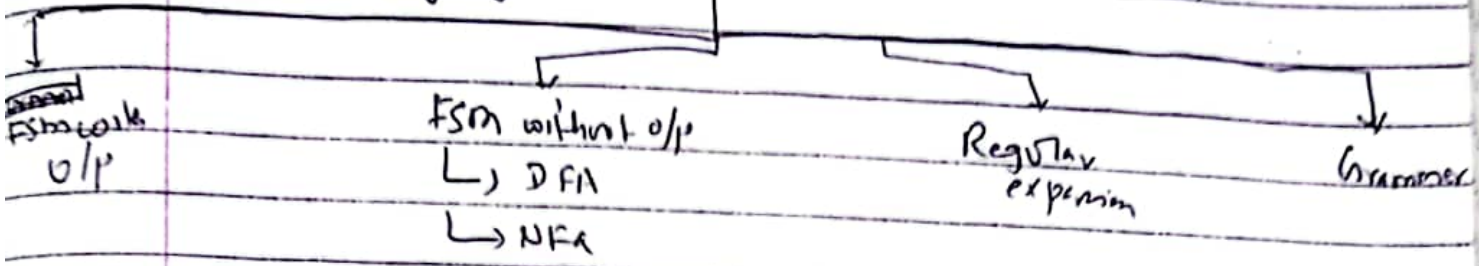
Chapter: VI

Language Grammer and Automata.



Formal definition of finite state machine with output:

=> A finite state machine with output is defined $M = (I, O, S, \delta, f, g)$ where,

(Non empty) I = finite set of input symbols

(Non empty) O = finite set of output symbols

(Non-empty) S = finite set of states

(Non-empty) δ = An initial state $\delta \in S$

f = state transition function (STF)

$f: S \times I \to S$

g: output function or machine function defined as

$g: S \times I \to O$

* let,

$I = \{a, b\}$

$O = \{0, 1\}$

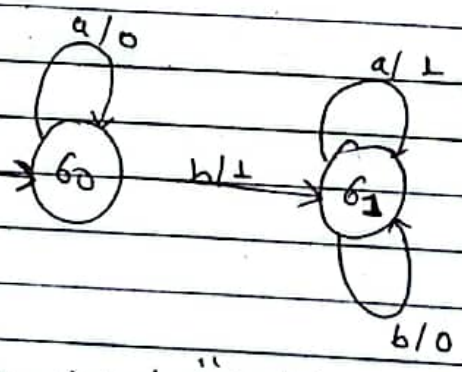
$S = \{S_0, S_1\}$

$S = S_0$ (Initial state)

f and g are given by following table.

I / S	f		g	
	a	b	a	b
S_0	S_0	S_1	0	1
S_1	S_1	S_1	1	0

Find the output for the input string "aabbaba"



now, input = "aabbaba"

current state	Input	next output state	output
S_0	a	S_0	0
S_0	a	S_0	0
S_0	b	S_1	1
S_1	b	S_1	0
S_1	a	S_1	1
S_1	b	S_1	0
S_1	a	S_1	1

output = 0010101.

(*) Draw the transition diagram for the given given finite state machine where

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

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

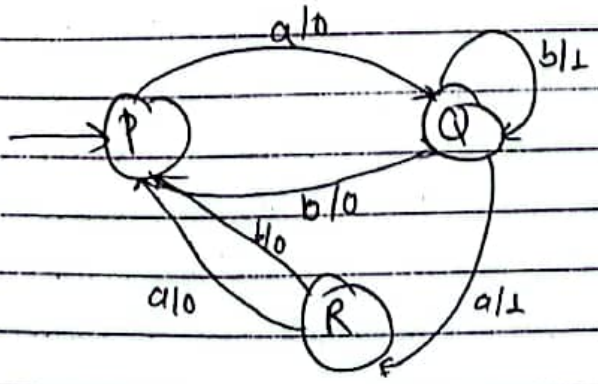
$S = \{P, Q, R\}$

$G = P$

and f and g are defined by following table. Find the output string corresponding to input string "aabbaba".

S	f		g	
Σ	a	b	a	b
P	Q	Q	0	1
Q	R	Q	1	1
R	P	P	0	0

Solution Input = "aabbaba"



Difference between

Non-Deterministic finite automata (NFA/NOFA)

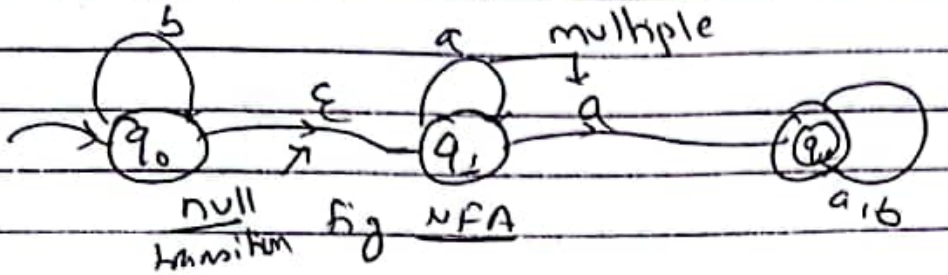


fig NFA

V.V.I

The key difference between DFA and NFA are

a) - NFA can have null transition whereas DFA cannot

b) - NFA can have multiple transition for same input

c) NFA doesnot necessarily required transition for each input whereas DFA Requires transition for each input.

The formal definition of NFA is:

$$m = (\varepsilon, S, \delta, A, f)$$

where,

ε =

S =

δ =

A =

f = state transition function defined as $f: S \times \varepsilon \sim 2^{|S|}$

Regular Expression.

operators:

(a) Union (+) = a+b

(b) Concatenation = a.b.

c) klenee closure (*) = a^*

d) positive closure (+) = a^+.

* Write regulara expremion for following language

a) Starts with ab.

$R.E = a.b(a+b)^*$

b) Ends with ab.

$R.E = (a+b)^* a b$

c) Starts and ends with same symbol.

$R.E = a(a+b)^*a + b(a+b)^*b + a + b$.

* language containing the shing with exactly 1 b.

a) $R.E = a^* b a^*$

b) Exactly a's b

$R.E = a^* b a^* b a^*$

(c) number of b is less than or equal to 2.

$R.E = a^* + a^* b a^* + a^* b a^* b a^*$

(d) first symbol of shing is a and third symbol is b.

$R.E = a(a+b) \cdot b(a+b)^*$

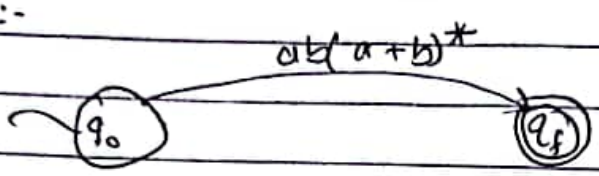
* Every 0 is followed by 11?

R.E = [illegible] 1* (011)* 1*

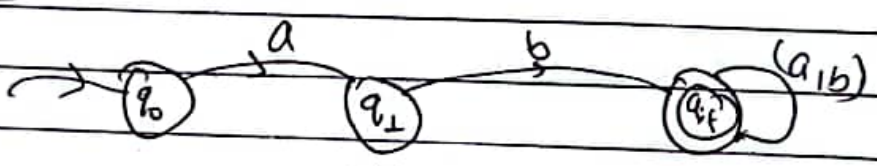
Regular expression to finite automata.

Q1. R.E :- ab(a+b)*

Step I :-



Step II



NFA -

$(a+b)^*$

$A \to aA$

$A \to bA$

$A \to \varepsilon$

Grammer

$G = (V, T, P, S)$

V → set of variables

T → set of terminals ie by lowercase, digit

P → set of production rule

$\alpha \to \beta$

left hand, right hand

S → start symbol.

* Write a grammer that

i) starts with 'ab', $\Sigma = (a, b)$

R.E = $ab(a+b)^*$

i) $S \to abA$

ii) $A \to aA$

iii) $A \to bA$

iv) $A \to \varepsilon$

$S \to abA$

$= abbA$

$= abbaA$

$= abba\varepsilon$

$= abba$.

The required grammer is,

$T = \{a, b\}$

$V = \{S, A\}$

$S = S$

P: $S \to abA$

$A \to aA$

$A \to bA$

(*) Length of string is exactly two $\Sigma=(a,b)$.

Solution

$R.E = (a+b).(a+b)$

$= (aa+ab+ba+bb)$

$S \longrightarrow A.A$

$A \longrightarrow a$

$A \longrightarrow b$ } or $A \rightarrow a/b$

(a) Ends with 'ba' over $\Sigma=(a,b)$

(*) Starts and ends with same symbol over $\Sigma=(a,b)$

(*) Starts and ends with different symbol over $\Sigma=(a,b)$

$R.E: a(a+b)^* b + b(a+b)^* a$

$S \rightsquigarrow aAb \,/\, bAa$

$A \rightarrow aA/bA/\varepsilon$

(*) Let G be the grammer with vocabulary
$V = \{S, A, a, b\}$
$T = \{a, b\}$
start symbol = S
P: $S \to aA$
$S \to b$
$A \to aa$
what is $L(G)$ of this grammer.
Soln
(i) $S \to b$
(ii) $S \rightsquigarrow aA$
$\to aaa$

(*)
$G = \{V, S, V_0, \to\}$
where,
$V = \{V_0, V_1, V_2, a, b, c\}$
$S = \{a, b, c\}$
$V_0 \Rightarrow aaV_0$
$V_0 \Rightarrow bV_1$
$V_1 \Rightarrow cV_2b$
$V_1 \to cb$
$V_2 \to bbV_2$
Determine whether the following string is language or not:
a) aabcb b) abbcb c) aaaabcbb d) aaaabcbbb
Solution
$V_0 \to aaV_0$
$\to aabV_1$
$\to aabcb$
This is language.

b) abbcb.

$V_0 \rightarrow aaV_0$

This is not language

c) aaaabcbb.

$V_0 \rightarrow aaV_0$

$= aaaaV_0$

$= aaaabV_1$

$= aaaabcV_2b$

$= aaaabcbbb.$

d) aaaabcbbb.

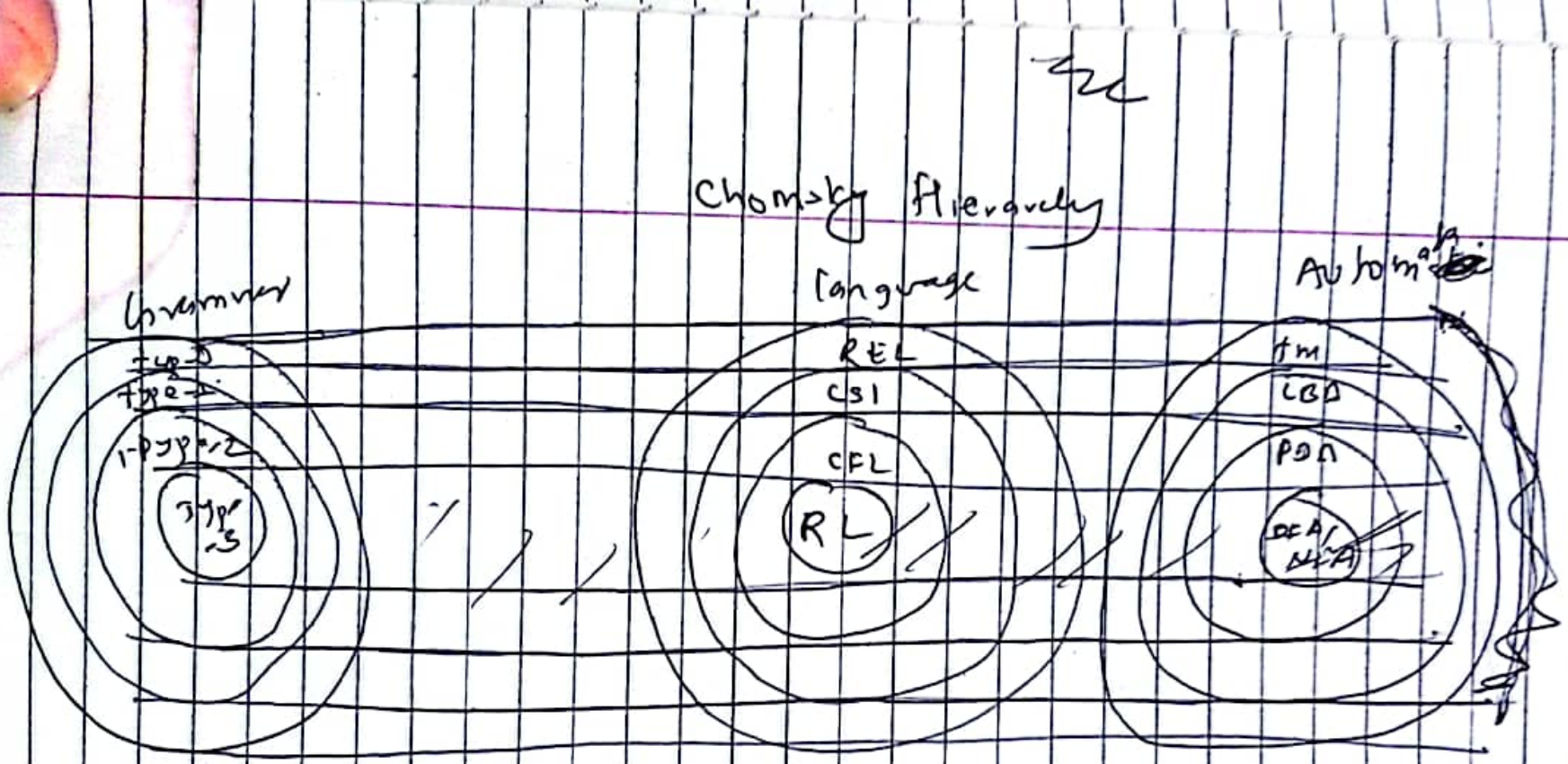
$V_0 \rightarrow aaV_0$

$aaaaV_0$

$aaaabV_1$

$aaaabcV_2b$

$aaaabcbbV_2b$


Chomsky Hierarchy
Grammar
Language
Automata
Type-0
Type-1
Type-2
Type-3
REL
CSL
CFL
RL
TM
LBA
PDA
DFA/NFA

a) Type-0 Grammer

The type 0 Grammer gives

$\gamma \rightarrow \beta$

$\gamma \in (T+N)^* N (T+N)^*$

$\beta = (T+N)^*$

Type -0 grammer is also known as recursively enumerable grammer that generate recursively enumerable language. which is accepted by turing machine

b) Type-1: Grammer

The production is given by.

$\alpha \rightarrow \beta$;

$\alpha \in (T+N)^* N (T+N)^*$

$\beta \in (T+N)^+$

$|\alpha| \leq |\beta|$

Type-1 grammer is also known as context sensitive grammer that generates context sensitive language which is accepted by linearly bound automata.

c) Type-2 Grammer

The production rule is given by.

$\alpha \rightarrow \beta$;

$\alpha \in N$

$\beta \in (T+N)^*$

$|\alpha| = 1$.

Type grammer is also know context free grammer that generate context free language which is accepted by push down automata

d) Type-3 Grammer

The production is given by

or

$A \to a$ $\quad$ $A \to a$

$A \to Ba$ $\quad$ $A \to aB$

$A, B \in N$

$|A| = |B| = 1$

$a \in T^*$

Type-3 grammer is also known as regular grammer that generates regular language which is accepted by DFA/NFA.