Date 4-1-202
Uploaded 11-1-2021
TCS-402

Theory of Automata

Unit-I

Three fundamental Ideas of This chapter
(1) languages (2) grammars (3) Automata

(1) Languages:— we start with a finite, non empty set Σ of symbols, called The alphabet. From The individual symbols we construct strings, which are finite sequence of symbols from The alphabet.

For Example:— if The alphabet $\Sigma = \{a, b\}$, Then abab and aaabbba are strings on Σ.

With few exceptions, we will use lowercase letters a, b, c --- for elements of Σ and u, v, w --- for strings names. we will write, for example,

$$w = abaaa$$

to indicate that the String named w has the specific Value abaaa

The concatenation of Two strings w and v is the string obtained by appending the symbols of v to The right end of w, that is, if

$$w = a_1 a_2 \text{ --- } a_n$$

and

$$v = b_1 b_2 \text{ --- } b_m,$$

Then the concatenation of w and v denoted by wv, is

$$wv = a_1 a_2 \text{---} a_n b_1 b_2 \text{---} b_m$$

The reverse of a string is obtained by writing the symbols in reverse order; if w is a strings as shown above, Then its reverse w^R is

$$w^R = a_n \text{---} a_2 a_1$$

The length of a string w, denoted by $|w|$, is the number of symbols in the string.

$$|abaaa| = 5$$

we will frequently need to refer to the empty string, which is a string with no symbol at all. it will be denoted by λ. The following simple Relations

$$|\lambda| = 0$$

$$\lambda w = w\lambda = w$$

Power of an alphabet :-

Σ - an alphabet

Σ^K - set of strings of length K, each of whose symbols ~~is~~ in Σ.

Example :-

- $\Sigma = \{a, b, c\}$ - alphabet
- $\Sigma^0 = \lambda/\epsilon$: λ is the only string of length 0
- $\Sigma^1 = \{a, b, c\}$ - strings of length 1.

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

Σ^3 = Strings of length 3

Σ^* – set of all strings over an alphabet Σ

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

$$= \{\lambda\} \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$$

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

so

$$\Sigma^* = \{\lambda\} \cup \Sigma^+$$

<u>languages</u> :→

$\Rightarrow L \subseteq \Sigma^* \rightarrow$ a language over Σ

<u>Example</u>:– 1. languages of all strings consisting of n 0's followed by n 1's, for some $n \geq 0$ is

$\{\lambda, 01, 0011, 000111\}$

$\rightarrow \{0^n 1^n \mid n \geq 0\}$

2. The set of strings of 0's and 1's with equal number of each $\{\lambda, 01, 10, 0011, 0101, 1001, \ldots\}$

3. The set of binary numbers whose value is a prime

– $\{w \mid w$ is a binary integer that is prime$\}$

– $\{10, 11, 101, 111, \ldots\}$

4. Σ^* – a language over Σ

5. ϕ - The empty language (a language over any ~~of~~ alphabet)

6. $\{\lambda\}$ - a language over any alphabet

$$\phi \neq \{\lambda\} / \{\epsilon\}$$

no string — are string of length 0

<u>language</u>:- may contain an infinite number of strings, but strings are drawn from one fixed, finite alphabet.

~~The set~~

<u>Ex</u>:- let $\Sigma = \{a, b\}$. Then

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab \ldots$$

The set $\{a, aa, aab\}$

is a language on Σ. Because it has a finite number of sentences, we call it a finite language. The set

$$L = \{a^n b^n : n \geq 0\}$$

is also a language on Σ. The strings $aabb$ and $aaaabbbb$ are in the language L, but the string abb is not in L. This language is infinite. Most interesting languages are infinite.

The Reverse of L is easily described in set notation as

$$L^R = \{b^n a^n : n \geq 0\};$$

Grammars:- →

A grammar G is defined as a quadruple $G = (V, T, S, P)$,

where V is a finite set of objects called variables

T is a finite set of objects called terminal symbols,

$S \in V$ is a special symbol called the start variable,

P is a finite set of productions.

It will be assumed without further mention that the set V and T are non-empty and disjoint.

Consider the grammar:-

$$G = (\{S\}, \{a, b\}, S, P),$$

with P given by

$$S \to aSb$$

$$S \to \lambda.$$

Then $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

so we can write

$$S \overset{*}{\Rightarrow} aabb$$

The string aabb is a sentence in the language generated by G, while aaSbb is a sentential form.

A grammar G completely defines $L(G)$, but it may not be easy to get a very explicitly description of the language from the grammar.

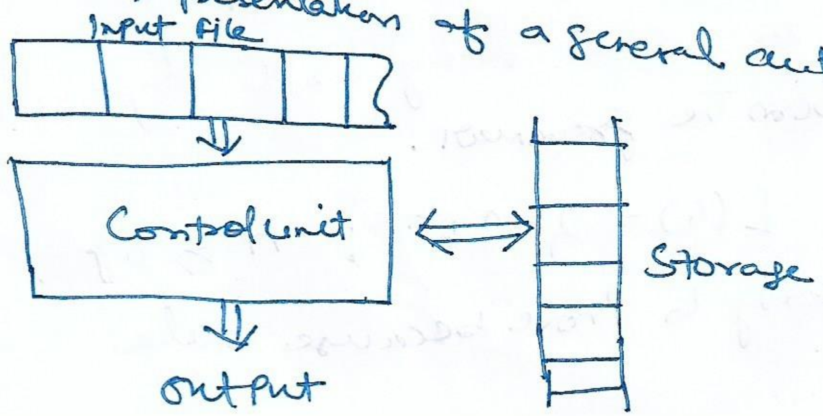
$$L(G) = \{a^n b^n ; n \geq 0\}.$$

and it is easy to prove because rule $S \to aSb$ is recursive.

Automata: → An ~~autonomous~~ Automation is an abstract model of a digital computer. As such, every automation includes some essential features. It has a Mechanism for reading input. It will be assumed that the input is a string over a given alphabet, written on an input file, which the automation can read but not change.

The I/P file is divided into cells, each of which can hold one symbol. The I/P mechanism can read the The I/P file from left to right, one symbol at a time. The I/P Mechanism can also detect the end of the I/P string (by sensing an end of file condition).

The automation can produce output of some form. It may have a temporary storage device, consisting of an unlimited number of cells, each capable of holding a single symbol from an alphabet (not necessarily the same as the input alphabet). The automation can read and change the contents of the storage cells. Finally, the automation can read and change the contents of the Storage cells. Finally, The automaton has a control unit, which can be any one of finite number of internal states, and which can change state in some defined manner. Figure shows a schematic representation of a general automation.



Finite automata

Deterministic automata

non-deterministic automata

A deterministic automaton is one in which each move is uniquely determined by The current configuration. The term Configuration will be used to refer to a Particular State of the control unit, Input file, and temporary storage. This Transition of The automaton from one Configuration to the next will be called a Move.

in Deterministic automata, if we know The internal State, The Input, and The contents of temporary storage, we can Predict The future behaviour of The automaton exactly.

in a non-Deterministic automation, This is not so. At each point, a non-deterministic automaton may have several Possible Moves, so we can only Predict a set Possible actions.

State :- Summarizes the information concerning Past inputs that is needed to determine the behaviour of The systems on Subsequent Inputs.

Example :- * Elevator (control Mechanism)

- Current floor
- up/down
- Collection ~~of~~ not yet satisfied
- Request for service

} States

Deterministic finite automaton (DFA) :→

A 5-tuple $(Q, \Sigma, \delta, q_0, F)$

Q : a finite set of states

Σ : a finite input alphabet

$\delta : Q \times \Sigma \to Q$, The Transition function :

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

$$q \xrightarrow{a} p$$

$q_0 \in Q$: The initial state

$F \subseteq Q$: The set of final / accepting states

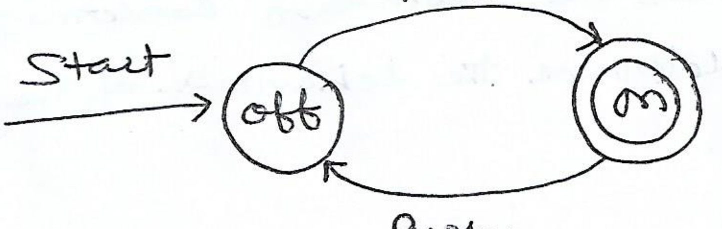
$\Sigma = \{0, 1\}$ binary alphabets

$\Sigma = \{a, b, \text{---} z\}$ → lower case letters

Σ . set of all ASCII characters

$\Bigg\}$ input alphabets

<u>Example:-</u> A finite automata modeling an on / off switch



(Transition diagram)

States: on, off → Q

Input: Push → Σ

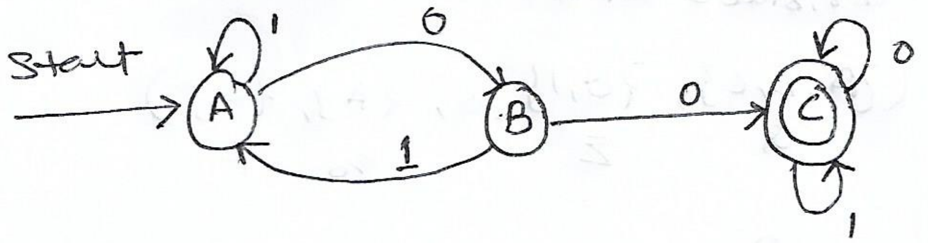
accepting state : on → F

initial state : off → q_0

δ	Push
→ off	on
* on	off

Transition table

Example:-



States : $A, B, C \rightarrow Q$

Input : $0, 1 \rightarrow \Sigma$

accepting state : $C \rightarrow F$

initial state : $A \rightarrow q_0$

Transition table

δ	0	1
$\rightarrow$ A	B	A
B	C	A
* C	C	C

How a DFA Process Strings :-

Consider a DFA

$A = \{Q, \Sigma, \delta, q_0, F)$ and a string $w = a_1 a_2 \cdots a_n$

$q_1 = \delta(q_0, a_1)$

$q_2 = \delta(q_1, a_2)$

$\vdots$

$q_i = \delta(q_{i-1}, a_i)$

$q_n = \delta(q_{n-1}, a_n)$

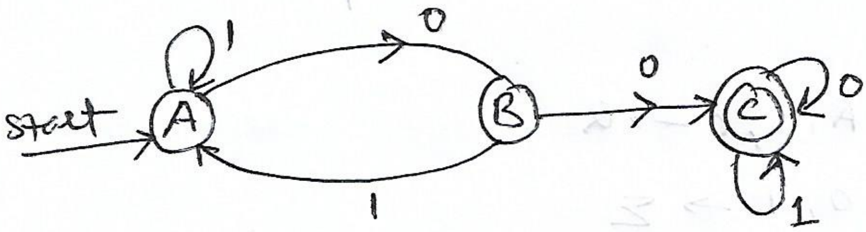
DFA A accepts the string $w = a_1 a_2 \cdots a_n$ if $q_n \in F$

if not, A Rejects w.

Example:—

Consider DFA:

$$(\underbrace{\{A,B,C\}}_{Q}, \underbrace{\{0,1\}}_{\Sigma}, \delta, \underbrace{\{A\}}_{q_0}, \underbrace{\{C\}}_{F})$$



Consider string $w = 101001$

Current state	Symbol read	New state
A	1	A
A	0	B
B	1	A
A	0	B
B	0	C
C	1	C

→ final state which is accept state

Also, The above DFA

- does not accept 11101
- accepts 0001
- accepts all strings of 0's and 1's with Two Consecutive zeros somewhere.

* language accepted by the given DFA

↳ Regular language