After going through this chapter, you should be able to understand :

- Alphabets, Strings and Languages
- Mathematical Induction
- Finite Automata
- Equivalence of NFA and DFA
- NFA with $\in$ - moves

1.1 ALPHABETS, STRINGS & LANGUAGES

Alphabet

An alphabet, denoted by Σ , is a finite and nonempty set of symbols.

Example:

1. If Σ is an alphabet containing all the 26 characters used in English language, then Σ is finite and nonempty set, and $\Sigma = \{a, b, c, \ldots, z\}$.
2. $X = \{0,1\}$ is an alphabet.
3. $Y = \{1,2,3,...\}$ is not an alphabet because it is infinite.
4. $Z = \{ \}$ is not an alphabet because it is empty.

String

A string is a finite sequence of symbols from some alphabet.

Example :

"*xyz*" is a string over an alphabet $\Sigma = \{a, b, c, \ldots, z\}$. The empty string or null string is denoted by $\in$.

Length of a string

The length of a string is the number of symbols in that string. If w is a string then its length is denoted by $|w|$.

Example :

1. $w = abcd$, then length of w is $|w| = 4$
2. $n = 010$ is a string, then $|n| = 3$
3. ϵ is the empty string and has length zero.

The set of strings of length K (K ≥ 1)

Let Σ be an alphabet and $\Sigma = \{a, b\}$, then all strings of length K ($K \geq 1$) is denoted by Σ^K.

$\Sigma^K = \{w : w \text{ is a string of length } K, K \geq 1\}$

Example:

1. $\Sigma = \{a,b\}$, then

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

   $\Sigma^2 = \{aa, ab, ba, bb\}$,

   $\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

   $|\Sigma^1| = 2 = 2^1$ (Number of strings of length one),

   $|\Sigma^2| = 4 = 2^2$ (Number of strings of length two), and

   $|\Sigma^3| = 8 = 2^3$ (Number of strings of length three)

2. $S = \{0,1,2\}$, then $S^2 = \{00, 01, 02, 11, 10, 12, 22, 20, 21\}$, and $|S^2| = 9 = 3^2$

Concatenation of strings

If w_1 and w_2 are two strings then concatenation of w_2 with w_1 is a string and it is denoted by w_1w_2. In other words, we can say that w_1 is followed by w_2 and $|w_1w_2| = |w_1| + |w_2|$.

Prefix of a string

A string obtained by removing zero or more trailing symbols is called prefix. For example, if a string $w = abc$, then a, ab, abc are prefixes of w.

Suffix of a string

A string obtained by removing zero or more leading symbols is called suffix. For example, if a string $w = abc$, then c, bc, abc are suffixes of w.
A string a is a proper prefix or suffix of a string w if and only if $a \neq w$.

Substrings of a string

A string obtained by removing a prefix and a suffix from string w is called substring of w. For example, if a string $w = abc$, then b is a substring of w. Every prefix and suffix of string w is a substring of w, but not every substring of w is a prefix or suffix of w. For every string w, both w and $\in$ are prefixes, suffixes, and substrings of w.
Substring of $\mathrm{w} = \mathrm{w} - (\text{one prefix}) - (\text{one suffix})$.

Language

A Language L over Σ, is a subset of Σ^*, i. e., it is a collection of strings over the alphabet Σ. ϕ, and $\{\in\}$ are languages. The language ϕ is undefined as similar to infinity and $\{\in\}$ is similar to an empty box i.e. a language without any string.

Example:

1. $L_1 = \{01, 0011, 000111\}$ is a language over alphabet $\{0,1\}$
2. $L_2 = \{\in, 0, 00, 000, ...\}$ is a language over alphabet $\{0\}$
3. $L_3 = \{0^n 1^n 2^n : n \geq 1\}$ is a language.

Kleene Closure of a Language

Let L be a language over some alphabet Σ. Then Kleene closure of L is denoted by L^* and it is also known as reflexive transitive closure, and defined as follows :

$L^* = \{Set\ of\ all\ words\ over\ \Sigma\}$

= {word of length zero, words of length one, words of length two, ...}

$= \bigcup_{K=0}^{\infty}(\Sigma^K) = L^0 \cup L^1 \cup L^2 \cup \ldots$

Example:

1. $\Sigma = \{a,b\}$ and a language L over Σ. Then

   $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$

   $L^0 = \{\in\}$

   $L^1 = \{a,b\}$,

   $L^2 = \{aa, ab, ba, bb\}$ and so on.

   So, $L^* = \{\in, a, b, aa, ab, ba, bb \ldots\}$
2. $S = \{0\}$, then $S^* = \{\in, 0, 00, 000, 0000, 00000, \ldots\}$

Positive Closure

If Σ is an alphabet then positive closure of Σ is denoted by Σ^+ and defined as follows :

$\Sigma^+ = \Sigma^* - \{\in\} = \{Set\ of\ all\ words\ over\ \Sigma\ excluding\ empty\ string\ \in\}$

Example :

if $\Sigma = \{0\}$, then $\Sigma^+ = \{0, 00, 000, 0000, 00000, \ldots\}$

1. 2 MATHEMATICAL INDUCTION

Based on general observations specific truths can be identified by reasoning. This principle is called mathematical induction. The proof by mathematical induction involves four steps.

Basis : This is the starting point for an induction. Here, prove that the result is true for some n = 0 or 1.

Induction Hypothesis : Here, assume that the result is true for n = k.

Induction step : Prove that the result is true for some n = k + 1.

Proof of induction step : Actual proof.

1.3 FINITE AUTOMATA (FA)

A finite automata consists of a finite memory called input tape, a finite - nonempty set of states, an input alphabet, a read - only head , a transition function which defines the change of configuration, an initial state, and a finite - non empty set of final states.

A model of finite automata is shown in figure 1.1.

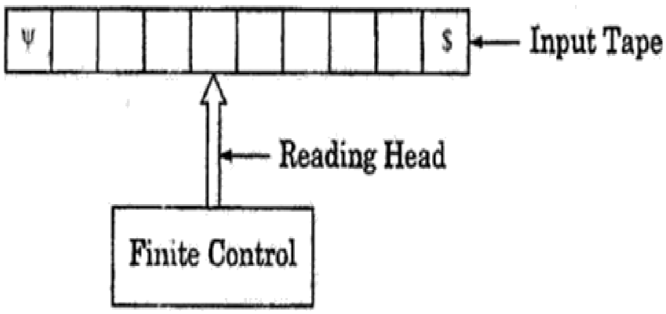


FIGURE 1.1 : Model of Finite Automata

The input tape is divided into cells and each cell contains one symbol from the input alphabet. The symbol 'ψ' is used at the leftmost cell and the symbol '$' is used at the rightmost cell to indicate the beginning and end of the input tape. The head reads one symbol on the input tape and finite control controls the next configuration. The head can read either from left - to - right or right - to -left one cell at a time. The head can't write and can't move backward. So , FA can't remember its previous read symbols. This is the major limitation of FA.

Deterministic Finite Automata (DFA)

A deterministic finite automata M can be described by 5 - tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is finite, nonempty set of states,
2. Σ is an input alphabet,
3. δ is transition function which maps $Q \times \Sigma \rightarrow Q$ i. e. the head reads a symbol in its present state and moves into next state.
4. $q_0 \in Q$, known as initial state
5. $F \subseteq Q$, known as set of final states.

Non - deterministic Finite Automata (NFA)

A non - deterministic finite automata M can be described by 5 - tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is finite, nonempty set of states,
2. Σ is an input alphabet,
3. δ is transition function which maps $Q \times \Sigma \rightarrow 2^Q$ i. e., the head reads a symbol in its present state and moves into the set of next state (s). 2^Q is power set of Q,
4. $q_0 \in Q$, known as initial state, and
5. $F \subseteq Q$, known as set of final states.

The difference between a DFA and a NFA is only in transition function. In DFA, transition function maps on at most one state and in NFA transition function maps on at least one state for a valid input symbol.

States of the FA

FA has following states :

1. **Initial state :** Initial state is an unique state ; from this state the processing starts.
2. **Final states :** These are special states in which if execution of input string is ended then execution is known as successful otherwise unsuccessful.
3. **Non - final states :** All states except final states are known as non - final states.
4. **Hang - states :** These are the states, which are not included into Q, and after reaching these states FA sits in idle situation. These have no outgoing edge. These states are generally denoted by ϕ. For example, consider a FA shown in figure1.2.

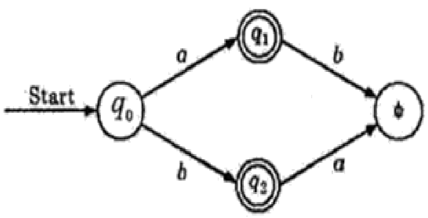


FIGURE 1.2 : Finite Automata

q_0 is the initial state, q_1, q_2 are final states, and ϕ is the hang state.

Notations used for representing FA

We represent a FA by describing all the five - terms $(Q, \Sigma, \delta, q_0, F)$. By using diagram to represent FA make things much clearer and readable. We use following notations for representing the FA:

1. The initial state is represented by a state within a circle and an arrow entering into circle as shown below:

(Initial state q_0)

2. Final state is represented by final state within double circles:

(Final state q_f)

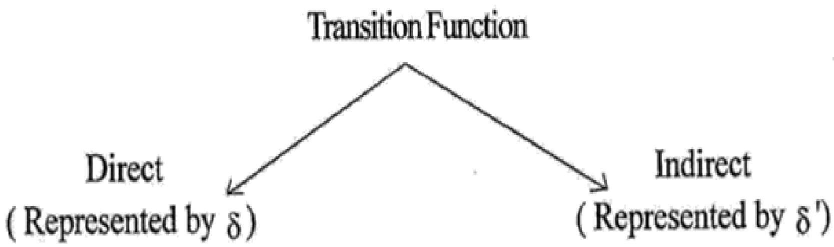
3. The hang state is represented by the symbol 'ϕ' within a circle as follows:

4. Other states are represented by the state name within a circle.
5. A directed edge with label shows the transition (or move). Suppose p is the present state and q is the next state on input - symbol 'a', then this is represented by

6. A directed edge with more than one label shows the transitions (or moves). Suppose p is the present state and q is the next state on input - symbols 'a_1' or 'a_2' or ... or 'a_n' then this is represented by

Transition Functions

We have two types of transition functions depending on the number of arguments.



Direct transition Function (δ)

When the input is a symbol, transition function is known as direct transition function.

Example : $\delta(p, a) = q$ (Where p is present state and q is the next state).
It is also known as one step transition.

Indirect transition function (δ')

When the input is a string, then transition function is known as indirect transition function.

Example : $\delta'(p, w) = q$, where p is the present state and q is the next state after | w | transitions. It is also known as one step or more than one step transition.

Properties of Transition Functions

1. If $\delta(p,a) = q$, then δ (p, ax) = δ(q, x) and if δ' (p, x) = q , then δ' (p, xa) = δ'(q, a)
2. For two strings x and y ; $\delta(p,xy) = \delta(\delta(p,x),y)$, and $\delta'(p,xy) = \delta'(\delta'(p,x),y)$

Example : 1. A DFA $M = (\{q_0, q_1, q_2, q_f\}, \{0,1\}, \delta, q_0, \{q_f\})$ is shown in figure 1.3 .

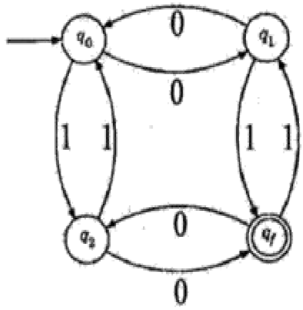


FIGURE 1.3 : Deterministic finite automata

Where δ is defined as follows :

	0	1
→ q_0	q_1	q_2
q_1	q_0	q_f
q_2	q_f	q_0
q_f	q_2	q_1

2. A NFA $M_1 = (\{q_0, q_1, q_2, q_f\}, \{0,1\}, \delta, q_0, \{q_f\})$ is shown in figure 1.4.

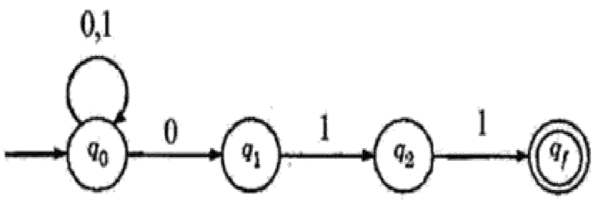
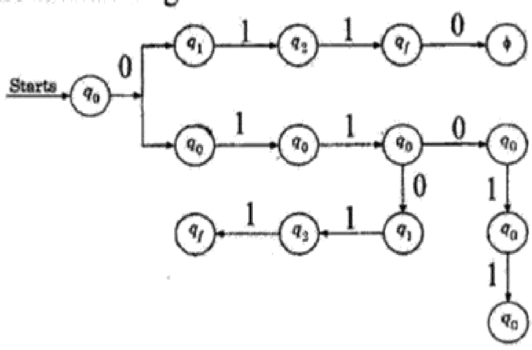


FIGURE 1.4 : Non - deterministic finite automata

3. Transition sequence for the string "011011" is as follows :



One execution ends in hang state ϕ, second ends in non - final state q_0, and third ends in final state q_f hence string "011011" is accepted by third execution.

Difference between DFA and NFA

Strictly speaking the difference between DFA and NFA lies only in the definition of δ. Using this difference some more points can be derived and can be written as shown :

DFA	NFA
1. The DFA is 5 - tuple or quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where Q is set of finite states Σ is set of input alphabets $\delta : Q \times \Sigma$ *to* Q q_0 is the initial state $F \subseteq Q$ is set of final states	The NFA is same as DFA except in the definition of δ. Here, δ is defined as follows : $\delta : Q \times (\Sigma \cup \epsilon)$ to subset of 2^Q
2. There can be zero or one transition from a state on an input symbol	There can be zero, one or more transitions from a state on an input symbol
3. No ϵ – transitions exist i.e., there should not be any transition or a transition if exist it should be on an input symbol	ϵ – transitions can exist i. e., without any input there can be transition from one state to another state.
4. Difficult to construct	Easy to construct

The NFA accepts strings a, ab, abbb etc. by using $\in$ path between q_1 and q_2 we can move from q_1 state to q_2 without reading any input symbol. To accept ab first we are moving from q_0 to q_1 reading a and we can jump to q_2 state without reading any symbol there we accept b and we are ending with final state so it is accepted.

Equivalence of NFA with $\in$– Transitions and NFA without $\in$– Transitions

Theorem : If the language L is accepted by an NFA with $\in$– transitions, then the language L_1 is accepted by an NFA without $\in$– transitions.

Proof : Consider an NFA 'N' with $\in$– transitions where $N = (Q, \Sigma, \delta, q_0, F)$

Construct an NFA N_1 without $\in$– transitions $N_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$

where $Q_1 = Q$ and

$$F_1 = \begin{cases} F \cup \{q_0\} & if \ \in\text{-}closure(q_0)\ contains\ a\ state\ of\ F \\ F & otherwise \end{cases}$$

and $\delta_1(q,a)$ is $\hat{\delta}(q,a)$ for q in Q and a in Σ.

Consider a non - empty string ω. To show by induction $|\omega|$ that $\delta_1(q_0, \omega) = \hat{\delta}(q_0, \omega)$

For $\omega = \in$, the above statement is not true. Because

$$\delta_1(q_0, \in) = \{q_0\},$$

while $$\hat{\delta}(q_0, \in) = \in -closure\ (q_0)$$

Basis :

Start induction with string length one.

i.e., $|\omega| = 1$

Then w is a symbol a, and $\delta_1(q_0, a) = \hat{\delta}(q_0, a)$ by definition of δ_1.

Induction : $|\omega| > 1$

Let $\omega = xy$ for symbol a in Σ.

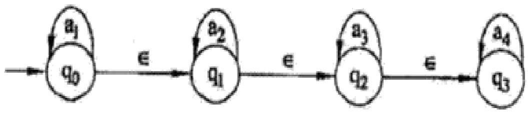
Then $\delta_1(q_0, xy) = \delta_1(\delta_1(q_0, x), y)$

Calculation of $\in$ - closure :

$\in$ - closure of state ($\in$-closure (q)) defined as it is a set of all vertices p such that there is a path from q to p labelled $\in$ (including itself).

Example :

Consider the NFA with $\in$ - moves



$\in -$ closure $(q_0) = \{ q_0, q_1, q_2, q_3 \}$

$\in -$ closure $(q_1) = \{ q_1, q_2, q_3 \}$

$\in -$ closure $(q_2) = \{ q_2, q_3 \}$

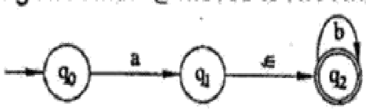
$\in -$ closure $(q_3) = \{ q_3 \}$

Procedure to convert NFA with $\in$ moves to NFA without $\in$ moves

Let $N = (Q, \Sigma, \delta, q_0, F)$ is a NFA with $\in$ moves then there exists $N' = (Q, \in, \hat{\delta}, q_0, F')$ without $\in$ moves

1. First find $\in -$ closure of all states in the design.
2. Calculate extended transition function using following conversion formulae.
   (i) $\hat{\delta}(q, x) = \in -$ closure $(\delta(\hat{\delta}(q, \in), x))$
   (ii) $\hat{\delta}(q, \in) = \in -$ closure (q)
3. F' is a set of all states whose $\in$ closure contains a final state in F.

Example 1 : Convert following NFA with $\in$ moves to NFA without $\in$ moves.



Solution : Transition table for given NFA is

δ	a	b	$\in$
$\rightarrow q_0$	q_1	ϕ	ϕ
q_1	ϕ	ϕ	q_2
(q_2)	ϕ	q_2	ϕ

(i) Finding $\in$ closure :

$$\in -\text{closure}(q_0) = \{q_0\}$$

$$\in -\text{closure}(q_1) = \{q_1, q_2\}$$

$$\in -\text{closure}(q_2) = \{q_2\}$$

(ii) Extended Transition function :

$\hat{\delta}$	a	b
$\rightarrow q_0$	$\{q_1,q_2\}$	ϕ
(q_1)	ϕ	$\{q_2\}$
(q_2)	ϕ	$\{q_2\}$

$$\begin{aligned}\hat{\delta}(q_0, a) &= \in -closure\ (\delta(\hat{\delta}(q_0,\in),a)) \\ &= \in -closure\ (\delta(\in -closure\ (q_0),\ a)) \\ &= \in -closure\ (\delta(q_0,\ a)) \\ &= \in -closure\ (q_1) \\ &= \{q_1,q_2\}\end{aligned}$$

$$\begin{aligned}\hat{\delta}(q_0, b) &= \in -closure\ (\delta(\hat{\delta}(q_0,\in),b)) \\ &= \in -closure(\delta(\in -closure\ (q_0), b)) \\ &= \in -closure(\delta(q_0, b)) \\ &= \in -closure(\phi) \\ &= \phi\end{aligned}$$

$$\begin{aligned}\hat{\delta}(q_1, a) &= \in -closure(\delta(\hat{\delta}(q_1, \in), a)) \\ &= \in -closure(\delta(\in -closure(q_1), a)) \\ &= \in -closure(\delta((q_1, q_2), a)) \\ &= \in -closure(\delta(q_1, a) \cup \delta(q_2, a)) \\ &= \in -closure\ (\phi) \\ &= \phi\end{aligned}$$

$$\hat{\delta}(q_1, b) = \in\text{-}closure\ (\delta(\hat{\delta}(q_1, \in), b))$$
$$= \in\text{-}closure\ (\delta(\in\text{-}closure(q_1), b))$$
$$= \in\text{-}closure\ (\delta((q_1, q_2), b))$$
$$= \in\text{-}closure\ (\delta(q_1, b) \cup \delta(q_2, b))$$
$$= \in\text{-}closure\ (q_2)$$
$$= \{q_2\}$$

$$\hat{\delta}(q_2, a) = \in\text{-}closure\ (\delta(\hat{\delta}(q_2, \in), a))$$
$$= \in\text{-}closure\ (\delta(\in\text{-}closure(q_2), a))$$
$$= \in\text{-}closure\ (\delta(q_2, a))$$
$$= \in\text{-}closure\ (\phi)$$
$$= \phi$$

$$\hat{\delta}(q_2, b) = \in\text{-}closure\ (\delta(\hat{\delta}(q_2, \in), b))$$
$$= \in\text{-}closure\ (\delta(\in\text{-}closure(q_2), b))$$
$$= \in\text{-}closure\ (\delta(q_2, b))$$
$$= \in\text{-}closure\ (q_2)$$
$$= \{q_2\}$$

(iii) Final states are q_1, q_2, because

$\in\text{-}closure\ (q_1)$ contains final state

$\in\text{-}closure\ (q_2)$ contains final state

(iv) NFA without $\in$ moves is