

UNIT - I

Name: Prof Anand Ghau
college: PVGCOE, NASHIK
class: TE COMPUTER
subject: TOC

Introduction to formal Language And FA.

8087777-708

* Formal Language :-

"A formed language is set of string of symbol together with a set of rules that are specific to it."

* Alphabet :-

- A subset of string over an alphabet is a language.
- An alphabet is a finite, non empty set of symbols.
- Σ symbol is used for an alphabets.
- Common Alphabet include:
 - $\Sigma \{0, 1\}$, the binary alphabet
 - $\Sigma \{A, B, \dots, Z\}$, set of uppercase letters.
 - $\Sigma \{0, 1, 2, \dots, 9\}$, the decimal alphabets.

* String :-

- A string is a finite sequence of symbols from alphabets.
- e.g. 10110 is a string from binary alphabet
- 520 is a string from decimal alphabet.
- 'MAN' is a string from roman alphabet.

* Kleene closure :-

Given alphabet Σ , the Kleene closure of Σ is a language given by

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \dots$$

\downarrow \downarrow \downarrow
 ϵ string of length 1 string of length 2.

PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

e.g.

1. If $\Sigma = \{x\}$ then
 $\Sigma^* = \{\epsilon, x, xx, \dots\}$

2. If $\Sigma = \{0, 1\}$ then

* Finite Automata :

→ "Finite automata is also called as finite state machine."

A finite state machine is mathematical model for actual physical process. By considering possible inputs on which these m/c can work, we can analyse their strength & weakness."

Finite automata is used for solving several common types of computer algorithm. Some of them are (Applications) of FA

1. Design of digital circuit
2. String matching
3. Comm² protocol for info² exchange.
4. Lexical analyzer of compiler.
5. Control of lift & working m/c.

* Limitation of Finite Automata :

- it can't be used for computation
- It can't be modified its own input
- It can't be used for context free Grammar/language.
- It can't be used for recursive.

* DFA - Deterministic Finite Automata :

"Deterministic finite Automata is a quintuple.

$$M = (Q, \Sigma, \delta, q_0, F), \text{ where.}$$

Q is a set of state

Σ is a set of alphabet.

$q_0 \in Q$ is the initial state.

$F \subseteq Q$ is the set of final state. and

δ is the transition function, is fun² from $Q \times \Sigma$ to Q .

Representation of DFA.

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

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

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

$$F = \{q_1\}$$

* NFA - Non-deterministic finite Automata.

→ " A Non-deterministic finite Automata is a 5 tuple
 $M = (Q, \Sigma, \delta, q_0, F)$

where, Q = A finite set of state.

Σ = A finite set of input.

δ = A transition function from $Q \times \Sigma$.

q_0 = start / initial state.

F = set of final / accepting state.

Representation of NFA :

$(\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$

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

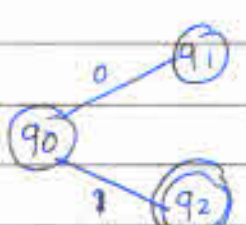
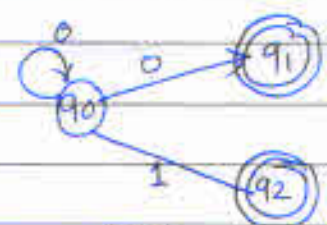
$\Sigma = \{0, 1\}$

δ = trans function

q_0 = q_0 start state.

$F = \{q_2\}$ final state.

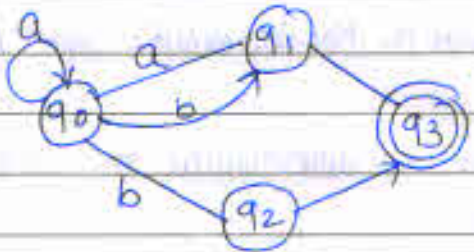
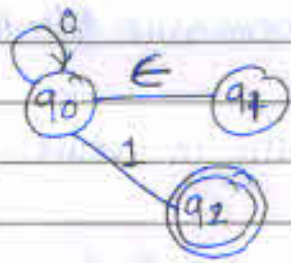
* Compare DFA and NFA.

Sr No	DFA	Sr No	NFA
1.	it stands for Deterministic finite Automata	1.	It stands for Non-deterministic finite Automata
2.	it is deterministic in nature	2.	It is non-deterministic in nature
3.	DFA may have one final state	3.	NFA can have multiple final state.
4.	DFA has single transition for each i/p.	4.	NFA can have multiple transition for i/p.
5.	 <p>as there are single transition for 0 input</p> <p>DFA.</p>	5.	 <p>As there are 2 transition for 0 input } 2 final state</p> <p>NFA.</p>
6.	it has more nos of state	6.	It has fewer nos of state
7.	easy to design	7.	difficult to design

* NFA with ϵ transition

- NFA stand for Non-deterministic Finite Automata
- ϵ is a null symbol.
- An ϵ transition allows transition on ϵ (or no input)
- ϵ (epsilon) is also called as empty string
- It means that machine can make transition without any i/p.

* Compare NFA and NFA- ϵ

Sr No	NFA	Sr No	NFA- ϵ
1.	it stands for Non-deterministic finite Automata without Epsilon ϵ	1.	it stands for Non-deterministic Finite Automata with
2.	NFA is non-deterministic in nature	2.	NFA- ϵ is non-deterministic in nature.
3.	There are no Epsilon ϵ or Empty transition	3.	There is Epsilon ϵ or empty transition.
4.	NFA occupies less space	4.	NFA- ϵ occupies more space
5.		5.	

UNIT - I

Introduction to formal Lang & finite Automata

Name: Anand N. Gharu
College: PVGCOE, Nashik
Class: TE COMPUTER
Subject: TOC

8087777708

* Mealy Machine :

- A mealy m/c M is defined as: o/p associated with transition:

$$M = \{Q, \Sigma, O, \delta, \lambda, q_0\}$$

where,

Q = A finite set of state.

Σ = A finite set of i/p state.

O = A finite set of o/p state.

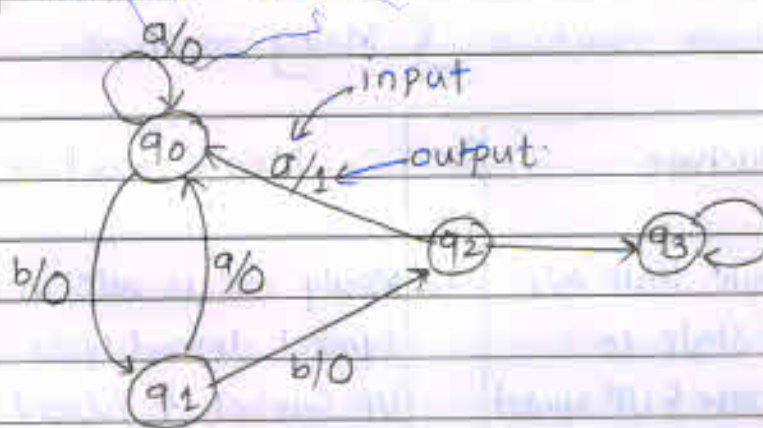
δ = A transition function $\Sigma \times Q \rightarrow Q$

λ = An output function $\Sigma \times Q \rightarrow O$

$q_0 = q_0 \in Q$ is an initial state.

“Mealy machine is a machine in which output symbol depends upon the present i/p symbol and present state of machine.”

e.g.



PROF. ANAND GHARU
PVGCOE NASHIK
anandgharu.wordpress.com

* Moore Machine :

A moore machine M is defined as

$$M = \{Q, \Sigma, O, \delta, \lambda, q_0\}$$

where,

Q = A finite set of state.

Σ = A finite set of i/p alphabet.

O = A finite set of o/p alphabet.

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

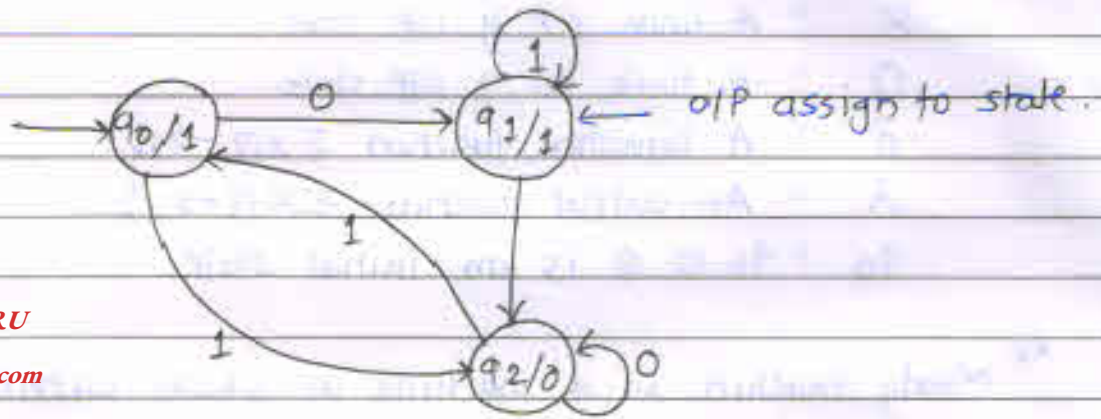
λ = An output function $Q \rightarrow O$

$q_0 = q_0 \in Q$ is an initial state.

* Moore machine

“ Moore machine is a finite state machine in which the next state is decided by current state & current i/p symbol ”
 The output symbol at given time depends only on present state of the machine.

e.g.



* Compare Moore machine & Mealy machine.

Sl. No	Moore machine	Sl. No	Mealy machine.
1.	moore m/c is finite state m/c in which the next state is decided by current state & i/p symbol	1.	Mealy m/c is m/c in which o/p symbol depend upon the present i/p symbol & present state m/c.
2.	output is associated with state. $\lambda: Q \rightarrow O$	2.	output is associated with transition. $\lambda: \Sigma \times Q \rightarrow O$
3.	length of moore machine is one longer by mealy	3.	length of mealy m/c is shorter than moore.
4.	Difficult to implement	4.	Easy to implement
5. e.g.		5. e.g.	