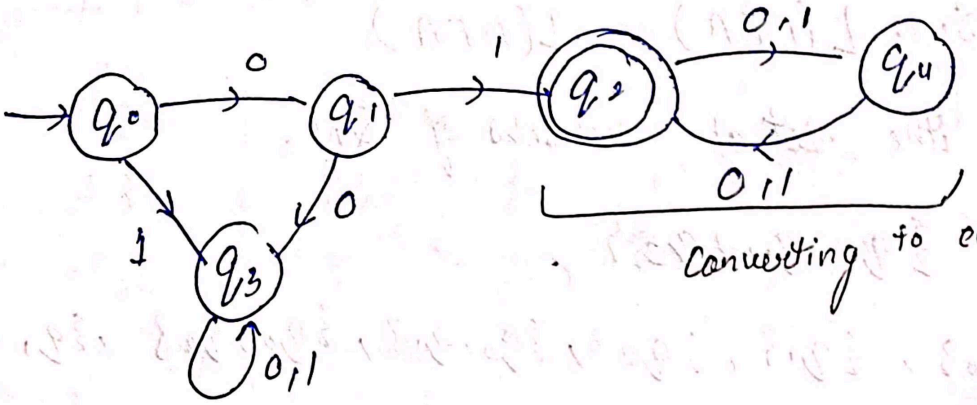
$\cancel{L = \{ axa \mid a \in \Sigma, x \in \Sigma^* \}}$





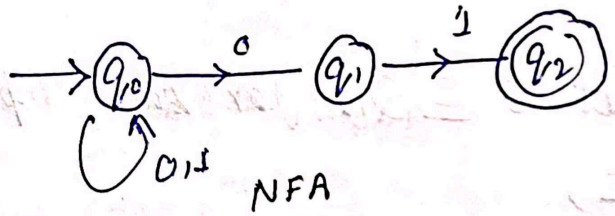
Ex: Build a DFA, to accept the following languages:

$L = \{ w \mid w \in \Sigma^*$ of even length & begins with $01 \}$



Construct a NFA, that accepts the language as

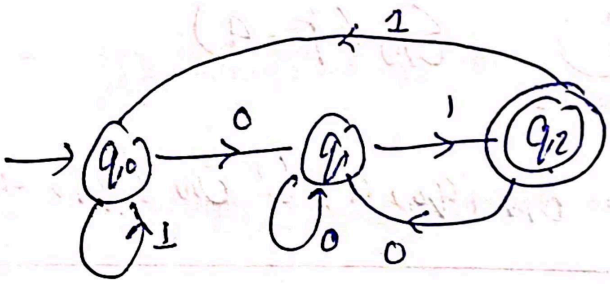
$L = \{ w01 \mid w \in \Sigma^* \}$



NFA

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	ϕ

DFA



Language accepted by NFA $\equiv$ Language accepted by DFA

→ Language accepted by NFA / DFA is Regular Language.

Equivalence b/w NFA & DFA

Given a NFA $= (Q_N, \Sigma, \delta_N, q_0, F_N)$

Design DFA $= (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$

s.t. $L(NFA) = L(DFA)$

↳ Q_D is the set of subsets of Q_N.

$Q_N = \{q_0, q_1, q_2\}$

$Q_D = \{ \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}, \phi \}$

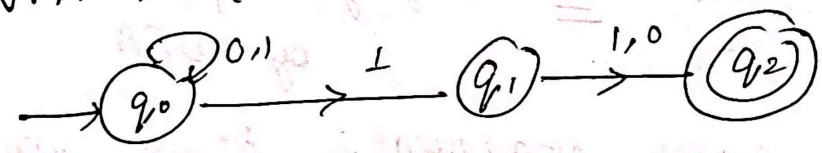
↳ F_D = set of subsets S of Q_N, s.t $S \cap F_N \neq \phi$

↳ δ_D = for each subset $S \subseteq Q_n$ for each input symbol $a \in \Sigma$,

$$\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a)$$

$$\delta_D(\{q_0, q_1\}, a) = \delta_N(q_0, a) \cup \delta_N(q_1, a)$$

Ex: NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$



δ_D	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ

$DFA = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$

	δ_D	0	1
A	ϕ	ϕ	ϕ
→ B	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
C	$\{q_1\}$	$\{q_2\}$	$\{q_2\}$
* D	$\{q_2\}$	ϕ	ϕ
E	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
* F	$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
* G	$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
* H	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

Reachable: B, E, F, H

Unreable: A, C, D, G

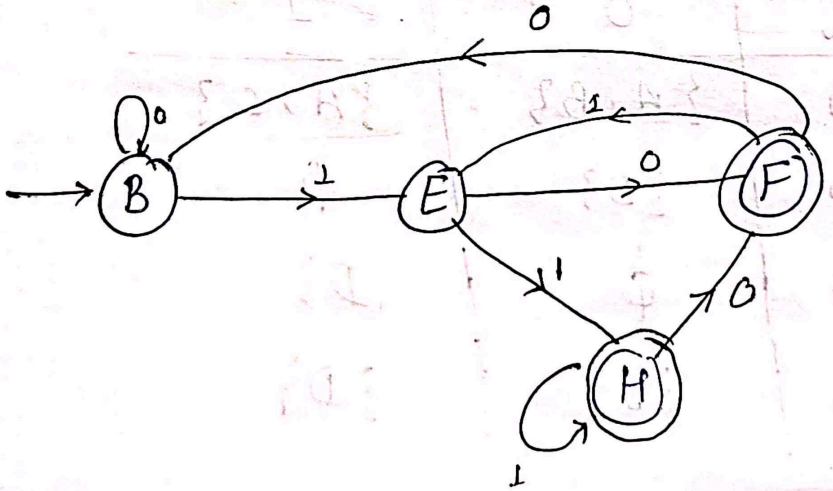
Renaming the states

	0	1
A	A	A
→ B	B	E
C	D	D
* D	A	A
E	F	H
* F	B	E
* G	D	D
* H	F	H

→

δ_D	0	1
→ B	B	E
E	F	H
* F	B	E
* H	F	H

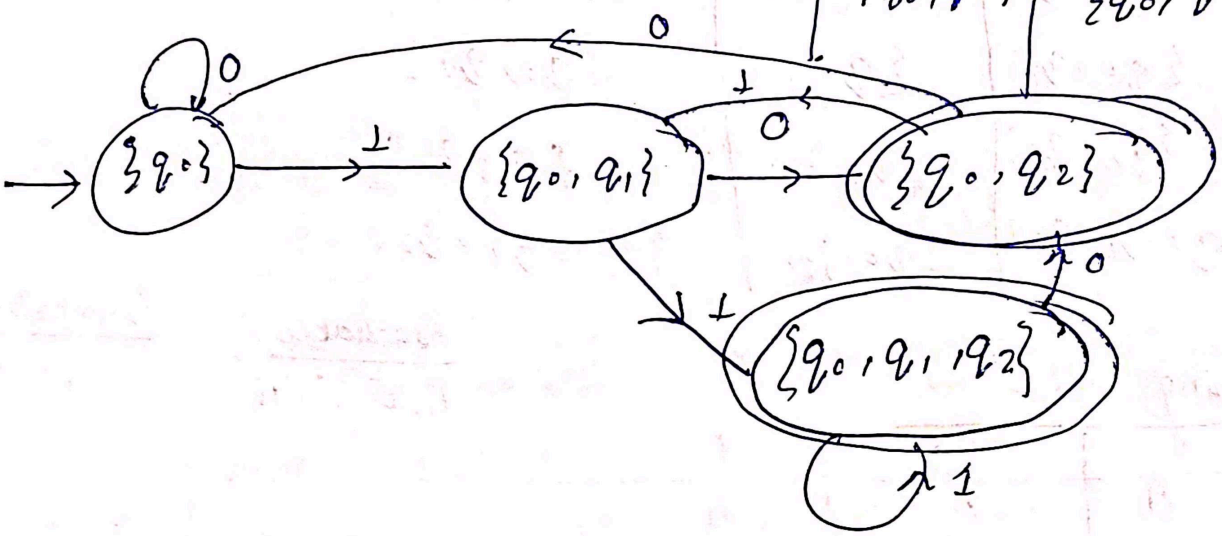
DFA



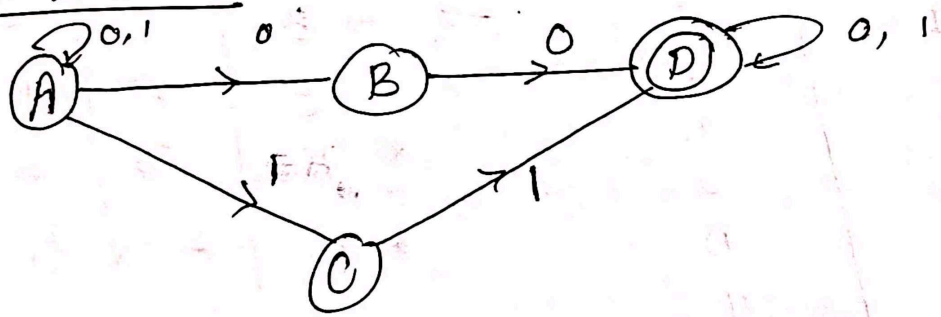
Short way

δ_N	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ

δ_D	0	1
$\rightarrow \{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$* \{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
$* \{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

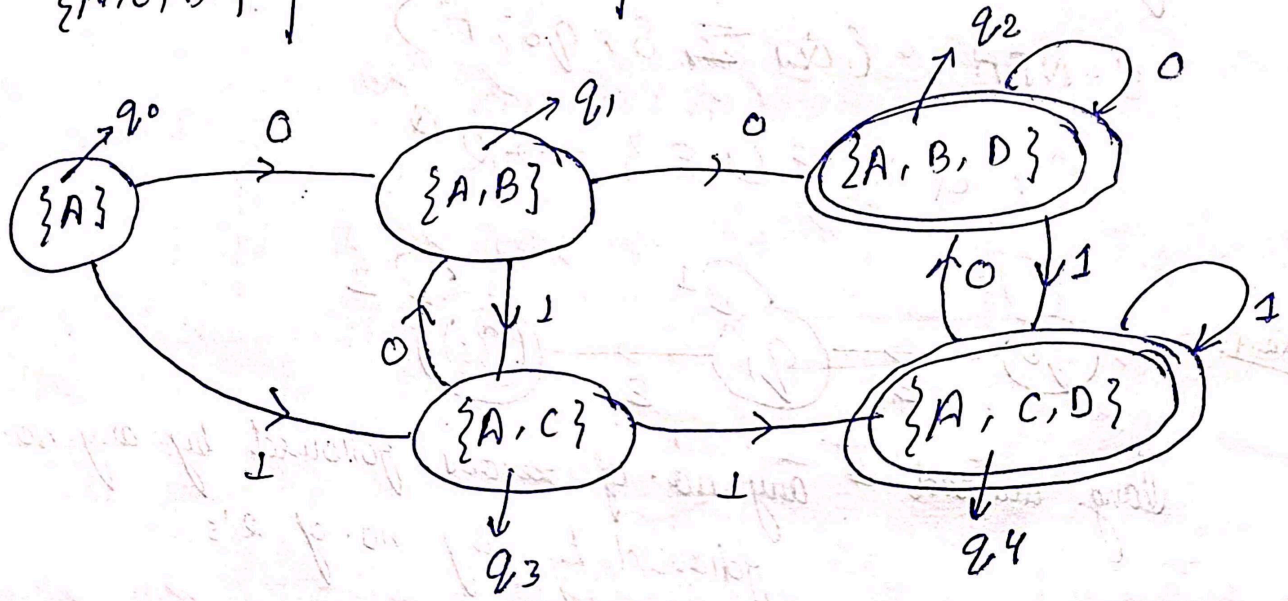


NFA to DFA



δ_N	0	1
$\rightarrow A$	$\{A, B\}$	$\{A, C\}$
B	$\{D\}$	ϕ
C	ϕ	$\{D\}$
$* D$	$\{D\}$	$\{D\}$

S_D	0	1
$\rightarrow \{A\}$	$\{A, B\}$	$\{A, C\}$
$\{A, B\}$	$\{A, B, D\}$	$\{A, C\}$
$\{A, C\}$	$\{A, B\}$	$\{A, C, D\}$
* $\{A, B, D\}$	$\{A, B, D\}$	$\{A, C, D\}$
* $\{A, C, D\}$	$\{A, B, D\}$	$\{A, C, D\}$



$DFA = (Q, \Sigma, S_D, q_0, F)$

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

$q_0 = \{A\}$

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

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

$L = \{\varepsilon, 012, 00112, 0001112222 \ldots\}$

$L = \{0^* 1^* 2^*$

FA with ε-transition (Epsilon transition)

or ε-NFA

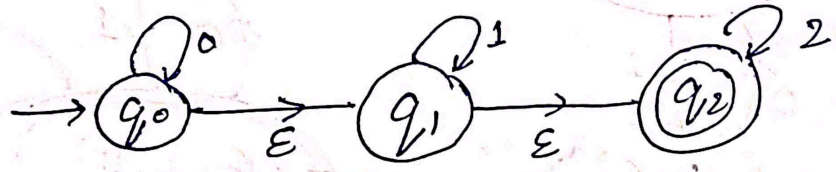
↳ No ε-transition for DFA.

↳ Using ε we can move from one state to other state.

$\varepsilon\text{-NFA} = (Q, \Sigma, \delta, q_0, F)$

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

Example



lang accepted = any no. of zeroes followed by any no. of ones followed by any no. of 2's

$= \{0^* 1^* 2^*\}$

Epsilon Closure (ECLOSE(q))

ECLOSE(q) = Set of states that are reachable from q using only ε transitions.

eg: ECLOSE$(q_0) = \{q_0, q_1, q_2\}$

ECLOSE$(q_1) = \{q_1, q_2\}$

ECLOSE$(q_2) = \{q_2\}$

Extended transition function of ε-NFA

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

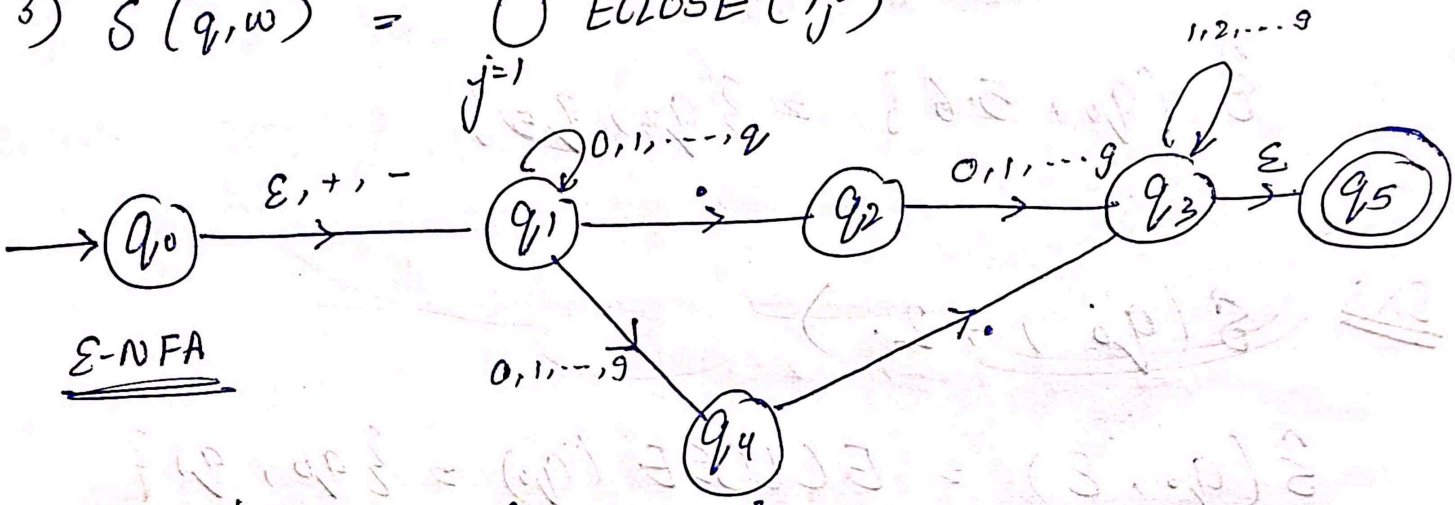
$\hat{\delta}(q, w)$ let $w = xa$, $a \in \Sigma$

follow steps:

1.) Suppose $\hat{\delta}(q, x) = \{p_1, p_2, p_3, \ldots, p_k\}$

2.) Let $\bigcup_{j=1}^{k} (\delta(p_j, a)) = \{r_1, r_2, \ldots r_m\}$

3) $\hat{\delta}(q, w) = \bigcup_{j=1}^{m} ECLOSE(r_j)$



ε-NFA

$\Sigma = \{0, 1, \ldots, 9, +, -, \cdot\}$

$w = 5.6$? $\longrightarrow$ $\hat{\delta}(q_0, 5.6)$

First symbol (5).

$\hat{\delta}(q_0, 5) \rightarrow w = xa$, $x = \varepsilon$, $a = 5$

$\hat{\delta}(q_0, \varepsilon) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 5)$ (i) $\delta(q_0, 5) \cup \delta(q_1, 5)$

$= \{\phi, q_1, q_4\} = \{q_1, q_4\}$

ii) $ECLOSE(q_1) \cup ECLOSE(q_4)$

$= \{q_1, q_4\}$

$\hat{\delta}(q_0, \underline{5}\cdot) \quad \{q_1, q_4\}$

↳ (i) $\delta(q_1, \cdot) \cup \delta(q_4, \cdot) = \{q_2, q_3\}$

(ii) $ECLOSE(q_2) \cup ECLOSE(q_3) = \{q_2, q_3, q_5\}$

$\hat{\delta}(q_0, \underline{5}\cdot 6) =$ (i) $\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6)$

$= \{q_3\}$

ii) $ECLOSE(q_3) = \{q_3, q_5\}$

$\hat{\delta}(q_0, 5\cdot 6\} = \{q_3, q_5\}$

<u>Ex 2</u> $\hat{\delta}(q_0, +2\cdot)$

$\hat{\delta}(q_0, \varepsilon) = ECLOSE(q_0) = \{q_0, q_1\}$

$\hat{\delta}(q_0, +) \longrightarrow$ (1) $\delta(q_0, +) \cup \delta(q_1, +)$

$= \{q_1\}$

↳ (11) $ECLOSE(q_1) = \{q_1\}$

$\hat{\delta}(q_0, +2) \longrightarrow$ (1) $\delta(q_1, 2) = \{q_1, q_4\}$

↳ (2) $ECLOSE(q_1) \cup ECLOSE(q_4)$

$= \{q_1, q_4\}$

$\hat{\delta}(q_0 \underline{+2}\cdot) =$ (1) $\delta(q_1, \cdot) \cup \delta(q_4, \cdot) = \{q_2, q_3\}$

(2) $ECLOSE(q_2) \cup ECLOSE(q_3) = \boxed{\{q_2, q_3, q_5\}}$

Eliminating ε-transitions

Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be a ε-NFA.

Its equivalent DFA is $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$

① Q_D is set of subsets of Q_E,

Subset $S \subseteq Q_E$, $S = ECLOSE(S)$

② $q_D = ECLOSE(q_0)$

③ F_D is those sets of states that contains atleast one accepting states of E.

$F_D = \{ S \mid S \text{ is in } Q_D \text{ and } S \cap F_E \neq \phi \}$

④ $\delta_D(S, a)$, for all $a \in \Sigma$

Suppose $S = \{p_1, p_2, \ldots, p_K\}$

ⓐ Compute $\bigcup_{i=1}^{K} \delta(p_i, a)$, let this set is $\{r_1, \ldots, r_m\}$

ⓑ Then $\delta_D(S, a) = \bigcup_{j=1}^{m} ECLOSE(r_j)$

Ex:

ε-NFA

δ_E	0	1	2	ε
→ q_0	$\{q_0\}$	ϕ	ϕ	$\{q_1\}$
q_1	ϕ	$\{q_1\}$	ϕ	$\{q_2\}$
* q_2	ϕ	ϕ	$\{q_2\}$	ϕ

DFA

$q_0 = ECLOSE(q_0)$

$= \{q_0, q_1, q_2\}$

DFA

	0	1	2
→* $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
* $\{q_2\}$	ϕ	ϕ	$\{q_2\}$
* $\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
ϕ	ϕ	ϕ	ϕ

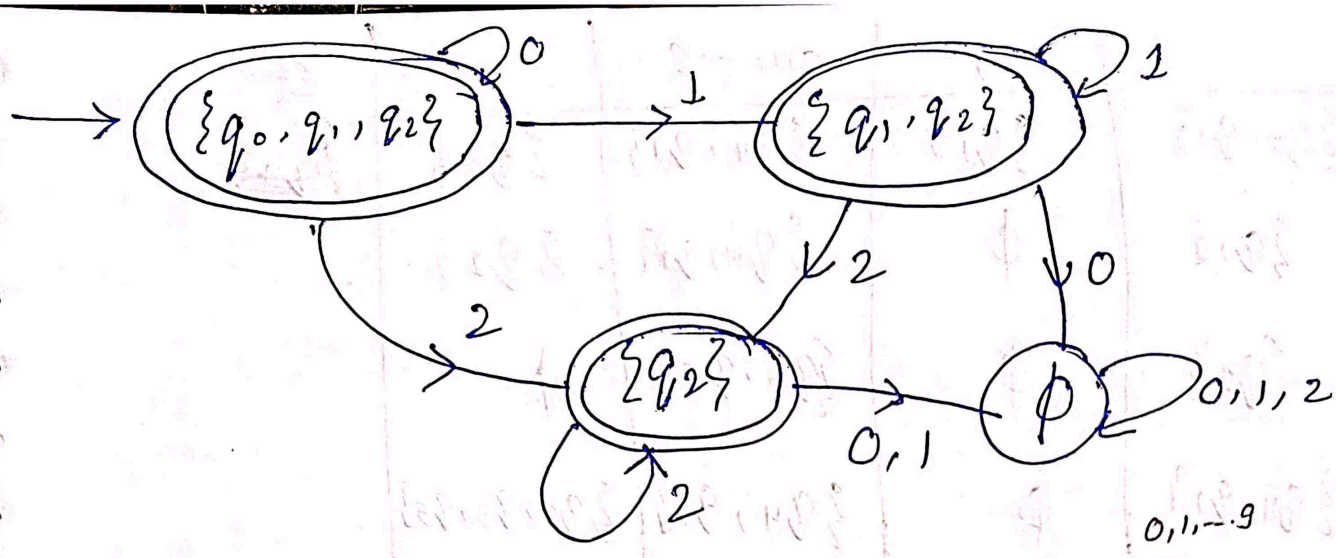
$S = \{q_0, q_1, q_2\} = \delta_D(S, 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$

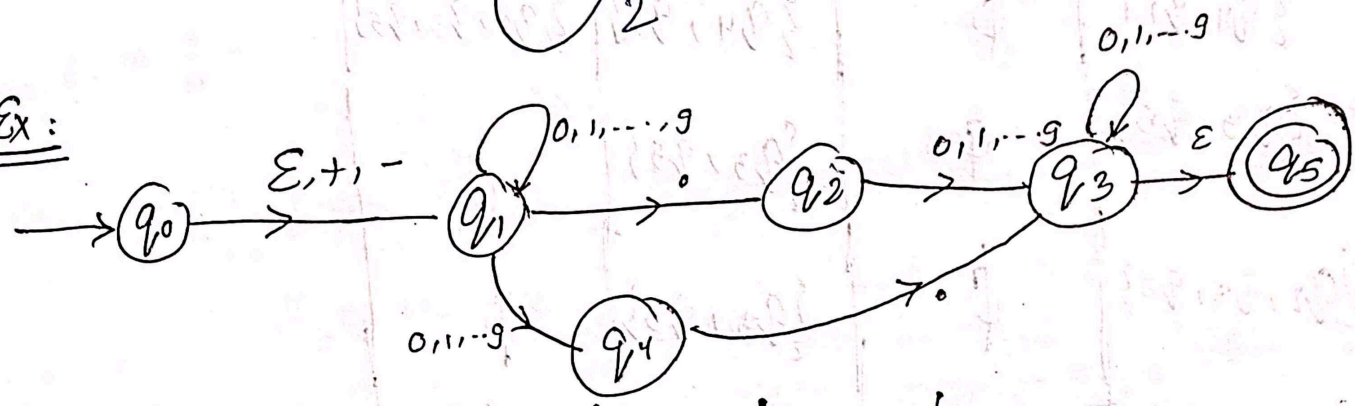
$= \{q_0\}$

$\delta_D(S, a) = ECLOSE(q_0)$

$= \{q_0, q_1, q_2\}$



Ex:

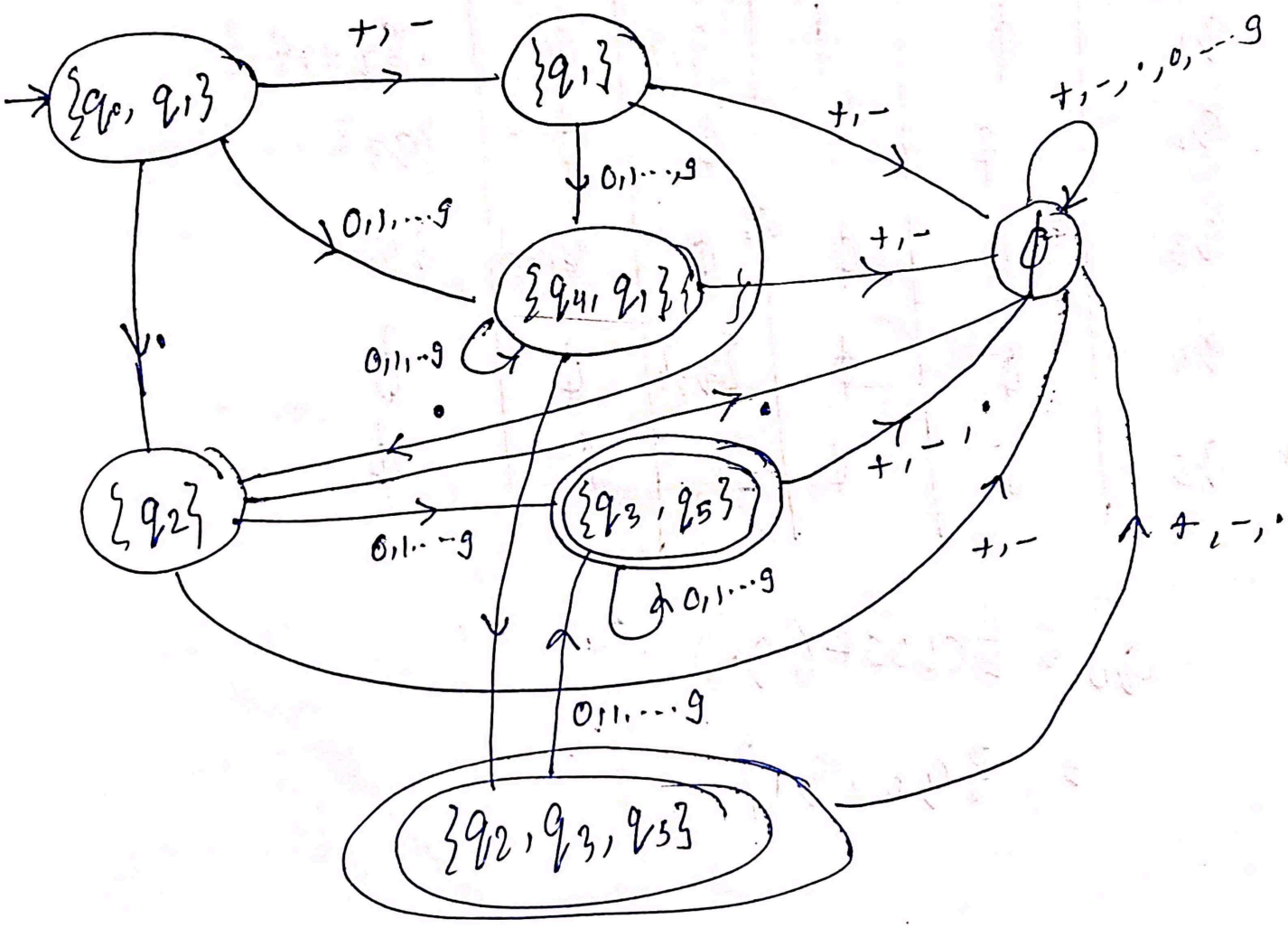


δ_E	+	−	.	ε	0, 1, ... 9
→ q_0	$\{q_1\}$	$\{q_1\}$	ϕ	$\{q_1\}$	ϕ
q_1	ϕ	ϕ	$\{q_2\}$	ϕ	$\{q_4, q_1\}$
q_2	ϕ	ϕ	ϕ	ϕ	$\{q_3\}$
q_3	ϕ	ϕ	ϕ	$\{q_5\}$	$\{q_3\}$
q_4	ϕ	ϕ	$\{q_3\}$	ϕ	ϕ
* q_5	ϕ	ϕ	ϕ	ϕ	ϕ

$q_0 = ECLOSE(q_0)$

$= \{q_0, q_1\}$

	$+,-$	$0,1,\cdots 9$	$\cdot$	ϵ
$\rightarrow \{q_0, q_1\}$	$\{q_1\}$	$\{q_4, q_1\}$	$\{q_2\}$	
$\{q_1\}$	ϕ	$\{q_4, q_1\}$	$\{q_2\}$	
$\{q_2\}$	ϕ	$\{q_3, q_5\}$	ϕ	
$\{q_4, q_1\}$	ϕ	$\{q_4, q_1\}$	$\{q_2, q_3, q_5\}$	
$*\ \{q_3, q_5\}$	ϕ	$\{q_3, q_5\}$	ϕ	
$*\ \{q_2, q_3, q_5\}$	ϕ	$\{q_3, q_5\}$	ϕ	
ϕ	ϕ	ϕ	ϕ	



Regular Expression & Regular Language.

L is regular if there exists a FA (NFA/DFA) M s.t.

$$L(M) = L$$

Language $L \subseteq \Sigma^*$

Suppose,

$L = \{0, 01, 011, 0111, \ldots, 1, 10, 100, \ldots\}$

Regular Expression : $\{01^* + 10^*\}$

↳ Algebraic description of Regular Language.

Properties of Regular Expression

(1) ϕ is a regular expression

$$L(\phi) = \{\}$$

(2) ε is a regular expression

$$L(\varepsilon) = \{\varepsilon\}$$

(3) If $a \in \Sigma$ is a regular exp^n

$$L(a) = \{a\}$$

(4) If $\{a+b\}$ is a regular exp^n

$$L(a+b) = \{a, b\}$$

(5) If 'r' & 's' are regular expr^n, then

$r+s$, $r.s$, $(r)^*$, $(s)^*$ are also regular exp^n.

$r+s \rightarrow L(r) \cup L(s)$

$r.s \rightarrow L(r) ; L(s)$ (concatenate)

$(r)^* \rightarrow (L(r))^*$

$(s)^* \rightarrow (L(s))^*$

Ques.

Lang: Set of all strings 'a' & 'b' containing 'aab' as substring.

$L = \{ \underline{aab}, a\underline{aab}, a\underline{aab}b, ab\underline{aab}ab, \ldots$

Corresponding regular expn

$$\underbrace{\Sigma^*}\ \underline{aab}\ \underbrace{\Sigma^*}$$

$\Sigma = \{a, b\} \Rightarrow (a+b)^*\ \underline{aab}\ (a+b)^*$

Ques.

L = Set of all strings ~~not having~~ starting with aa and ending with bb

$\rightarrow aa(a+b)^* bb$

Ques.

L = Set of all strings starting with aa or ending with bb

$\rightarrow aa(a+b)^* + (a+b)^* bb$

Ques.

L = Set of all strings over $\Sigma = \{0, 1\}$ not having two consecutive 0's.

r is regular expr

$r + r = r$

$(r^*)^* = r^*$

$\varepsilon^* = \varepsilon$

$\phi^* = \varepsilon$

$(r+s)^* = (r^* s^*)^*$

$r^* = r^+ + \varepsilon$

$r^+ = r \cdot r^*$

$\phi + r = r + \phi = r$, ϕ is Union Identity.

$\phi \cdot r = r \cdot \phi = \phi$, ϕ is left & right annihilator for concatenation.

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

$L = \{ w \in \Sigma^* \mid$ number of a's in $w <$ number of b's in $w\}$

is not regular. Prove it.

→ ~~Let no. of a's~~

Suppose "n" satisfying the condⁿ of pumping lemma.

~~Let $w = a^p b^q$, $|w| = p+q \geq n$~~

~~$p < q$~~

~~$w = xyz$, $y \neq \varepsilon$, $|xy| \leq n$~~

Let $w = a^n b^{n+1}$, $|w| = 2n+1 \geq n$

$w = xyz$, $y \neq \varepsilon$, $|xy| \leq n$

→ $\{x, y$ are a's$\}$

$\Rightarrow \{ x = a^i, y = a^j, i+j = n \}$

$\{ j \neq 0, y \neq \varepsilon \}$

→ $xy^k z$, $k \geq 0$

$\underline{k=0}$ $\quad xy^0 z = xz = a^i b^{n+1} a^{2j}$

$= a^{n+j} b^{n+1} \quad 1 \leq j \leq n$

$= \boxed{a^{n+1} b^{n+1}}$

Both a & b are equal

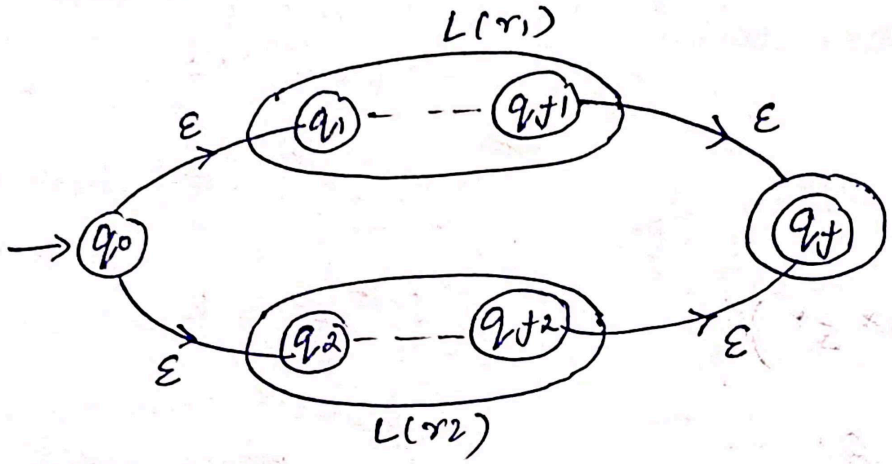
$xy^k z \notin L$

$\therefore L$ is not regular.

(4) $r = r_1 + r_2$

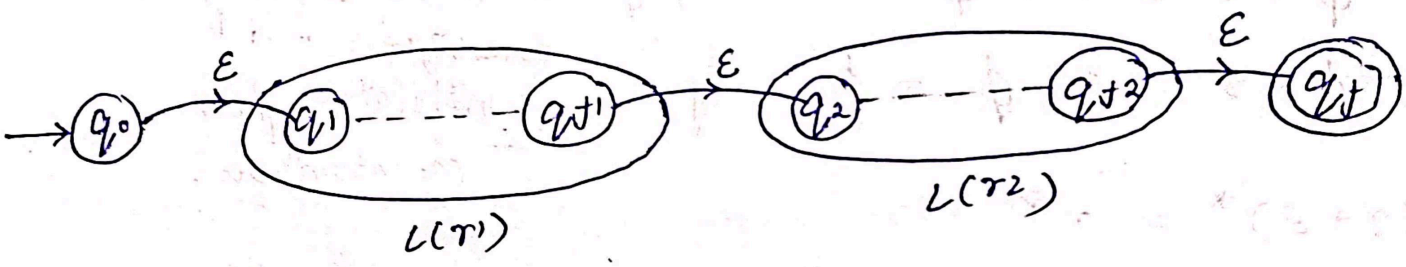
$\downarrow$

$M \rightarrow L(M) = L(r) = L(r_1) \cup L(r_2)$.

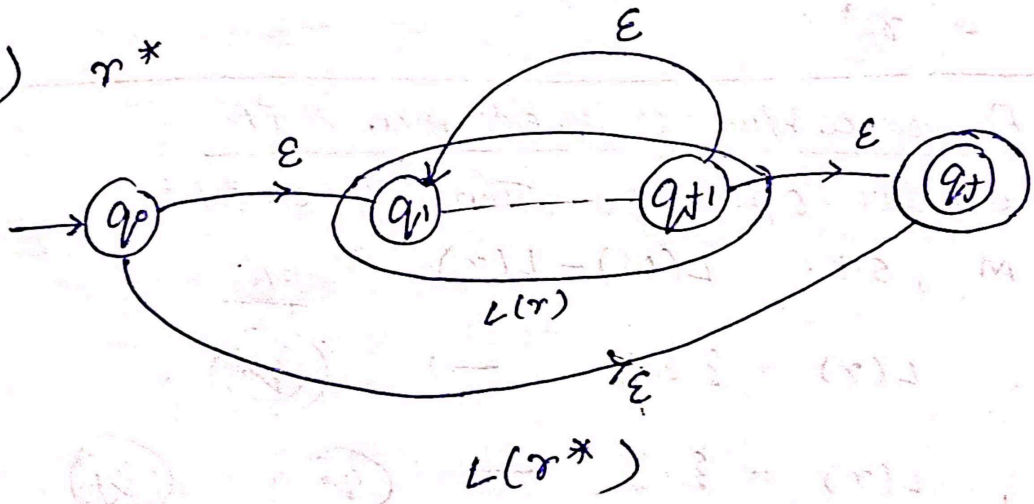


(5) $r = r_1 \cdot r_2$

$L(r) = L(r_1) \cdot L(r_2)$



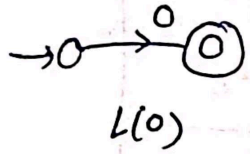
(6) r^*



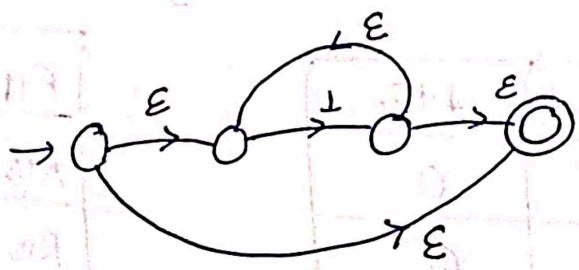
Let us construct an NFA, for all the regular expression

$$r = \underbrace{01^*}_{r1} + \underbrace{1}_{r2}$$

$$r1 = \underset{r3}{0} \cdot \underset{r4}{1^*}$$

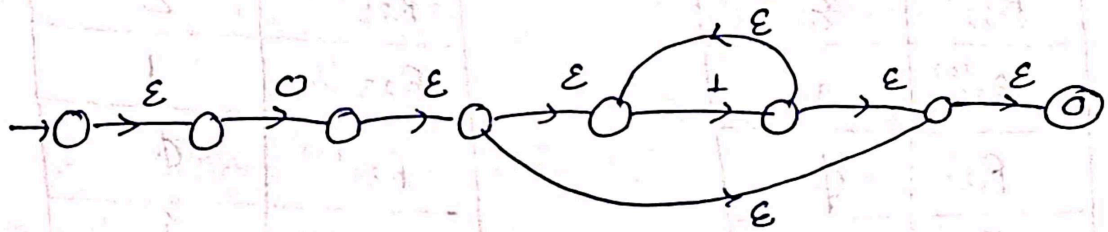


L(0)

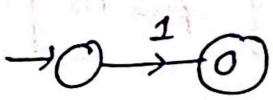


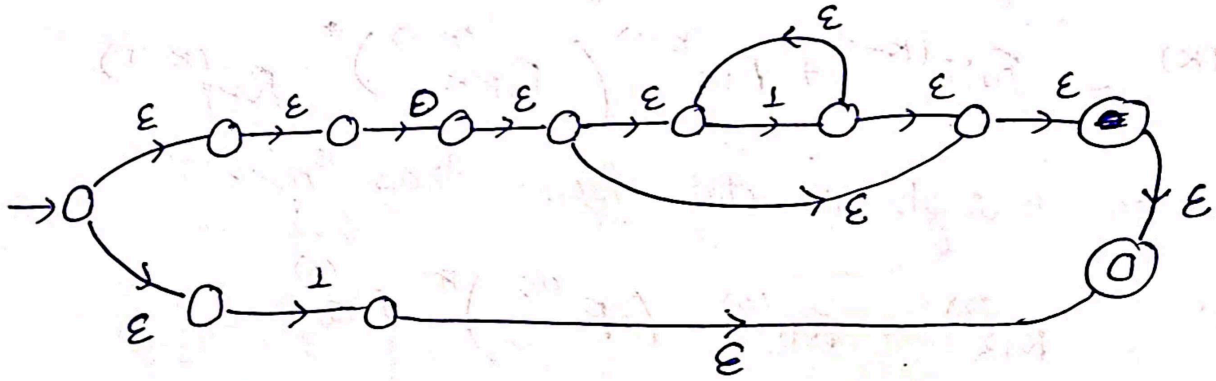
L(1)*

L(01*)



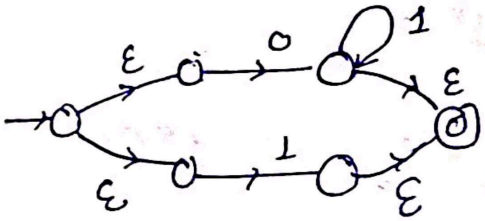
L(1)





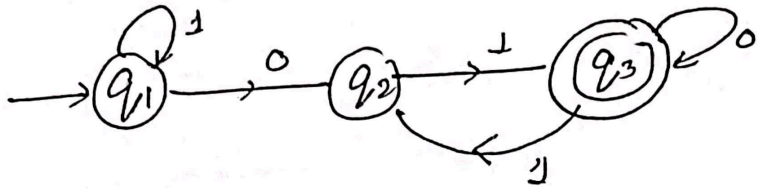
L(01* + 1)

Short-Cut



Convert FA to regular expression

Ex:



$R_{11}^{(0)}$	$1+\varepsilon$
$R_{12}^{(0)}$	0
$R_{13}^{(0)}$	ϕ
$R_{21}^{(0)}$	ϕ
$R_{22}^{(0)}$	ε
$R_{23}^{(0)}$	1
$R_{31}^{(0)}$	ϕ
$R_{32}^{(0)}$	1
$R_{33}^{(0)}$	$0+\varepsilon$

$R_{11}^{(1)}$	1^*
$R_{12}^{(1)}$	1^*0
$R_{13}^{(1)}$	ϕ
$R_{21}^{(1)}$	ϕ
$R_{22}^{(1)}$	ε
$R_{23}^{(1)}$	1
$R_{31}^{(1)}$	ϕ
$R_{32}^{(1)}$	1
$R_{33}^{(1)}$	$0+\varepsilon$

Regular exprn from state i to j:

$$R_{ij}^{(K)} = R_{ij}^{(K-1)} + R_{iK}^{(K-1)} \left(R_{KK}^{(K-1)} \right)^* R_{Kj}^{(K-1)}$$

go through no state higher than "K".

$$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} \left(R_{11}^{(0)} \right)^* R_{12}^{(0)}$$

$$= 0 + (1+\varepsilon)(1+\varepsilon)^* \cdot 0$$

$$= 0 + 1^* 0$$

$$= 0(\varepsilon + 1^*) = 1^* 0$$

$R_{11}^{(2)}$	1^*
$R_{12}^{(2)}$	1^*0
$R_{13}^{(2)}$	1^*01
$R_{21}^{(2)}$	ϕ
$R_{22}^{(2)}$	ε
$R_{23}^{(2)}$	1
$R_{31}^{(2)}$	ϕ
$R_{32}^{(2)}$	1
$R_{33}^{(2)}$	$0+\varepsilon+11$

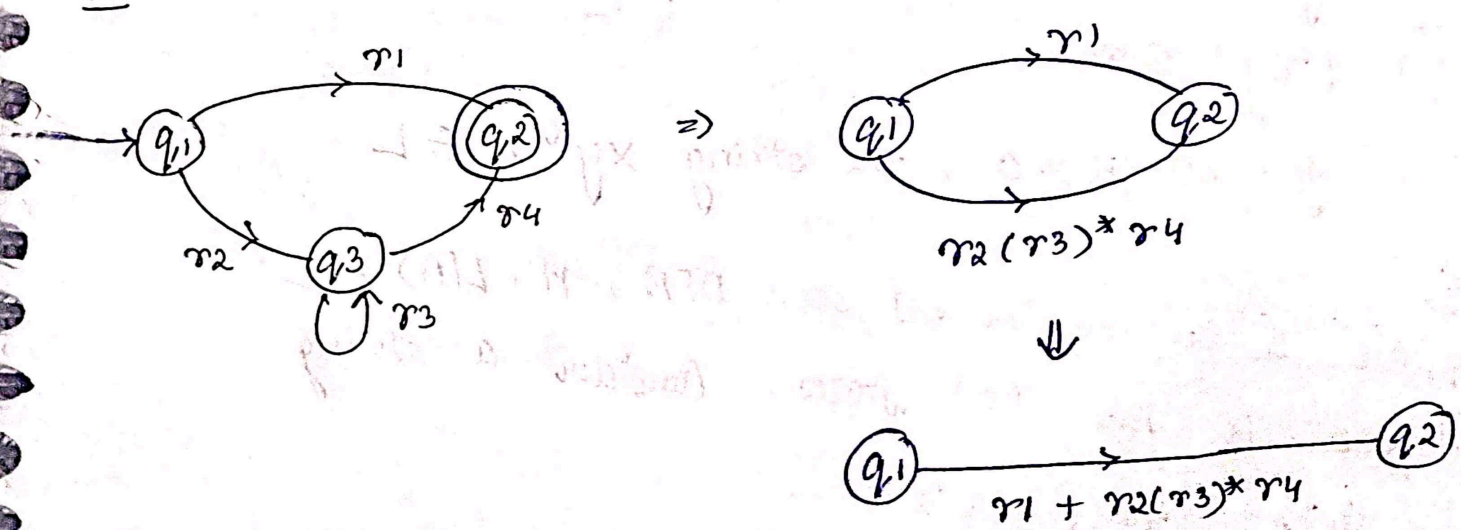
$$R_{13}^{(3)} = R_{13}^{(2)} + R_{13}^{(2)} \left(R_{33}^{(2)}\right)^* \cdot R_{33}^{(2)}$$

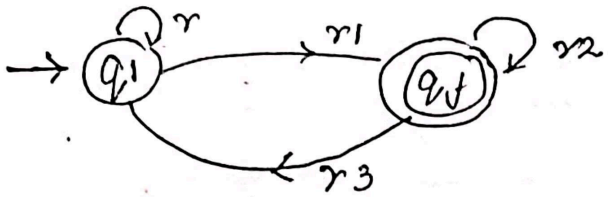
$$= 1^*01 + 1^*01\,(0+\varepsilon+11)^*(0+\varepsilon+11)$$

$$= 1^*01\,(0+\varepsilon+11)^*$$

$$= \boxed{1^*01\,(0+11)^*}$$ Ans.

Ex: Shorten path





$\Rightarrow (r + r1 \cdot r2^* r3)^* \, r1 \, r2^*$

Pigeon Hole Principle

m - object , n - boxes , $m > n$

at least one box must have more than one objects.

Pumping Lemma

Can be used to prove a lang. is not regular but vice-versa not true.

Theorem: Let L be a regular language. Then there exists a constant "n" (depends on L), s.t., for every string $w \in L$, s.t., $|w| \geq n$

We can break w into three substrings.

$w = xyz$, s.t.,

(i) $y \neq \varepsilon$ (i.e., not empty)

(ii) $|xy| \leq n$

(iii) for all $K \geq 0$, the string $xy^K z \in L$

Proof:

Suppose L is regular. $\exists$ some DFA, A. $L(A) = L$

Suppos A has "n" states. Consider a string

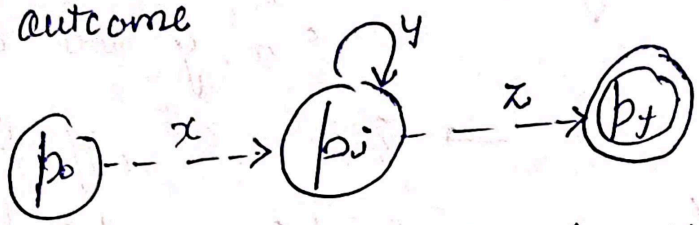
$w = a_1 a_2 \cdots a_m$, $m \geq n$.

$\hat{\delta}(q_0, a_1 a_2 \cdots a_i) = p_i$ for $i = 0, 1, 2 \cdots n$

q_0 = starting state

p_i = state after reading first ith symbols of w ($p_0 = q_0$).

$(n+1)$ states (n-states)
DFA
outcome



We find diff. integers i, j with

$0 \leq i < j \leq m$ then the substring

are :

$x = a_1 \text{---} a_i$

$y = a_{i+1} \text{---} a_j$

$z = a_{j+1} \text{---} a_m$

<u>Check</u> $xy^k z$, $k \geq 0$

<u>$k=0$</u> $p_0 \longrightarrow p_i \longrightarrow$ final state

<u>$k \geq 1$</u> $p_0 \longrightarrow p_i \longrightarrow p_f$

<u>Examples</u>

① L = consists of all palindromes over Σ^* , $\Sigma = \{0, 1\}$,

prove that L is not a regular language.

Suppose "n" satisfying the condⁿ of pumping lemma.

Let $w = \underline{0^n 1 0^n}$, $|w| \geq n$

$w = xyz$, $y \neq \varepsilon$, $|xy| \leq n$

$\{x, y$ are 0's$\}$

$\rightarrow \{x = 0^i, y = 0^j, i+j = n\}$

$\{j \neq 0, y \neq \varepsilon\}$

$\rightarrow xy^k z$

$\rightarrow k \geq 0 \Rightarrow \underline{k = 0}$ $\quad xy^0 z = 0^i 1 0^n$

$= 0^{n-j} 1 0^n \notin L$

$\therefore$ <u>L is not regular.</u>

<u>Q.</u> Prove that the language $L = \{0^n 1^n \mid n \geq 1\}$ is not regular.

=) Let $w = 0^n 1^n$, $|w| \geq n$

$w = xyz$, $y \neq \varepsilon$, $|xy| \leq n$

$\{x, y$ are 0's$\}$

$\rightarrow \{x = 0^i, y = 0^j, i+j = n\}$

$\rightarrow \{j \neq 0, y \neq \varepsilon\}$

$\rightarrow xy^k z$

$\rightarrow k \geq 0 \rightarrow \underline{k = 2}$ $\quad xy^0 z = 0^i 0^{2j} 1^n$

$= 0^{i+j} 0^j 1^n$

$= 0^n 0^j 1^n \notin L$

$\because j \neq 0$

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

$L = \{ w \in \Sigma^* \mid$ number of a's in $w <$ number of b's in $w\}$

is not regular. Prove it.

→ ~~Let no. of a's~~

Suppose "n" satisfying the condⁿ of pumping lemma.

~~Let $w = a^p b^q$, $|w| = p+q \geq n$~~

~~$p < q$~~

~~$w = xyz$, $y \neq \varepsilon$, $|xy| \leq n$~~

Let $w = a^n b^{n+1}$, $|w| = 2n+1 \geq n$

$w = xyz$, $y \neq \varepsilon$, $|xy| \leq n$

→ $\{x, y$ are a's$\}$

$\Rightarrow \{ x = a^i, y = a^j, i+j = n \}$

$\{ j \neq 0, y \neq \varepsilon \}$

→ $xy^k z$, $k \geq 0$

$\underline{k=2}$ $xy^0 z = xz = a^i b^{n+1} a^{2j}$

$= a^{n+j} b^{n+1}$ $1 \leq j \leq n$

$= \boxed{a^{n+1} b^{n+1}}$

Both a & b are equal

$xy^k z \notin L$

$\therefore L$ is not regular.

Closure Properties of Regular Language

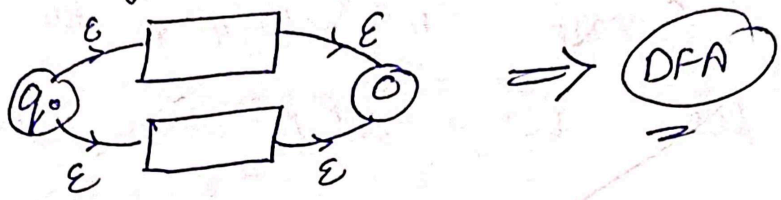
1.) Union

Union of two regular languages is regular.

If L & M are regular langs. then $L \cup M$ is also regular.

$L \rightarrow DFA(A)$

$M \rightarrow DFA(B)$



ε-NFA

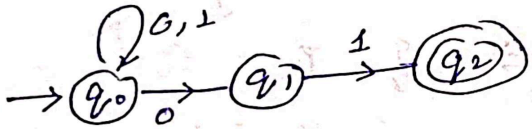
2.) Complement

Complement of regular language is also a regular lang.

$L \rightarrow$ is a lang.

$\overline{L} \rightarrow \Sigma^* - L$ $\Rightarrow$ L is regular $\rightarrow$ $\overline{L}$ is regular.

Ex: $L = (0+1)^* 01$ (r.l)

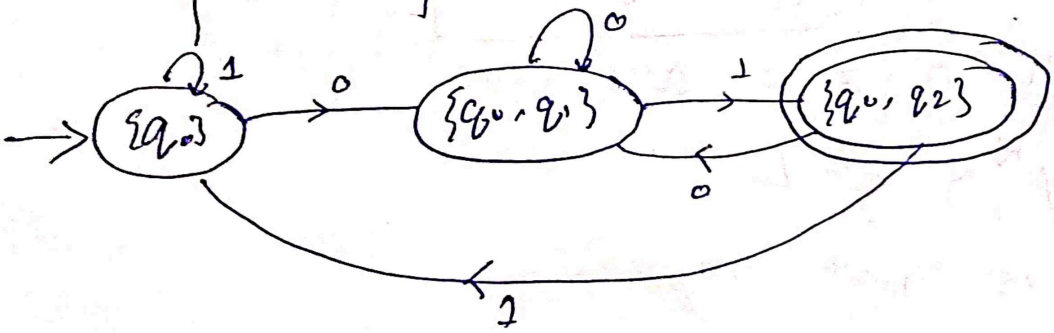


NFA

$\overline{L} = \Sigma^* - (0+1)^* 01$

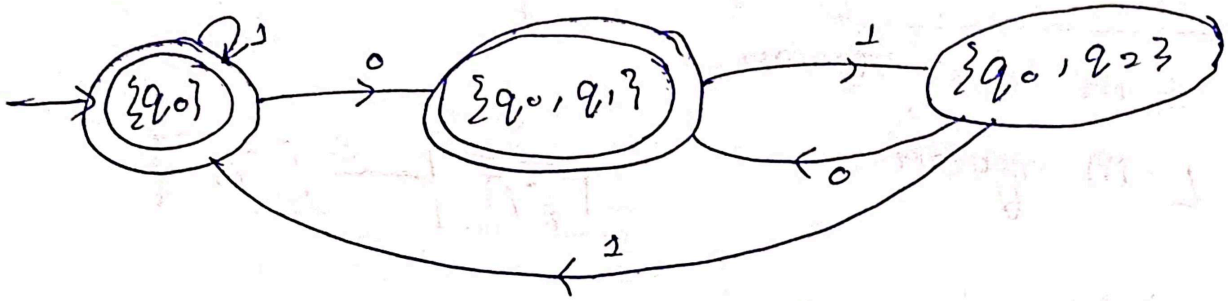
Convert NFA to DFA

	0	1
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$



Let $L(A)$ for some DFA $A(Q, \Sigma, \delta, q_0, F)$.
The $\overline{L} = L(B)$ where B is the DFA $(Q, \Sigma, \delta, q_0, Q-F)$
accepting state of A have become non-accepting state of B & vice-versa.

i.e. DFA (B)



$$\overline{L} = \Sigma^* - (0+1)^* \underline{01}.$$

3) Intersection

Intersection of two regular lang. is also regular.
L & M are regular $\Rightarrow L \cap M$ is regular.

$$L \cap M = \overline{\overline{L} \cup \overline{M}}$$

4) Difference

Difference of two regular lang. is also regular lang.
L & M are regular lang.

$$\Rightarrow L - M = L \cap \overline{M}.$$

5) Reverse

The reverse of a regular lang is also a regular lang.

$L \rightarrow$ regular lang.

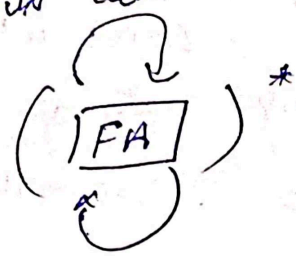
$$L = \{011, 01101, 00001\}$$

$$L^R = \{110, 10110, 10000\}$$

6.) Star Closure (*)

The star closure of a regular lang. is also a regular lang.

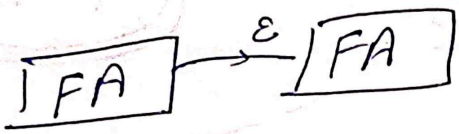
$L = \{0\} \longrightarrow L^* = \{0^*\}$



(7) The Concatenation of regular lang. is also regular.

L & M are regular.

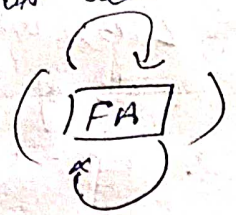
L · M regular.



6.) Star Closure (*)

The star closure of a regular lang. is also a regular lang.

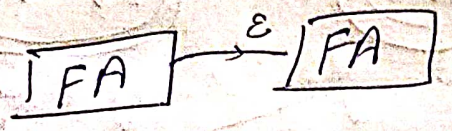
$L = \{0\} \rightarrow L^* = \{0^*\}$



(7) The Concatenation of regular lang. is also regular.

L & M are regular.

L·M regular.



Finite Automata with output.

2-types of FA with o/p

(1) Moore Machine (o/p is associated with the state)

(2) Mealy Machine (" " " transition (state, i/p))

Moore Machine

A moore machine is six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Same as DFA:

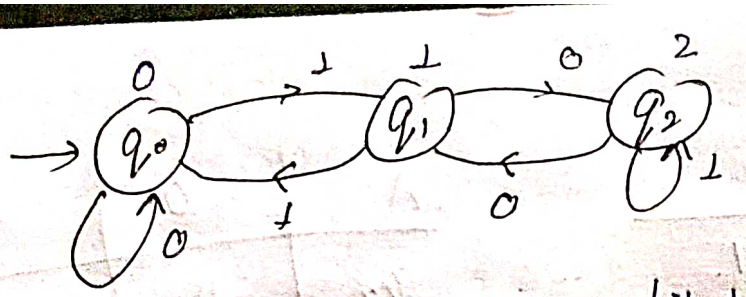
- Q = finite set of states
- Σ = Input alphabet
- δ = Transition function $(\delta : Q \times \Sigma \rightarrow Q)$
- q_0 = Starting / Initial state

Δ = output alphabet

λ = output function

$\lambda : Q \rightarrow \Delta$

Ex:



i/p string : $w: 1010$ $\quad |i/p| = 4$

$$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_1$$

o/p string: 0 1 2 2 1

$$|o/p| = 5 = \boxed{|i/p| + 1}$$

Ans: $\boxed{\text{Residue mod } 3}$

$\rightarrow 10\%3 = 1$ (10) $\quad 1010 \rightarrow 1$

$\rightarrow 5\%3 = 2$ (5) $\quad 101 \rightarrow 2$

$2\%3 = 2$ (2) $\quad 10 \rightarrow 2$

$1\%3 = 1$ (1) $\quad 1 \rightarrow 1$

$0\%3 = 0$ (0) $\quad 0 \rightarrow 0$

Q) Conversion from Moore to Mealy & vice-versa

Mealy Machine

A mealy machine consists of 6 tuples $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

⇒ All are same except for λ

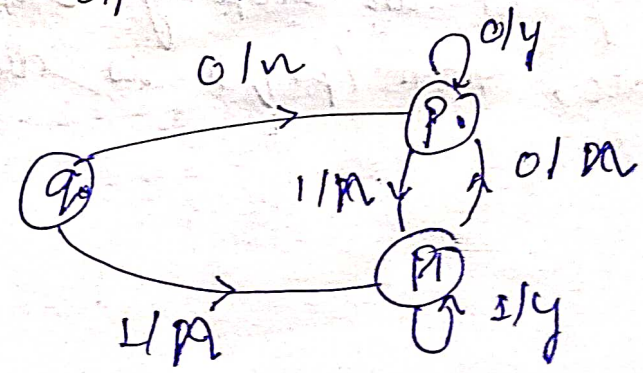
$$\lambda : Q \times \Sigma \rightarrow \Delta$$

Σ : i/p alphabet $\quad \Delta$: o/p alphabet.

$Q = \{q_0, p_0, p_1\}$

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

$\Delta = \{n, y\}$



Transition function

δ	0	1
q_0	p_0	p_1
p_0	p_0	p_1
p_1	p_0	p_1

λ	0	1
q_0	n	n
p_0	y	n
p_1	n	y

i/p : 0 1 1 0 0

o/p : ??

$$q_0 \xrightarrow{0} p_0 \xrightarrow{1} p_1 \xrightarrow{1} p_1 \xrightarrow{0} p_0 \xrightarrow{0} p_0$$

$p_0 \downarrow n$, $p_1 \downarrow n$, $p_1 \downarrow y$, $p_0 \downarrow n$, $p_0 \downarrow y$

$|i/p| = 5$ $\quad |o/p| = 5$

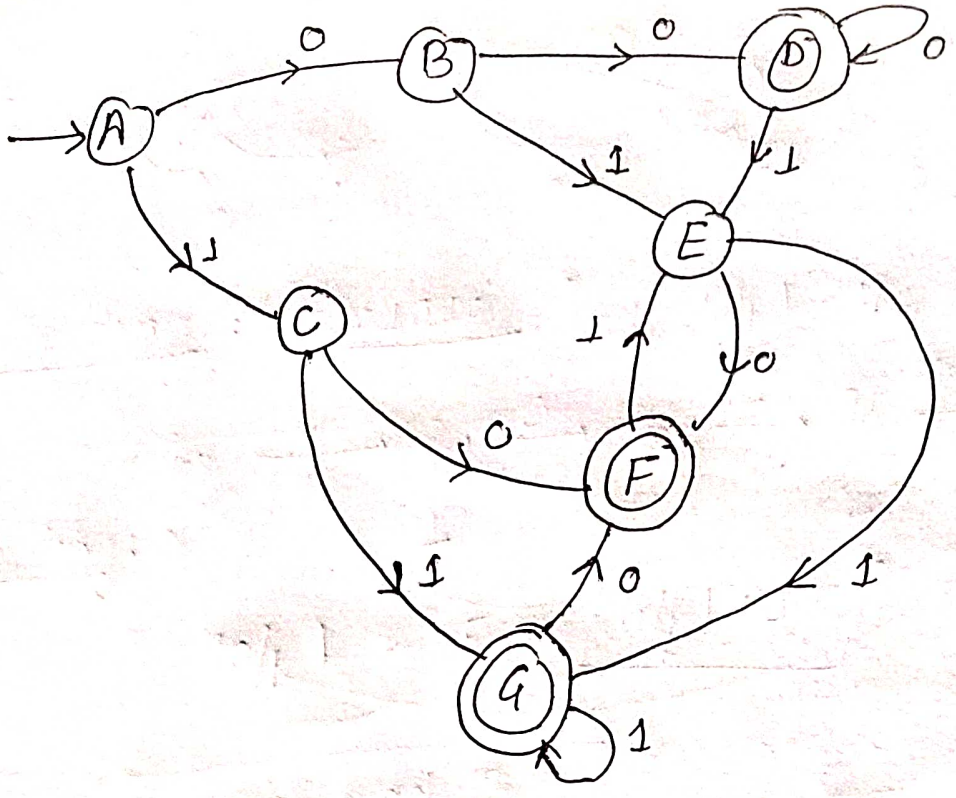
Minimization of DFA

Equivalence b/w 2 states (p, q)

for all i/p strings ω, $\hat{\delta}(p, \omega)$ is an accepting state iff $\hat{\delta}(q, \omega)$ is an accepting state.

If two states are not equivalent, then they are distinguishable states.

If p is an accepting state & q is a non-accepting state, then (p, q) is distinguishable.



	0	1
A	B	C
B	D	E
C	F	G
* D	D	E
E	F	G
* F	D	E
* G	F	G

Table filling Order

B	X					
C	X	X				
* D	X	X	X			
E	X	X	O	X		
* F	X	X	X	O	X	
* G	X	X	X	X	X	X
	A	B	C	D *	E	F *

X → distinguishable
O → equivalent.

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

(F, G) : i/p 1 E, G

(D, G) : i/p 1 E, G

(C, E) : i/p = 0 (F, F)
i/p = 1 (R, G)