Introduction

FLAT

Formal Languages and Automata Theory

- It is also known as Automata Theory and formal Languages (ATFL)
- Why this course "Formal language and automata theory"?
- Study of languages, Grammar and Automata.
- Design of translators.
- Compiler ~~des~~. Construction - Programming languages.

Automata Theory

Automata Theory is the study of abstract machines and automata, as well as the computational problems that can be solved using them. It is a theory in theoritical computer science.

- The word automata comes from the Greek word, it means "self-acting", "self-willed", "self-moving".
- In other words, we can say it recognize the grammar or it accepts the language.

Computational devices

computing devices are the electronic devices which are inputs, process the inputs and then calculate results from the inputs.

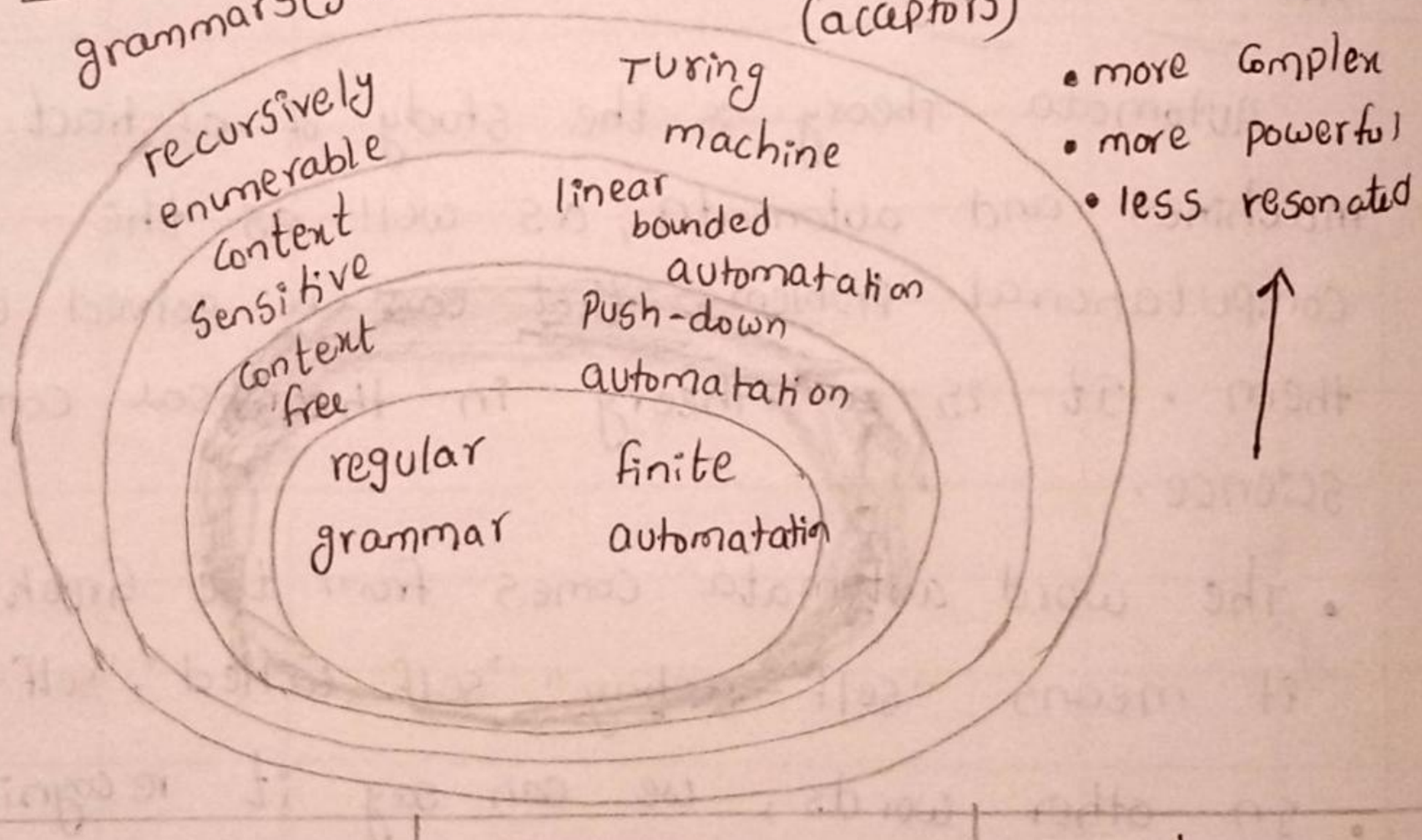
Automation :-

An automation is an abstract self - propelled computing device which follows a predetermined sequence of operations automatically.

Major themes of this course
1. Languages
2. Grammers
3. Automata.

- Types and hierarchy.
- Relation b/w Language, Grammers and Automata
- Design of Automata and conversions.

Heirarchy :-



Grammars	Language	Machine
unrestricted Grammar	Recursively enumerable (RE)	Turing machine (TM)
Context sensitive Grammar (CSG)	Context Sensitive Language (CSL)	Linear bounded automation (LBA)
Context free Grammar (CFG)	Context free Language (CFL)	push - down automation (PDA)
Regular Grammar (RG)	Regular language (RL)	Finite automation (FA)

Levels

0-Level ⇒ RE (Recursively enumerable)

1-Level ⇒ CS (Content sensitive)

2-Level ⇒ CF (Context Free)

3-Level ⇒ RG (Regular Grammer)

Grammar will ~~be~~ generate the language and language will be acapted by the automata.

- If something is regular it may be context free.
- If something is context free it may be a context sensitive.
- If something is context sensitive it may be a recursively enumerable.

Unit - I

Introduction to Finite Automata

overview

- Basics : Alphabets, Strings and Languages
- Finite state Machine
- Acceptance of strings and Machines
- DFA and NFA
- Transition diagrams
- Language recognizers

Basics

Alphabet:- A finite nonempty set of symbols denoted by Σ

→ Elements are some kind of characters.

→ Members/Elements of an alphabet are called

→ Symbols letters and digits

→ Ex: $\Sigma = \{a\}$, $\Sigma = \{a, b, x, y\}$, $\Sigma = \{0\}$, $\Sigma = \{0, 1\}$

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

String : Finite sequence of symbols from the ~~seq~~ alphabet.

→ written adjacent without commas.

→ Ex: over $\Sigma = \{a, b, c, d\}$: aa, aba, bac, abab, abbba

→ Ex: over $\Sigma = \{0, 1\}$: 000, 01011, 11001., etc.

→ Typically represent strings with letters like

u, v, w, x, y, z.

Ex: u = abc, x = 10110

operations on strings

* Concatenation :- Join two or more strings and/or symbols.

Ex: If u = abab, v = abbba then uv = ababbbba.

* Reverse of a string u is obtained by writing symbols in the reverse order. Denoted by u^R.

Ex: The reverse of u : u^R = baba

* Length of a string u is number of symbols in u and is denoted as |u|.

Ex: |u| = 4 ; |v| = 5 ; |uv| = 9

- Empty String : String with no symbols is denoted by $\wedge$ (or $\in$ or λ) and its length is zero.
- If u and v are two strings it is true that |uv| = |u| + |v|.
- Kleen absure and kleen plus : If Σ is an alphabet, Σ^* is the set of strings obtained by concatenating zero or more symbols from Σ

And $\Sigma^+ = \Sigma^* - \{\wedge\}$

Ex: If $\Sigma = \{a, b\}$ then $\Sigma^* = \{\wedge, a, b, aa, ab \ldots\}$

and $\Sigma^+ = \{a, b, aa, ab \ldots\}$.

Ex:

$\Sigma = \{a\}$

$\Sigma^* = \{\in, a, aa, aaa, \ldots\}$

$\Sigma^+ = \{a, aa, aaa, \ldots\}$

Language

- A set of strings (over an alphabet)
- Language is defined as a subset of Σ^*.

However we define in a more meaningful way in the comming sessions using some conditions.

- $L_1 \subset \Sigma^*$

  L_1 is subset of Σ^*

Ex:

→ over alphabet $\{a, b, c\}$:

- $L_1 = \{aaa, aba\}$; $L2 = \{a, aa, aaa\}$

→ over alphabet $\{0, 1\}$;

- $L_1 = \{0\}$; $L2 = \{0, 1, 010\}$.

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

$L = \{\omega \mid \omega \in \{ab\}^*\}$

$L = \{\epsilon, ab, abab, ababab..\}$

- set of all binary strings ending with 1.

  Ex: $L = \{1, 01, 001, 011, 101, \ldots\}$

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

- set of all strings over $\{a, b\}$ having number of a's equal to number of b's.

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

  Ex: $L = \{\epsilon, ab, ba, abba, aabb, baab, \ldots\}$

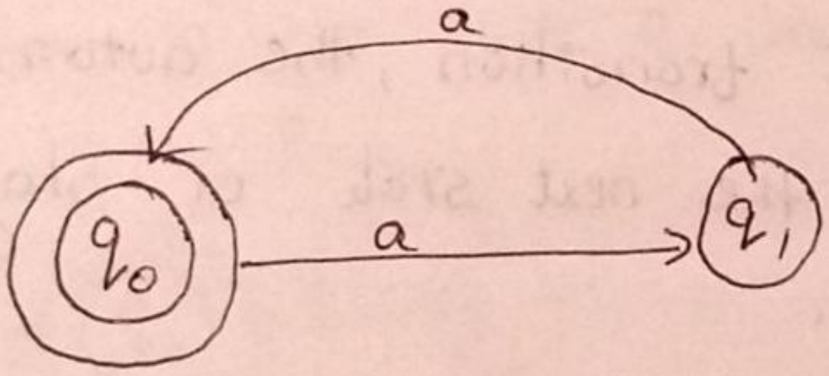
- $L : \{a^n / n \geq 1\}$ over $\{a\} \rightarrow \{a, aa, aaa, aaaa, \ldots\}$
- $L : \{a^n b^n / n \geq 1\}$ over $\{a, b\} \rightarrow \{ab, aabb, aaabbb, \ldots\}$
- $L : \{a^n b^n c^n / n \geq 1\}$ over $\{a, b, c\} \rightarrow \{abc, aabbcc, \ldots\}$

- set of valid variable names in Fortran.
- set of valid arithmetic expressions in C.

Automata

- An abstract machine - to solve computation problems.
- we use state diagrams to represent automata transition.
- Automaton consists of states and transitions. the state is represented by circles, and the transitions is represented by arrows.
- Take some string as input and this input goes through a finite number of states and may enter in the final state.

Ex:



Here q_0 is the final state.

So it is represented using double circle.

states: q_0 & q_1

transitions: $q_0 \rightarrow q_1$

$q_1 \rightarrow q_0$

IF $\omega = \{aaaa\}$,

the transition is

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_0$$

The transition is accepted.

The above state diagram can be represent as

$L = \{a^n \mid n \geq 0 \ \& \ n \text{ is even}\}$

It is accepted.

For $L = \{a^n \mid n \geq 0 \ \& \ n \text{ is odd}\}$

It is rejected.

Finite Automata

- Finite automata are used to recognize Strings.
- Take the string of symbol as input and changes its state accordingly. when the desired symbol is found, then the transition occurs.
- At the time of transition, the automata can either move to the next state or stay in the same state.
- Finite automata have two states, Accept state or Reject state. when the input string is processed successfully, and the automata reached its final state, then it will accept.

Finite Automation

- A finite automation is a 5-tuple

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

Q : a finite non empty set of states

Σ : input alphabet

δ : transition function, $Q \times \Sigma \Rightarrow Q$

q_0 : initial state

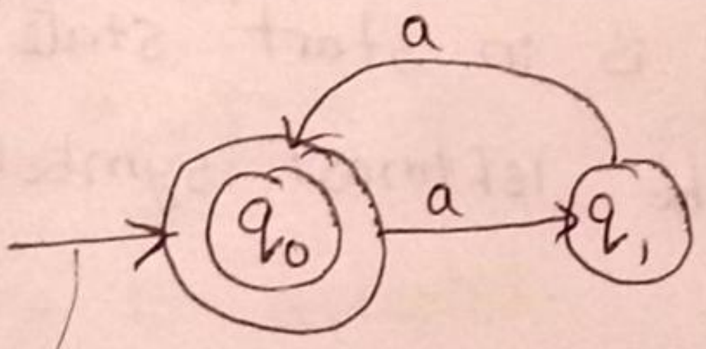
F : set of final states, $F \subseteq Q$.

- δ is defined for each $q \in Q$ and for each $a \in \Sigma$
- Thus with each input alphabet the automation moves from one state to another state as per δ.

~~$\delta(q, a) = q$~~

- $\delta(q_i, a) = q_j$ indicates, with input a the automation in the state q_i moves to state q_j.

Ex



$Q = \{q_0, q_1\}$

$\Sigma = \{a\}$

$q_0 = q_0$

$F = \{q_0\}$

δ	a
q_0	q_1
q_1	q_0

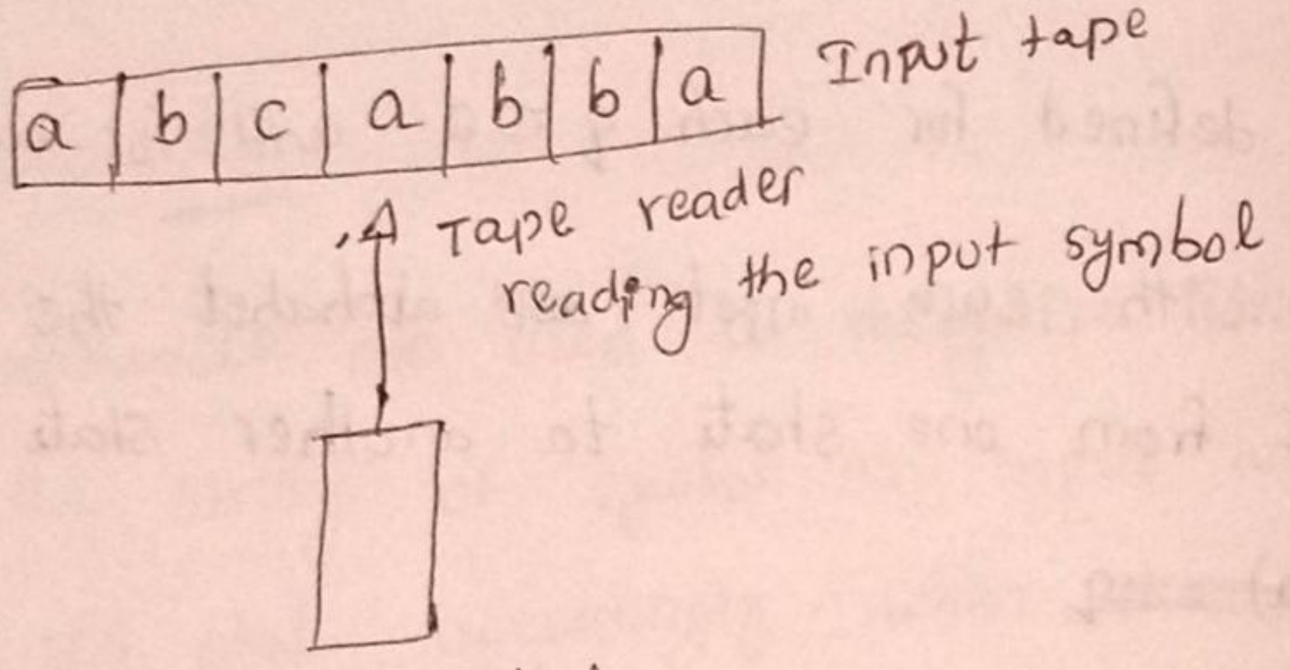
Finite Automata Model

- Input tape, finite control and tape reader.

Input tape : It is a linear tape having some number of cells. Each input symbol is placed in each cell.

Finite Control :- The finite control decides the next state on receiving particular input from input tape.

Tape reader :- Tape reader reads the cells one by one from left to right, and at a time only one input symbol is read.

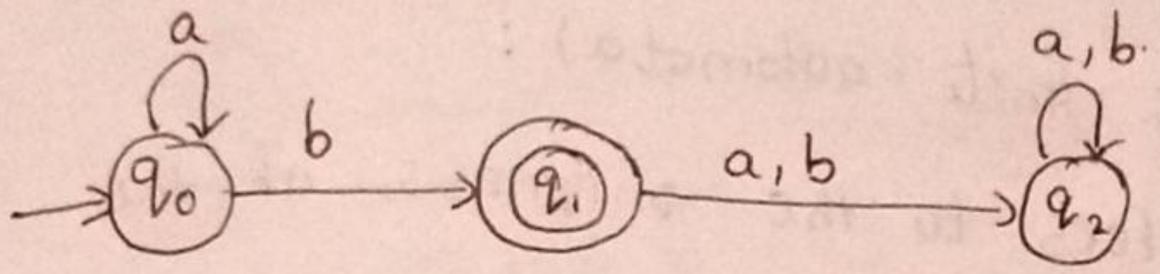


Working of Finite Automata

* Initially Finite Automata is in start (initial) state and the input head is on the leftmost symbol of input.
* The input symbol is read (input head moves to the next character) and FA assumes a new state, depending on the present state and the read input symbol (transition function).
* Repeat the above step until the input is consumed.
* If finite automata is in a final state, announce accept, otherwise announce reject.

Ex:

a a b



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

where

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

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

δ	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2

$q_0 = q_0$

$F = \{q_1\}$.

$\omega = aab$,

$q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1$ final state

∴ String is accepted.

Examples for the above finite automata :-

b, ab, aab, aaab, aaaab → accepted strings.

∴ $L = \{a^n b / n \geq 0\}$.

Not Rejected string :- i) abab

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_2 \xrightarrow{b} q_2$ → non-finate state.

ii) bab.

$q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_2$

rejected.

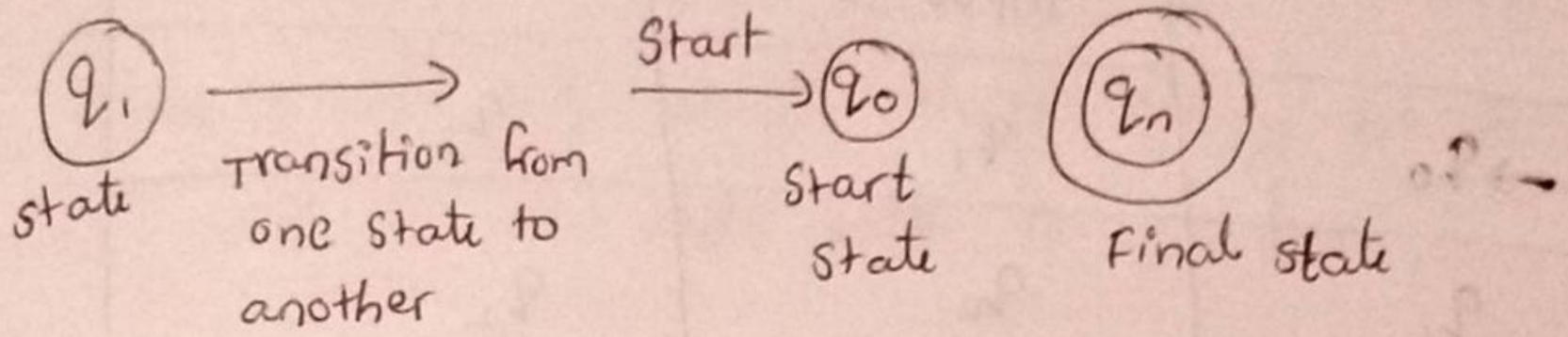
Types of Automata

- There are two types of finite automata:
  - DFA (deterministic finite automata):
    - Deterministic refers to the uniqness of the computation.
    - machine goes to one state only for a particular input character.
    - DFA does not accept the null move.
  - NFA (non-deterministic finite automata):
    - transmit any number of state for a particular input.
    - It can accept the null move.
- Every DFA is NFA, but NFA is not DFA
- Multiple final states in both NFA and DFA.

Transition Diagram

- A transition diagram or state transition diagram is a directed graph which can be constructed as follows
  - There is a node for each state in Q, which is represented by the circle.
  - There is a directed edge from node q to node p labded a if $\delta(q,a) = p$.
  - In the start state, there is an arrow with no sourle.

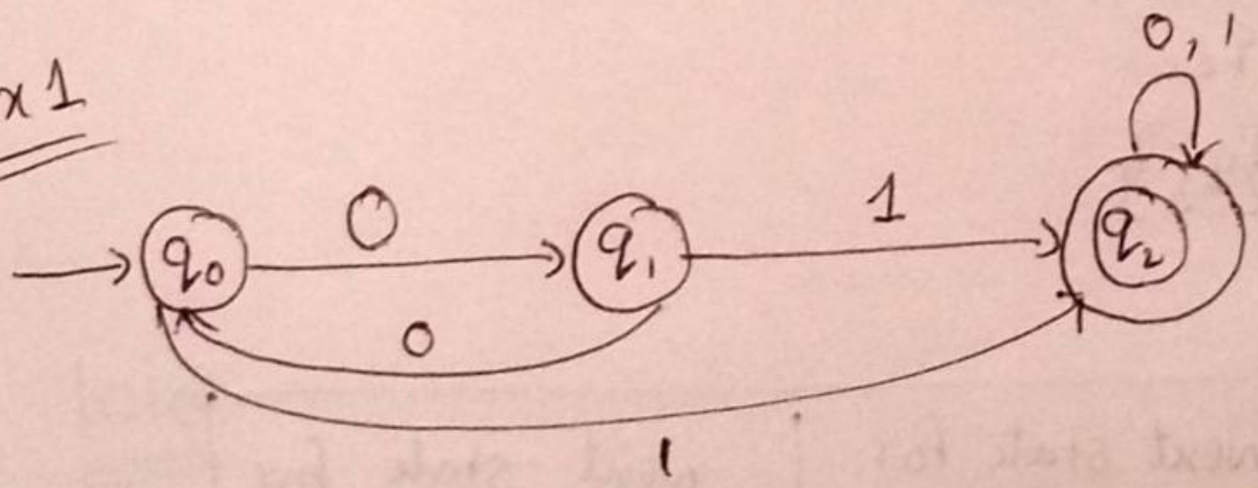
- Accepting states or final states are indicating by a double circle.



Transition table

- tabular representation of the transition function.
- It takes two arguments (a state and a symbol) and returns a state (the next state)
- A transition table is represented by the following
  - → columns correspond to input symbols
  - → Rows correspond to states
  - → Entries correspond to the next state
  - ⇒ start state is denoted by an arrow with no source.
  - ⇒ The accept state is denoted by a star.

Ex 1



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

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

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

$\delta = (q_0, 0) = q_1$ $\quad (q_1, 1) = q_2$

$(q_0, 1) = q_2$ $\quad (q_2, 0) = q_2$

$(q_1, 0) = q_0$ $\quad (q_2, 1) = q_2$

$q_0 = q_0$

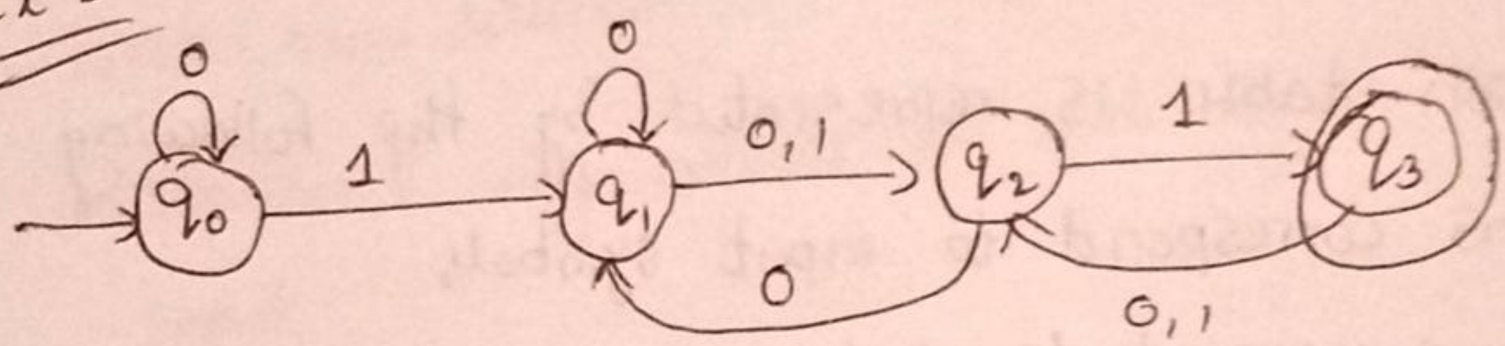
$F = \{q_2\}$

δ :

Present state	Next state for Input 0	Next state for Input 1
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_2
$*q_2$, (q_2)	q_2	q_2

It is an example of DFA.

Ex 2:



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

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

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

$q_0 = q_0$

$F = \{q_3\}$

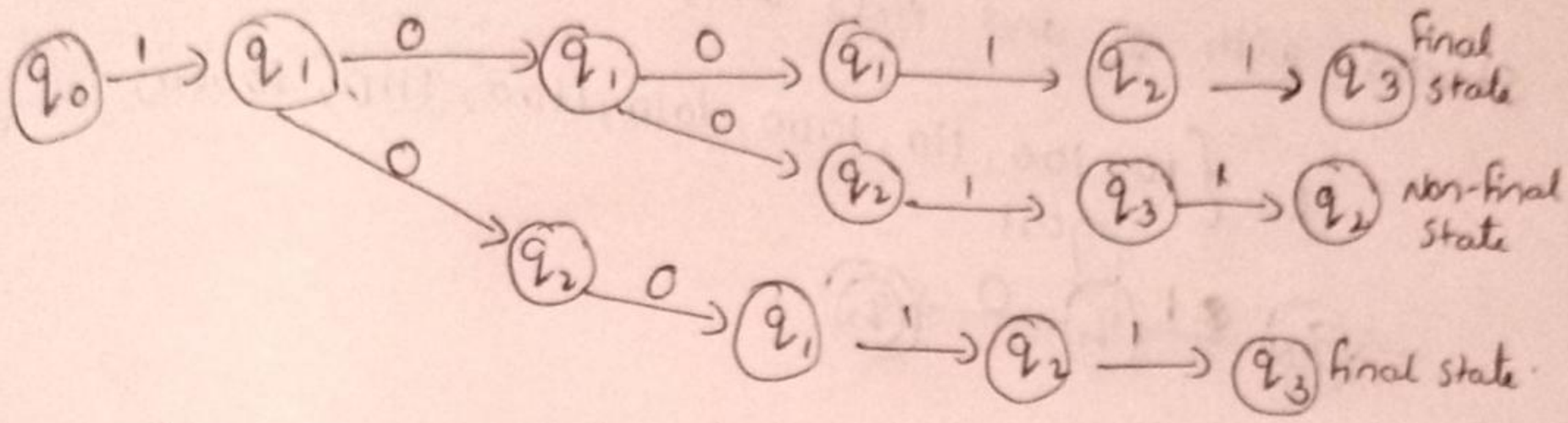
δ :

present state	Next state for Input 0	Next state for Input 1
$\rightarrow q_0$	q_0	q_1
q_1	q_1, q_2	q_2
q_2	q_1	q_3
$*q_3$, (q_3)	q_2	q_2

* It is an example of NFA.

- FA with $\Sigma = \{0, 1\}$ accepts the only input 101.
- FA with $\Sigma = \{0, 1\}$ accepts those string which starts with 1 and ends with 0.

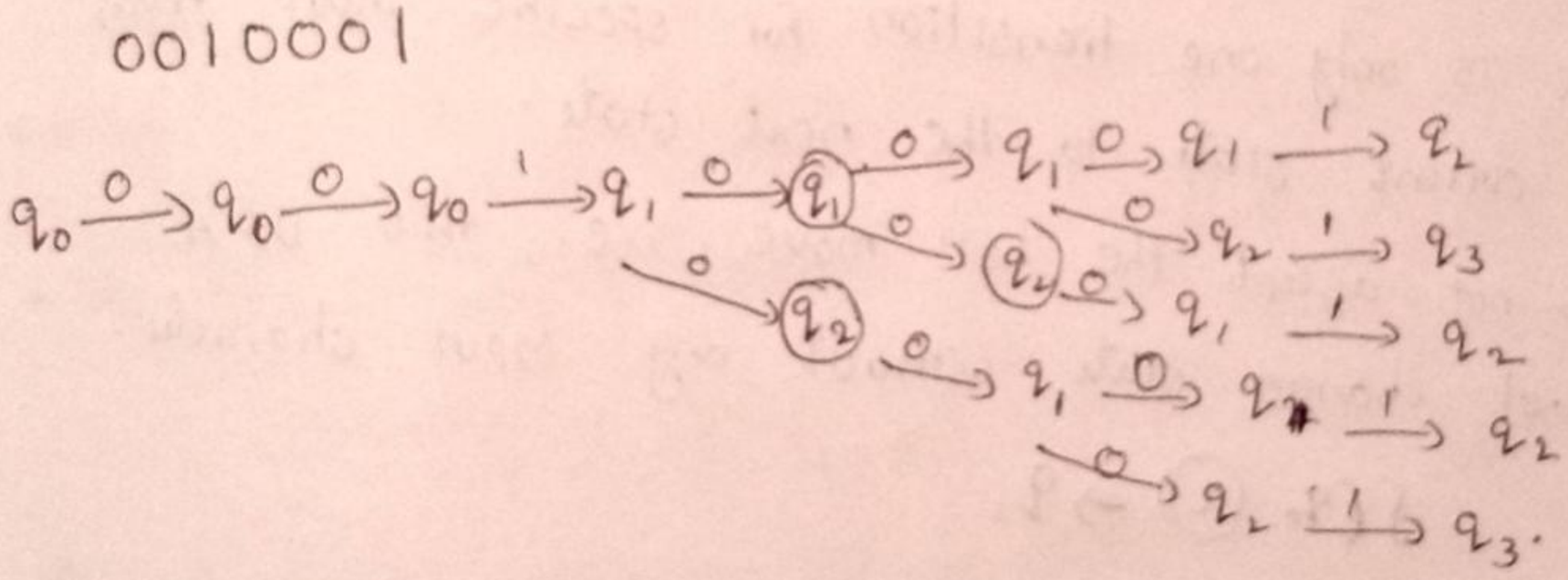
$\omega = 10011$



The string is accepted.

Examples of strings accepted by the ex2: finite automata

0010001



10101

Rejected:

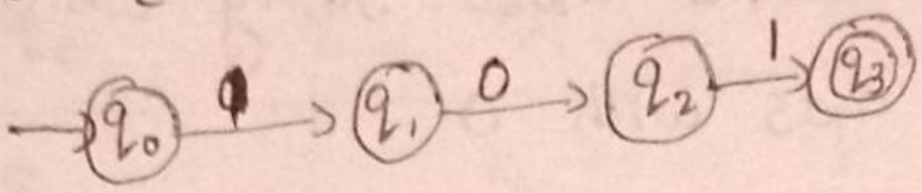
→ 10101

→ 1010.

Example:

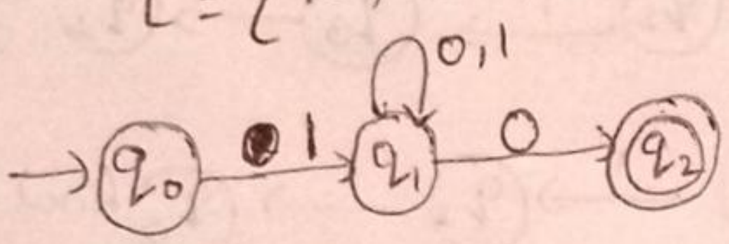
- FA with $\Sigma = \{0, 1\}$ accepts the only input 101.

$$L = \{101\}$$



- FA with $\Sigma = \{0, 1\}$ accepts those string which starts with 1 and ends with 0.

$$L = \{10, 100, 110, 1000, 1010, 1100, 1110, 10000, \ldots\}$$

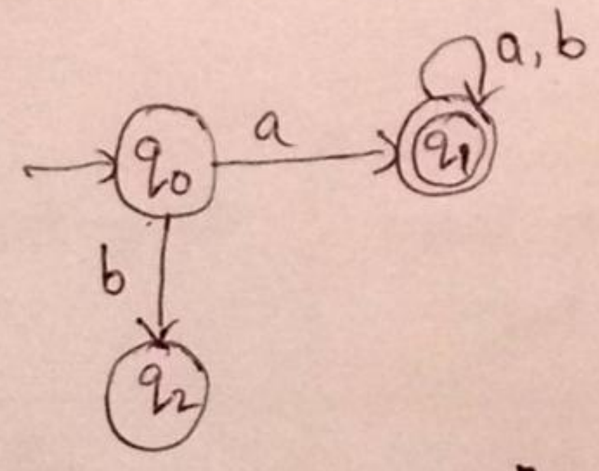


DFA

- Deterministic - uniqueness of the computation
- there is only one transition for specific input from the current state to the next state.
- Does not accept the null move, i.e., the DFA cannot change state without any input character.

$$\delta\{q_0, (a)\} \Rightarrow q_1$$

Ex1:-



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

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

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

$q_0 = q_0$

$F = \{q_1\}$

δ :-

	a	b
q_0	q_1	q_2
q_1	q_1	q_1
q_2	ϕ / -	ϕ / -

Language for the example :-

$L = \{a, aa, ab, aab, aba, aaa, abb...\}$

Ex2:

$L = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$.

$\delta =$

	a	b
q_0	q_0	q_0

$L = \{(a,b)^n \mid n \geq 0\}$

$n = 0 \rightarrow \epsilon$

Ex: $L = \{a, aa, ab, aaa, aab, aba, aabb...\}$

$L = \{a(a,b)^n \mid n \geq 0\}$

Formal Definition of DFA

- A DFA is a 5-tuple.

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

Q : a finite nonempty set of states

Σ : input alphabet

δ : transition function $\quad q_0 \times a \rightarrow q_1$

$Q \times \Sigma \Rightarrow Q$.

q_0 : initial state

F : set of final states. $\quad F \subseteq Q$

Formal Definition of NFA

- A NFA is a 5-tuple

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

Q : a finite non-empty set of states

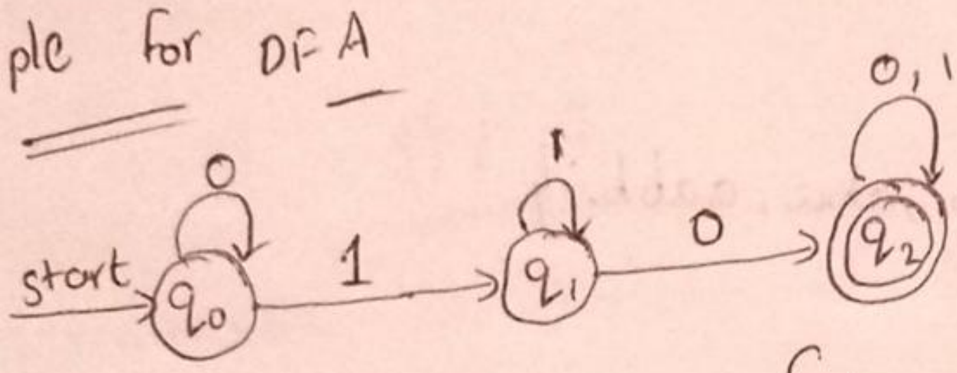
Σ : input alphabet

δ : transition function, $Q \times \Sigma = 2^Q$

q_0 : initial state

F : set of final states, $F \subseteq Q$

Example for DFA



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

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

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

$q_0 = q_0$

$F = \{q_2\}$

δ :-

	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1
$*q_2$	q_2	q_2

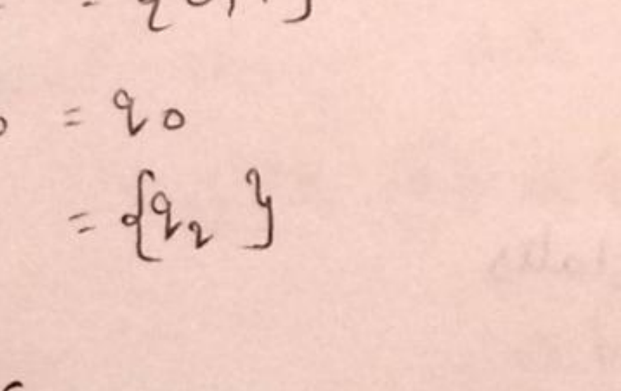
Examples

- DFA with $\Sigma = \{0, 1\}$ accepts all strings starting with 0.

$L = \{0, 01, 00, 000, 001, 010, 011; \ldots\}$

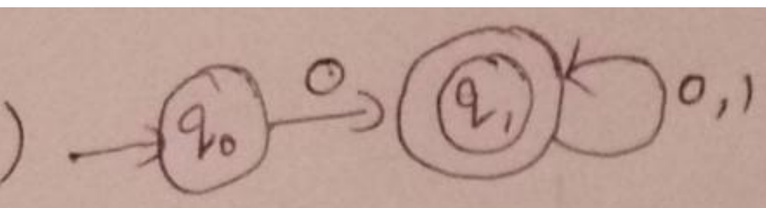
$L = \{0(0,1)^n \mid n \geq 0\}$

1)



	0	1
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_2	q_2

ii)



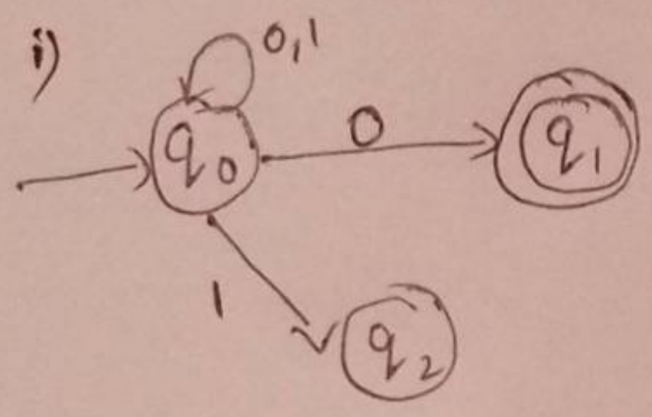
	0	1
q_0	q_1	–
q_1	q_1	q_1

- DFA with $\Sigma = \{0, 1\}$ accepts all ending with 0.

$L = \{0, 00, 10, 000, 010, 100, 110, 0000, \ldots\}$

$L = \{(0,1)^n 0 \mid n \geq 0\}$

i)

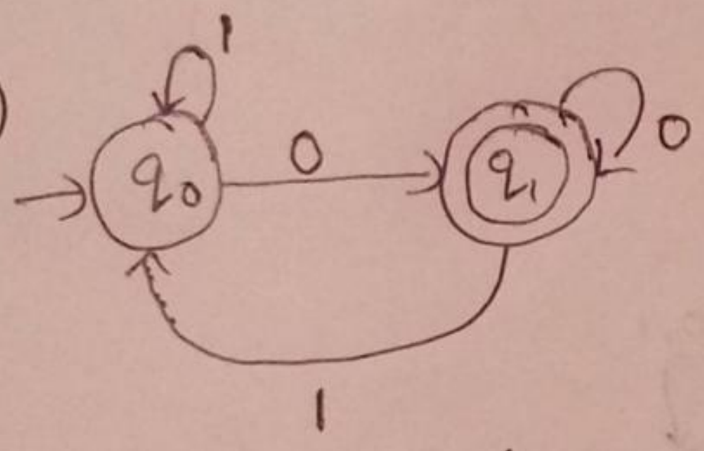


} FA
not DFA

δ:

	0	1
q_0	$\{q_0, q_1\}$	q_0
q_1	–	–

ii)



δ:

	0	1
q_0	q_1	q_0
q_1	q_1	q_0

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

$Q = \{q_0, q_1\}$

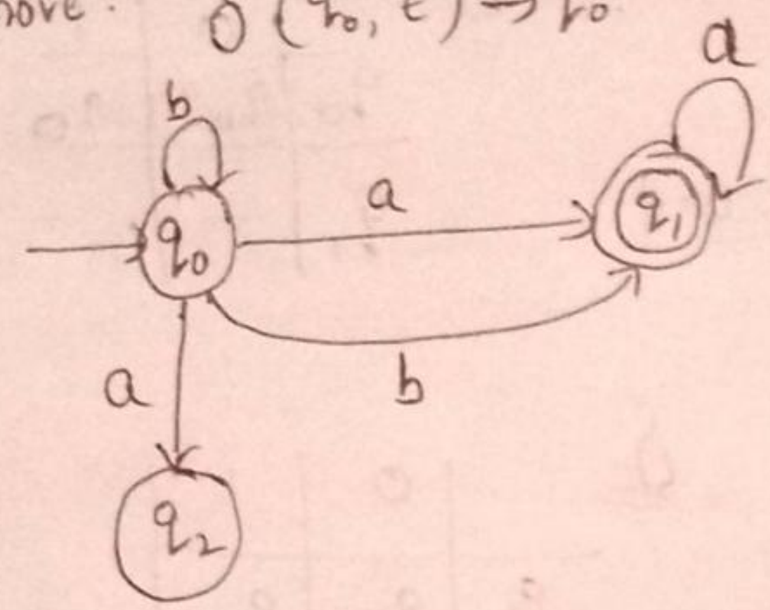
$\Sigma = \{0, 1\}$

$q_0 = q_0$

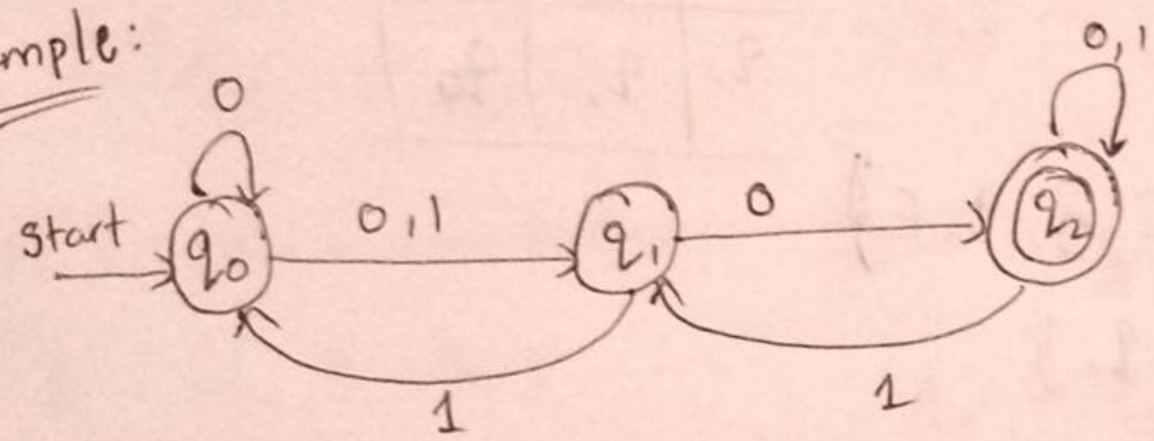
$F = \{q_1\}$

NFA

- The exact state, to which the machine moves cannot be determined.
- For a particular input symbol, the machine can move to any combination of the states in the machine.
- NULL move. $\delta(q_0, \epsilon) \Rightarrow q_0$



Example:



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

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

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

$q_0 = q_0$

$F = \{q_2\}$

δ :-

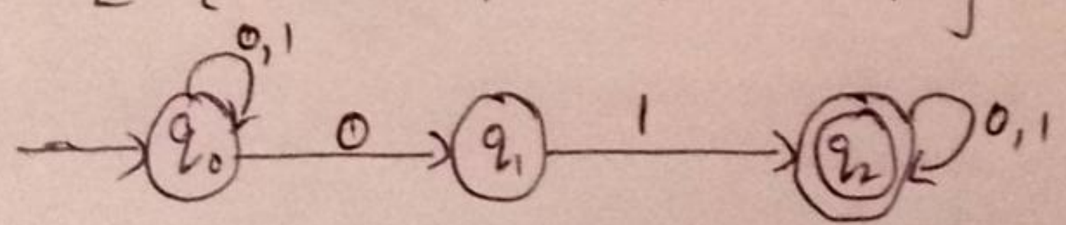
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_1
q_1	q_2	q_0
$*q_2$	q_2	$\{q_1, q_2\}$

Example:

- NFA with $\Sigma = \{0, 1\}$ accepts all strings with 01.

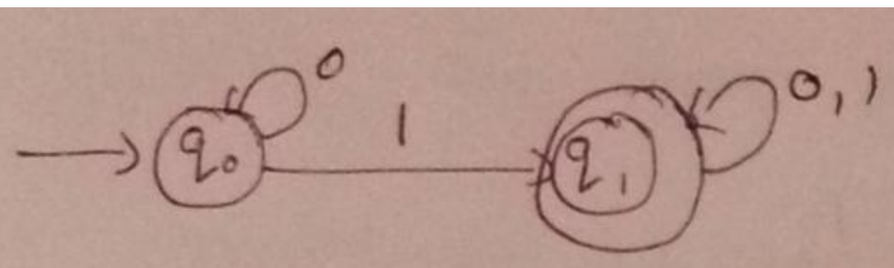
$L = \{01, 001, 0101, 011 \ldots\}$

i)



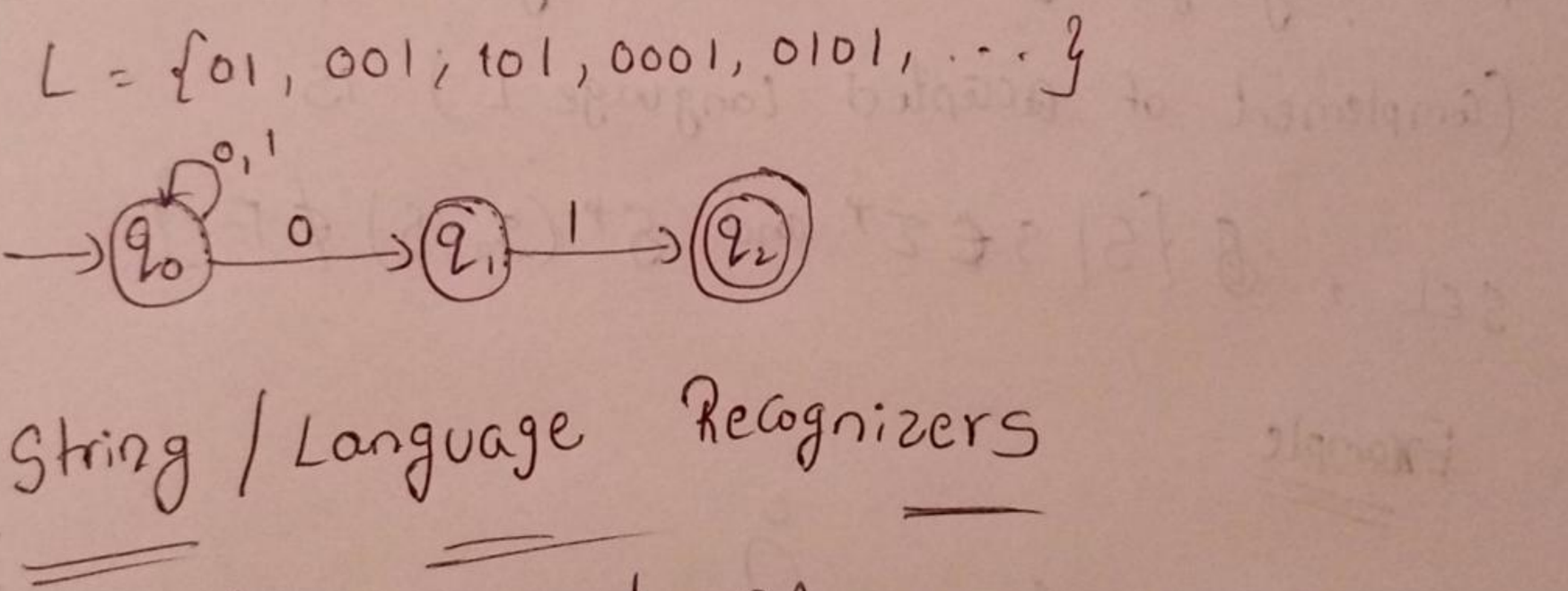
NFA

ii)



- NFA with $\Sigma = \{0, 1\}$ and accept all string of length atleast 2.

$L = \{01, 00, 010, 001, 000, \ldots\}$



- NFA with $\Sigma = \{0, 1\}$ accepts all string ending with 01.

$L = \{01, 001, 101, 0001, 0101, \ldots\}$

String / Language Recognizers

Acceptability by DFA and NFA.

→ A string is accepted by a DFA/NFA iff the DFA/NFA starting at the initial state ends in an accepting state (any of the final states) after reading the string wholy.

A string S is accepted by a DFA/NFA $(Q, \Sigma, \delta, q_0, F)$, if and only if

$$\delta^*(q_0, S) \in F$$

$\forall S \in L$

$\delta^*(q_0, S) \in F$

$\forall S' \notin L$

$\delta^*(q_0, S') \notin F$

The Language L is accepted by DFA/NFA is

$\forall S \in L, \{S \mid S \in \Sigma^* \text{ and } \delta^*(q_0, S) \in F\}$

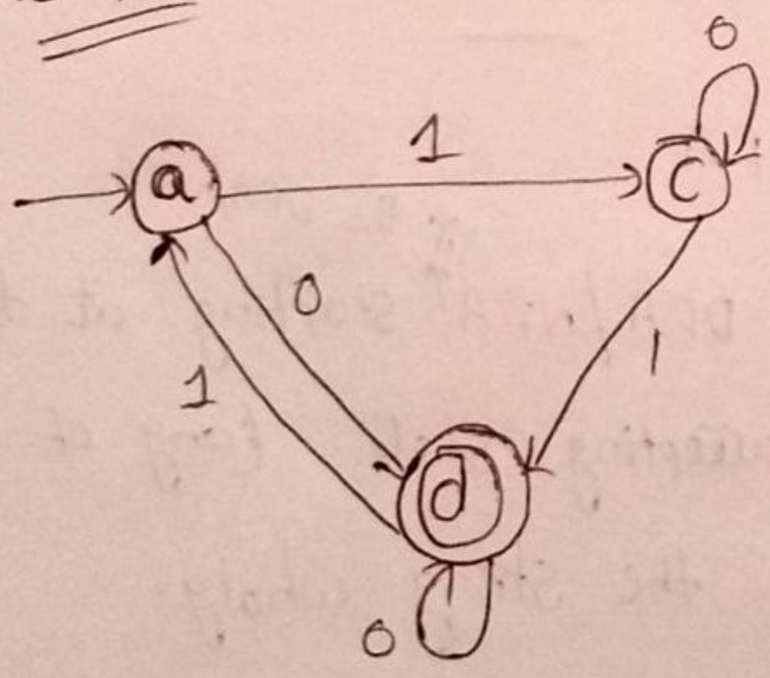
A string S' is not accepted by a DFA/NFA$(Q, \Sigma, \delta, q_0, F)$ if and only if

$\forall S' \notin L, \delta^*(q_0, S') \notin F$

The language L' not accepted by DFA/NFA (Complement of accepted language L) is

$S \in L', \{S \mid S \notin \Sigma^* \text{ and } \delta^*(q_0, S) \notin F\}$

Example



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

$Q : \{a, c, d\}$

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

$q_0 : a$

$F : \{d\}$

δ:

	0	1
→a	d	c
c	c	d
*d	d	a

Strings accepted by the above DFA $\{0, 00, 11, 010, 101, \ldots\}$

Strings not accepted by the above DFA

$\{1, 011, 111, \ldots\}$.

Example - String Acceptance

$\delta^*(q_0, S) \in F$

$S = 10010$

$Q = \{a, c, d\}$

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

$q_0 = a$

$F = \{d\}$

δ :-

$\delta(a, \underset{\uparrow}{1}0010)$

$= \delta(c, \underset{\uparrow}{0}010)$

$= \delta(c, \underset{\uparrow}{0}10)$

$= \delta(c, \underset{\uparrow}{1}0)$

$= \delta(d, \underset{\uparrow}{0})$

$= \delta(d, \epsilon)$

↳ Final state i.e. $d \in F$

~~∴ String S is accepted.~~

∴ String S is accepted.

String Rejection:

$\delta^*(q_0, S') \notin F$

$S' = 111100)$

$Q = \{a, c, d\}$

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

$q_0 = a$

$F = \{d\}$

δ :-

$\Rightarrow \delta(a, \underset{\uparrow}{1}11100)$

$\delta(c, \underset{\uparrow}{1}1100)$

$\delta(d, \underset{\uparrow}{1}100)$

$\delta(a, \underset{\uparrow}{1}00)$

$\delta(c, \underset{\uparrow}{0}0)$

$\delta(c, \underset{\uparrow}{0})$

$\delta(c, \epsilon)$

↘ c is not final state.

$C \notin F$.

∴ String S^1 is not accepted.

It is rejected.