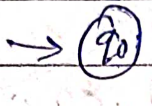
TOC

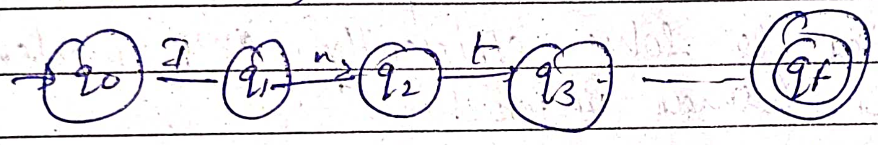
→ Initial State :-

denoted by single circle and arrow

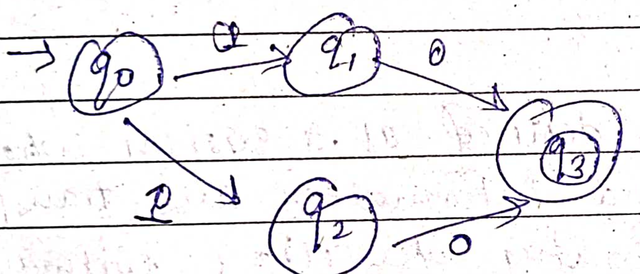


→ Finale state :- denoted by double circle.

int a, b;



← Final State



set Q contains no. of state

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

→ Initial State = q_0. Finale State = q_3

→ Input symbol = $\{0, 1\} = \Sigma$

transaction → in which state which input is applied - give detail about path.

represent with delta

$\delta(q_0, 1) \; q_2$

Previous State, Input, Next state

$\delta(q_0, 1) = q_1$

$\delta(q_2, 0) = q_3$

$\delta(q_1, 0) = q_3$

table of previous transition :-

rows = state
column = input

		0	1
1	$\rightarrow q_0$	q_1	q_2
2	q_1	q_3	—
3	q_2	q_3	—
4	$* q_3$	—	—

~~language accepted~~

$L\{10, 00\}$

01/10/22

Defination → It is a branch of computer science and mathematics to solve efficiently any problem without human interaction.

Automation :-

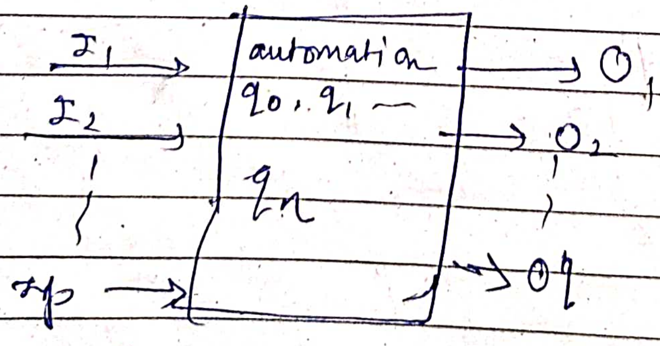
An automation is defined as a system where energy, material and information are transferred and used for performing functions without direct participation of man.

or

It is a system where information and knowledge is transferred for solving problem without human interaction.

Model of automation →

Automation



Date
Page

L A G → Grammar
Automata

Language →

Symbol → Building block of the language
smallest unit of the language.

= {a, b, c, 0, 1, 2, ..}

Alphabate → Σ - sigma
finite set of symbols → It can't be infinite

String → Collection of Alphabates
Sequence of Alphabates

Σ {a, b} → {a, b, aa, bb, ba}

length of the string → How many symbols are present in string.

If length is 2 then

Σ {a, b} = {aa, ab, ba, bb}

If w = 01101 = |W| = 5
W = abcdbg = |W| = 6

Language → Collection of all the possible strings
language can be infinite also

Ex → String starts with a end with a.

Ex → L1 = ~~Two~~ Strings of length 3

$\Sigma^0 = \varepsilon$ - Epsilon

= length of string can be also 0 and it is defined by

Automata → (model, machine)

→ In this its a mathematical model which helps us to know whether the string is ~~present in~~ a part of language or not.

Power of Σ (sigma) :—

minimum power can be 0.

Σ^0 = set of all strings with length '0' = ε, λ

Σ^1 = ——— " ——— '1' = {a, b}.

Σ^2 = ——— " ——— '2' =

$\Sigma \cdot \Sigma = \{a,b\}\{a,b\} = \{aa, ab, ba, bb\}$

Σ^3 = ——— " ——— '3' =

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

$= \{aa, ab, ba, bb\}\{a,b\}$

$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

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

Σ^* (Kleene closure)

→ (set of all strings of all length possible) on

→ Infinite language

Σ^+ → positive ~~closure~~ closure → all string possible except ε

identity element exists which is null ← $= \boxed{\Sigma^* - \varepsilon = \Sigma^+}$

$\{\Sigma, Q, P, \delta,$

⇒ .

Grammar :—

~~Construct DFA in TOC~~ →

Grammar :→

A grammar G is defined as quadruple.

$G = \{V, T, P, S\}$

→ It's a standard way of representing a language.

→ Rules used to generate string of language L.

4 ~~tuple~~ :—

(V, T, P, S)

variable, Terminal, Production rule, Starting symbol

$S \rightarrow SS \mid aAS \mid a,$

$A \rightarrow SbA \mid ba$

~~"aabaa"~~ "aabaa"

$\Rightarrow S \rightarrow SS$

$\rightarrow a\,aAS$

$\rightarrow aabaS$

$\rightarrow$ aabaa"

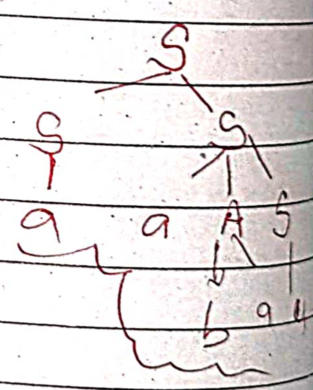
$S \rightarrow SS$

aS

$a\,aAS$

$a\,abaS$

"aabaa"



Date
Page

* Characteristic of automation are now described as

① Input → at each of the discretes instant of time $t_1, t_2 \ldots t_n$ the input value $I_1, I_2 \ldots$ which can take finite number of Input be fixed value from input alphabet Σ.

② output → $O_1, O_2 \ldots O_q$ are the output of model each of which can take a finite number of fixed value from an output O.

③ State → at any instant of time the automation can be one of states q_1, q_2 ---

④ State relation → the next state of an automation at any instant of time is determined by the present state and present input.

⑤ Output relation →

depends on previous state

the output is related to either state only or to both the input and state. only one state can be taken as once on reading "input" symbol the automation moves to next state → see Amair pdf

* Alphabets = finite non-empty set of symbols denoted by Σ

eg = $\Sigma = \{a, b, c\}$ ← set of alphabet this is known as termination

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

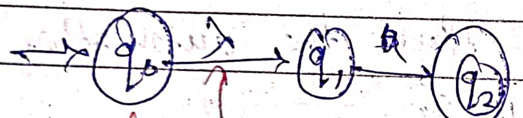
Date ____
Page ____

String :- A sequence of symbols from the alphabets.

Ex :- $w = 01101$ $\Sigma = \{0,1\}$ $w = abcbca$ $\Sigma = \{a,b,c\}$

Empty string :- represented by ϵ or λ

→ also string with no symbol defined as empty string.



here we are not consuming any symbol while going from q_0 to q_1

Length of a String :- No. of symbols present in the string

If $w = 01101$ $|w| = 5$

$w = abccba$ $|w| = 6$

Substring :- X is a substring if it appears consecutively in a string W.

X can be at front portion and middle, end.

$L_1 = \{\varepsilon, a, aa, b\}$ $L_1 \cdot L_2$

$L_2 = \{\cancel{\varepsilon} b, ab, c\}$ ε

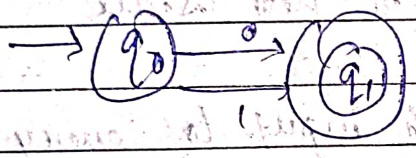
Operation on string :-

(1) Concatenation :-

$X = 00 \quad y = 11$

After concatenation $xy = 0011$

(2) OR → This makes parallel path while transition

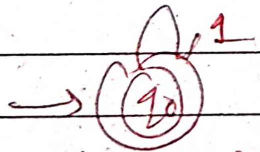


} at a time only one input will be accepted

(3) Closure (Repetation) :- It can be repeated many times no limit

→ Closure means Repeatation of any symbol.

→ denoted as Σ^* or 1^*



$= 1\ 1\ 1 ---$

set of this $= \{$ null, 1, 1,1, 111 --- $\}$

If we remove null it'll become equivalence and denoted by $1^+ = \{1, 11, 111 --- \}$

Date
Page

12/10/22

DFA to NFA
[DFA & NFA]

1. **DFA** it refers to Deterministic finite Automation

A finite automation is said to be deterministic if corresponding to an input symbol there is a single resultant state i.e. there is only one transition.

or

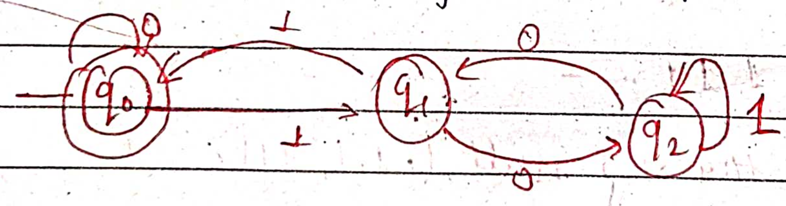
For every single input there will be next state.

→ there should have path for each input to every next state

every input there is 1 next state.

→ automation is represented with M, Q is set of state, Σ is set of alphabets. δ shows mapping or transition q_0 is for initial state f is for final state.

eg -



$Q = \{q_0, q_1, q_2\}$ $\Sigma = \{0, 1\}$ Check → $W = 0010010$ Accepted string

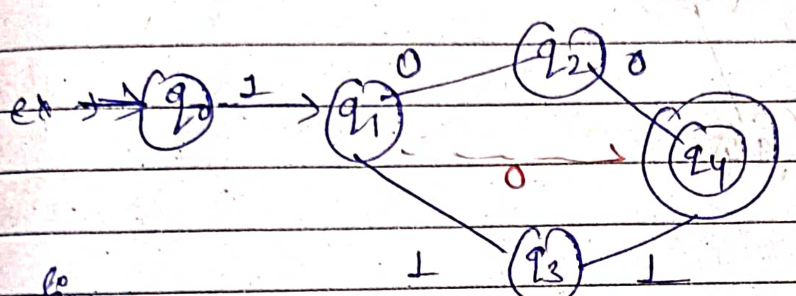
$\delta(q_0, 0010010)$
$\delta(q_0, 010010)$
$\delta(q_0, 10010)$
$\delta(q_1, 0010)$
$\delta(q_2, 010)$

$\delta(q_1, 10)$
$\delta(q_0, 0)$
$\delta(q_0, \lambda)$ — when nothing left in string take λ and if state is == final state then we found the string

Qs M 8 T P

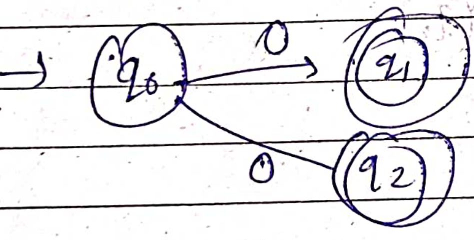
10

2. NFA :- Non-deterministic finite automation
it is said if their is more than one possible transition from one state on same input.



Since there is two path for $q_1 \to q_4$ this NFA
→ even if in Σ there exist more symbol which is not present in automation then it is NFA

→ This also has same 5 tuples as DFA but it does have difference in δ (delta)



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

$Q = \{q_0, q_1, q_2\}$ then will get power set for all the state

i.e. $Q \times \Sigma = 2^Q$ $= \{\{\phi\}, \{q_1\}, \{q_0\}, \{q_2\}, \{q_0 q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

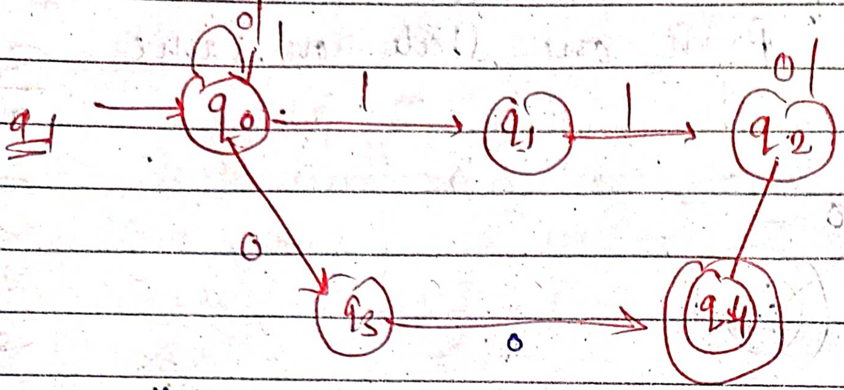
$q_0 \; 0 \longrightarrow \phi$

$\{a \; b \; c\}$

$q_0 \} \longrightarrow q_0 \; q_1$

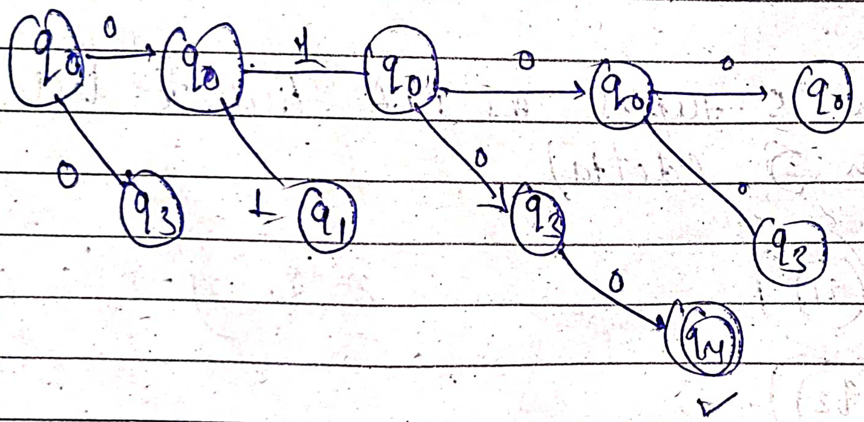
powers = $\{\phi, a, b, c, ab, bc, ca, abc\}$

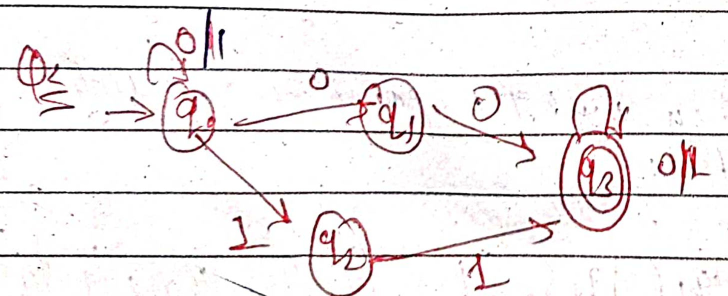
Checking whether string is ~~acep~~ accepted or not



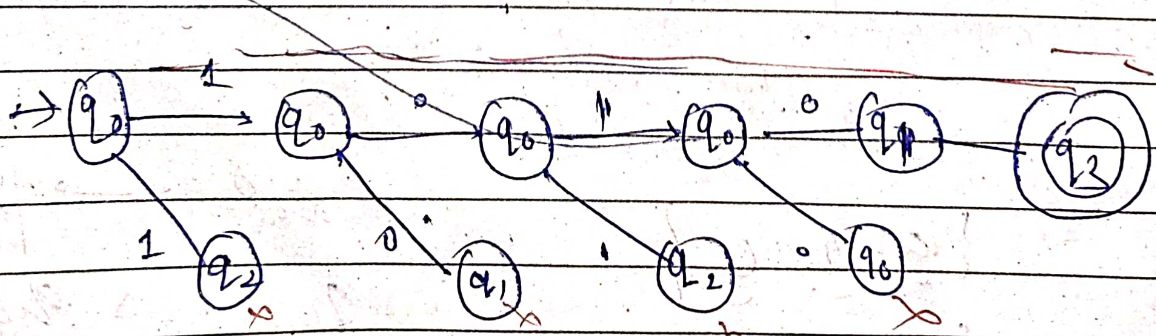
$w = 0100$

= transition system for a ~~NFA~~ non-deterministic auto





$w = 10100$

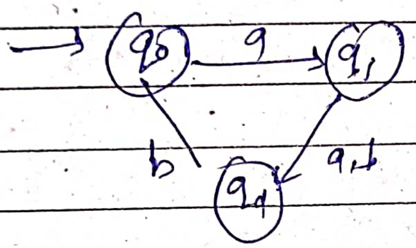


13/10/22 **Conversion of NFA to DFA**

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

transition system - $\rightarrow (q_0) \xrightarrow{a} (q_1)$

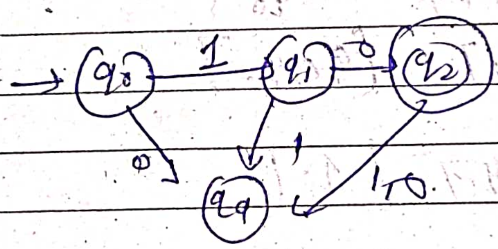
Since its NFA we need to have path with all input and send then to dummy state $(q_d) \rightarrow \{\phi\}$



} Still its output will be ϕ only

This is easy to implement when no multiple path available in NFA.

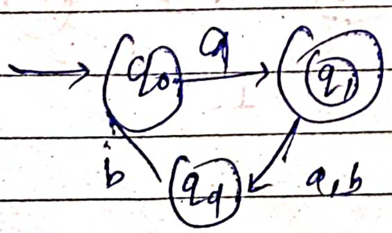
Ex $\Sigma = \{0, 1\}$



→ Now steps of conversion

(1) Convert into tabular form.

State → row Q Symbol → column Σ

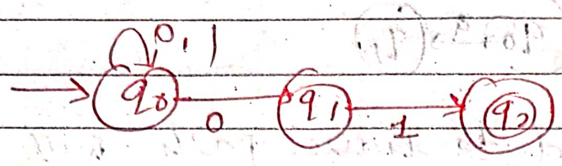


Tabular

	a	b
$\rightarrow q_0$	q_1	ϕ
q_1	ϕ	ϕ

2nd step copy first row as it is and then evaluate the unevaluated expression.

Ex-2



tabular form

	0	1
$\rightarrow q_0$	$[q_0, q_1]$	q_0
q_1	ϕ	q_2
$* q_2$	ϕ	ϕ

Now copy first row as it

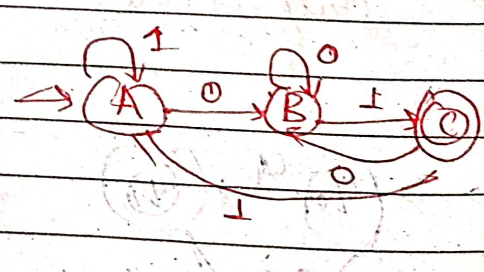
	0	1
$\rightarrow q_0$	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$* [q_0, q_2]$	$[q_0, q_1]$	$[q_0]$

→ evaluate the unevaluated expression
→ start with each element
→ don't write ϕ term

now changing the variable

$q_0 \rightarrow A$ $[q_0\ q_1] \rightarrow B$ $[q_0\ q_2] \rightarrow C$

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



14/10/22

Regular Expression :—

The regular expressions are useful for representing certain sets of strings in an algebraic fashiod.

→ It is a common representation of all strings in a language.

Set L = { all accepted string }

→ Initial state can be also final state

Ex: $a = e^*$ = { null, a, aa ---- }

collection of all strings. ⇒ (a^4) a null

loop → *

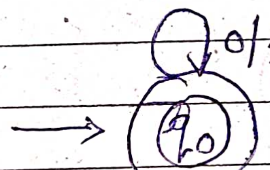
① $\Sigma = \{0, 1\}$

Write RE for binar no.

→ at a time use 0 or 1

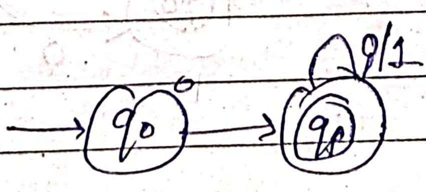
~~min|w| = 1~~

$(0+1)^*$



② Binary numbers starting with string 0.

$0(0+1)^*$

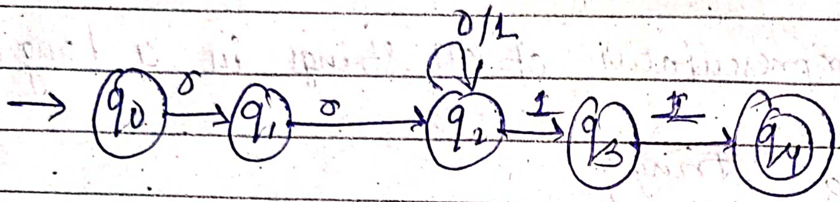


min|w| = 1 ←

length of the min string

(3) Binary no's starting with 00 & ending with 11

$\Rightarrow$ $00(0+1)^*11$



Length of min string $min|w| = 4$

(4) Binary string with 111 as a substring

$\Rightarrow$ $(0+1)^* 111 (0+1)^*$

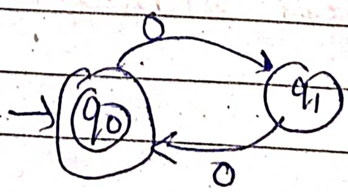
(5) $\Sigma = \{a, b\}$

String of a,b with ~~all~~ atleast two a's

$\Rightarrow$ $(a+b)^* aa (a+b)^*$

(6) $\Sigma = \{0\}$ even no. of 0's

$(00)^*$



doubt ?

Date
Page

7) Even number of a's followed by odd no' of b's

$$(aa)^* \, b \, (bb)^*$$

8) $\Sigma = \{0, 1\}$ At least two 0's any where not mandatory X001

$$(0+1)^* \, 0 \, (0+1)^* \, 0 \, (0+1)^*$$

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

At least One 0 any where

$$\therefore (0+1)^* \, 0 \, (0+1)^*$$

10) $\Sigma = \{0, 1\}$ All strings of binary no. where 1^{st} & last bit is alway 1

$$1 \, (0+1)^* \, 1$$

11) $\Sigma = \{0, 1\}$ 2^{nd} bit from left is 1 3^{rd} bit from right is 0.

single bit 0/1

$$(0+1) \, 1 \, (0+1)^* \, 0 \, (0+1) \, (0+1)$$

min $|w| = 5$

12) 1^{st} bit from left is 0 and 2^{nd} bit Right is 1

$$0 \, (0+1)^* \, 1 \, (0+1) \qquad |w|_{min} = 3$$

18|10|22

★ Operations on Languages! →

$L_1 = \{\epsilon, a, aa ---\}$

$L_2 = \{a, aa . --\}$

set — all possible string

→ Regular expression means common representation

① ① Concatenation → means with itself.

If L_1 and L_2 are two regular languages their concatenation will also be regular.

Let - $L_1, L_2 \in \Sigma^*$

$L_1 L_2 = \{ xy \mid x \text{ is in } L_1 \text{ and } y \text{ is in } L_2 \}$

Ex → $\Sigma = \{a, b\}$

$L = \{a, b\} = L^1$ → concatation with itself (original string)

L^0 = null (not taken any symbol or string)

$L^2 = L \cdot L^1 = \{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$

$L^3 = L^2 \cdot L = \{aa, ab, ba, bb\}\{a, b\}$

$= \{aaa, aab, aba, abb, baa, bbb\}$

bba, bbb }

$\boxed{L^i = L \cdot L^{i-1} \text{ For all } i \geq 1}$

Kleene Closure :-

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

$L\{a\}$

symbol = a

= {null, a, aa, aaa ---} - with null value

Positive closure :-

(a^+) $L^+ = \{L^1 \cup L^2 \cup$ ---}

↓ = {a, aa, aaa ---}

without null.

(The set Σ^+ is the infinite set of all possible strings of all possible length over Σ excluding λ.

Representation - $\Sigma^+ = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup$ ---

Ex - If Σ = {a, b}

Σ^+ = { a, b, aa, ab, ba, bb, - - - }

Closure properties of regular set :-

① Union :-

The union of the regular set is also regular

$RE_1 = a(aa)^*$ L_1 = { a, aaa, aaaaa - - }

$RE_2 = (aa)^*$ L_2 = { ϕ, aa, aaaa, aaaaaa, aaaaaaaa - - }

$L_1 \cup L_2$ = { ϕ, a, aa, aaa, aaaa, aaaaa, aaaaaa - - }

RE_3 = $(a)^*$

Date
Page

② **Intersection :-** The Intersection of regular set is also regular

$RE_1 = (aa)^*$ $L_1 = \{null, aa, aaaa, ---\}$
$RE_2 = aa(aa)^*$ $L_2 = \{aa, aaaa, ---\}$

$L_1 \cap L_2 = \{aa, aaaa, aaaaaa, aaaaaaaa ---\}$

$RE_3 = aa(aa)^*$

③ **Closure :-**
Closure of regular set is also regular.

$RE = a^*$ $L = \{null, a, aa, aaa ---\}$

$(a^*)^* = \{null, a, aa, aaa ---\}^*$
$= \{null, a, aa, aaa, aaaa ---\}$
$\downarrow$
a^*

④ **Difference :-** A-B
Difference of a regular set is also regular.

$RE_1 = a(a^*) = L_1 \{a, aa, aaa ---\}$ $= (a, b, c) - (b, c)$
$RE_2 = (aa)^*$ $L_2 = \{null, aa, aaaa ---\}$ $= (a)$

$L_1 - L_2 = \{a, aaa, aaaaa, aaaaaaa, ---\}$
$RE_2 = a(aa)^*$

Date
Page

(5) Reverse:-

Reverse of regular expression is also regular.

$L = \{01, 10, 11, 00\}$

$RE = 01 + 10 + 11 + 00$

$L^R = \{10, 01, 11, 00\}$

$RE = 10 + 01 + 11 + 00$

(6) Complement:-

Complement of regular expression is also regular.

$RE(L) = a(aa)^*$

$L = \{a, aaa, aaaaa, 7a's, 9a's \ldots\}$

Complement of $L = \{null, aa, aaaa, aaaaaa, \ldots 8's \ldots\}$

$RE = (aa)^*$

(7) Concatenation is also regular:-

$a(a)^*$

~~a(a, aaa,~~

a(null, a, aa, aa . .

(a, aa, aaa, aa . .)

$RE_1 = (0+1)^*0$

$RE_2 = 01(0+1)^*$

$L_1 L_2 = (0+1)^* 001 (0+1)^*$

$\rightarrow aa(aa)^*$

aa(null, aa, aaaa . . . }

aa, aaaa, aaaaaa.

$\rightarrow a\{aa\}^*$

{null, a, aaa}, aaaaa }

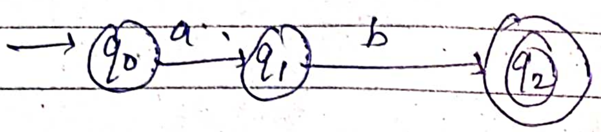
19/10/22

Grammar:

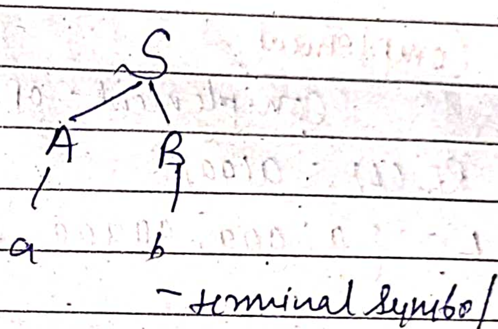
Rules used to generate strings of language L.

Example → $L = \{a, b\}$

RE = a b



derivation tree / parse tree



$S \to AB$
$A \to a$
$B \to b$

} example of grammar.

Defination :- It is represented with 4 tuple,

(S, V_n, Σ, P)

- V_n: All capital letters
- Σ: Small letters
- P: production rule

S = starting symbol

V_n = set of variables (capital L) (S, A, B)

Σ = set of input signal or terminals (small L) (a, b)

P = production rule (no. of arrow that much production rule)

$A \to \alpha$ where A is a variable $\alpha \in (V_n \cup \Sigma)^*$

Q→ Derive the string from the grammer:

$S \rightarrow SS \mid aAS \mid a$.

$A \rightarrow SbA \mid ba$

"aabaa"

$\Rightarrow S \rightarrow SS$

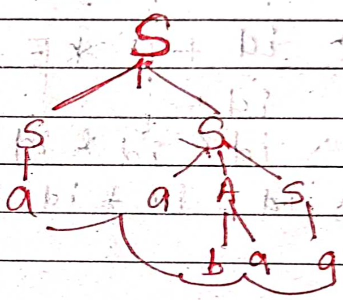
$\rightarrow aS \quad (S \rightarrow a)$

$\rightarrow aaAS \quad (S \rightarrow aAS)$

$\rightarrow aabaS \quad (A \rightarrow \cancel{SbA})$

$\rightarrow aabaa \quad (S \rightarrow a)$

Derivation tree



Q→ $S \rightarrow aCb$

$C \rightarrow aCb$

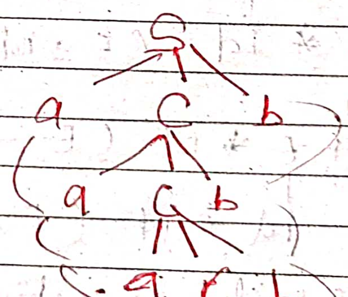
$C \rightarrow c$ "aaacbbb"

$S \rightarrow aCb$

$\rightarrow aaCbb \quad (C \rightarrow aCb)$

$\rightarrow aaaCbbb \quad (C \rightarrow aCb)$

$\rightarrow aaacbbb \quad (C \rightarrow c)$



left to right → left most derivation

Q → E → E+E
E → E*E
E → id

"id + id * id"

→ E → E + E

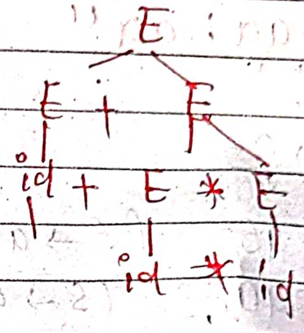
→ id + E (E → id)

→ id + E * E (E → E*E)

→ id +

→ id + id * id (E → id)

→ id + id * id (E → id)



→ Now Right to left! –

For same question

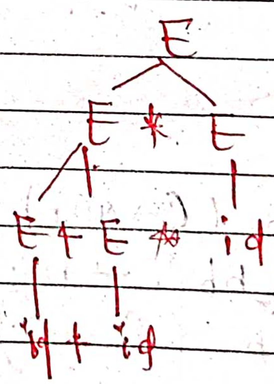
E → E * E

→ E * id (E → id)

→ E + E * id (E → E+E)

→ id * id (E → id)

→ id + id * id (E → id)



Right to left

Q → $S \to SbS \mid a$

(abababa)

→ $S \to SbS$

→ Sba $(S \to a)$

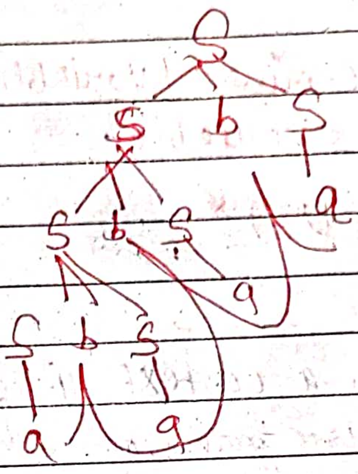
→ $SbSba$ $(S \to SbS)$

→ $Sbaba$ $(S \to a)$

→ $SbSbaba$ $(S \to SbS)$

→ $Sbababa$ $(S \to a)$

→ $abababa$ $(S \to a)$



= Left to right

→ $S \to SbS$

→ abS → $(S \to a)$

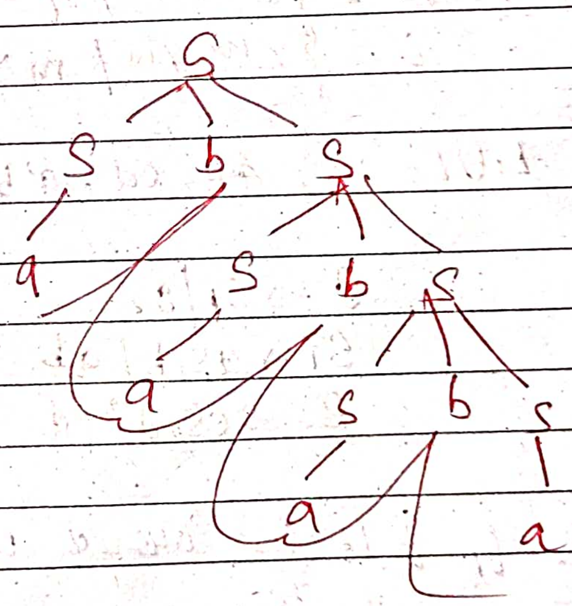
→ $abSbS$ → $(S \to SbS)$

→ $ababS$ → $(S \to a)$

→ $ababSbS$ → $(S \to SbS)$

→ $abababS$ $(S \to a)$

→ $abababa$ $(S \to a)$



⇒ G is ambigeous (Because tree is different)

01/11/22

~~Definition of Grammar~~ :- A Grammar in which each production is of the form

Properties of context free Grammar:-

$A \to \alpha$

$\alpha \to$ Terminal / Variable

A is a variable.

α is $(V_n \cup \Sigma)^*$

1) If L_1 is a context free grammar and L_2 is also a context free grammar then

$L_1 \cup L_2$ is also a context free grammar (closed).

Ex → $L_1 = \{a^n b^n \mid n \geq 1\}$ → $S_1 \to ab\ a^2b^2\ a^3b^3$ ------

$L_2 = \{c^m d^m \mid m \geq 1\}$ $S_2 \to cd\ c^2d^2\ c^3d^3$ ------

$L_1 \cup L_2 = \{ab, cd, a^2b^2, c^2d^2 \ldots\}$

$S \to S_1 / S_2$

$S_1 \to aS_1b / ab$

$S_2 \to cS_2d / cd$

2) L_1 & L_2 is also closed under concatenation

$L_1L_2 = \{abcd, abc^2d^2, abc^3d^3 \ldots$
$a^2b^2cd, a^3b^3cd \ldots\}$

$a^n b^n c^m d^m$ pattern

$S \to S_1 S_2$

$S_1 \to aS_1b / ab$

$S_2 \to cS_2d / cd$

3) Kleene closure →

$S_1 \to aSb$
$S_1 \to null$

$L = \{ a^n b^n \mid n \geq 0 \}$ null included

$L = \{ null, ab, a^2b^2, a^3b^3 \ldots \}$

Ex) $L^* = \{ab, aba^2b^2, a^4b^4, abababab \ldots\}$

Grammar → $S_1 \to S_1 S_1$

4) Intersection of $L_1 \cap L_2$ may or may not be context free Grammar

let $L_1 = \{ a^n b^n c^m \mid n, m > 0 \}$
$L_2 = \{ a^m b^n c^n \} \; n, m > 0 \}$

L_1	L_2
abc	abc
ab^2c^2	abc^2
ab^3c^3	abc^3
⋮	⋮
a^2bc	a^2b^2c
$a^2b^2c^2$	$a^2b^2c^2$
$a^2b^3c^3$	$a^2b^2c^3$
⋮	⋮
a^3bc	a^3b^3c
$a^3b^2c^2$	$a^3b^3c^2$
$a^3b^3c^3$	$a^3b^3c^3$

$L_1 \cap L_2 = \{abc, a^2b^2c^2, a^3b^3c^3 \ldots\}$
$= \{a^n b^n c^n \mid n > 0\}$

It does not belong to a C.F.G

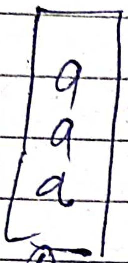
02/11/22

[PUSH Down Automation;]—

$a^n b^n$ → should be same . take (a) same as b) .

↑ regular set fails. here and this is solved by using push down automation .

Stack will be used for this



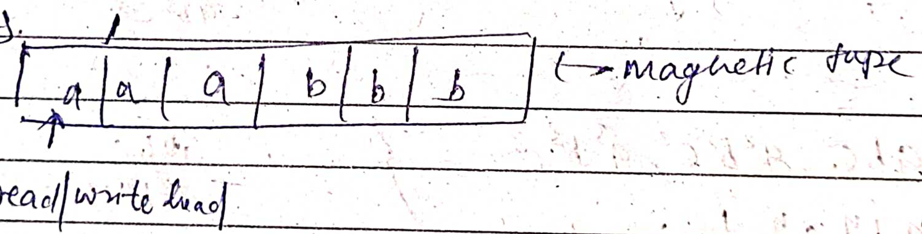
One by one will pop from stack and take the same no. of b

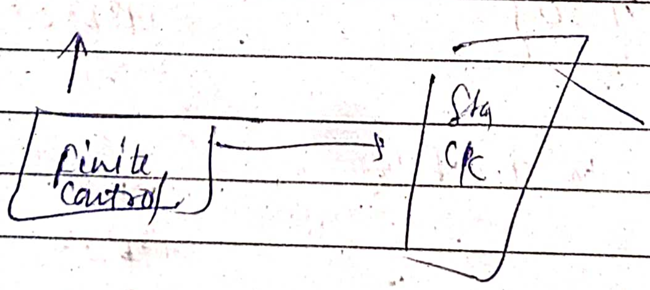
↑ for memorising (a)

$a^n b^n$

any one from them can be pushed into stack.

model →





✪ finite control

(, ,) = (,),

(Current State , Current input , top of stack) = (next State , top of stack)

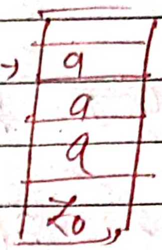
a^3b^3 ⇓

Initially q_0

1^{st} a $(q_0, a, Z_0) = (q_0, a)$ (push)

initially null

since same process repeated



for remaining a

2. $(q_0, a, a) = (q_0, a)$ push

3. $(q_0, a, a) = (q_0, aa)$

Now ~~from~~ for b reading b POP (a)

$(q_0, b, a) = (q_1, \text{null})$ λ

(for POP)

$(q_1, b, a) \rightarrow (q_1, \lambda)$

(for all remaing b

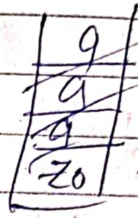
$(q_1, \lambda, Z_0) \rightarrow (q_F)$ - final state reacher

Design of PDA for $L = a^n b^n, n \geq 0$

(A) $a^3 b^3$ pattern

Rule 1 $\delta(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$

Rule 2. $\delta(q_0, a, a) \rightarrow (q_0, aa)$



Rule 2; $\delta(q_0, a, a) \rightarrow (q_0, aa)$

Rule 3; $\delta(q_0, b, a) \rightarrow (q_1, \lambda)$

Rule 4 $\delta(q_1, b, a) \rightarrow (q_1, \lambda)$:

rule 4 = $\delta(q_1, b, a) \rightarrow (q_1, \lambda)$

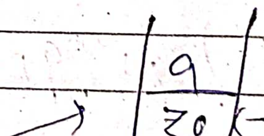
rule 5 $\delta(q_1, \lambda, Z_0) \Rightarrow q_f$

Push Down Automation for $a^n b^{2n}$

$a^n b^{2n} \rightarrow aabbbb \; (n=2)$

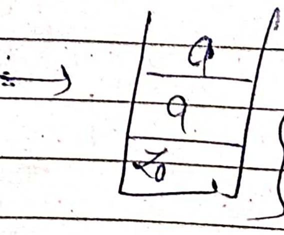
single a - two b's

now for first



$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$



(a) push

now if we find b the state will change but will skip that

$\delta(q_0, b, a) = (q_1, a)$

bb bb (pop a, pop a)

$\delta(q_1, b, a) = (q_2, \lambda)$

$\delta(q_2, b, a) = (q_1, a)$

$\delta(q_2, \lambda, Z_0) = (q_f, Z_0)$

?

03/11/22 Defination of PDA

$\delta \rightarrow$ set of finite non-empty stack symbol

$\delta(Q, \Sigma, \Gamma) \rightarrow (Q, \Gamma^*)$ — mean many symbols appears

Transition diagram of PDA

$a^n b^n \mid n \geq 0$

$\delta(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$

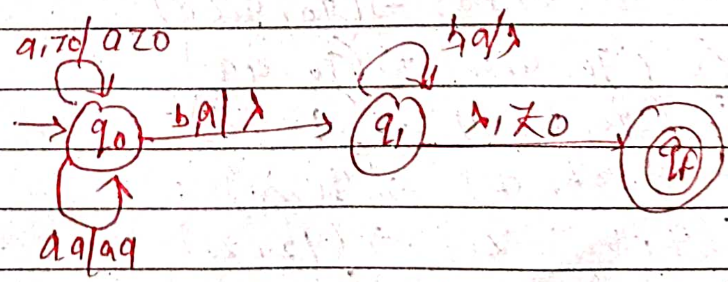
$\delta(q_0, a, a) \rightarrow (q_0, aa)$

$\delta(q_0, b, a) \rightarrow (q_1, \lambda)$

$\delta(q_1, b, a) \rightarrow (q_1, \lambda)$

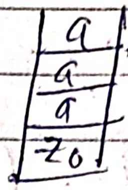
$\delta(q_1, \lambda, Z_0) \rightarrow q_f$

now drawing automatia



Transition diagram for the pattern:

$a^n c b^n \quad a^3 c b^3$



$\delta(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$

$\delta(q_0, a, a) \rightarrow (q_0, aa)$

$\delta(q_0, c, a) \rightarrow (q_1, aa)$

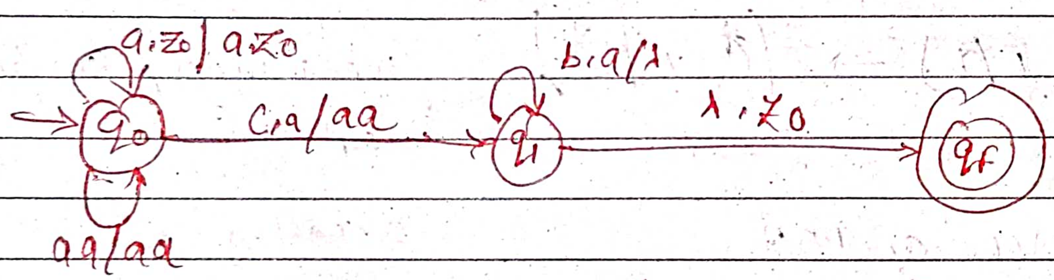
C is only reading and moving forward

$\delta(q_1, b, a) \rightarrow (q_1, \lambda)$

It is not associated with POP or PUSH so it can be used in b.

$\delta(q_1, b, a) \rightarrow (q_1, \lambda)$

$\delta(q_1, \lambda, Z_0) \rightarrow q_f$



Pattern $a^n b^n c^m$

$\delta(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$

$\delta(q_0, a, a) \rightarrow (q_0, aa)$

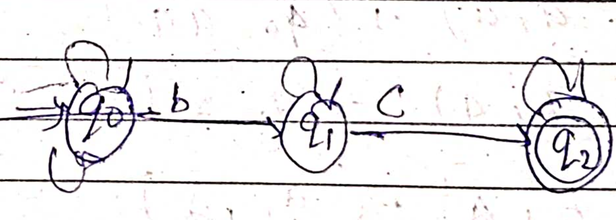
$\delta(q_0, b, a) \rightarrow (q_1, \lambda)$

$\delta(q_1, b, a) \rightarrow (q_1, \lambda)$

$\delta(q_1, c, Z_0) \rightarrow (q_2, Z_0)$

$\delta(q_2, c, Z_0) \rightarrow (q_2, Z_0)$

$\delta(q_2, \lambda, Z_0) \rightarrow q_f$



Ratio Pattern:

$a^n b^{3n}$ → Here dependency is where ever the last symbol is dependent on previous one will write λ/Z_0 last

aaa bbb bbb bbb

a → 3 time

Starting for a and rest of a

$\delta(q_0, a, Z_0) \to (q_0, aZ_0)$

$\delta(q_0, a, a) \to (q_0, aa)$

$\delta(q_0, b, a) \to (q_1, aa)$

$\delta(q_1, b, a) \to (q_2, aa)$

$\delta(q_2, b, a) \to (q_3, aa)$

$\delta(q_3, b, a) \to (q_4, aa)$

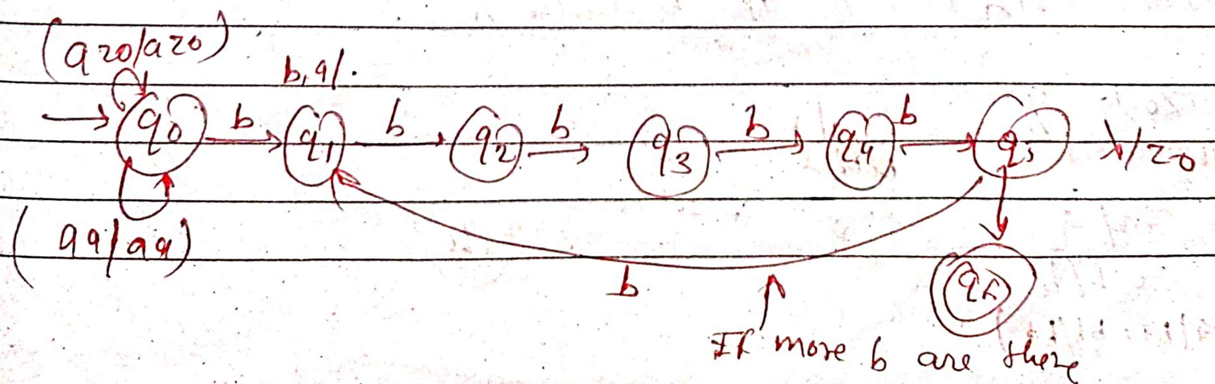
$\delta(q_4, b, a) \to (q_5, \lambda)$

$\delta(q_3, \lambda, Z_0) \to q_f$

aa → bbbbb bbbbb

$q_0 \to$ $q_5 \to$

$\delta(q_5, b, a) \to (q_1, a)$



04/4/22

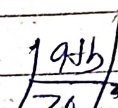
Design a PDA for :-

$L = \{ wcw^R \mid w \in \{a,b\}$, c is a terminal $\}$

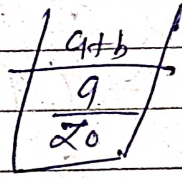
The above can be written as

a a c a a
b b c b b
a b c b a
b a c a b

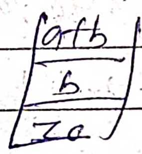
$R_1 \quad \delta(q_0, a, z_0) = (q_0, az_0)$
$R_2 \quad \delta(q_0, b, z_0) = (q_0, bz_0)$



$R_3 \quad \delta(q_0, a, a) = (q_0, aa)$
$R_4 \quad \delta(q_0, b, a) = (q_0, ba)$



$R_5 \quad \delta(q_0, a, b) = (q_0, ab)$
$R_6 \quad \delta(q_0, b, b) = (q_0, bb)$



for c :-
$\delta(q_0, c, a) = (q_1, a)$
$\delta(q_0, c, b) = (q_1, b)$

$\delta(q_1, a, a) = (q_1, \lambda)$ — If a with a
$\delta(q_1, b, b) = (q_1, \lambda)$ — If b with b

$\delta(q_1, \lambda, z_0) = q_f$