Theory Of Automata

I – Finite automata ← DFA, NFA
II – Regular grammars
III – context free grammars
IV – push down automata
V – Turing machine

problem:- Whether the string is in given Language

TOA:- Representation of abstract machine

- **Alphabets (Σ):** Set of Symbols ex:- for binary System $\Sigma = \{0, 1\}$
- **Strings:-** Combination of Symbols from these alphabets
- **Language:-**

alphabets (Σ)

- binary $\Sigma = \{0, 1\}$
- Octal $\Sigma = \{0, 1 - - , 7\}$
- Lower Case $\Sigma = \{a, - - , z\}$
- Upper case $\Sigma = \{A - - Z\}$

String:- Combination of Symbols from the alphabets (Σ^L)

(L — length of String, Σ — alphabet Set)

$\Sigma^1 = \{0, 1\}$ = 2 String

$\Sigma^2 = \{00, 01, 10, 11\}$

⇒ 4 String with length = 2

Note :- 1) '*' represents Kleene closure

* $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots \cup \Sigma^n$

2) '+' represents positive closure

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

3) Kleene closure includes null string (Σ^0) whereas positive closure contains atleast one string with non zero length.

* $\Sigma^* = \Sigma^0 \cup \Sigma^+$

Language :- $L \subseteq \Sigma^*$: language is a subset of set of all string over 'Σ'

Example :- 1) even number over $\{0, 1\}$

$\{0, 10, 100, 110, \ldots\}$

2) Odd number over $\Sigma = \{0, 1\}$

$\{1, 11, 101, \ldots\}$

3) String with substring '010' over $\Sigma = \{0, 1\}$

$\{010, 0100, 00100 \ldots\}$

Alphabet :- is a finite, non empty set of symbol
- We use the symbol 'Σ' for an alphabet.

Ex :- Alphabet of binary Σ = {0, 1}

String :- (or) Sometimes called as blocks.
- Is a finite set of symbol chosen from the same set of alphabet.

- String over Σ = {0, 1} are {00, 01, 011 - - - }

Empty String :- Is a string with zero occurenc of symbol.
- This string is represent by 'ϵ'.
- 'ϵ' is a string that may be chosen fro any alphabet.

Length of a string :- The length is the no. of symbol (or) position in the symbol.
- Represented as |w|

Language with Σ = {a, b}

i) String ending with 'bab'

L = { bbab, abbabbab, - - - }

regular so...

* Powers of an alphabet:- If 'Σ' is an alphabet the set of all string of a certain length from that alphabet by using exponential notation.

Σ^k - set of strings of length k, each of symbol in Σ

ε - is the only string of length '0' (Epsilon)

→ The set of all string over an alphabet 'Σ' is conventionally denoted by Σ^*

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \cdots$

→ The set of non empty strings from alphabet 'Σ' by represented by Σ^+

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cdots$

→ $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$ (Kleene closure)

Concatenation

$\epsilon w = w\epsilon = w$

ε is the identity for concatenation.

* Language:- a set of string of which are chosen from some Σ^*, where 'Σ' is a particular alphabet is called a language

→ If Σ is an alphabet and if $L \subseteq \Sigma^*$ then $\Sigma - \Sigma^\circ$ 'L' is a language over 'Σ'. ϕ

→ The language over 'Σ' need not include strings with all the given symbols of 'Σ'

→ L is language over 'Σ' then 'L' is a language over any alphabet i;e superset of Σ.

Example:

1) Prime no. over $\Sigma = \{0, 1\}$

$L = \{10, 11, 101 - - -\}$

→ 'Σ^*' is a language for any alphabet Σ

→ 'ϕ' is the empty language (it doesn't ever include (Σ° (or) ε) empty String)

→ 'ϕ' is the empty language which exist in every language.

→ The language which contains empty string is also a language over any alphabet

i;e $\phi \neq L \in \{\}$

Problem in Automata Theory

* In AT the problem is the question of deciding whether the given string is the member of some particular language.

Regular Sets

* → If Σ is an alphabet and 'L' is a language over Σ then the problem is given a string 'w' in Σ^* decide whether or not 'w' is in 'L'.

FINITE AUTOMATA

Regular languages are the languages accepted by Finite automata.

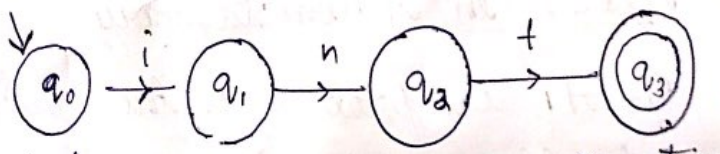
- They are represented by FA (or) Regular expression

Deterministic FA :

- * It has 5 component :-

1. Start State
2. Set of States – Q
3. Accepting States (Final State) – F
4. Set of alphabets – Σ
5. Transition function – δ

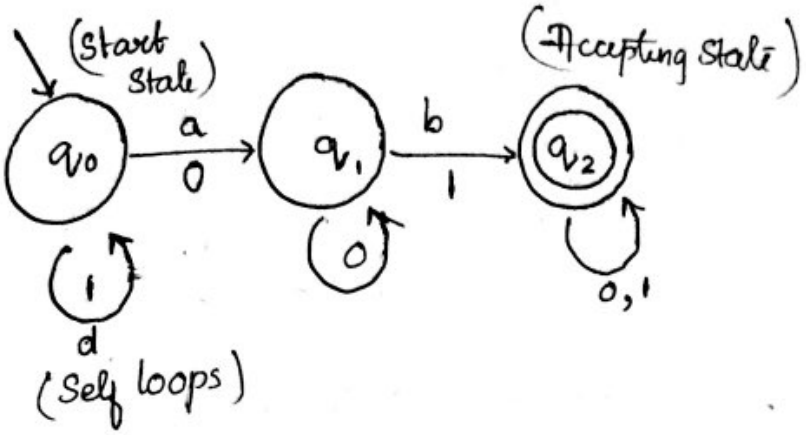
Ex :-



* Start State — Acccepting State

1) $Q = \{ q_0, q_1, q_2, q_3 \}$

Finite Automata



→ Start State = q_0, ~~Finite St~~ Accepting State = $q_2 = F$,

→ Set of States = $Q = \{q_0, q_1, q_2\}$

→ $\Sigma = \{0, 1\}$

→ $\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$, $\delta(q_2, 1) = q_2$

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

note: $F \subseteq Q$

→ If the state on a particular external input change to

Determinester: refers to the fact that on each input there is one and only one state to which automaton can transition from its current state.

Definition of DFA

1. A finite set of states denoted by Q.
2. A finite set of i/p symbol denoted by Σ.
3. A transition function that takes arguments as a state & an input symbol and returns a state and denoted by 'δ'.
   - δ represented by arcs between states & label on the arc.
   - If q is a state and a is an i/p then $\delta(q, a)$ is that state p such that there is an arc labeled a from q to p.
4. Start state is one of the states in Q
5. A set of final (or) accepting states 'F'.
   - $F \subseteq Q$.

DFA in five-tuple Notation

$$A = (Q, \Sigma, \delta, q_0, F)$$

↑	↑	↑	↑	↑	↑
DFA	Set of State	i/p Symbol	trans fun	Start State	Final State

Note: Self loop is a Kleene closure '*'

* DFA to accept substring "01" over $\Sigma = \{0,1\}$

$W = \{01, 001, 0001, 101, 1101 - - -\}$

Ex: i) 010110

$\delta(q_0,0) = q_1$	$\delta(q_2,1) = q_2$
$\delta(q_1,1) = q_2$	$\delta(q_2,1) = q_2$
$\delta(q_2,0) = q_2$	$\delta(q_2,0) = q_2$

accepted

ii) 1000

$\delta(q_0,1) = q_0$

$\delta(q_0,0) = q_1$

$\delta(q_1,0) = q_1$

not accepted

iii) 10001

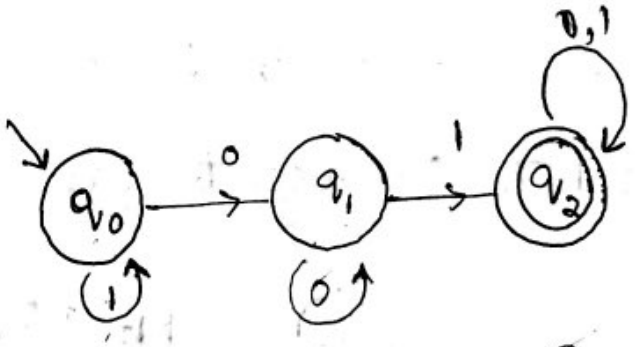
$\delta(q_0,1) = q_0$

$\delta(q_0,0) = q_1$

$\delta(q_1,0) = q_1$

$\delta(q_1,1) = q_2$

accepted



11100001 10

Transition Diagram : for a DFA :

$A = (Q, \Sigma, \delta, q_0, F)$ is a graph defined as follows:-

i) For each state in 'Q' there is a node

ii) For each state 'q' in set of states 'Q' & each i/p symbol a in Σ let $\delta(q,a) = p$ then the transition diagram has an arc labeled a.

- If there are several i/p symbols that cause transition from q to p then the transition diagram can have one arc labeled by the list of these symbols.

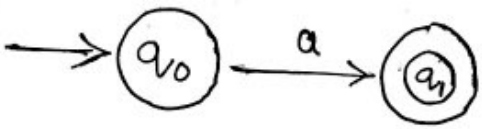
3) There is an arrow into the start state q_0 labeled as start. This arrow doesn't originate at any node.

4) Nodes corresponding to accepting states (F) are marked by a double circle.
- States not in F have a single circle.

Q) DFA to accept an empty language.

No Final State in empty language

Q) DFA to accept exactly one 'a' over the $\Sigma = \{a\}$

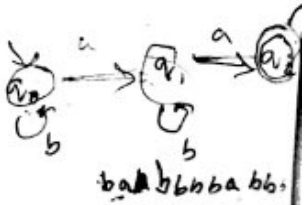


input: aa

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = ?$ (undefined)

(Not accepted)



Start State = q_0 3> Accepting State = q_3

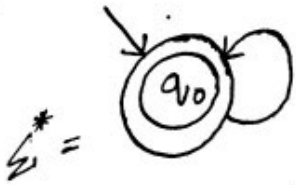
for

DFA accepting ~~words~~ Σ^* ~~above~~

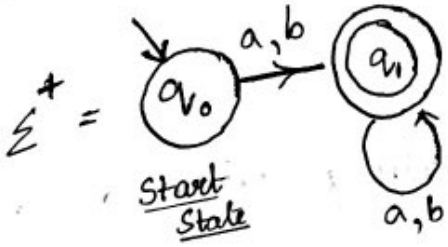
Q) Construct DFA for the language Σ^* over the $\Sigma = \{a, b\}$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 - - - -$$

Sol^n

$\Sigma^* =$ 

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 - - -$$

$\Sigma^+ =$ 

Language accepted for $\Sigma = \{a, b\}$ construct DFA

for language $= \{ (a,b)^n / n \geq 0 \}$

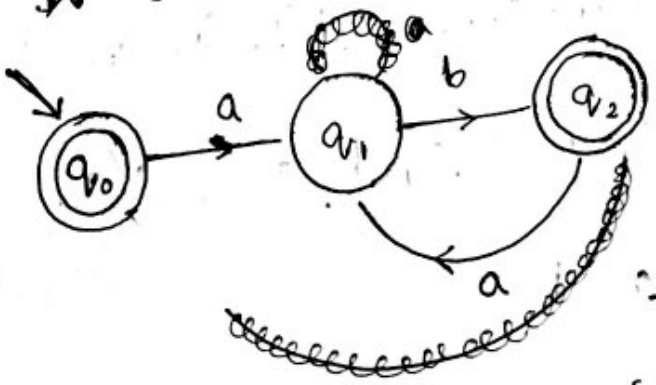
$= \{ \epsilon (ab)^0, (ab)^1, (ab)^2, (ab)^3 - - - \}$

abab

aaabbb

aaabbb

Sol^n



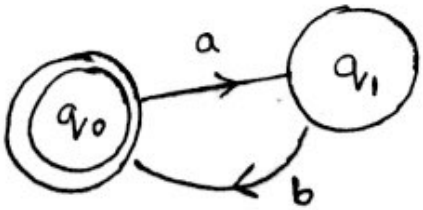
Transition

$\delta(q_0, a) = q_1$

$\delta(q_1, b) = q_0$

$\delta(q_0, b) =$ not defined

$\delta(q_1, a) =$ not defined

DFA $\Rightarrow$ 

* Start State = q_0 ; Final State = q_0 , $Q = \{q_0, q_1\}$; Σ

$$A = \left(\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_0\} \right)$$

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

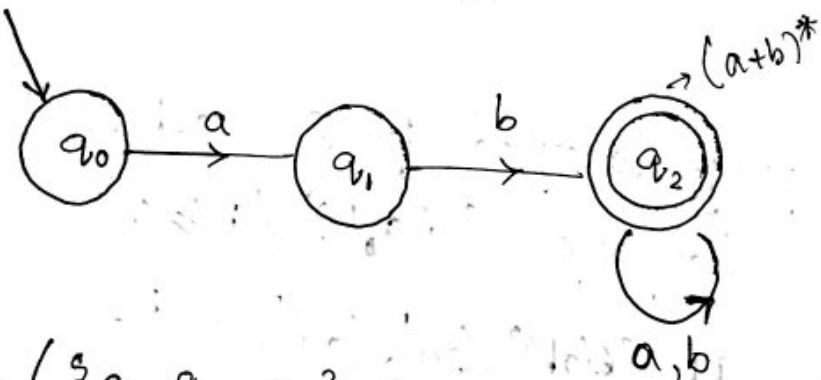
* Table representation

	δ	a	b (input)
→ *	q_0	q_1	-
	q_1	-	q_0

(State)

Q) Draw the DFA for starting with 'ab' over.

A



$$A = \left(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\} \right)$$

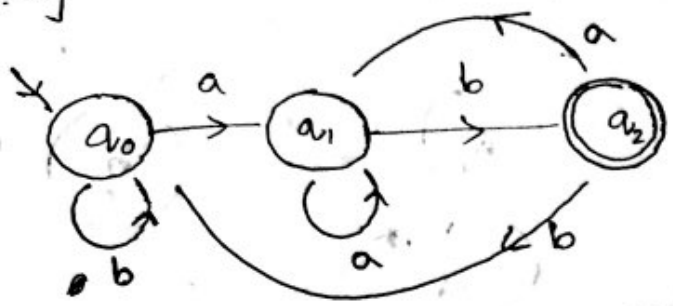
$$\delta(q_0, a) = q_1 \ ; \ \delta(q_1, b) = q_2 \ , \delta(q_2, a) = q_2$$

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

δ	a	b
→ q_0	q_1	-
q_1	-	q_2
* q_2	q_2	q_2

Q) Ending with 'ab' over $\Sigma = \{a, b\}$

aadbaab



$A = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$+ q_2$	q_1	q_0

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

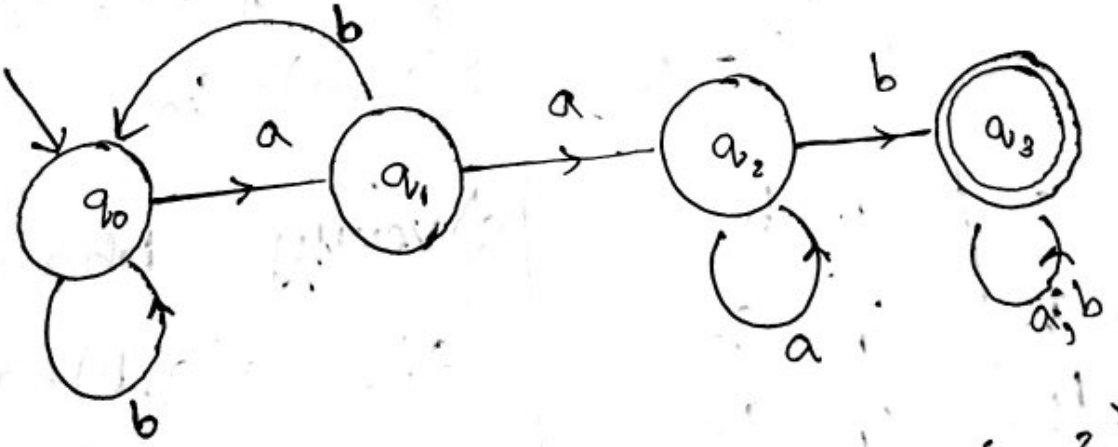
$\delta(q_0, b) = q_0$; $\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_1$; $\delta(q_2, b) = q_0$

aaaaaaa

babaabaaa

Q) Substring 'aab'



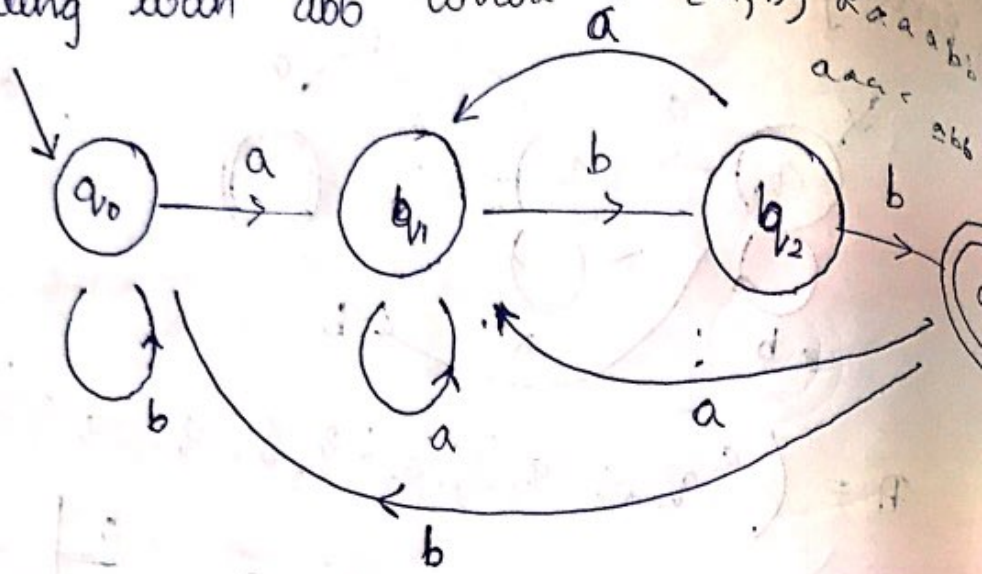
$A = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\})$

$\delta(q_0, a) = q_1$; $\delta(q_0, b) = q_0$; $\delta(q_1, a) = q_2$

$\delta(q_1, b) = q_0$; $\delta(q_2, a) = q_2$; $\delta(q_2, b) = q_3$

$\delta_3(q_3, b) = q_3$; $\delta_3(q_3, b) = q_3$

Q) Ending with 'abb' where $\Sigma = \{a, b\}$



$$A = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\})$$

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

$\delta(q_1, b) = q_2$; $\delta(q_2, a) = q_1$; $\delta(q_2, b) = q_3$

$\delta(q_3, a) = q_1$; $\delta(q_3, b) = q_0$

δ	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_3	q_1
* q_3	q_1	q_0

Q) verify bbaababab

Sol: q_0

$\delta(q_0, b) = q_0$

$\delta(q_0, b) = q_0$

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

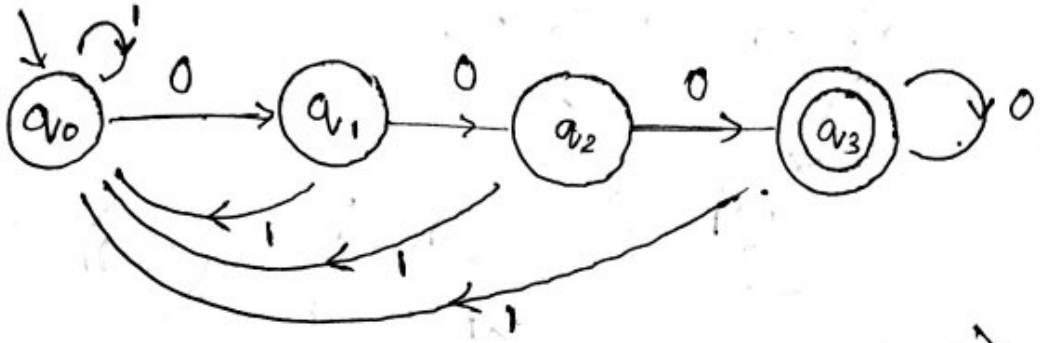
$\delta(q_2, a) = q_1$

$\delta(q_1, b) = q_2$

(not accepted)

Q) Substring '000'

A)



$A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_0$

$\delta(q_2, 0) = q_3$

$\delta(q_2, 1) = q_0$

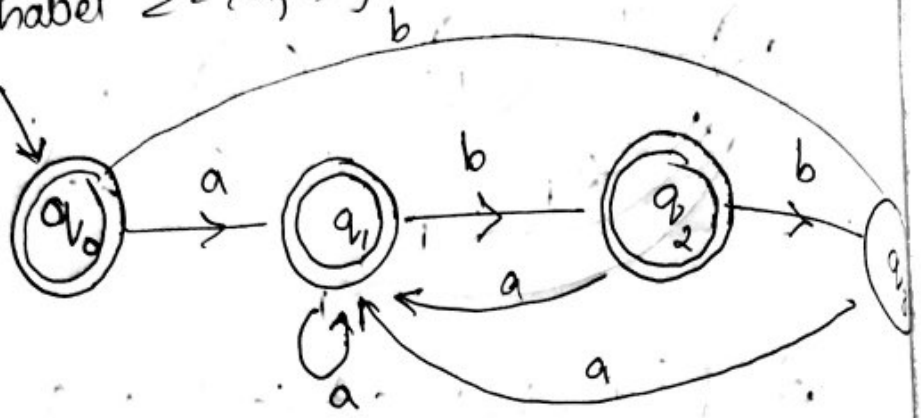
$\delta(q_3, 0) = q_3$

$\delta(q_3, 1) = q_0$

δ	0	1
→ q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
* q_3	q_3	q_0

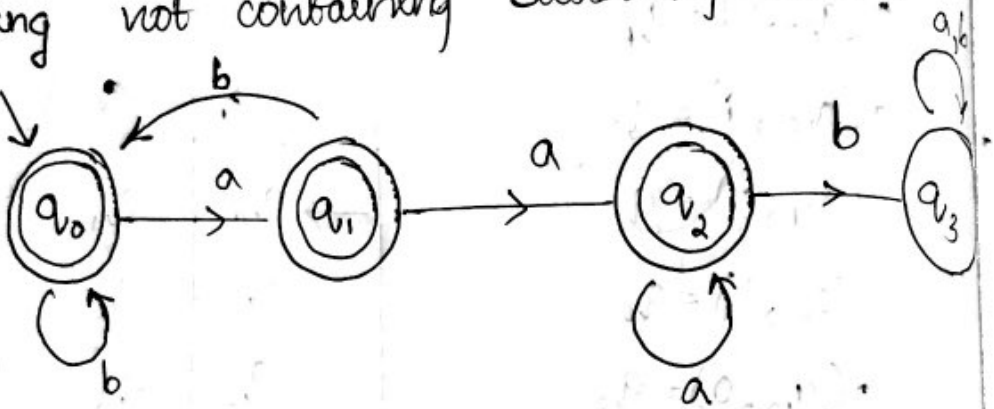
Q) Design DFA not ending with abb over alphabet $\Sigma = \{a, b\}$

Sol:



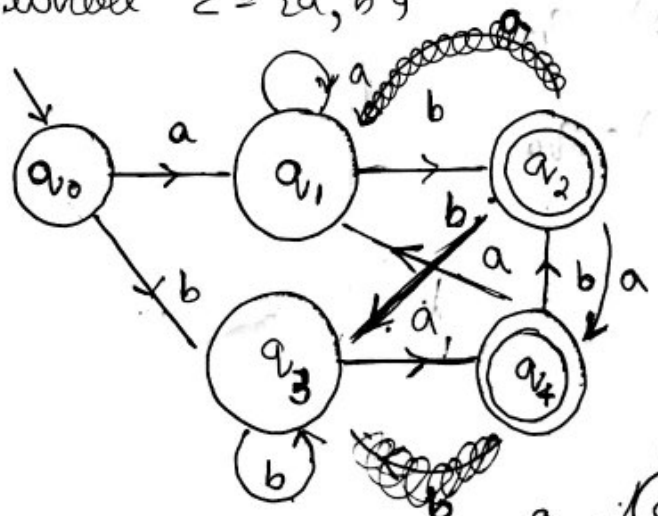
Q) String not containing substring 'aab'.

Sol:



Q) DFA for strings ending with ab (or) ba where $\Sigma = \{a, b\}$

Sol:

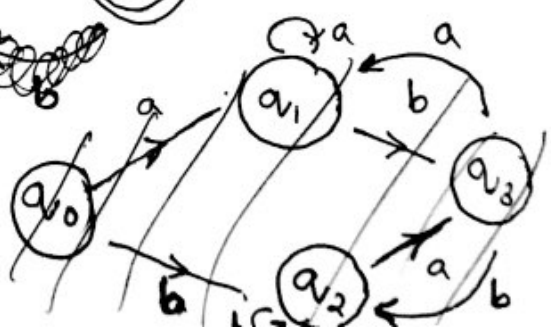


aaaba

bbbab

aaabg

(or)



$A = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_0, \{q_2, q$

$\delta(q_0, a) = q_1$

$\delta(q_0, b) = q_3$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_4$

$\delta(q_3, a) = q_4$

$\delta(q_3, b) = q_3$

$\delta(q_4, a) = q_1$

$\delta(q_4, b) = q_2$

$\delta(q_2, b) = q_3$

	δ	a	b
→	q_0	q_1	q_3
	q_1	q_1	q_2
*	q_2	q_4	q_3
	q_3	q_4	q_3
*	q_4	q_1	q_2

Verify:- a a b a b b

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_1$

$\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_4$

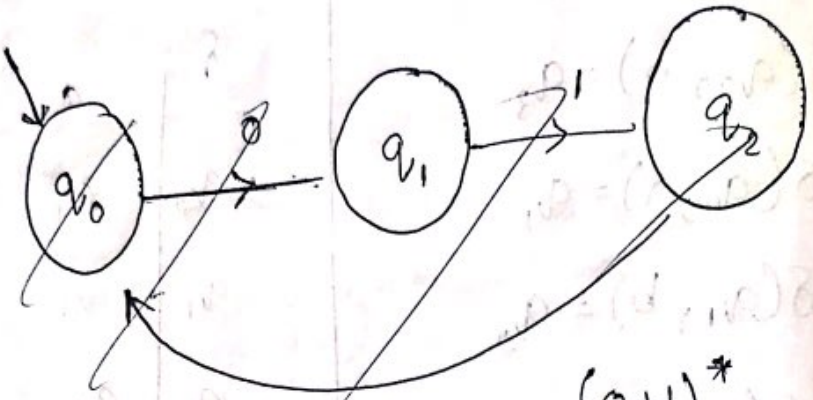
$\delta(q_4, b) = q_2$

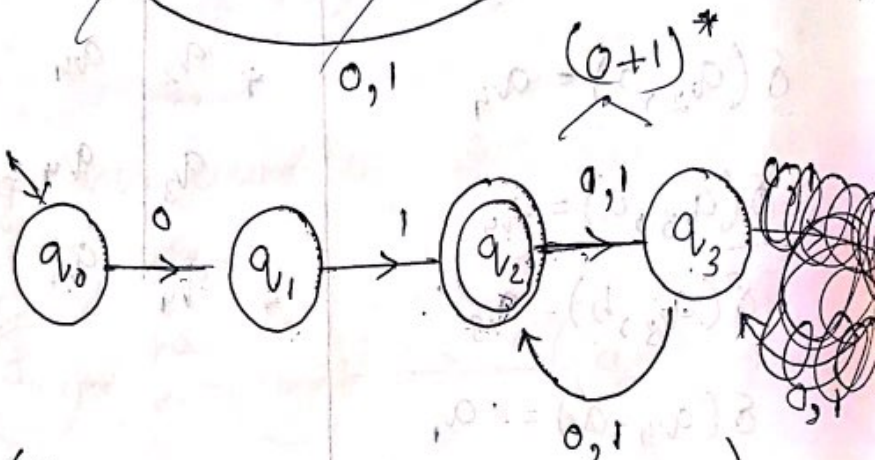
$\delta(q_2, b) = q_3$

(not accepted)

Q) $L = \{ w / w$ is of even length and begi
with 01 $\}$ over $\Sigma = \{0, 1\}$

Sol.





$A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_2\})$

$\delta(q_0, 0) = q_1$

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_3$

$\delta(q_2, 1) = q_3$

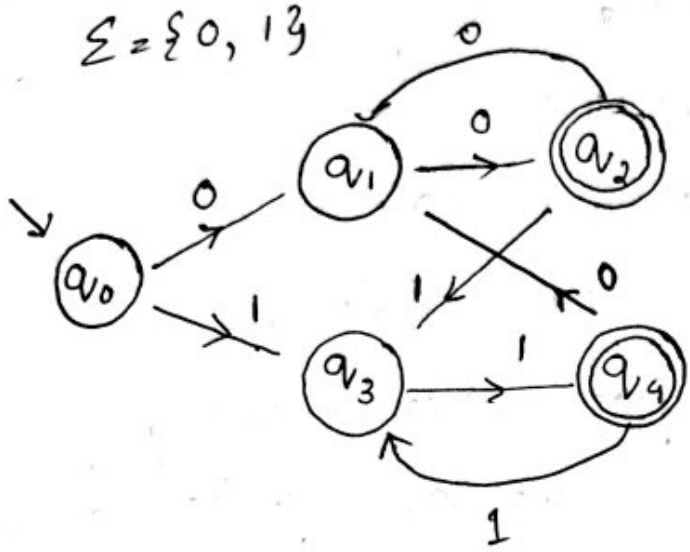
$\delta(q_3, 0) = q_2$

$\delta(q_3, 1) = q_2$

δ	0	1
q_0	q_1	-
q_1	-	q_2
q_2	q_3	q_3
q_3	q_2	q_2

8) even no. of 0's & even no. of 1's where

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



~~0011~~ 001100

100101

1000

$q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \rightarrow$

Extended Transition Function describes what happens when we start in any state and follow any sequence of i/p.

→ If 'δ' is one transition function then the extended transition function constructed from δ will be called as $\hat{\delta}$.

→ The extended transition function is a function that takes a state 'q' and a word 'w' and returns a state p. i;e The state the automaton reaches when starting state 'q' & the processing the sequence of i/p 'w'

→ We define $\hat{\delta}$ by induction on the length of i/p string as follows.

Basis: $\hat{\delta}(q_0, \epsilon) = q_0$

i;e if you are in state 'q' & read no i/p then we are still in state 'q'.

Induction: Suppose 'w' is a string of the form xa, i;e a is the last symbol of w, and x is the string consisting of all but the last symbol..

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$$

Q) Verify whether string ends with 'ab' (or) 'ba'.

Sol^n

String: aaabab

Basis: $\hat{\delta}(q_0, \epsilon) = q_0$

Induction: i) $\hat{\delta}(q_0, a) = \hat{\delta}(q_0, \epsilon a) = \delta(\hat{\delta}(q_0, \epsilon), a)$

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

ii) $\hat{\delta}(q_0, aa) = \delta(\hat{\delta}(q_0, a), a) = \delta(q_1, a) = q_1$

iii) $\hat{\delta}(q_0, aaa) = \delta(\hat{\delta}(q_0, aa), a) = \delta(q_1, a) = q_1$

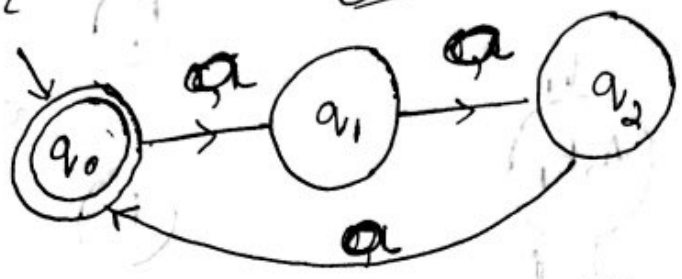
iv) $\hat{\delta}(q_0, aaab) = \delta(\hat{\delta}(q_0, aaa), b) = \delta(q_1, b) = q_2$

v) $\hat{\delta}(q_0, aaaba) = \delta(\hat{\delta}(q_0, aaab), a) = \delta(q_2, a) = q_3$

vi) $\hat{\delta}(q_0, aaabab) = \delta(\hat{\delta}(q_0, aaaba), b) = \delta(q_3, b) = q_2$

Q) Obtain DFA to accept the language

$L = \{ w : |w| \bmod 3 = 0 \}$ (i.e length = 3)



$A = (\{q_0, q_1, q_2\}, \{a\}, \delta, q_0, \{q_0\})$

→ find $\hat{\delta}(q_0, aaaaa) = ?$

Basis:- $\hat{\delta}(q_0, \epsilon) = q_0$

1> $\hat{\delta}(q_0, a) = \delta(\hat{\delta}(q_0, \epsilon), a) = \delta(q_0, a) = q_1$

2> $\hat{\delta}(q_0, aa) = \delta(\hat{\delta}(q_0, a), a) = \delta(q_1, a) = q_2$

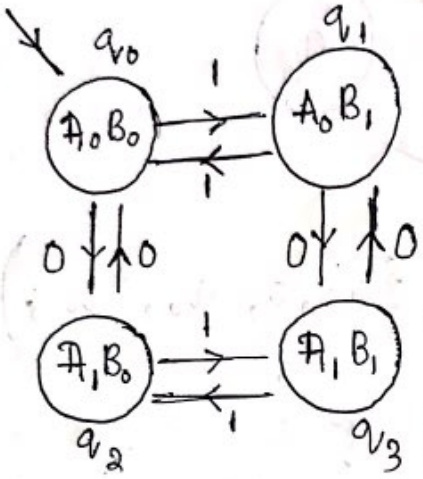
3> $\hat{\delta}(q_0, aaa) = \delta(\hat{\delta}(q_0, aa), a) = \delta(q_2, a) = q_0$

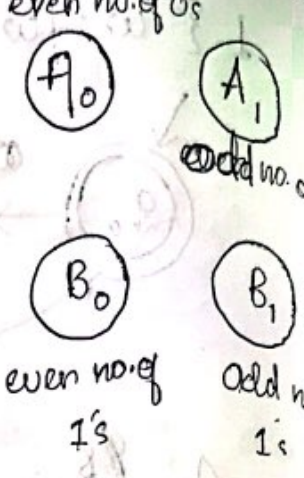
4> $\hat{\delta}(q_0, aaaa) = \delta(\hat{\delta}(q_0, aaa), a) = \delta(q_0, a) = q_1$

5> $\hat{\delta}(q_0, aaaaa) = \delta(\hat{\delta}(q_0, aaaa), a) = \delta(q_1, a) = q_2$

Regular ...

8) Construct DFA to accept even no. of 0's & even no. of 1's.





A_0B_0 - even 0's, even 1's

A_0B_1 - even 0's, odd 1's

A_1B_0 - odd 0's, even 1's

A_1B_1 - odd 0's, odd 1's

Verify:- $\hat{\delta}(q_0, 0101010)$

1> Bases:- $\hat{\delta}(q_0, \epsilon) = q_0$

2> $\delta(\hat{\delta}(q_0, \epsilon), 0) = q_2 = \hat{\delta}(q_0, 0)$

3> $\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_2, 1) = q_3$

4> $\hat{\delta}(q_0, 010) = \delta(\hat{\delta}(q_0, 01), 0) = \delta(q_3, 0) = q_1$

5> $\hat{\delta}(q_0, 0101) = \delta(\hat{\delta}(q_0, 010), 1) = \delta(q_1, 1) = q_0$

6> $\hat{\delta}(q_0, 01010) = \delta(\hat{\delta}(q_0, 0101), 0) = \delta(q_0, 0) = q_2$

7> $\hat{\delta}(q_0, 010101) = \delta(\hat{\delta}(q_0, 01010), 1) = \delta(q_2, 1) = q_3$

8> $\hat{\delta}(q_0, 0101010) = \delta(\hat{\delta}(q_0, 010101), 0) = \delta(q_3, 0) = q_1$

- Central Concepts of Automata Theory
- Kleene Closure, +ve closure
- Definition of DFA.
- Transition fun of DFA $\Rightarrow Q \times \Sigma = Q$
- Ext. " " " " $\Rightarrow Q \times \Sigma^* = Q$

a) Construct DFA $L = \{ w / w \in (a+b)^* N_a(w) \bmod 3 \ \& \ N_b(w) \bmod 2 = 0 \}$

Sol:

A_0 — a's - remainder '0'

A_1 — a's - rem 1

A_2 — a's - rem 2

B_0 — b's - rem '0'

B_1 — b's - rem 1

~~aba~~ ~~aabb~~ ~~aabba~~

A_0B_0 - a%3=0, b%2=0

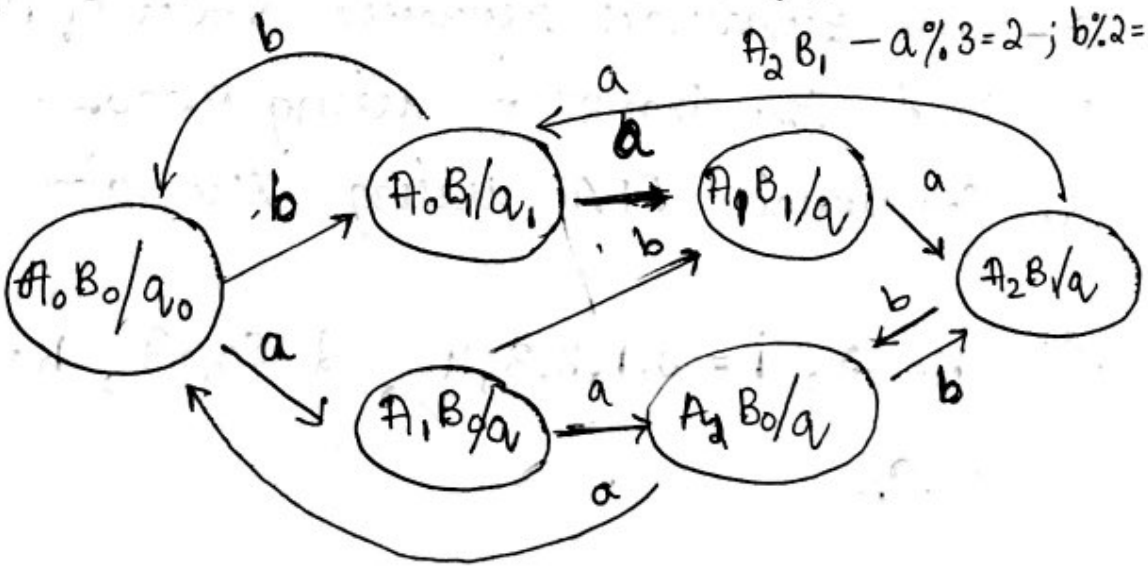
A_0B_1 - a%3=0 ; b%2=1

A_1B_0 - a%3=1 ; b%2=0

A_1B_1 - a%3=1 ; b%2=1

A_2B_0 - a%3=2 ; b%2=0

A_2B_1 - a%3=2 ; b%2=1



Q) Design DFA to accept binary strings divisible by.

000000
00104
1010

2|10
2|5 - 0
2|2 - 1
1 - 0

2|15
2|7 - 1
2|3 - 1
1 - 1

★ Divisible by k-problem

Transition can be obtained using the following relation

* $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$

$r \rightarrow$ radix of i/p

$i \rightarrow$ remainder obtained after dividing by k

$d \rightarrow$ represents digits for binary $d = \{0, 1\}$

$k =$ divisor

Steps:

1> Identify the radix, input alphabets & divisor

2> Compute the possible remainders

3> These remainders represent states of DFA.

3> Find the transition using above relation

4> Construct DFA using the above transition

Sol: $r = 02$, $i = 0, 1, 2, 3, 4$, $d = \{0, 1\}$, $k = 5$

Remainder (i)	d	$j = (r*i+d) \bmod k$
0	0	$j = (0\times 2+0) \bmod 5 = 0$
	1	$j = (2\times 0+1) \bmod 5 = 1$
1	0	$j = (2\times 1+0) \bmod 5 = 2$
	1	$j = (2\times 1+1) \bmod 5 = 3$
2	0	$j = (2\times 2+0) \bmod 5 = 4$
	1	$j = (2\times 2+1) \bmod 5 = 0$
3	0	$j = (2\times 3+0) \bmod 5 = 1$
	1	$j = (2\times 3+1) \bmod 5 = 2$
4	0	$j = (2\times 4+0) \bmod 5 = 3$
	1	$j = (2\times 4+1) \bmod 5 = 4$

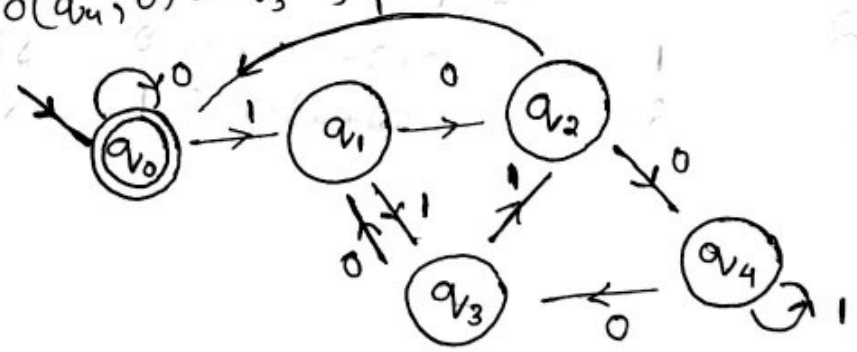
$\delta(q_0,0) = q_0$, $\delta(q_0,1) = q_1$,

$\delta(q_1,0) = q_2$, $\delta(q_1,1) = q_3$,

$\delta(q_2,0) = q_4$, $\delta(q_2,1) = q_0$,

$\delta(q_3,0) = q_1$, $\delta(q_3,1) = q_2$,

$\delta(q_4,0) = q_3$, $\delta(q_4,1) = q_4$

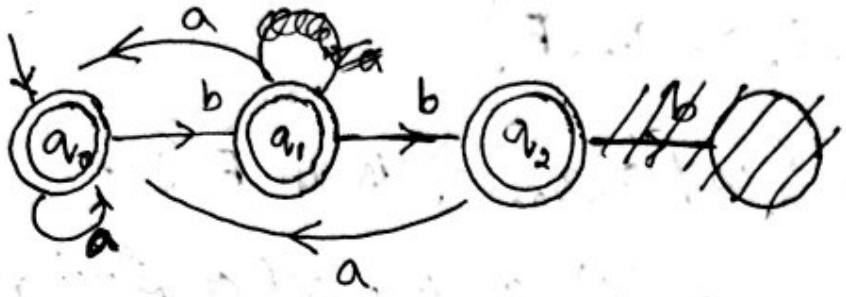


8) DFA to accept decimal Strings divisible by '3' (decimal)

Sol: $r=10$, $d=\{0,1,2,\ldots,9\}$; $i=0,1,2$ ($k=3$)

Remainder (i)	d	$j=(r*i+d) \bmod k$	δ_1
0	0	$j=(10*0+0)=0$	$\delta(q_0,0)=q_0$
	1	$j=(10\times0+1) \bmod 3=1$	$\delta(q_0,1)=q_1$
	2	$j=(10\times0+2) \bmod 3=2$	$\delta(q_0,2)=q_2$
	3	$j=(10\times0+3) \bmod 3=0$	$\delta(q_0,3)=q_0$
	4	$j=(0+4) \bmod 3=1$	$\delta(q_0,4)=q_1$
	5	$j=(0+5) \bmod 3=2$	$\delta(q_0,5)=q_2$
	6	$j=(6) \bmod 3=0$	$\delta(q_0,6)=q_0$
	7	$j=7 \bmod 3=1$	$\delta(q_0,7)=q_1$
	8	$j=8 \bmod 3=2$	$\delta(q_0,8)=q_2$
	9	$j=9 \bmod 3=0$	$\delta(q_0,9)=q_0$
1	0	$j=(10\times1+0) \bmod 3=1$	$\delta(q_1,0)=q_1$
	1	$j=(10\times1+1) \bmod 3=2$	$\delta(q_1,1)=q_2$
	2	$j=(10\times1+2) \bmod 3=0$	$\delta(q_1,2)=q_0$
2	0	$j=(10\times2+0) \bmod 3=2$	$\delta(q_2,0)=q_2$
	1	$j=(10\times2+1) \bmod 3=0$	$\delta(q_2,1)=q_0$
	2	$j=(10\times2+2) \bmod 3=1$	$\delta(q_2,2)=q_1$

8) design DFA to accept string of a's & b's with atmost 2 consecutive b's.



8) Design DFA to accept $L = \{ \omega : n_a(\omega) \geq 1, n_b(\omega) = 2 \}$

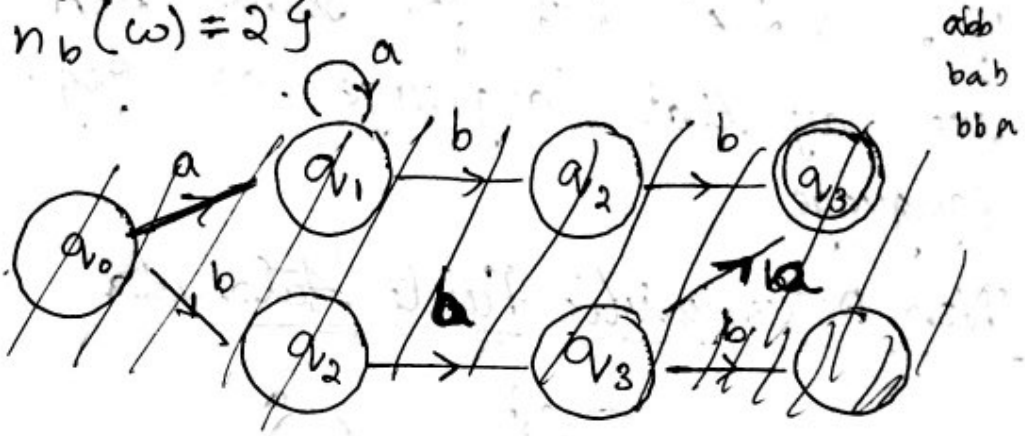
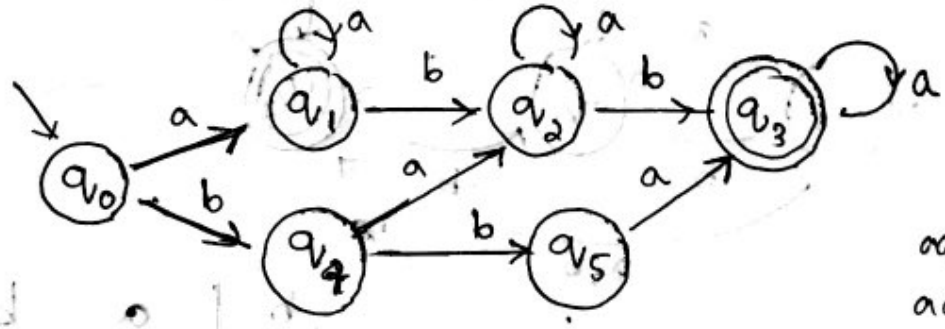


abb
bab
bba



aabb
aabaab
baabac

$A = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \delta, q_0, \{q_3\})$

$\delta(q_0, a) = q_1$
$\delta(q_0, b) = q_4$
$\delta(q_1, a) = q_1$
$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_2$
$\delta(q_2, b) = q_3$
$\delta(q_3, a) = q_3$
$\delta(q_4, a) = q_2$
$\delta(q_4, b) = q_5$
$\delta(q_5, a) = q_3$

16-07-18

Language of a DFA

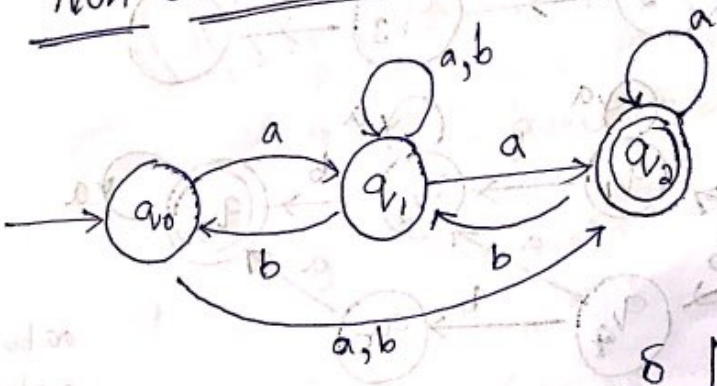
$A = (Q, \Sigma, \delta, q_0, F)$

$L(A) = \{ \omega / \hat{\delta}(q_0, \omega) \text{ is in } F \}$

→ The language of 'A' is the set of strings 'ω' that take the start state 'q_0' to one of the accepting state.

→ If L is L(A) then for some DFA A the we say that language as regular language.

Non Deterministic Finite Automata



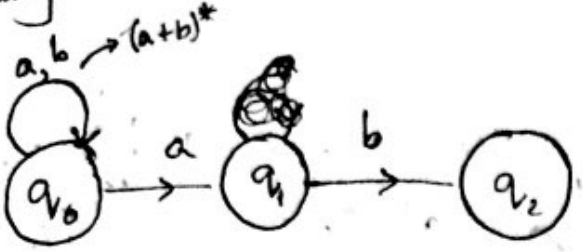
δ	a	b
q_0	$\{q_1, q_2\}$	$\{q_2\}$
q_1	$\{q_1, q_2\}$	$\{q_1\}$
q_2	$\{q_2\}$	$\{q_1\}$

$\delta(q_0, a) = \{q_1, q_2\}$

$\delta(q_0, a) = P$

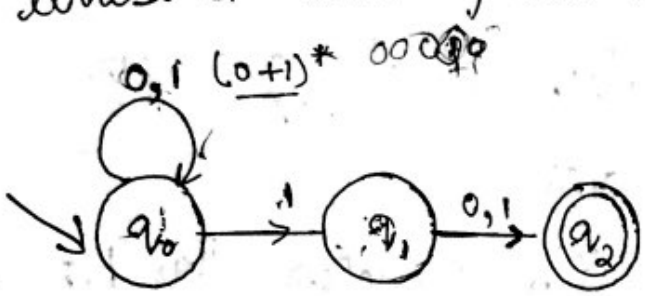
$|Q \times \Sigma = 2^Q|$

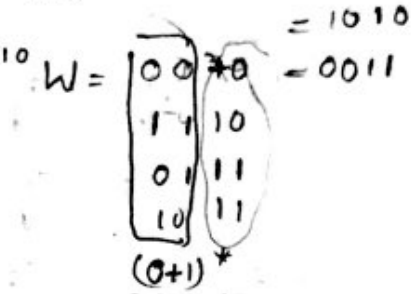
Ending with 'ab'.



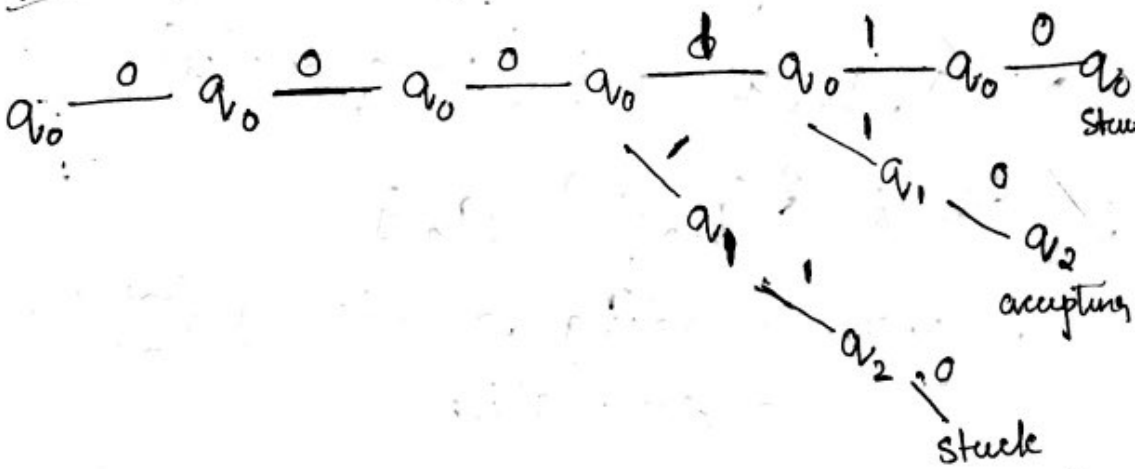
(a+b)

* Draw NFA accepting set of all string whose 2nd last symbol is '1'.





verify: 000110



Non-Deterministic Finite Automata :- Definition

$A = (Q, \Sigma, \delta, q_0, F)$

Q = Finite set of State

Σ = Finite set of i/p symbol

q_0 = Start state. & is a member of Q

F = Subset of Q is the set of final & accepting state.

δ = Transition function is a function that takes a state q_0 & i/p sym 'a' from set of Q symbol in Σ as arguments and returns a subset of Q.

The type of value that δ returns is set of states

Extended transition function for NFA

→ It is represented by $\hat{\delta}$

→ $\hat{\delta}(q_0, w) = ?$ (Set of states)

Proof Basis $\hat{\delta}(q_0, \epsilon) = \{q_0\}$

→ Without reading any i/p symbol we are on in the same state.

Induction :

→ Suppose w is the form $\boxed{w = xa}$

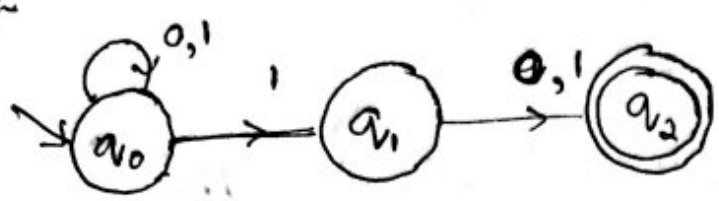
* Where 'a' is finale symbol of 'w' & 'x' is rest of 'w'.

→ Also suppose that $\hat{\delta}(q, x) = \{p_1, p_2 \cdots p_k\}$

let $\bigcup_{k=1}^{k} \delta(p_i, a) = \{r_1, r_2 \cdots r_m\}$

then $\hat{\delta}(q, \omega) = \{r_1, r_2 \ldots r_m\}$

Ex:



verify :: 010010

Basis: $\hat{\delta}(q_0, \epsilon) = q_0$

Induction::

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0) = \delta(q_0, 0) = q_0$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_0, 1) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 010) = \delta(\hat{\delta}(q_0, 01), 0) = \delta(\{q_0, q_1\}, 0) =$

$= \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_2$

$\hat{\delta}(q_0, 0100) = \delta(\hat{\delta}(q_0, 010), 0) = \delta(\{q_0, q_2\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0\} \cup \{\} = q$

$\hat{\delta}(q_0, 01001) = \delta(\hat{\delta}(q_0, 0100), 1) = \delta(q_0, 1) = \{q_0, q$

$\hat{\delta}(q_0, 010010) = \delta(\hat{\delta}(q_0, 01001), 0) = \delta(\{q_0, q_1\}, 0)$

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

$\Rightarrow$ accepted State

A language of a DFA

$A = (Q, \Sigma, \delta, q_0, F)$ is an NFA then

$$L(a) = \{ w / \hat{\delta}(q_0, w) \cap F \neq \emptyset \}$$

$L(A)$ is the set of strings 'w' in Σ^+ (Kleen closure) such that $\hat{\delta}(q_0, w)$ contains atleast one accepting state.

Equivalence of NFA & DFA

- Subset Configuration

The subset construction starts from an NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

Its goal is the description of DFA

$$D = (Q_0, \Sigma, \delta_D, \{q_0\}, F_0)$$

that the language of $D : L(D) = L(N)$

that is input alphabets of two automatas are same and start state of D is the set containing only the start state of N.

→ Q_D is the set of subsets of Q_N

Q_D is the power set of Q_N.

$Q_D = N$ —— Q_D 2^N states

often not all the states are accessible from the start state of Q_D.

~~Traversable sta~~

Traversable states can be thrown away.

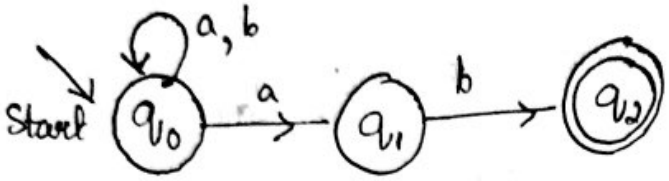
→ F_D is the set of subsets S of Q_N such that $S \cap F_n \neq \emptyset$

that is F_D is all sets of N's states that include atleast one accepting state of N (Delta Transition).

→ For each set $S \subseteq Q_N$ and for each i/p symbol a in Σ

$$\delta_D(\delta, a) = \bigcup_{P \text{ in } S} \delta_N(P, a)$$

Q) Convert NFA to DFA



$\Sigma_N = \Sigma_D = \{a, b\}$

Start State $= \{q_0\}$

$Q_D = \{q_0\}$

$= \{q_1\}$

$= \{q_2\}$

$= \{q_0, q_1\}$

$= \{q_1, q_2\}$

$= \{q_0, q_2\}$

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

$\delta_D(\emptyset, a) = \emptyset$

$\delta_D(\emptyset, b) = \emptyset$

$\delta_D(q_0, a) = \{q_0, q_1\}$

$\delta_D(q_0, b) = \{q_0\}$

$\delta_D(q_1, a) = \emptyset$

$\delta_D(q_1, b) = q_2$

$\delta_D(q_2, a) = \emptyset$

$\delta_D(q_2, b) = \emptyset$

$\delta_D(\{q_0, q_1\}, a) = \delta_0(q_0, a) \cup \delta_0(q_1, a) = \{q_0, q_1\}$

$\delta_D(\{q_0, q_1\}, b) = \delta_0(q_0, b) \cup \delta_0(q_1, b) = \{q_0, q_2\}$

$\delta_D(\{q_1, q_2\}, a) = \delta_0(q_1, a) \cup \delta_0(q_2, a) = \{\emptyset\}$

$\delta_D(\{q_1, q_2\}, b) = \delta_0(q_1, b) \cup \delta_0(q_2, b) = \{q_2\}$

$\delta_D(\{q_0, q_2\}, a) = \delta_0(q_0, a) \cup \delta_0(q_2, a) = \{q_0, q_1\}$

$\delta_D(\{q_0, q_2\}, b) = \delta_0(q_0, b) \cup \delta_0(q_2, b) = \{q_0\}$

$\delta_D(\{q_0, q_1, q_2\}, a) = \delta_0(q_0, a) \cup \delta_0(q_1, a) \cup \delta_0(q_2, a)$

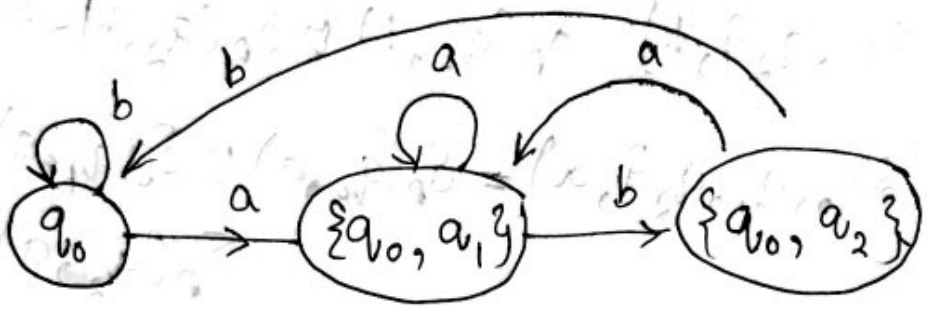
$= \{q_0, q_1\}$

$\delta_D(\{q_0, q_1, q_2\}, b) = \delta_0(q_0, b) \cup \delta_0(q_1, b) \cup \delta_0(q_2, b)$

$= \{q_0, q_2\}$

δ	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{\emptyset\}$	q_2
q_2	$\emptyset$	$\emptyset$
q_0, q_1	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1, q_2	$\{\emptyset\}$	$\{q_2\}$
q_0, q_2	$\{q_0, q_1\}$	$\{q_0\}$
* q_0, q_1, q_2	$\{q_0, q_1\}$	$\{q_0, q_2\}$

	a	b
q_0	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

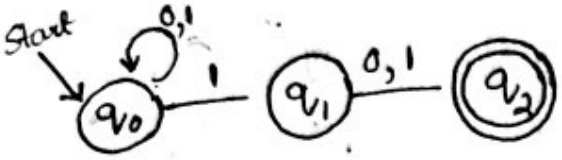


NFA $= (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$

DFA $= (\{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}\}, \{a, b\},$

$\delta, \{q_0\}, \{\{q_0, q_2\}\})$.

6) Convert NFA to DFA



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

Start State = $\{q_0\}$

Q_D (Power Set) = $\{q_0\}$

$= \{q_1\}$

$= \{q_2\}$.

$= \{q_0, q_1\}$

$= \{q_0, q_2\}$

$= \{q_1, q_2\}$

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

$\delta_D(\emptyset, 1) = \emptyset$

$\delta_D(\emptyset, 0) = \emptyset$

$\delta_D(\{q_0\}, 0) = \delta_N(q_0, 0) = \{q_0\}$

$\delta_D(\{q_0\}, 1) = \delta_N(q_0, 1) = \{q_0, q_1\}$

$\delta_D(\{q_1\}, 0) = \delta_N(q_1, 0) = \{q_2\}$

$\delta_D(\{q_1\}, 1) = \delta_N(q_1, 1) = \{q_2\}$

$\delta_D(\{q_2\}, 0) = \delta_N(q_2, 0) = \emptyset$

$\delta_D(\{q_2\}, 1) = \delta_N(q_2, 1) = \emptyset$

$\delta_D(\{q_0, q_1\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

$= \{q_0, q_2\}$

$\delta_D(\{q_0, q_1\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$

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

$\delta_D(\{q_0, q_2\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_2, 0)$

$= \{q_0\}$ $\not\emptyset$

$\delta_D(\{q_0, q_2\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_2, 1)$

$= \{q_1\}$

$\delta_D(\{q_1, q_2\}, 0) = \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_2\}$

$\delta_D(\{q_1, q_2\}, 1) = \delta_N(q_1, 1) \cup \delta_N(q_2, 1)$

$= \{q_2\}$

$\delta_D(\{q_0, q_1, q_2\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_0, q_2\}$

$\delta_D(\{q_0, q_1, q_2\}, 1) = \delta_N(\{q_0, 1\}) \cup \delta_N(q_1, 1) \cup \delta_N(q$

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

	0	1
q_0	q_0	$\{q_0, q_1\}$
q_1	q_2	q_2
q_2	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	q_0	$\{q_0, q_1\}$
$\{q_1, q_2\}$	q_2	q_2
$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

Final transition table :

Subset method

	0	1
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	q_0	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

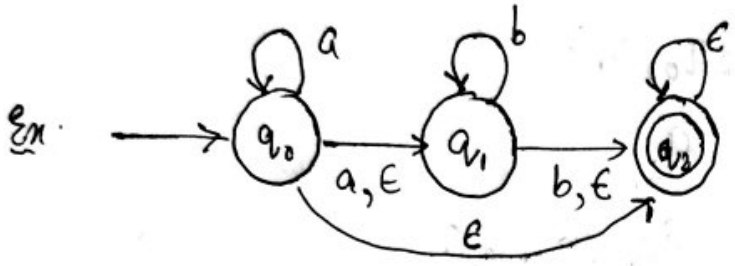
(q_0)

NFA with ϵ transitions

DFA : $\delta : Q \times \Sigma \longrightarrow Q$

NFA : $\delta : Q \times \Sigma \longrightarrow 2^Q$

ϵ-NFA : $\delta : Q \times (\Sigma \cup \epsilon) \longrightarrow 2^Q$



δ	ϵ	a	b	$\emptyset$
q_0	$\{q_1, q_2\}$	$\{q_0, q_1\}$	$\emptyset$	$\emptyset$
q_1	q_2	$\emptyset$	$\{q_1, q_2\}$	$\emptyset$
q_2	q_2	$\emptyset$	$\emptyset$	

Definition : An NFA with one (or) more ϵ transition is called ϵ-NFA.

ϵ transition : A transition with an empty i/p string is called ϵ-transition

→ If there is a change of state from

q_i to q_j on ϵ then we write it as

$$\boxed{\delta(q_i, \epsilon) = q_j}$$

Epsilon Closure (E closure):

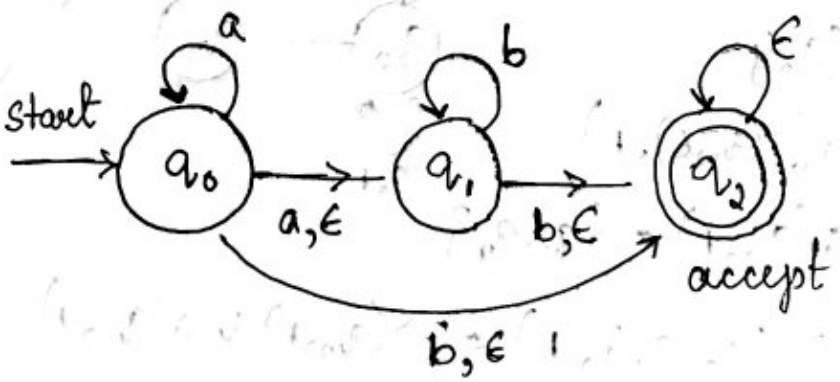
→ E-closure of q is denoted by Eclose(q)

→ E close(q) is the set of all states which are reachable from q on ϵ-transitions only.

1) State q is in E close of q ⇒ ECLOSE(q)
   i;e ECLOSE(q) = q

2) If ECLOSE(q) contains p and if there is a transition from state 'p' to state 'r' labeled 'ε' then state 'r' is also in ECLOSE(q)

Ex

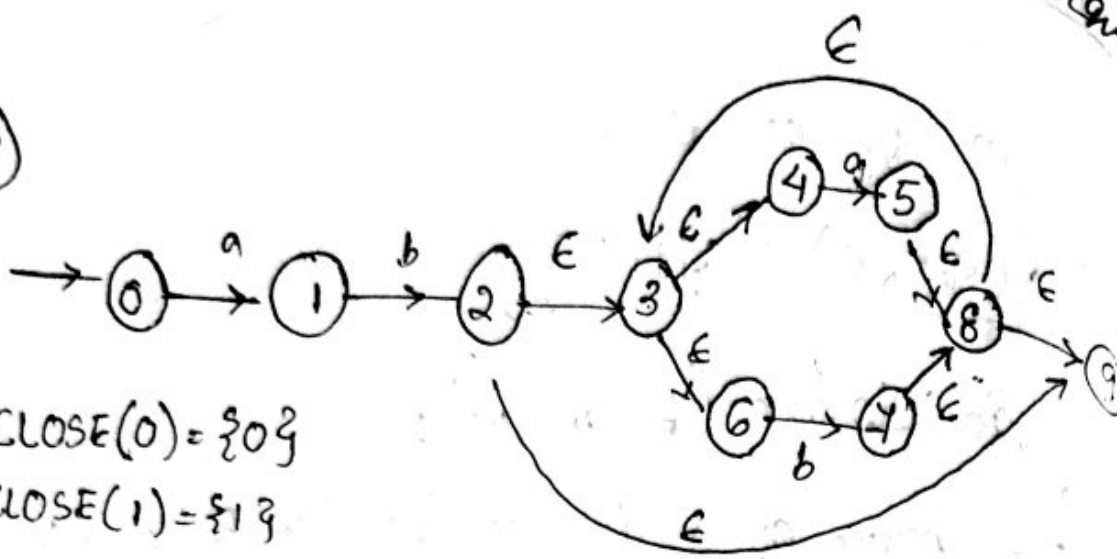


ECLOSE(q_0) = { q_0, q_1, q_2 }

ECLOSE(q_1) = { q_1, q_2 }

ECLOSE(q_2) = { q_2 }

2)



ECLOSE(0) = {0}

ECLOSE(1) = {1}

ECLOSE(2) = {2, 3, 4, 6}

ECLOSE(3) = {3, 4, 6}

ECLOSE(4) = {4}

ECLOSE(5) = {5, 8, 3, 4, 6}

ECLOSE(6) = {6}

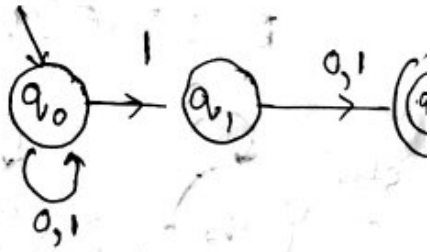
ECLOSE(7) = {7, 8, 3, 4, 6}

ECLOSE(8) = {8, 3, 4, 6}

ECLOSE(9) = {9}

Q)

verify $\hat{\delta}(q_0, 11111010)$



1> $\hat{\delta}(q_0, \epsilon) = \{q_0\}$

2> $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1)$

$= \delta(q_0, 1) = \{q_0, q_1\}$

3> $\hat{\delta}(q_0, 11) = \delta(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$

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

4> $\hat{\delta}(q_0, 111) = \delta(\{q_0, q_1, q_2\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta($

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

6) $\hat{\delta}(q_0, 1111) = \delta(\{q_0, q_1, q_2\}, 1) = \{q_0, q_1, q_2\}$

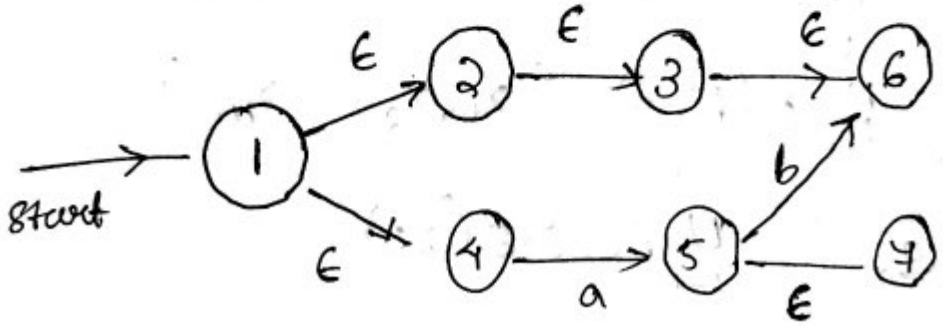
7) $\hat{\delta}(q_0, 11111) = \delta(\{q_0, q_1, q_2\}, 1) = \{q_0, q_1, q_2\}$

8) $\hat{\delta}(q_0, 111110) = \delta(\{q_0, q_1, q_2\}, 0) = \{q_0, q_2\}$

$\hat{\delta}(q_0, 1111101) = \delta(\{q_0, q_2\}, 1) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 11111010) = \delta(\{q_0, q_1\}, 0) = \{q_0, q_2\}$

(8)



Sol^n

ECLOSE(1) = {1, 2, 3, 6, 4}

ECLOSE(2) = {2, 3, 6}

ECLOSE(3) = {6, 3}

ECLOSE(4) = {4}

ECLOSE(5) = {5, 7}

ECLOSE(6) = {6}

ECLOSE(7) = {7}

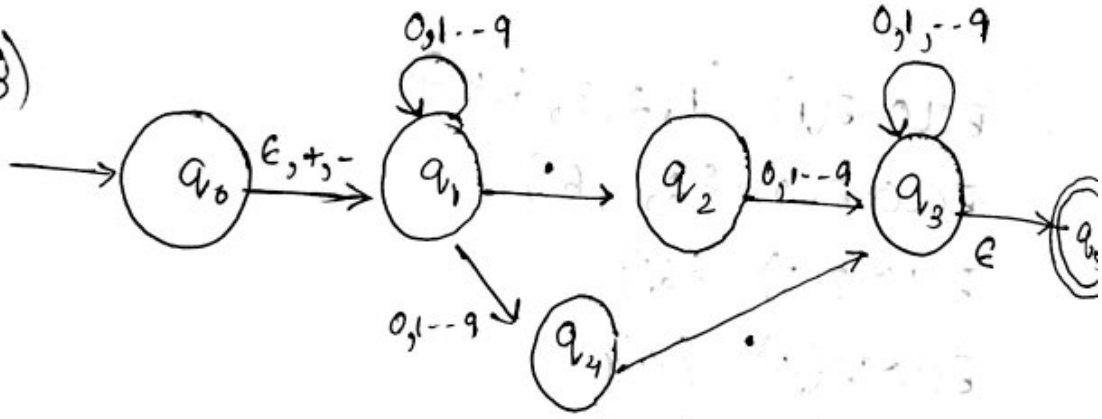
Extended Transition Function – String – ϵ-NFA

Base Condition : $\hat{\delta}(q_0, \epsilon) = ECLOSE(q_0)$

Induction: 'w' is of the form 'xa', 'a' is the last symbol of 'w'.

1. Let $\{P_1, P_2, \ldots, P_k\}$ be $\hat{\delta}(q, x)$
2. Let $\bigcup_{i=1}^{k} \delta(P_i, a) = \{r_1, r_2 \ldots r_m\}$
3. $\hat{\delta}(q, w) = \bigcup_{j=1}^{m} ECLOSE(r_j)$

Q)



Compute $\hat{\delta}(q_0, 5.6)$

1) $\hat{\delta}(q_0, \epsilon) = ECLOSE(q_0) = \{q_0, q_1\}$

2) $\hat{\delta}(q_0, 5) = \delta(\hat{\delta}(q_0, \epsilon), 5) = \delta(\{q_0, q_1\}, 5) = \{q_1, q_2\}$

3) $= ECLOSE(q_1) \cup ECLOSE(q_4)$

$= \{q_1, q_4\}$

3) $\hat{\delta}(q_0, 5\cdot) = \text{ECLOSE}[\delta(\hat{\delta}(q_0, 5), \cdot)]$

$= \text{ECLOSE}[\delta(\{q_1, q_4\}, \cdot)]$

$= \text{ECLOSE}[\delta(q_1, \cdot) \cup \delta(q_4, \cdot)]$

$= \text{ECLOSE}(q_2, q_3) = \{q_2, q_3, q_5\}$

4) $\hat{\delta}(q_0, 5.6) = \text{ECLOSE}[\delta(\hat{\delta}(q_0, 5\cdot), 6]$

$= \text{ECLOSE}[\delta(\{q_2, q_3, q_5\}, 6)]$

$= \text{ECLOSE}[\delta(q_2, 6), \delta(q_3, 6), \delta(q_5, \epsilon)]$

$= \text{ECLOSE}[\{q_3\}] = \{q_3, q_5\}$

Converting ϵ NFA to DFA (or) Eliminating ϵ transition

* The language of an ϵ NFA $E = (Q, \Sigma, \delta, q_0, F)$

$L(E) = \{w / \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$

i.e the language of E is the set of strings w that takes the start state to atleast one accepting state.

Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ then the equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$

1. Q_D is the set of subsets of Q_E, only accessible states of D are the ϵ-closed subsets of Q_E, that is those sets $S \subseteq Q_E$ such that $S = ECLOSE(S)$
2. $q_D = ECLOSE(q_0)$
3. F_D is those sets of states that contains at one accepting state of E.

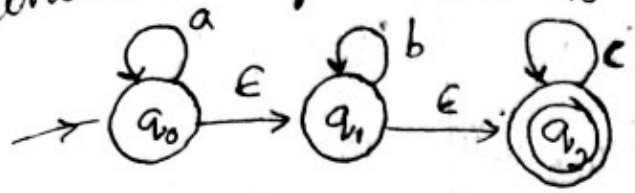
   that is $F_D = \{S / S \text{ is in } Q_D \text{ and } S \cap F_E \neq \emptyset\}$
4. $\delta_D(S, a)$ is computed for all a in Σ and sets S in Q_D by

   a) Let $S = \{p_1, p_2, \ldots, p_k\}$

   b) compute $\bigcup_{i=1}^{k} \delta(p_i, a)$ let this set be $\{r_1, r_2 \ldots r_m\}$

   c) Then $\delta_D(S, a) = \bigcup_{j=1}^{m} ECLOSE(r_j)$

Q) Convert the foll. NFA to its equivalent DFA.



Start State = ECLOSE (Start State of ∈NFA)

1) q_0 DFA $= ECLOSE(q_0) = \{q_0, q_1, q_2\}$

2) $\delta(\{q_0, q_1, q_2\}, a) = \overset{ECLOSE}{(}\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)$
$= ECLOSE\{q_0\} = \{q_0, q_1, q_2\}$

3) $\delta(\{q_0, q_1, q_2\}, b) = ECLOSE(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$
$= ELOSE(q_1) = \{q_1, q_2\}$

4) $\delta(\{q_0, q_1, q_2\}, c) = ECLOSE(\delta(q_0, c) \cup \delta(q_1, c) \cup \delta(q_2, c))$
$= ECLOSE(q_2) = \{q_2\}$

5) $\delta(\{q_1, q_2\}, a) = ECLOSE(\delta(q_1, a) \cup \delta(q_2, a))$
$= ECLOSE(\emptyset) = \emptyset$

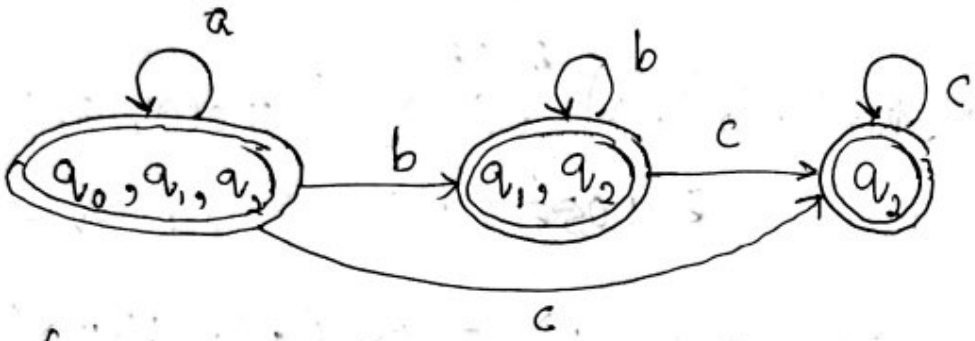
6) $\delta(\{q_1, q_2\}, b) = ECLOSE(\delta(q_1, b) \cup \delta(q_2, b)) =$
$= ECLOSE(q_1) = \{q_1, q_2\}$

7) $\delta(\{q_1, q_2\}, c) = ECLOSE(\delta(q_1, c) \cup \delta(q_2, c))$
$= ECLOSE(q_2) = q_2$

8) $\delta(q_2, a) = ECLOSE(\emptyset) = \emptyset$ 9) $\delta(q_2, b) = \emptyset$

10) $\delta(q_2, c) = ECLOSE(q_2) = \{q_2\}$

δ	a	b	c
→ * $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
* $\{q_1, q_2\}$	$\emptyset$	$\{q_1, q_2\}$	$\{q_2\}$
* $\{q_2\}$	$\emptyset$	$\emptyset$	$\{q_2\}$

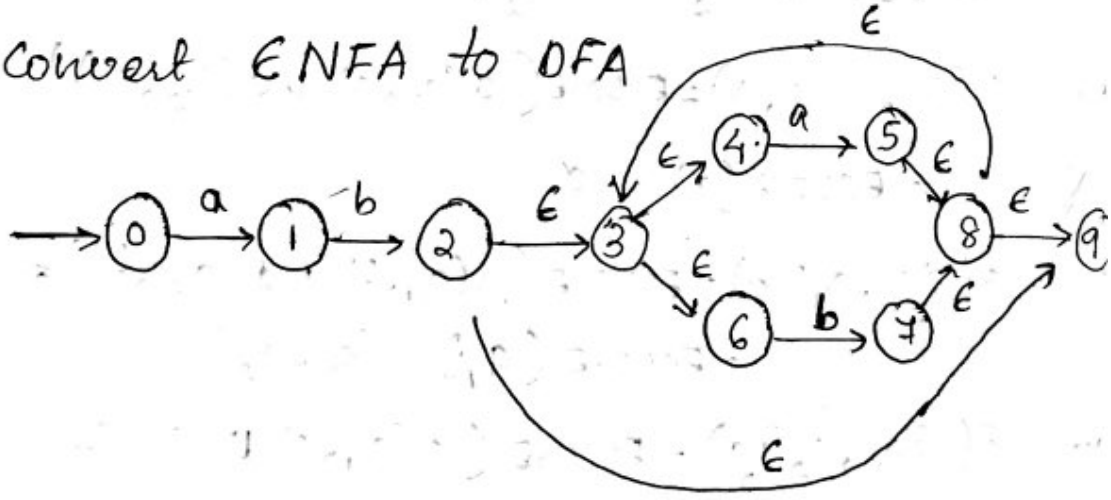


$E = (\{q_0, q_1, q_2\}, \{a, b, c\}, \delta, \{q_0\}, \{q_2\})$

$D = (\{\{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\}\}, \{a, b, c\},$

$\{q_0, q_1, q_2\}, \{\{q_0, q_1, q_2\}, \{q_1, q_2\}, \ldots$

Q) Convert ϵNFA to DFA



1> Start State = ECLOSE(0) = {0}

2> $\delta(0, a)$ = ECLOSE(1) = {1}

3> $\delta(0, b)$ = ECLOSE($\delta(0, b)$) = ECLOSE($\emptyset$) = $\emptyset$

4) $\delta(1, a) = \text{ECLOSE}(\delta(1,a)) = \text{ECLOSE}(2)$
$= \{2, 3, 4, 6, 9\}$

5) $\delta(1, b) = \text{ECLOSE}(\delta(1, b)) = \text{ECLOSE}(\emptyset) = \emptyset$

6) $\delta(\{2,3,4,6,9\}, a) = \text{ECLOSE}(\delta(\{2,3,4,6,9\}, a)$
$= \text{ECLOSE}(\{5\}) = \{5, 8, 3, 4, 6, 9\}$

7) $\delta(\{2,3,4,6,9\}, b) = \text{ECLOSE}(\delta(\{2,3,4,6,9\}, b)$
$= \text{ECLOSE}(\{7\}) = \{7, 8, 9, 3, 4, 6\}$

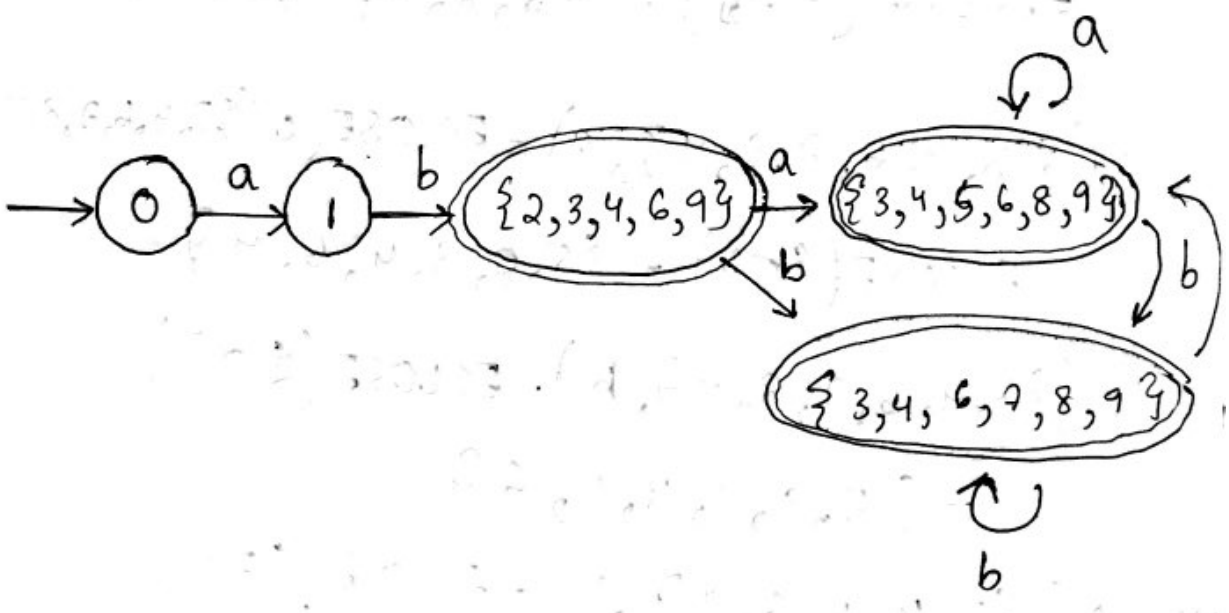
8) $\delta(\{3,4,5,6,9\}, a) = \text{ECLOSE}(\delta(\{3,4,5,6,9\}, a$
$= \text{ECLOSE}(\{5\}) = \{5, 8, 3, 4, 6, 9\}$

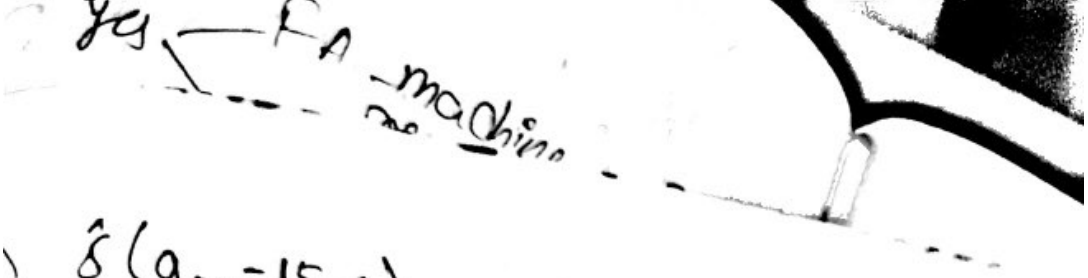
9) $\delta(\{3,4,6,7,8,9\}, a) = \text{ECLOSE}(\delta(\{3,4,6,7,8,9$
$= \text{ECLOSE}(\{5\}) = \{5, 8, 3, 4, 6, 9\}$

10) $\delta(\{3,4,6,7,8,9\}, b) = \text{ECLOSE}(\{7\})$
$= \{7, 8, 9, 3, 4, 6\}$

11) $\delta(\{3,4,5,6,9\}, b) = \text{ECLOSE}(\{7\})$
$= \{7, 8, 9, 3, 4, 6\}$

δ	a	b
→ 0	1	∅
1	∅	{2, 3, 4, 6, 9}
* {2,3,4,6,9}	{5,8,3,4,6,9}	{3, 4, 6, 7, 8, 9}
* {3,4,5,6,8,9}	{3,4,5,6,8,9}	{3, 4, 6, 7, 8, 9}
* {3,4,6,7,8,9}	{3,4,5,6,8,9}	{3, 4, 6, 7, 8, 9}





$\hat{\delta}(q_0, -15.6)$ verify:

$\hat{\delta}(q_0, \epsilon) = \text{ECLOSE}(q_0) = \{q_0, q_1\}$

$\hat{\delta}(q_0, -) = \text{ECLOSE}(\delta(\hat{\delta}(q_0, \epsilon), -))$

$= \text{ECLOSE}(\delta(\{q_0, q_1\}, -)) = \text{ECLOSE}(\{q_1\})$

$= \{q_1\}$

$\hat{\delta}(q_0, -1) = \text{ECLOSE}(\delta(\hat{\delta}(q_0, -), 1)) = \text{ECLOSE}(\delta(q_1,$

$= \text{ECLOSE}(\{q_1, q_4\}) = \{q_1, q_4\}$

$\hat{\delta}(q_0, -15) = \text{ECLOSE}(\delta(\hat{\delta}(q_0, -1), 5)) = \text{ECLOSE}(\delta($

$\delta(\{q_1, q_4\}, 5) = \text{ECLOSE}(\{q_1\})$

$= \{q_1\}$

$\hat{\delta}(q_0, -15.) = \text{ECLOSE}(\delta(\hat{\delta}(q_0, -15), .))$

$= \text{ECLOSE}(\delta(q_1, .)) = \text{ECLOSE}(q_2) = \{q_2$

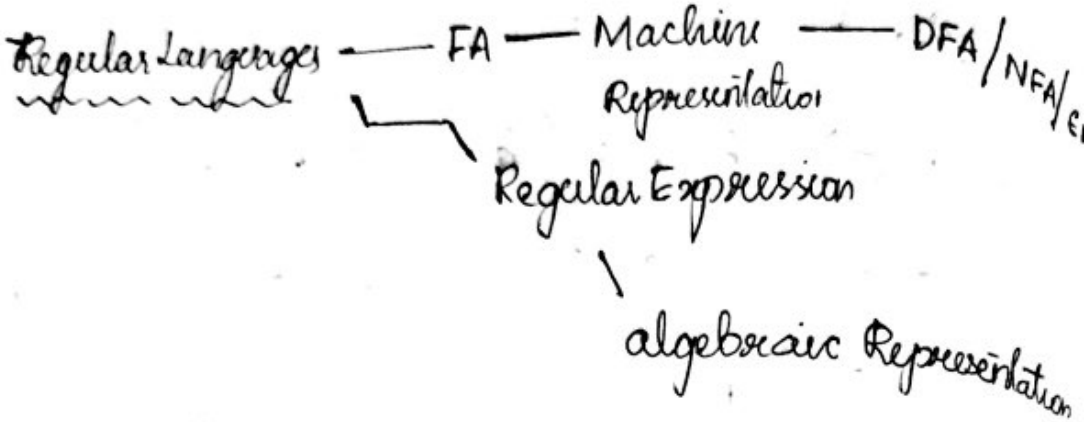
$\hat{\delta}(q_0, -15.6) = \text{ECLOSE}(\delta(\hat{\delta}(q_0, -15.), 6))$

$= \text{ECLOSE}(\delta(q_2, 6)) = \text{ECLOSE}(q_3)$

$= \{q_3, q_5\}$

Accepted//

Regular Expressions



FA : Application

- Text Search

Regular Exp^{n} : Application

- UNIX - GREP command
- Compiler design - Lexical analyzer

The operators of Regular Expression

1) Union of two languages

Let 'L' and 'M' be two language then union is represented as 'L' U M, is the set of string that are in L (or) M (or) both.

Eg:- $L = \{001, 10, 111\}$; $M = \{\epsilon, 001\}$

$L \cup M = \{\epsilon, 001, 10, 111\}$

2> concatenation of two languages: L & M
the set of strings that can be formed
by taking any string in 'L' and
concatenating with any string in 'M'
denote it either with '.' (or) no symbol

$L = \{001, 10, 111\}$ $M = \{\epsilon, 001\}$

$LM = \{001, 10, 111, 001001, 10001, 111001\}$

3> Closure of Language:
The closure of a language 'L' is
denoted by L^* and represents the
set of those string that can be formed
by taking any no. of string from 'L'
possibly with repitetion & concatenate
all of them.

Ex:- $L = \{0, 1\}$

$L^* = \epsilon \cup L^1 \cup L^2 \cdots$

$L = \{0, 11\}$

$L^* = \epsilon \cup L^1 \cup L^2 \cdots$

Building regular Expression

* **Basic** consist of 3 parts :-

1. ⟶ Constants, ϵ and $\emptyset$.
   - ϵ & $\emptyset$ are regular expression.
   - denotes the language $\{\epsilon\}$ and $\emptyset$

2. If a is any symbol then a is R.E.
   - This R.E denotes the language a i;e.
   i;e $L(a) = \{a\}$;

3. ~~A variable~~ Represent any language with Upper Case.

* Induction :-

1. If E & F are RE then $(E+F)$ is a R.E denoting the union of $L(E)$ and $L(F)$
   i;e $L(E+F) = L(E) \cup L(F)$

2. If E & F are regular Exp^n then EF is a denoting the concatenation $L(E)$ and $L(F)$
   i;e $L(EF) = L(E)L(F)$

3. If 'E' is the R.E then 'E*' is a R.E denoting the closure of L(E)
   i.e $L(E^*) = (L(E))^*$
4. If 'E' is the R.E then (E), a parenthesized E, also a Reg.Exp^n denoting the same language 'E'.

* Precedence of R.E operators

1. '*' has highest precedence. (Kleene Closure)
2. '.' – Concatenation
3. '+' – Union

→ a^* : String consist of any no. of a's.

$a^* = \{\epsilon, a, aa - - \}$

→ a^+ : string consist of a's atleast one time.

$a^+ = \{ a, aa - - \}$

→ (a+b) : String consist either one 'a' (or) one 'b'

→ $(a+b)^*$: set of String consist ~~either~~ of a (or) b of any length including ϵ

→ $(a+b)^*abb$: String set ending with 'abb'.

→ $ab(a+b)^*$: String set starting with 'ab'.

→ $(a+b)^*aa(a+b)^*$: String set containing substring 'aa'.

RE

1Q) Obtain RE for $\Sigma = \{a, b\}$ with length...

2Q) R.E representing representing $\Sigma = \{a, b\}$ len... ≤ 2

3Q) " " " length ≤ 10.

4Q) Obtain RE representing $\Sigma = \{a, b\}$ having even length.

5Q) " " " odd length

6Q) " " to accepts $\Sigma = \{a, b\}$ starting with 'a' and ending with 'b'.

7Q) " " second symbol from right is 'a'

8Q) 10th symbol from right end is 'a'

9Q) 3rd from right is 'a' & 4th from right is b

1A: $(a+b).(a+b) = (a+b)^2$

2A: ~~$(a+b)*(a+b)+a$~~ $(a+b)+(a+b).(a+b)$

3A: $(a+b)+((a+b).(a+b))+(a+b)^3+(a+b)^4+(a+b)^5+(a+b)^6$
$(a+b)^7+(a+b)^8+(a+b)^9+(a+b)^{10}$

4A: $((a+b)(a+b))^*$

5A: $((a+b))((a+b)(a+b))^*$

6A: $a\{a+b)^*b = a(\epsilon+a+b)^*b$

7A $(\epsilon+a+b)^*a\,(a+b+\epsilon)$

8A)

8A) $(a+b)(a+b)^* a(a+b)^9$

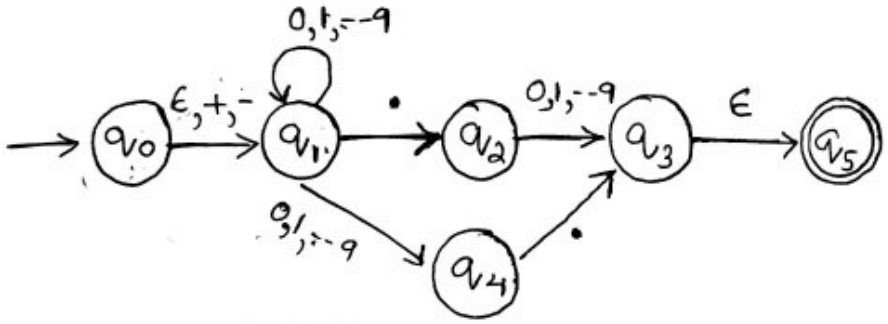
9A $(a+b)^* ba(a+b)(a+b)$

2A) $(\epsilon+a+b)(\epsilon+a+b) = (\epsilon+a+b)^2$

3A) $(\epsilon+a+b)^{10}$

---

9)



Convert ENFA to DFA

Sol 1> Start State = $ECLOSE(q_0) = \{q_0, q_1\}$

2> $\delta_D(\{q_0, q_1\}, 0) = ECLOSE(\delta(q_0, 0) \cup \delta(q_1, 0))$

$= ECLOSE(\{q_1, q_4\}) = \{q_1, q_4\}$

3> $\delta_D(\{q_1, q_4\}, +) = ECLOSE(\delta(q_1, +) \cup \delta(q_4, +))$

$= \{\emptyset\}$

4> $\delta_D(\{q_0, q_1\}, +) = ECLOSE(\delta(q_0, +) \cup \delta(q_1, +))$

$= ECLOSE(\{q_1\}) = \{q_1\}$

5> $\delta_D(\{q_0, q_1\}, .) = ECLOSE(\delta(q_0, .) \cup \delta(q_1, .))$

$= ECLOSE(\{\emptyset\} \cup \{q_2\}) = \{q_2\}$

5) $\delta(\{q_1, q_4\}, 0) = ECLOSE(\{$

Q) Write regular exp^n for the string start & end with same letter.

abab a
bbab

Sol^n

$$a(a+b)^*a + b(a+b)^*b$$

Q) String donot end with 01 over $\{0,1\}$

Sol^n

~~$(a+b)^*ab$~~ $\Rightarrow (0+1)^*00 + (0+1)^*10 + (0+1)^*1$

Q) Obtain a reg. exp^n to accept string of a's & b's length is either even (or) multiples of 3.

Sol^n ~~$(a+b)(a+b)$~~ $((a+b)^2)^* + ((a+b)^3)^*$

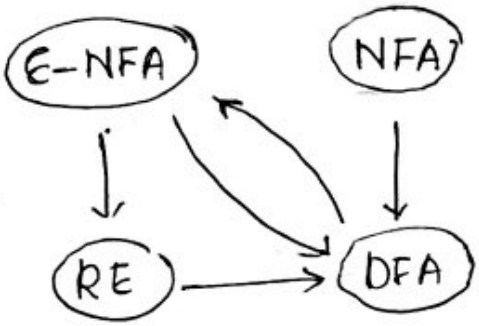
Q) Obtain a reg. exp^n to obtain strings of a's & b's such that every block of 4 consecutive symbol contains atleast 2a's.

Sol^n $(aa(a+b)^2 + a(a+b)^2a + a(a+b)a(a+b) + (a+b)a(a+b)a + (a+b)^2aa + (a+b)aa(a+b))^*$

8) $L\{a^m, b^n$ where / $m+n$ is even $\}$

Eg" $((a+b)^2)^* = ((a+b)(a+b))^*$ --

→ $(aa)^*(bb)^* + a(aa)^* b(bb)^*$



→ Every language defined by a regular exp^n is also defined by finite Automata.

→ $L = L(R)$ for R.E 'R'.

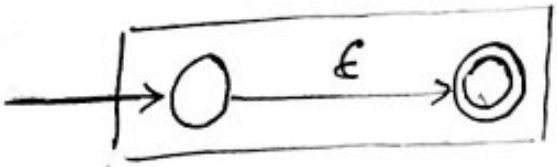
we can show that $L = L(E)$

for some ϵ-NFA E with.

1. Exactly one accepting state
2. No arcs into the initial state
3. No arcs out of the accepting state

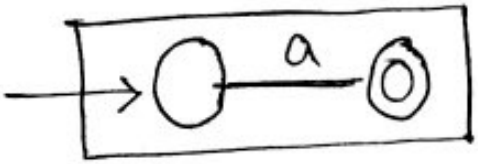
Basis :-

i) ~~$\emptyset$~~ ϵ :



ii) $\emptyset$:

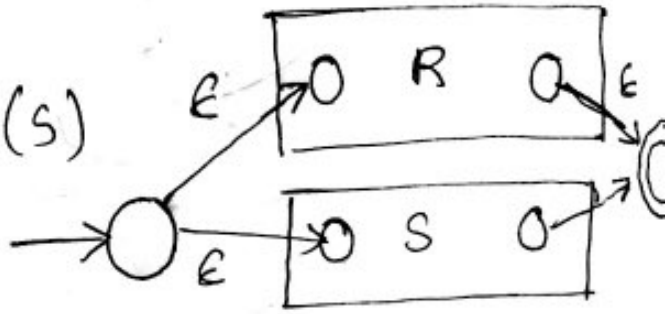
iii) a :



Union

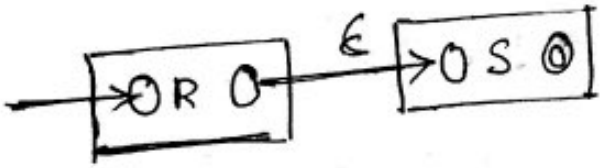
- $R+S$ — $L(R) \cup L(S)$

$L(R)$	$L(S)$

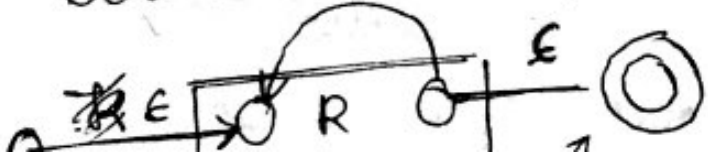


Concatenate

- RS — $L(R)\,L(S)$



Kleene Closure : R^*

- 

Equivalence of RE & FA

→ DFA to R.E

→ RE to ENFA

i) DFA to R.E

1) 

DFA

$\Rightarrow$ R.E : $1(0+1)^*$

* $R_{ij}^{(k)}$: is the name of the regular exp^n whose language is set of strings 'w' such that 'w' is the label of path from state i to state j in A and that path has no intermediate nodes whose no. is greater than k.

→ The beginning & end pts of the path are not intermediate

→ To construct exp^n $r_{ij}^{(k)}$ start at $k=0$ and finally reaching $k=n$.

Basis : $k=0$

⇒ The path has no intermediate state at all.

There are two types of path when $k=0$:

1. An arc from node(state) i to node j.

2. A path of length 0 that consist of only some node i.

→ If $i \neq j$ then

a) if there is no such a,

then $R_{ij}^{(0)} = \phi$

b) if there is exactly one such symbol a,

then $R_{ij}^{(0)} = a$

c) if there are symbols $a_1, a_2 -- , a_k$ that labels are from state i to state j then

then $R_{ij}^{(0)} = a_1 + a_2 + a_3 + -- + a_k$

→ If $i = j$ then the legal paths are the path length 0 and all loops from i to itself

→ The path of length 0 is represented by the R.E is 'ϵ'

→ The path of length one such symbol then

$RE = a + \epsilon$

→ if there are symbols $a_1, a_2 -- a_k$ then

$RE = a_1 + a_2 + -- + a_k + \epsilon$

Induction

Suppose there is a path from state i to j that goes no far from state 'k'.

1) Path doesn't go through state 'k' at all. In this label of path is the language of $R_{ij}^{(k-1)}$

2) Path pass through state 'k' atleast once then we can break the path into several pieces: i) The first goes from state i to k without passing through 'k'

ii) The second goes from 'i' to 'k' without 'j'

and all th

→ ij the path

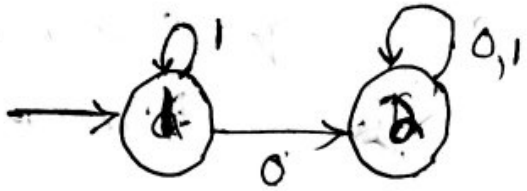
→ The set of labels of all the paths of this type is, represented by R.E that is

$$\left(R_{ik}^{(k-1)}\right)\left(R_{kk}^{(k-1)}\right)^{*}\left(R_{kj}^{(k-1)}\right)$$

Total RE $R_{ij}^{(k-1)} + R_{ik}^{(k-1)}\left(R_{kk}^{(k-1)}\right)^{*} R_{kj}^{(k-1)}$

Q) convert given DFA to R.E

Renam:



Bases $k=0$

$R_{11}^0 = 1+\epsilon$

$R_{22}^0 = 0+1+\epsilon$

$R_{21}^0 = \emptyset$

$R_{12}^0 = 0$

$k=1$

$R_{11}^{(1)} = R_{11}^0 + R_{11}^0 (R_{11}^0)^* R_{11}^0 = (1+\epsilon) + (1+\epsilon)(1+\epsilon)^*$

$i=1; j=1; k=1$

$= (\epsilon+1) + (\epsilon+1)^* = (\epsilon+1) + 1^* = 1^*$

$R_{12}^{(1)} = R_{12}^0 + R_{11}^1 (R_{11}^0)^* (R_{12}^0)$

$i=1; j=2; k=1$

$= \emptyset + 1^* (1+\epsilon)^* 0 = \emptyset 1^* 0$

$R_{21}^{(1)} = R_{21}^0 + R_{21}^0 (R_{11}^0)^* R_{11}^0$

$(i=2; j=1; k=1) = \emptyset + \emptyset (\quad) = \emptyset$

$R_{22}^{(1)} = R_{22}^{0} + R_{21}^{0}\left(R_{11}^{0}\right)^{*} R_{12}^{0} = (0+1+\epsilon) + \emptyset = (\epsilon+1+0)$

$i=2, j=2, k=1$

$k=2$

$R_{11}^{(2)} = R_{11}^{1} + R_{12}^{1}\left(R_{22}^{1}\right)^{*} R_{21}^{1} = 1^{*} + 1^{*}0\,(\epsilon+1)^{*}\emptyset = 1^{*}$

$i=1 j=1 k=2$

$R_{21}^{(2)} = R_{21}^{1} + R_{22}^{1}\left(R_{22}^{1}\right)^{*} R_{21}^{1} = \emptyset$

$j=2 j=1 k=2$

$R_{12}^{(2)} = R_{12}^{1} + R_{12}^{1}\left(R_{22}^{1}\right)^{*}\left(R_{22}^{1}\right)$

$i=1; j=2; k=2 = 1^{*}0 + 1^{*}0\,(1+\epsilon+0)^{*}(1+\epsilon)^{*}$

$= 1^{*}0 + 1^{*}0\,(1+0)^{*} = 1^{*}0 + 1^{*}0\,(0+1)^{*} = 1^{*}0(1+0)^{*}$

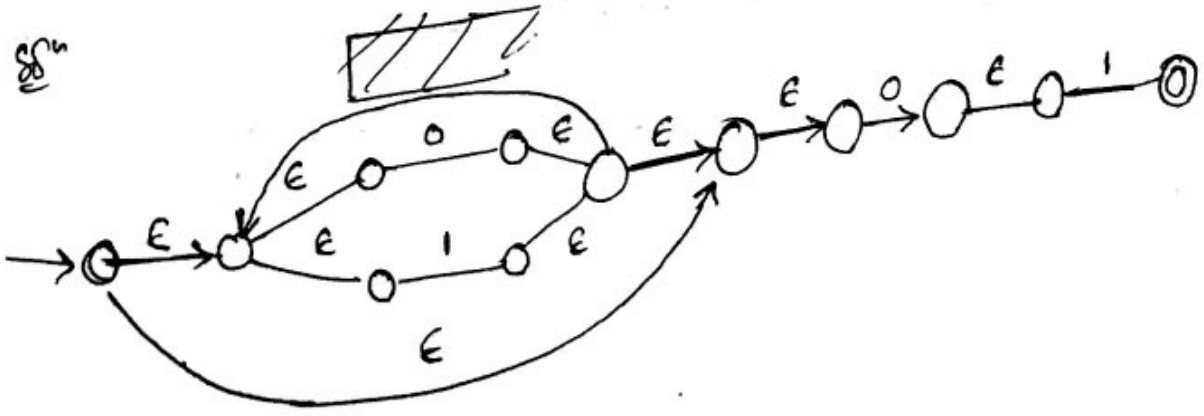
R_{22}^{2}

$i=2 j=2 k=2 = R_{22}^{1} + R_{22}^{1}\left(R_{22}^{1}\right)^{*} R_{22}^{1} = (0+1+\epsilon) + (0+1+\epsilon)^{*} = (0+1)^{*}$

$= (0+1+\epsilon)^{1} + (0+1+\epsilon)^{*} = (0+1)^{*}$

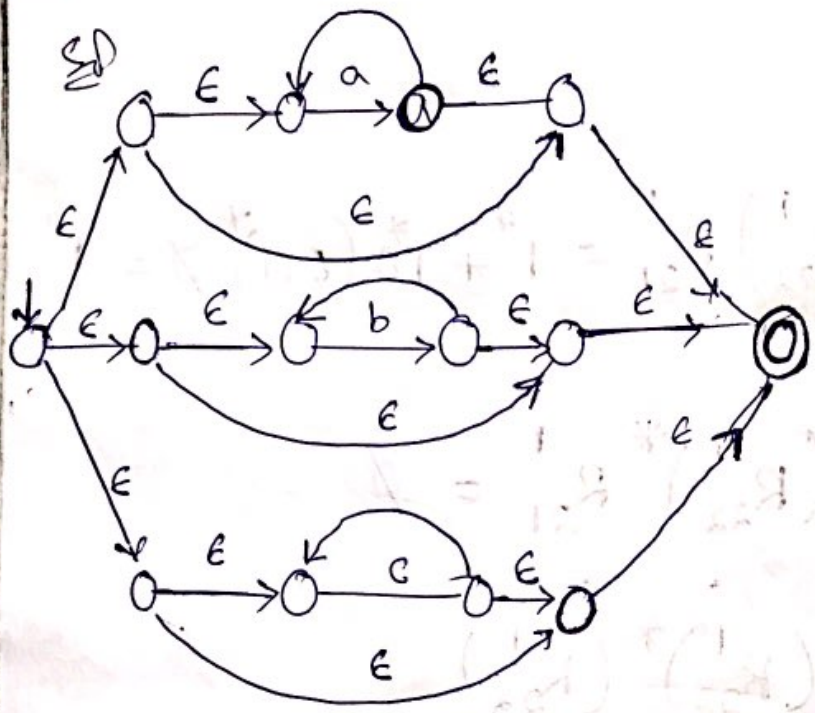
(Q) Construct NFA for $(0+1)^{*}01$

Sol^{n}



Q) $(a^* + b^* + c^*)$ construct NFA

Sol)



Applications of Regular Expressions

- R.E in unix
- Lexical Analysis
- Finding pattern in text

Algebraic laws in regular Exps

i) $L+M = M+L$ — Commutative law for Union

ii) $(L+M)+N = L+(M+N)$ — Associative law for Union

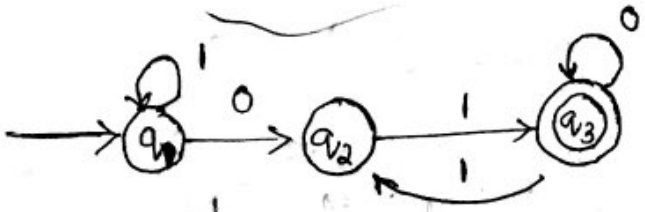
iii) $(LM)N = L(MN)$ - Associative law of Concatenation

iv)

Identities and Annihilator

1) $\emptyset + L = L + \emptyset$ (This law asserts that $\emptyset$ is the identity for union)
2) $\epsilon L = L\epsilon = L$ (" " " ϵ " for concatenation)
3) $\emptyset L = L\emptyset = \emptyset$ (" " " $\emptyset$ "annihilator for concatenation")

Laws involving closures

1) $(L^*)^* = L^*$
2) $(\emptyset)^* = \epsilon$
3) $\epsilon^* = \epsilon$
4) $L^+ = LL^* = L^*L$
5) $L^* = L^+ + \epsilon$
6) $L? = \epsilon + L$
7) $(L+M)^* = (L^*M^*)^*$
8) 

Convert DFA to R.E

Sol $R_{13}^{(3)} = R_{13}^{2} + R_{13}^{2}\left(R_{33}^{2}\right)^* R_{33}^{2}$

$i=1; j=3, K=3$

$$R_{13}^{2} = R_{13}^{1} + R_{12}^{1}\left(R_{22}^{1}\right)^{*} R_{23}^{1}$$

$i=1; j=3, k=2$

$\underline{k=0}$: $R_{11}^{0} = (\epsilon+1)^{*}$; $R_{12}^{0} = 0$, $R_{21}^{0} = \emptyset$, $R_{22}^{0} = \emptyset$

$R_{23}^{0} = 1$; $R_{32}^{0} = 1$, $R_{33}^{0} = 0^{*}$, $R_{13}^{0} = \emptyset$, $R_{31}^{0} = \emptyset$

$\underline{k=1}$

$$R_{11}^{1} = R_{11}^{0} + R_{11}^{0}\left(R_{11}^{0}\right)^{*}\left(R_{11}^{0}\right) = (\epsilon+1)^{*} = 1^{*}$$

$i=1; j=1; k=1$

$$R_{12}^{1} = R_{12}^{0} + R_{11}^{0}\left(R_{11}^{0}\right)^{*} R_{12}^{0} = 0 + (\epsilon+1)(\epsilon+1)^{*}0$$

$i=1; j=2; k=1$

$$= 0 + (\epsilon+1)^{*}0 = 1^{*}0$$

$$R_{13}^{(1)} = R_{13}^{0} + R_{11}^{0}\left(R_{11}^{0}\right)^{*}\left(R_{13}\right)^{0}$$

$i=1; j=3; k=1$ $= \emptyset + \emptyset = \emptyset$

$$R_{21}^{(1)} = R_{21}^{0} + R_{21}^{0}\left(R_{11}^{0}\right)^{*} R_{11}^{0}$$

$i=2; j=1; k=1$ $= \emptyset + \emptyset(\ ) = \emptyset$

$$R_{22}^{(1)} = R_{22}^{0} + R_{21}^{0}\left(R_{11}^{0}\right)^{*} R_{12}^{0} = \emptyset + \emptyset = \emptyset$$

$i=2; j=2; k=1$

$$R_{23}^{(1)} = R_{23}^{0} + R_{21}^{0}\left(R_{11}^{0}\right)^{*} R_{13}^{0} = 1 + \emptyset = 1$$

$i=2; j=3; k=1$

$R_{31}^{1} = R_{31}^{0} + R_{31}^{0}(R_{11}^{0})^{*} R_{13}^{0} = \phi + \phi = \phi$

$i=3; j=1; k=1$

$R_{32}^{1} = R_{32}^{0} + R_{31}^{0}(R_{11}^{0})^{*} R_{12}^{0} = 1 + \phi = 1$

$i=3; j=2; k=1$

$R_{33}^{(1)} = R_{33}^{0} + R_{31}^{0}(R_{11}^{0})^{*} R_{13}^{0} = 0 + \phi = 0$

$i=3; j=3; k=1$

$k=2$

$R_{13}^{(2)} = R_{13}^{1} + R_{12}^{1}(R_{22}^{1})^{*} R_{23}^{1} = \phi + 1^{*}0(\phi)^{*}\phi$

$i=1; j=3; k=2$

$= (1^{*}0)\epsilon 1 = 1^{*}01$

$R_{33}^{2} = R_{33}^{1} + R_{32}^{(1)}(R_{22}^{(1)})^{*}(R_{23}^{1})$

$i=3; j=3; k=2$ $= 0 + 1(\phi)^{*}1 = 11+0$

* $R_{13}^{(3)} = R_{13}^{2} + R_{13}^{2}(R_{33}^{2})^{*} R_{33}^{2}$

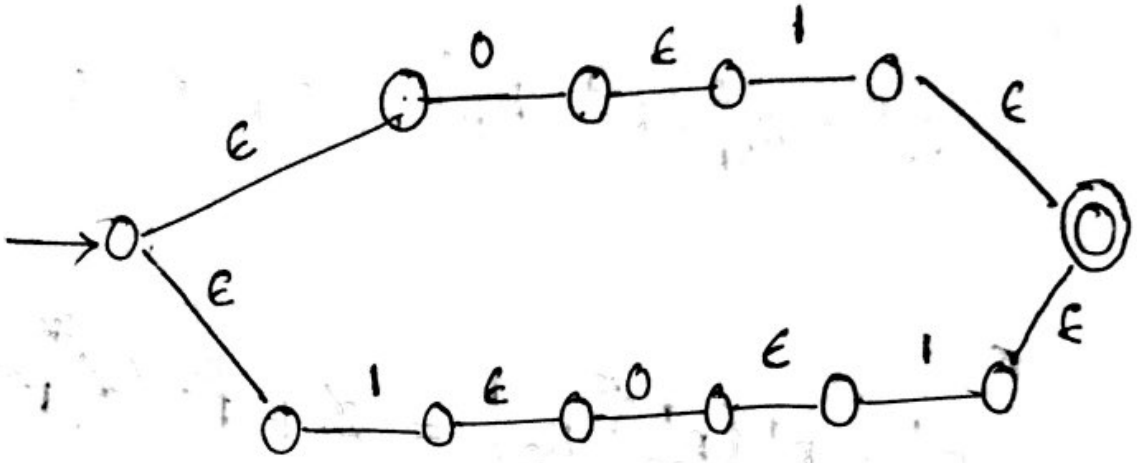
$= 1^{*}01 + (1^{*}01)(0+11)^{*}(0+11)$

$= (1^{+}01) + (1^{*}01)(0+11)^{*} = (1^{*}01)(0+11)^{*}$

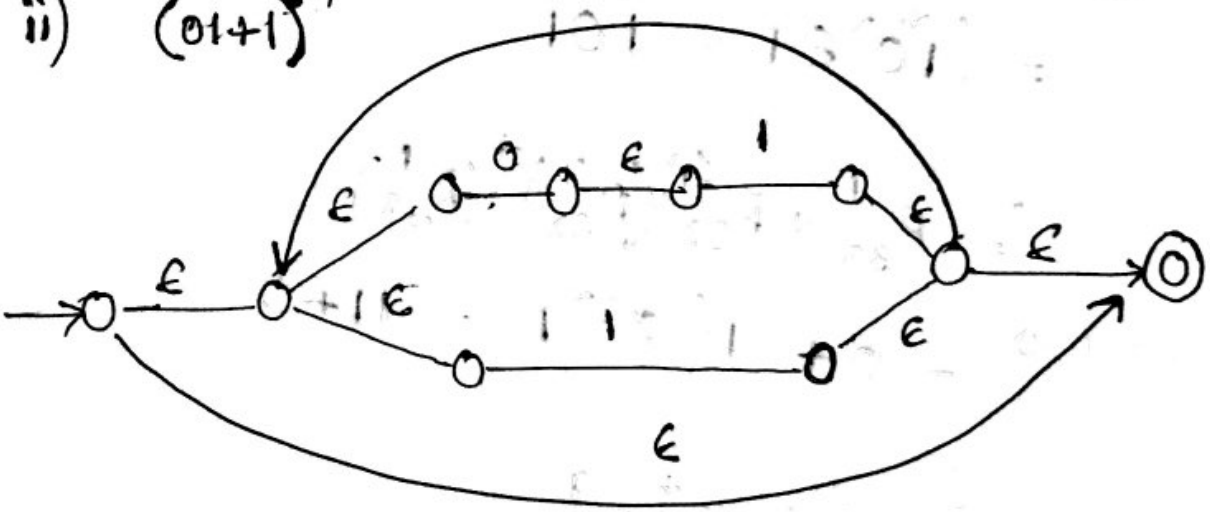
$R_{13}^{3} = (1^{*}01)(0+11)^{*}$

9) Convert R.E to ε-NFA

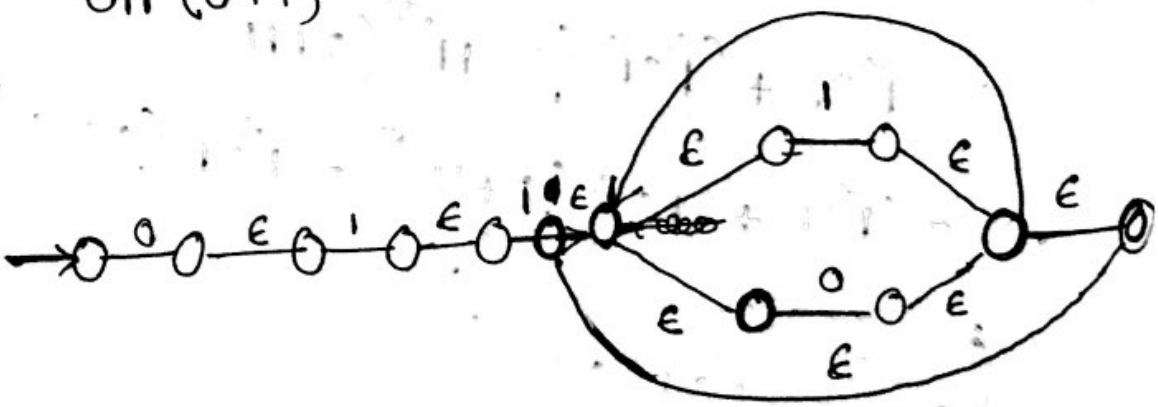
i) 01+101



ii) $(01+1)^*$



iii) $011(0+1)^*$



1Q) Show that $(0^*1^*)^* = (0+1)^*$

Sol" L.H.S $= (0^*1^*)^* = \{\epsilon, 0, 1, 00, 11, 01, 10, 001, 110, \}$

R.H.S $= (0+1)^* = \{\epsilon, 0, 1, 00, 11, ------ \}$

$\therefore$ L.H.S = L.H.S

2Q) $(ab)^* \neq a^*b^*$

Sol" L.H.S $= (ab)^* = \{\epsilon, ab, abab, ababab, abababab - - \}$

R.H.S $= (a^*b^*) = \{\epsilon, a, b, ab, aab, abb - - - \}$

3Q) S.T $(r+s)^* \neq (r^* + s^*)$

Sol" L.H.S $= (r+s)^* = \{\epsilon, r, s, rr, rs, sr, ss, - - \}$

R.H.S $= (r^* + s^*) = \{\epsilon, r, s, rr, ss - - \}$

Q4) prove $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1) = 1^*0(0+10^*1)^*$

Sol" L.H.S $= (1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$

$= (1+00^*1)(\epsilon + (0+10^*1)^*(0+10^*1))$

$= (1+00^*1)(0+10^*1)^*$

$= 1(\epsilon + 00^*)(0+10^*1)^*$

$= 10^*(0+10^*1)^*$

5Q) $\epsilon + (1^*(011)^*)(1^*(011)^*)^* = (1+011)^*$

Sol^n $= \epsilon + (1^*(011)^*)(1^*(011)^*)^*$ $[\epsilon + xx^* = x^*]$

$= (1^*(011)^*)^*$ $= (1+011)^*$ $[R^*S^* = (R+S)^*]$

Properties of Regular Languages

1) — pumping lemma :- To prove the language regular expression is not regular.

Theorem :- pumping lemma for regular languages

Let 'L' be regular language then there exist a constant 'n' which depends on 'L' such that for every string 'w' in 'L' such that the length of 'w' $\geq n$ then we can break 'w' into three strings $w = xyz$

Such that :

1) $y \neq \epsilon$ 2) length $(xy) <= n$

3) For all $k \geq 0$ the string xy^kz is also in L

Q) Perfect Square $= \{0, 1, 4, 9, 25, 36, 49, 64, \ldots\}$

$= \{0, 10, 100, 10000, \ldots\}$ (In binary)

Sol^n $10000 - 16$

$(x\ y\ z)$ $\xrightarrow{k=2}$ $(x\ y^2 z)$

$1|0000|0 \Longrightarrow 1|00000000|0\ \#_L$

$x\ \ y\ \ z$

→ Its not a regular language.

8) $\{0^n / n$ is a perfect cube $\}$

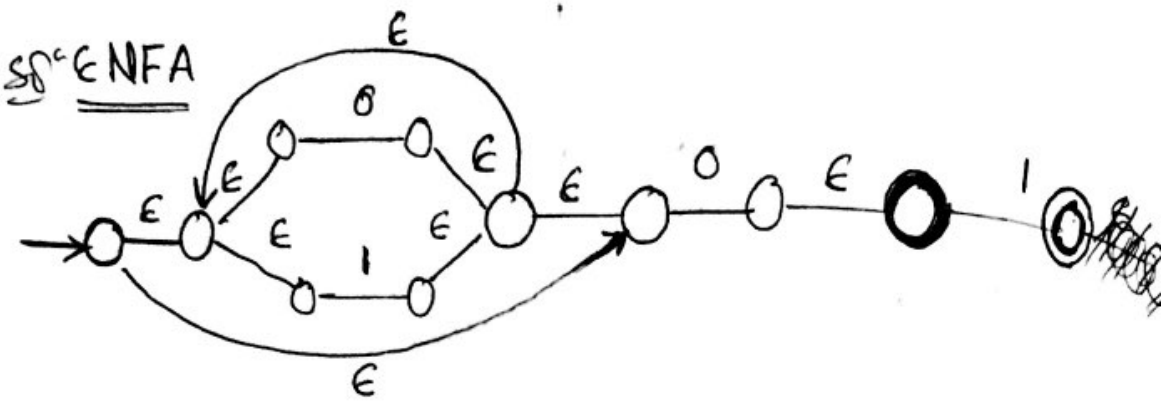
$= \{0, 1, 1000, \text{---}\}$

2) Closure properties of regular languages

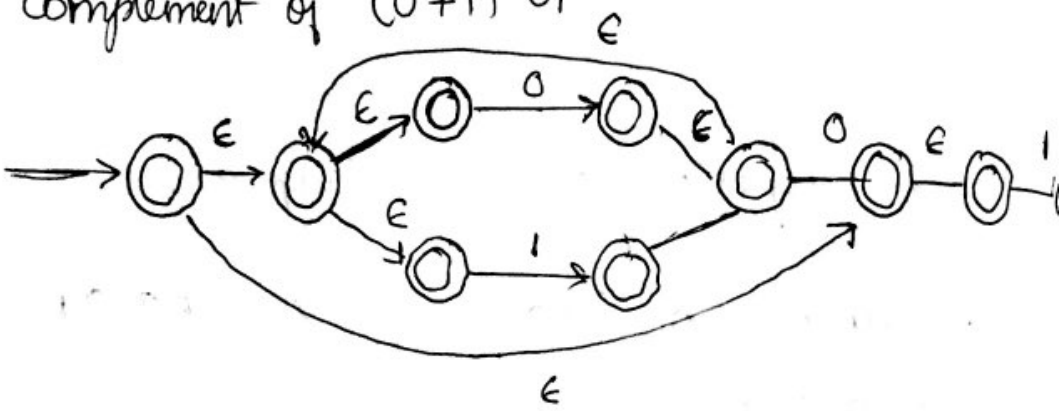
1. The union of two regular language is regular.
2. The intersection " " " "
3. The complement of a regular language is regular.
4. The diff. of two regular language is regular.
5. The reversal of a reg. language is regular.
6. The closure (star) of reg. lang is regular.
7. The concatenation " " " "
8. A homomorphism (Substitution of string for Symbol) of reg lang is regular.
9. The inverse homomorphism " " "

9) $(0+1)^*01$

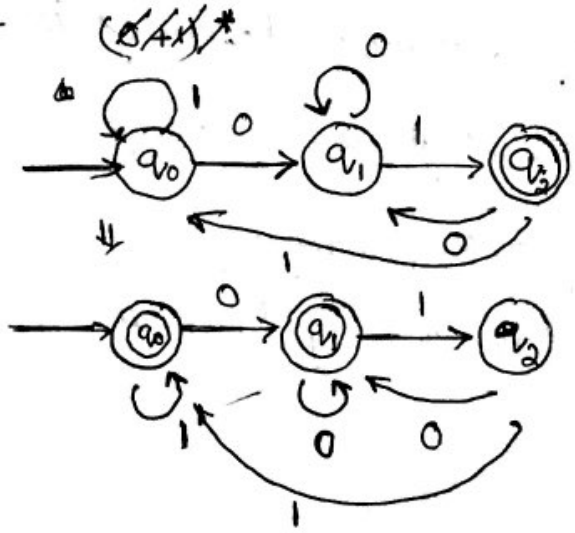
Soln: ∈ NFA



Complement of $(0+1)^*01$



DFA



$\Rightarrow$ Final States = $R_{11}^{(3)} + R_{12}^{(3)}$

$$R_{11}^{(3)} = R_{11}^{2} + R_{13}^{2}\left(R_{33}^{2}\right)^{*} R_{31}^{2}$$

$$R_{12}^{(3)} = R_{12}^{2} + R_{13}^{2}\left(R_{33}^{2}\right)^{*} R_{32}^{2}$$

$K=0$

$R_{11}^{(0)} = \emptyset(1+\epsilon)^* R_{12}^{(0)} = 0 \; ; \; R_{13}^{0} = \emptyset \; ; \; R_{21}^{(0)} = \emptyset$

$R_{22}^{0} = (0+\epsilon)^* \; ; \; R_{23}^{0} = 1 \; ; \; R_{31}^{0} = 1 \; , R_{32}^{0} = 0$

$R_{33}^{0} = \emptyset$

$K=1$

$R_{11}^{1} = R_{11}^{0} + R_{11}^{0}(R_{11}^{0})^* R_{11}^{0} = (1+\epsilon)+(1+\epsilon)(1+\epsilon)^*(1+\epsilon)$

$= 1^*$

$R_{12}^{1} = R_{12}^{0} + R_{11}^{0}(R_{11}^{0})^* R_{12}^{0} = 0 + (1+\epsilon)^*((1+\epsilon)^*)^* 0$

$= 0 + (1+\epsilon)^* 0 = 1^* 0$

$R_{13}^{1} = R_{13}^{0} + R_{11}^{0}(R_{11}^{0}) R_{13}^{0} = \emptyset + \emptyset = \emptyset$

$R_{21}^{1} = R_{21}^{0} + R_{21}^{0}(R_{11}^{0})^* R_{11}^{0} = \emptyset + \emptyset(1+\epsilon)^* = \emptyset$

$R_{22}^{1} = R_{22}^{0} + R_{21}^{0}(R_{11}^{0})^* R_{12}^{0} = (0+\epsilon)^* + \emptyset = 0^*$

$R_{23}^{1} = R_{23}^{0} + R_{21}^{0}(R_{31}^{0})^* R_{13}^{0} = 1 + \emptyset = 1$

$R_{31}^{1} = R_{31}^{0} + R_{31}^{0}(R_{11}^{0})^* R_{11}^{0} = 1 + 1((1+\epsilon)^*)^*(1+\epsilon)^*$

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

$R_{32}^{1} = R_{32}^{0} + R_{31}^{0}(R_{11}^{0})^* R_{12}^{0} = 0 + 1(1+\epsilon)^* \emptyset = 0 + 1^* 0$

$R_{33}^{1} = R_{33}^{0} + R_{31}^{0}(R_{31}^{0})^* R_{13}^{0} = \emptyset + 1(1+\epsilon)^* \emptyset = \emptyset$

$R_{11}^{(2)} = R_{11}^{1} + R_{12}^{1}(R_{22}^{1})^{*} R_{21}^{1} = 1^{*} + 1^{*}0(0^{*})^{*}\emptyset = 1^{*}$

$R_{13}^{2} = R_{13}^{1} + R_{12}^{1}(R_{22}^{1})^{*} R_{23}^{1} = \emptyset + 1^{*}0(0^{*})^{*}1$

$= 1^{*}00^{*}1$

$R_{33}^{2} = R_{33}^{1} + R_{32}^{1}(R_{22}^{1})^{*} R_{23}^{1} = \emptyset + (0+11^{*}0)(0^{*})1$

$= (0+11^{*}0)0^{*}1$

$R_{31}^{2} = R_{31}^{1} + R_{32}^{1}(R_{22}^{1})^{*} R_{21}^{1} = 1(1^{*}) + (0+11^{*}0)(0^{*})$

$= 1(1^{*})$

$R_{12}^{2} = R_{12}^{1} + R_{12}^{1}(R_{22}^{1})^{*} R_{22}^{1} = 1^{*}0 + 1^{*}0(0^{*})0^{*}$

$= 1^{*}0 + 1^{*}00^{*} = 1^{*}00^{*}$

$R_{32}^{2} = R_{32}^{1} + R_{32}^{1}(R_{22}^{1})^{*} R_{22}^{1} = (0+11^{*}0) + (0+11^{*}0)$
$(0^{*})^{*}0^{*}$

$= (0+11^{*}0) + (0+11^{*}0)0^{*} = (0+11^{*}0)0^{*}$

$R_{11}^{3} = 1^{*} + (1^{*}00^{*}1)((0+11^{*}0)0^{*}1)^{*}(1(1^{*}))$

$R_{12}^{3} = (1^{*}00^{*}) + (1^{*}00^{*}1)((0+11^{*}0)0^{*}1)^{*}((0+11^{*}$

Complementation

If L is regular language over alphabet Σ then

$\bar{L} = \Sigma^* - L$ is also a regular language

$L = L(A)$

DFA $A = (Q, \Sigma, \delta, q_0, F)$ then

$\bar{L} = L(B) = (Q, \Sigma, \delta, q_0, Q - F)$

Intersection

$L \cap M = \overline{\bar{L} \cup \bar{M}}$

If L and M are regular languages then

$L \cap M$ is also a regular language.

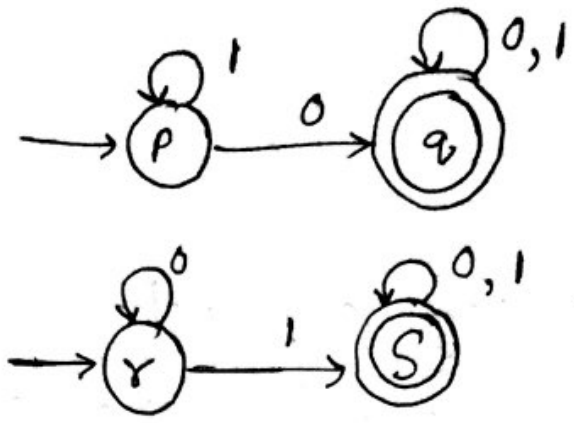
$A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$

$A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$

$* A = (Q_L \times Q_M, \Sigma, \delta, (q_L, q_M), F_L \times F_M)$

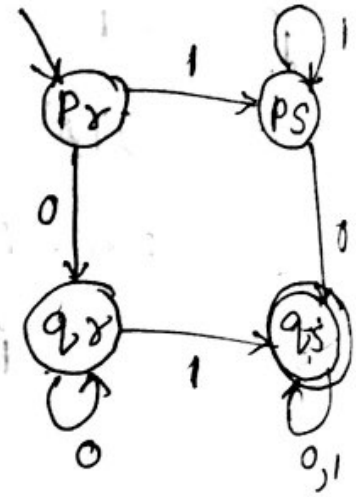
$\delta((p, q), a) = (\delta_L(p, a), \delta_M(q, a))$

Ex :-



$\delta(pr,0) = \delta(p,0), \delta(r,0) = (q,r)$

$\delta(pr,1) = \delta(p,1), \delta(r,1) = (ps)$



$\delta(ps,0) = \delta(p,0), \delta(s,0) = (q,s)$

$\delta(ps,1) = \delta(p,1), \delta(s,1) = ps$

$\delta(qr,0) = (q,r)$; $\delta(qs,0) = qs$

$\delta(qr,1) = (q,s)$ $\delta(qs,1) = qs$

Q) Substring is ab & starting with ab.

Ex: Starting with ab — $ab(a+b)^*$ — $\{ab, aba, abb, abaa\}$

Substring — $(a+b)^*ab(a+b)^*$ — $\{ab, aaba, aba\}$

$ab(a+b)^* \cap (a+b)^*ab(a+b)^* = ab(a+b)^*$

L-M: Set Difference

The set of strings in language L but not in language M.

If L and M are regular languages then so is

$$L - M = L \cap \bar{M}$$

Reversal

The reversal of a regular language is also a regular language.

Ex. $L = \{00, 1100, 001, 1101\}$

$L^R = \{00, 0011, 100, 1011\}$

$\epsilon^R = \epsilon$

$\emptyset^R = \emptyset$

$a^R = a$

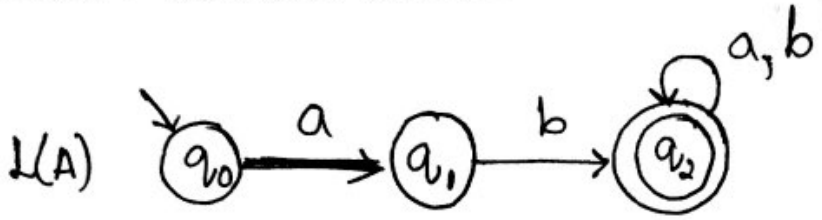
Given a language 'L' that is L(A) for some finite automaton with non deterministic & epsilon transaction may construct an automata for L^R by.

1. Reverse all the arcs in the transition diagram for A
2. Make the start state of A be the or

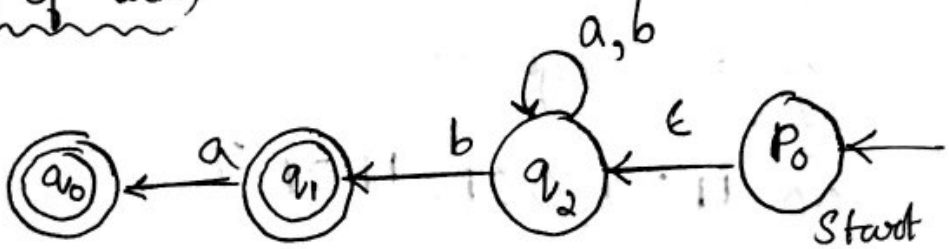
accepting state for the new automaton

3. Create a new start state P_0 with transition on ϵ to all the accepting state of A.

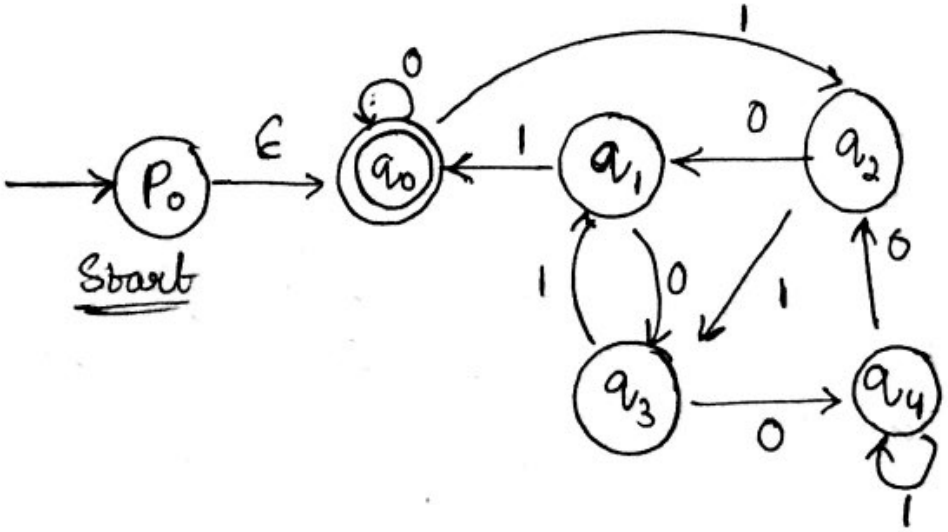
Ex: String starting with 'ab'



Reverse of L(A)



Q) Divisible by '5'.



Theorem: If L is a regular language so is L^R is also a regular language

$$L = L(E) \qquad L^R = L(E^R)$$

$$L(E^R) = (L(E))^R$$

Basis: If E is ϵ, $\emptyset$ or ~~$L(E^R) \neq L$, $L(E^R) = (L(E))^R$~~

a for symbol a

E^R is same as E

$$\{\epsilon\}^R = \{\epsilon\}$$

$$\emptyset^R = \emptyset$$

$$\{a\}^R = a$$

Induction: $E = E_1 + E_2$

$$E^R = E_1^R + E_2^R$$

Concatenation:

$$E = E_1 E_2$$

$$E^R = E_2^R E_1^R$$

$$L(E_1) = \{01, 111\}$$

$$L(E_2) = \{00, 10\}$$

$L(E_1E_2) = \{0100, 0110, 11100, 11110\}$

$L(E_2)^R = \{00, 01\}$; $L(E_1)^R = \{10, 111\}$

$L(E_1E_2)^R = \{0010, 00111, 0110, 01111\}$

$\therefore L(E_1E_2)^R = L(E_2)^R L(E_1)^R \cdot L(E_2^R E_1^R)$

closure

Homomorphism :- If 'L' is a Regular language ove alpha 'Σ' . 'h' is homomorphism on 'Σ' the h(L) is also regular.

L - Language over Σ ; h - homomorphism on then $h(L) = \{h(\omega) / \omega$ is in $L\}$

Union $L(h(E)) + L(h(R)) = h(E) + h(R)$

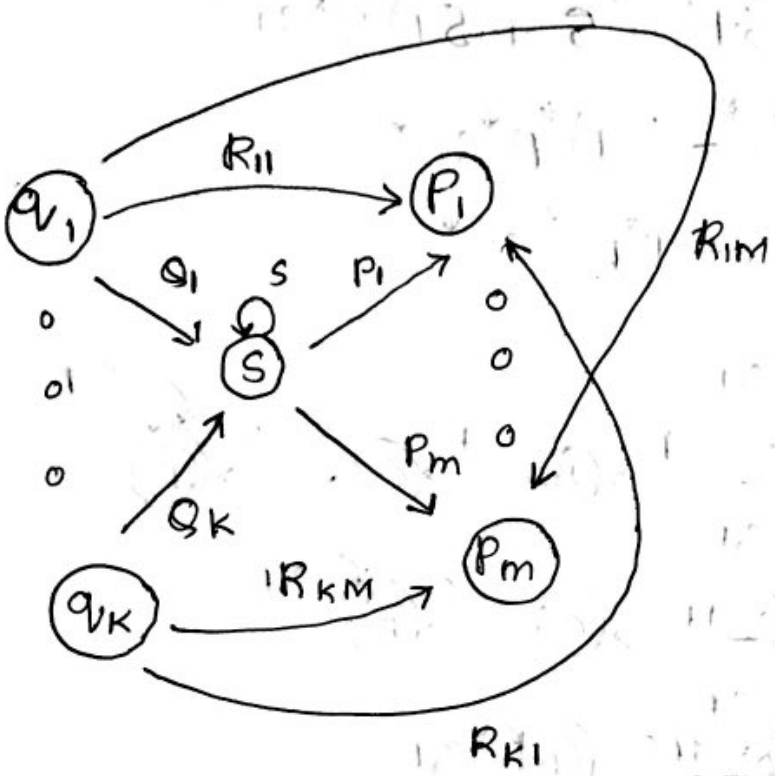
Concatenation $L(h(E)) \cdot L(h(R)) = h(E)h(R)$

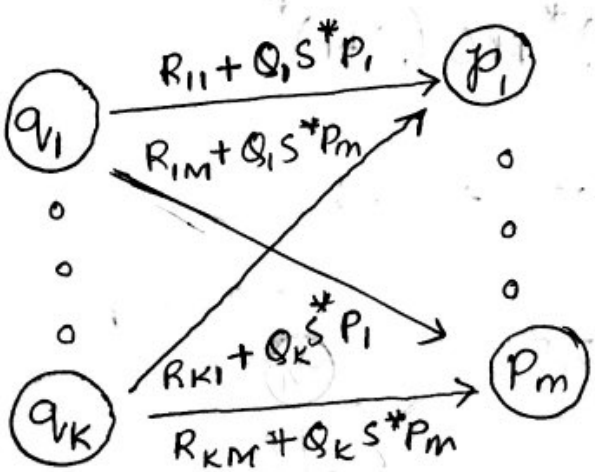
Decision

FA to Regular Exp^n with elimination of states

DFA $\longrightarrow$ R.E

n-state — time Complexity = n^3



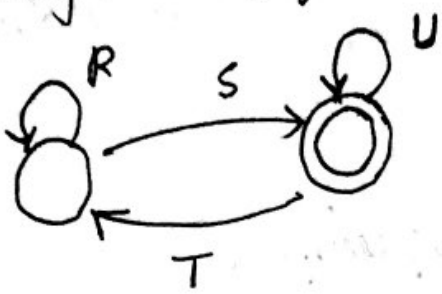


1> accepting state, start state is same

— then it contains only single Reg. Exp

min

2) accepting state, final state are different — min 2 states

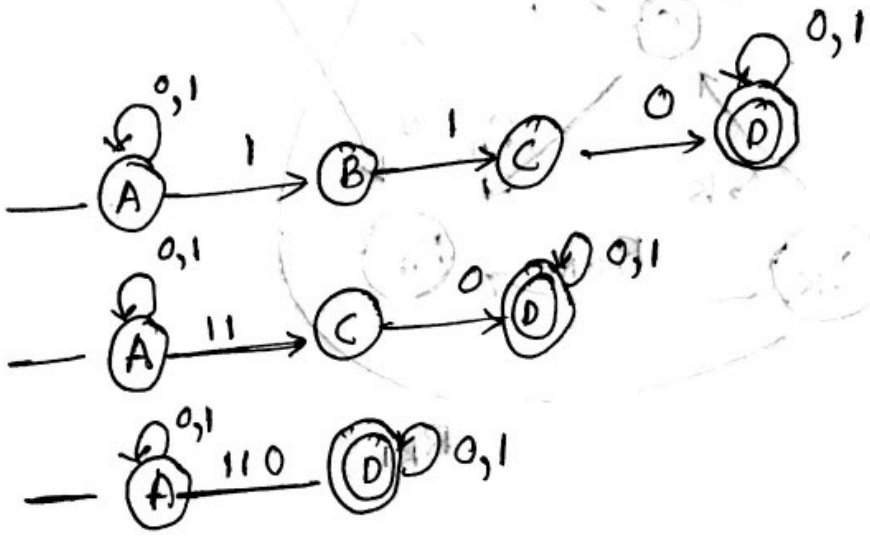


$$R.E = R^*S + (ST)^*S + (ST)^*R^*SU^*$$

$$= R^*S + (ST)^*R^*SU^*$$

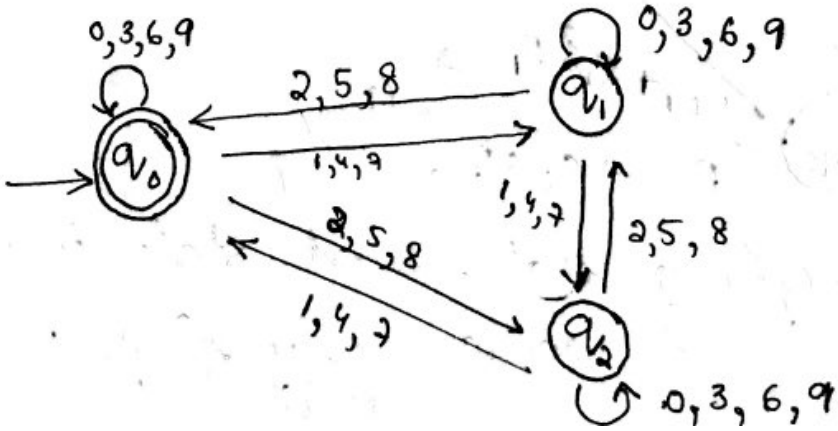
$$= (R + SU^*T)^*SU^*$$

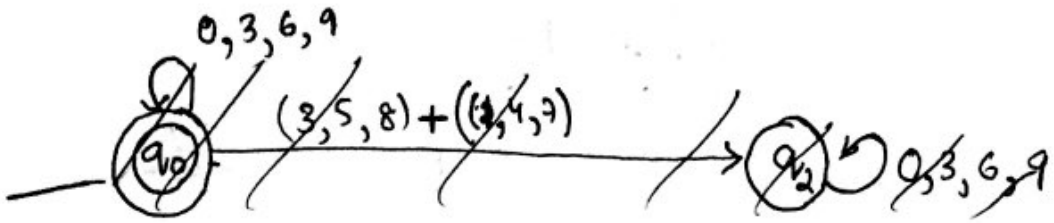
Ex.

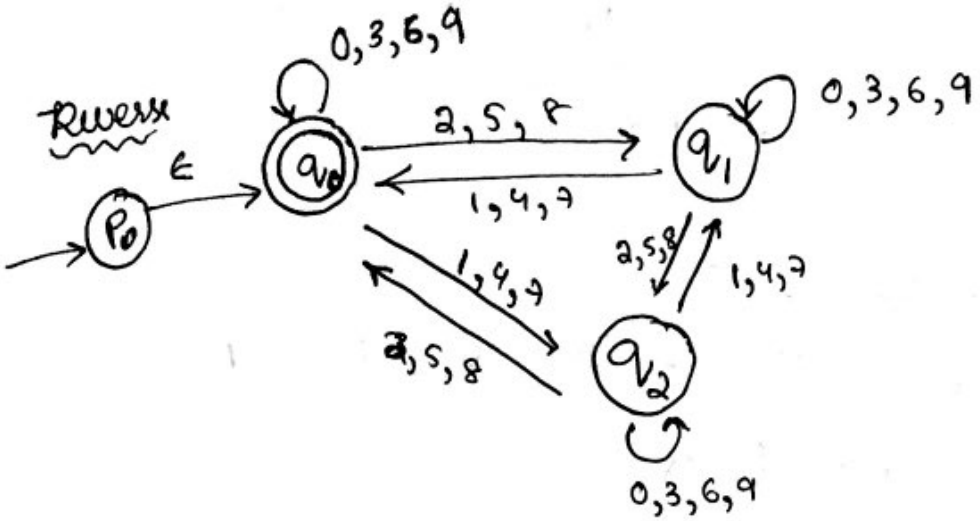


$$= (0+1)^*110(0+1)^*$$

3) Binarey Number ÷ by 3

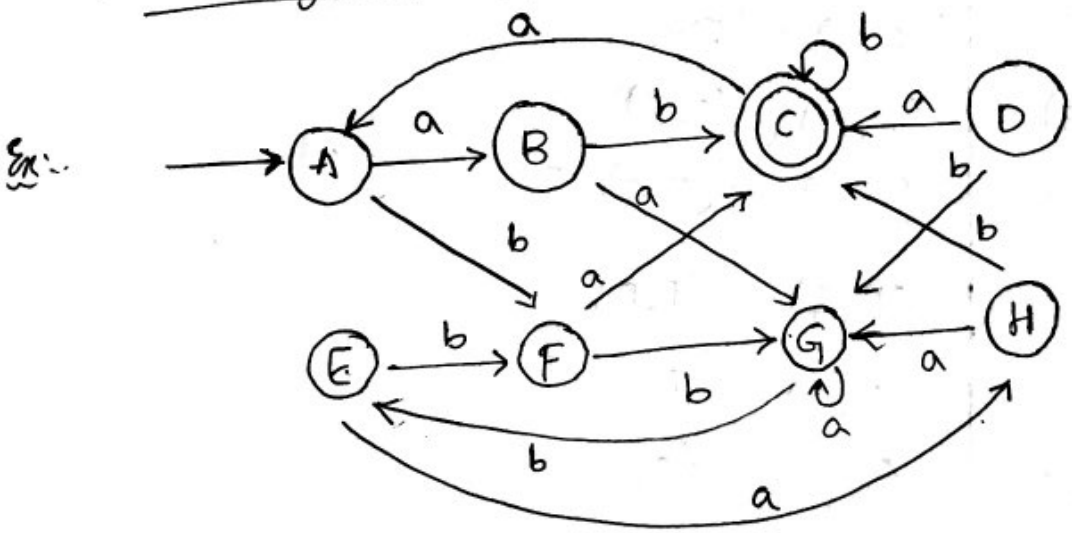






Minimization of DFA

Ex:-



	a	b
→ A	B	F
B	G	C
* C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

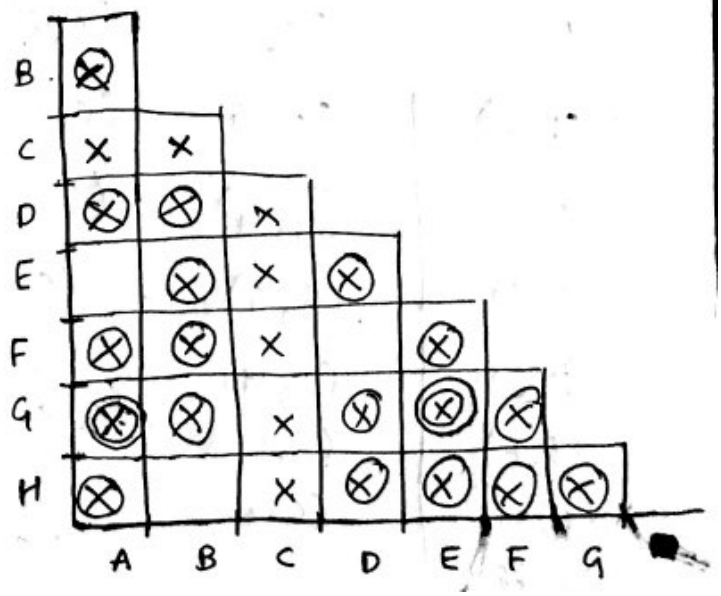


Table Filling Algorithm

→ For each pair (p, q) if $p \in F$ and $q \notin F$ (or) vice versa then the pair (p, q) is distinguish and Mark that pair (p, q) using X.

Accepting State – C.
Non Accepting State – A, B, D, E, F, G, H

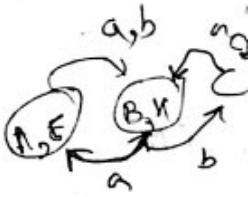
Indistenguishable States

distinguishable	a	b	distinguishable
× (A, B)	(B, G)	(F, C)	→ distinct
× (A, D)	(B, C)	(F, G)	
(A, E)	(B, H)	(F, F)	
× (A, F)	(B, C)	(F, G)	
(A, G)	(B, G)	(F, E)	
× (A, H)	(B, G)	(F, C)	
× (B, D)	(G, C)	(C, G)	
× (B, E)	(G, H)	(C, F)	
× (B, F)	(G, C)	(C, G)	
× (B, G)	(G, G)	(C, E)	
(B, H)	(G, G)	(C, C)	
× (D, E)	(C, H)	(G, F)	
(D, F)	(C, C)	(G, G)	
× (D, G)	(C, G)	(G, E)	
× (D, H)	(C, G)	(G, C)	

x (E,F)	(H,C)	(F,G)
(E,G)	(H,G)	(F,E)
x (E,H)	(H,G)	(F,C)
x (F,G)	(C,G)	(G,E)
x (F,H)	(C,G)	(G,C)
x (G,H)	(G,G)	(E,C)

Level 2

(A,E)	(B,H)	(F,F)
x (A,G)	(B,G)	(F,E)
(B,H)	(G,G)	(C,C)
(D,F)	(C,C)	(G,G)
(X) (E,G)	(H,G)	(F,E)



Level 3

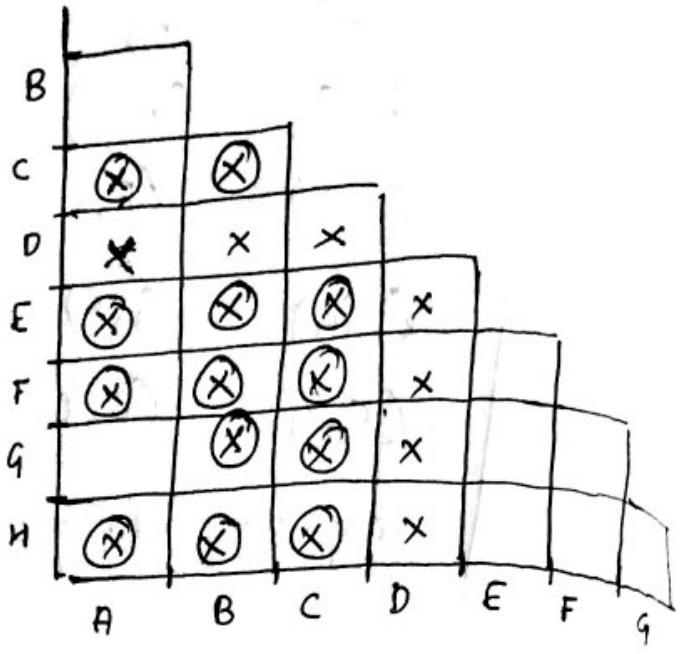
(A,E)	(B,H)	(F,F)
(B,H)	(G,G)	(C,C)
(D,F)	(C,C)	(G,G)

} Indistinguishable States

States: (A,E), C, (B,H), G, (D,F)

Ex 2

		0	1
	A	B	A
	B	A	C
	C	D	B
*	D	D	A
	E	D	F
	F	D	E
	G	F	G
	H	G	D



Level 1	0	1
(A, B)	(B, A)	(A, C)
✓ (A, C)	(B, D)	(A, B)
✓ (A, E)	(B, D)	(A, F)
✓ (A, F)	(B, D)	(A, E)
(A, G)	(B, F)	(A, G)
✓ (A, H)	(B, G)	(A, D)
✓ (B, C)	(A, D)	(C, B)
✓ (B, E)	(A, D)	(C, F)
✓ (B, F)	(A, D)	(C, E)
✓ (B, G)	(A, F)	(C, G)
✓ (B, H)	(A, G)	(C, D)
✓ (C, E)	(D, D)	(B, F)
✓ (C, F)	(D, D)	(B, E)
✓ (C, G)	(D, F)	(B, G)
✓ (C, H)	(D, G)	(B, D)

	0	1
(E, F)	(D, D)	(F, E)
(E, G)	(D, F)	(F, G)
(E, H)	(D, G)	(F, D)

5. Context – Free Grammar

Context-free Language → CFG (algebraic)
→ Pushdown Auto (machine)

CFG :- (Application)

- Syntax Analysis
- Parser (II nd phase of Compiler Design)
- DTD (in XML)

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

Context Free Grammar

1) For even palindrome

Sol^n W = 1001 ; For Palindrome ($W = W^R$)

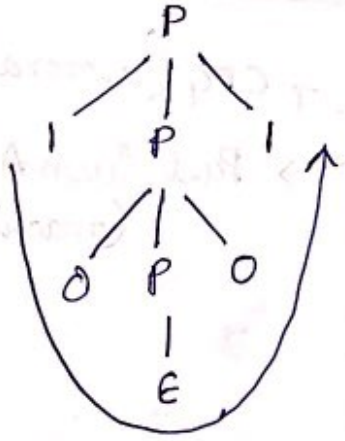
~~G = (V, T, P, S)~~

$1001 = W^R$

(*) P → ∈/0/1/0P0/1P1

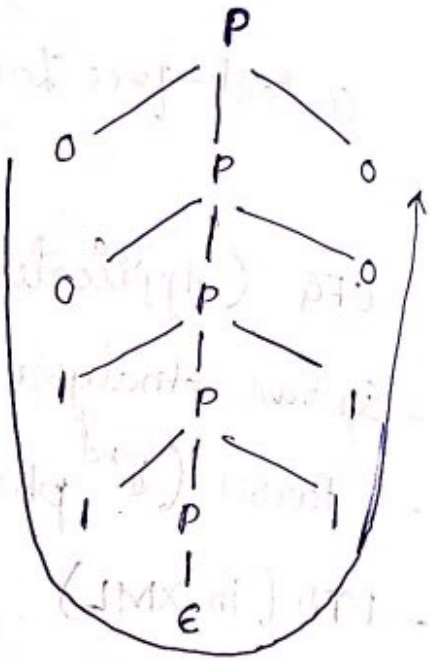
P → ∈	P → 0P0
P → 0	P → 1P0
P → 1	

① 1001



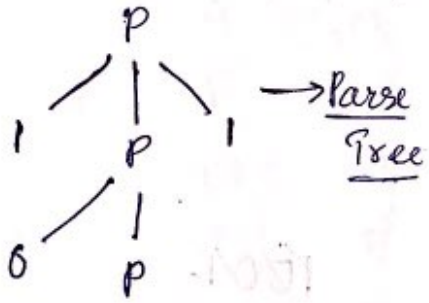
→ 1001 is palindrom

② 00111100



⇒ 00111100 is palindrom

③ 1011



→ Parse Tree

Content Free language

Content Free languages have a natural, recursive notation called Content Free Gramm

Definition

There are 4 component in the grammat description of language.

→ Finite set of symbols that form the stri

of the language being defined.
eg = {0, 1}

→ We called this alphabets as terminals (or) terminal symbols.

→ There is a finite set of variable also called non terminal (or) syntactic categories

→ Each variable represents a language i;e a set of strings.

Ex:- OPO

3.→ One of the variables represents the language being defined. It is called the start symbols.

→ Other variables represents auxilliary classes of string that are used to help define the language of start symbol

4.→ There is a finite set of products (or) rules that represent recursive definition of the language.

→a) Each production consist of a variable that is being partially defined

by the production. This variable is often called the head of the production

) The production symbol is $\longrightarrow$

) A string of zero (or) more terminals and variables this string called the body of the production, represents one way to represents form string in the language of the variable of the head

. We leave terminals unchanged & substitute for each variable of the body any string that is known to be the language of that variable.

1) arithematic expression :-

$E \longrightarrow Op1 + Op2 * Op3$

$E \longrightarrow id$

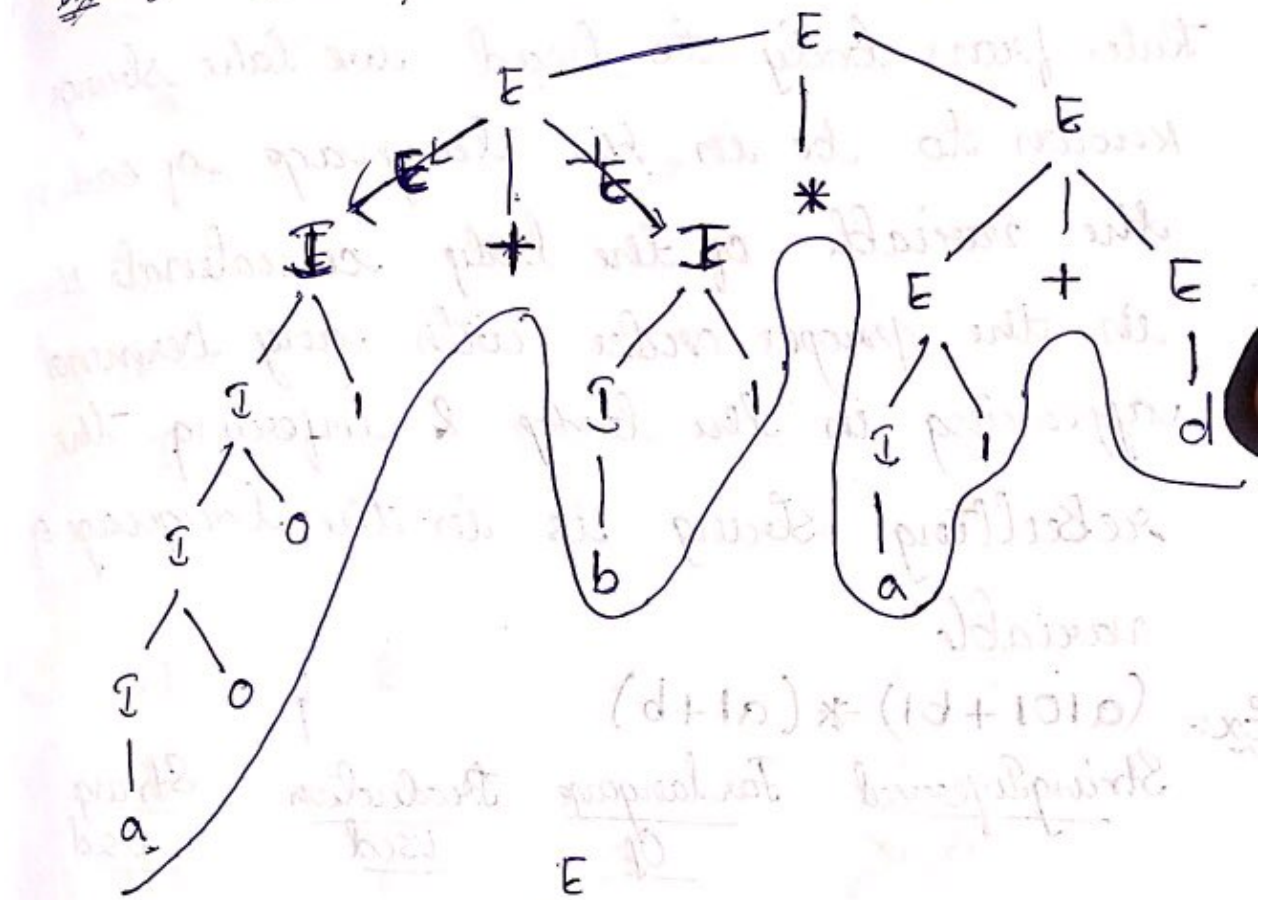
(Expression Grammar)

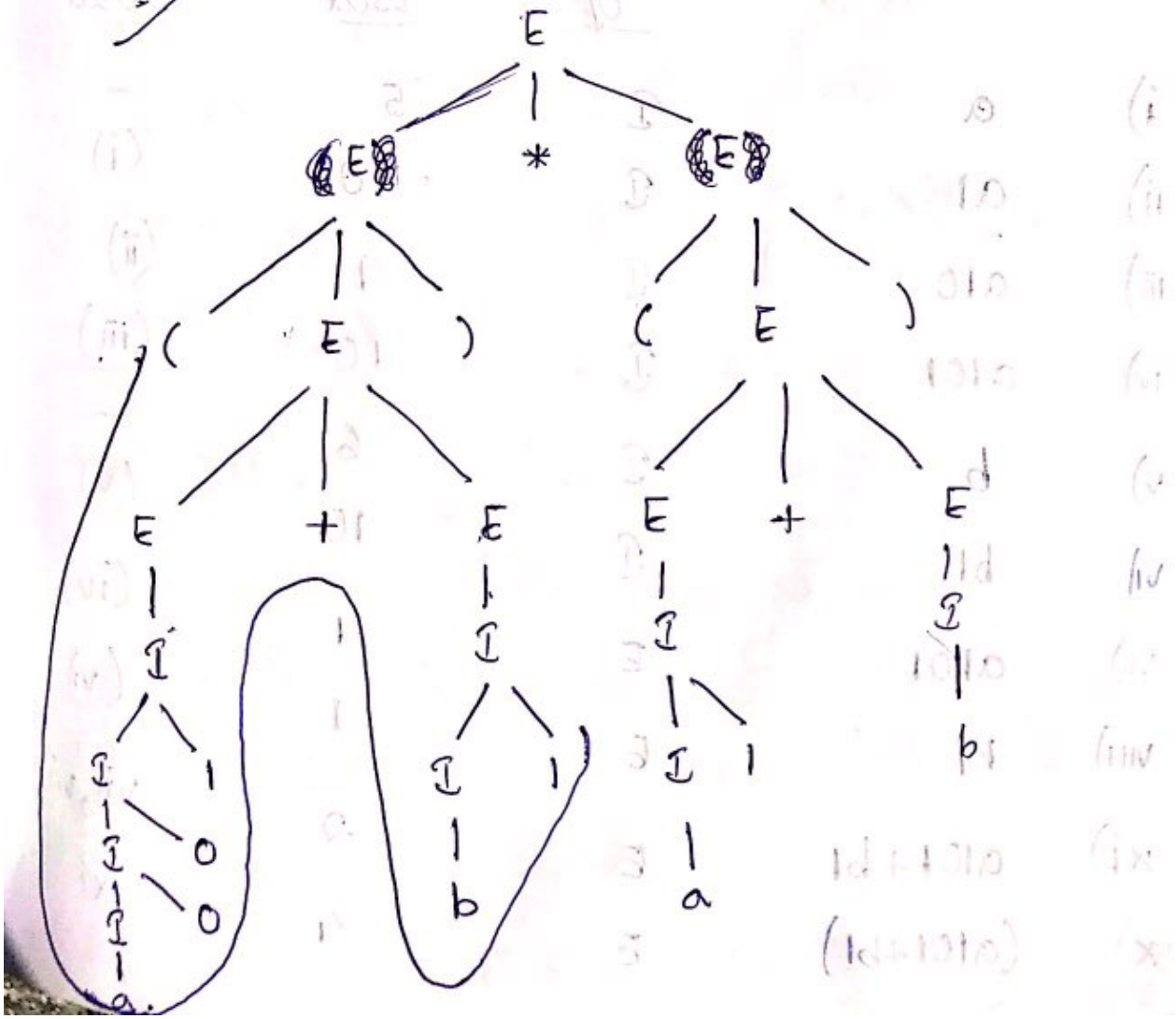
$E \longrightarrow I$
$E \longrightarrow E+E$
$E \longrightarrow E*E$
$E \longrightarrow (E)$

$I \longrightarrow a$
$I \longrightarrow b$
$I \longrightarrow Ia$
$I \longrightarrow Ib$
$I \longrightarrow I0$
$I \longrightarrow I1$

identifiers

Ex 1) (a101 + b1) * (a1 + b) — E





Rule of Inference :- (Recursive Inference)

Rules from body to head we take strings known to be in the language of each of the variable of the body concatenate them in the proper order with any terminal appearing in the body & inferring the resulting string is in the language of variable

Ex:- (a101 + b1) * (a1 + b)

	String Inferred	For language Of	Production Used	String Used
i)	a	I	5	–
ii)	a1	I	10	(i)
iii)	a10	I	9	(ii)
iv)	a101	I	10	(iii)
v)	b	I	6	–
vi)	b1	I	10	(v)
vii)	a|01	E	1	(iv)
viii)	b1	E	1	(vi)
xi)	a101 + b1	E	2	vii, viii
x)	(a101 + b1)	E	4	xi

Production

1. $E \rightarrow I$
2. $E \rightarrow E+E$
3. $E \rightarrow E*E$
4. $E \rightarrow (E)$
5. $I \rightarrow a$
6. $I \rightarrow Ib$
7. $I \rightarrow Ia$
8. $I \rightarrow Ib$
9. $I \rightarrow I0$
10. $I \rightarrow I1$

---

xi)	a1	E	1	ii
xii)	b	E	1	v
xiii)	a1+b	E	2	xi, xii
xiv)	(a1+b)	E	4	xiii
xv)	(a101+b1)* (a1+b)	E	3	x, xiv

Derivative Defenition

The process of deriving string by applying production from head to body requires the defenition of new relation symbol ($\Rightarrow$) derivation symbol.

* Suppose $G = (V, T, P, S)$ is a context free

Grammar. Let $\alpha A \beta$ be a string of terminal and variable with A as a variable.

→ let α, β are strings in $(V \cup T)^*$ and A in V:

→ Let $A \to \gamma$ be a production of G then we say $\alpha A \beta \underset{G}{\Longrightarrow} \alpha \gamma \beta$

→ $\alpha A \beta \Rightarrow \alpha \gamma \beta$ in one derivation step replaces any variable any where in the string by body of its production

Ex:- $(a|01 + b|) * (a| + b)$

$E \underset{lm}{\Rightarrow} E * E$ $(E \to E * E)$

$\Rightarrow (E) * E$ $(E \to (E))$

$\Rightarrow (E + E) * E$ $(E \to E + E)$

$\Rightarrow (I + E) * E$ $(E \to I)$

$\Rightarrow (I| + E) * E$ $(I \to I|)$

$\Rightarrow (I01 + E) * E$ $(I \to I0)$

$\Rightarrow (I101 + E) * E$ $(I \to I|)$

$\Rightarrow (a101 + E) * E$ $(I \to a)$

$\Rightarrow (a101 + I) * E$ $(E \to I)$

$\Rightarrow (a101 + I|) * E$ $(I \to I|)$

$\Rightarrow (a101 + b|) * E$ $(I \to b)$

$\Rightarrow$ (a|01+b|) * (E) (E → (E))

$\Rightarrow$ (a|01+b|) * (E+E) (E → E+E)

$\Rightarrow$ (a|01+b|) * (I+E) (E → I)

$\Rightarrow$ (a|01+b|) * (I1+E) (I → I1)

$\Rightarrow$ (a|01+b|) * (a1+E) (I → a)

$\Rightarrow$ (a|01+b|) * (a1+I) (E → I)

$\Rightarrow$ (a|01+b|) * (a1+b) (I → b)

$E \overset{*}{\underset{lm}{\Rightarrow}} (a|01+b|) * (a1+b)$

$\Longrightarrow$ Derivation Process

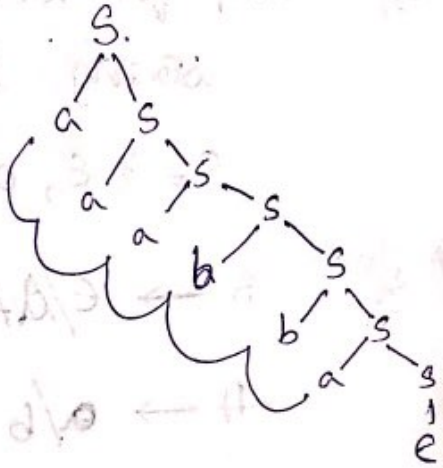
Q) CFG for a^* or any no. of a's

A) $S \rightarrow \epsilon / aS$

Q) CFG for any no. of a's & b's (or) $(a+b)^*$

$S \rightarrow \epsilon / aS / bS$

S - aaabba



Q) atleast one a (a+)

Sol: $S \rightarrow a/aS$

Q) Design context Free Grammar for atleast 2a

Sol: $\Sigma = \{a\}$

$S \rightarrow aa/aS$

Q) even no. of a's? ($S = \{\epsilon, aa, aaaa, \ldots\}$)

Sol: $S \rightarrow \epsilon/aaS$

Q) odd no. of a's? $S = \{a, aaa, aaaaa \ldots\}$

$S \rightarrow a/aaS$

Q) Multiples of 3. $S = \{\epsilon, aaa, aaaaaa \ldots\}$

$S \rightarrow \epsilon/aaaS$

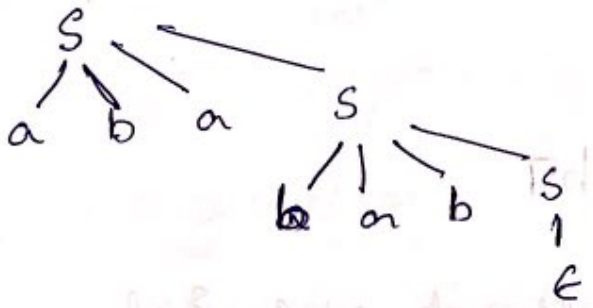
Q) CFG for strings of a's & b's such that string lenght is multiple of 3.

$S = \{\epsilon, abb, aab, aaa, bbb \ldots\}$

Sol: $S \rightarrow \epsilon/AAAS$

$A \rightarrow a/b$ (Non Terminals)

i) ababab



Q) Obtain CFG to generate string consisting of any no. of a's & b's with atleast one 'a' (or) one 'b'.

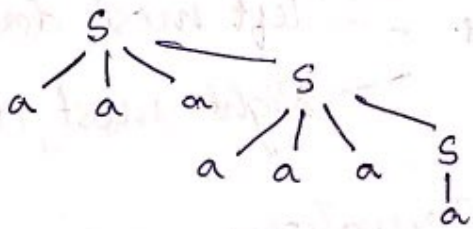
Sol^n $S \rightarrow a/b/aS/bS$

Q) Obtain CFG to accept the foll. language.

$L = \{ w : |w| \bmod 3 > 0$ where $w \in \{a\}^* \}$

Sol^n $S \rightarrow a/aa/aaaS$

(i) aaaaaaa



Q) Generate string of a's & b's having substring ab. $S = \{ab, aaab, aaabba, \ldots\}$

Sol^n $S \rightarrow ab/ASA$

$A \rightarrow aA/bA/\epsilon$

g) starting with ab ; $S = \{aba, abab, \ldots\}$

Sol: $S \Rightarrow ab/ST$

$T \rightarrow \epsilon, aT, bT$

g) ending with ab ; $S = \{ab, aaab, abab$

Sol $S \Rightarrow ab/TS$

$T \rightarrow \epsilon/aT/bT$

g) Obtain CFG to generate the following lan

i) $L = \{w : n_a(w) \bmod 2 = 0 \quad w \in \{a, b\}^*\}$

$S = \{\epsilon, aa, aaaab, \ldots\}$ $\Rightarrow b^*a.b^*ab^*$

$\Rightarrow (b^*ab^*ab^*)^*$

~~$S \rightarrow \epsilon/aa$~~

$\boxed{S \rightarrow \epsilon/SS/SaSaS}$

Derivations — left most derivation
— Right most derivation

Left Most Derivation:

→ At each step replace the left most varia by one of its production bodies.

→ Such derivation is called Left Most Der

→ To indicate left most derivation by us

the relation $\underset{lm}{\Longrightarrow}$ and $\overset{*}{\underset{lm}{\Longrightarrow}}$ for one or more steps respectively.

Right Most Derivation:-

At each step the right most variable is replaced by one of its bodies. Use the symbols $\underset{rm}{\Longrightarrow}$ and $\overset{*}{\underset{rm}{\Longrightarrow}}$ to indicate one (or) many rightmost derivation steps respectively.

Ex: $(a|0|+b|)*(a|+b)$

$E \underset{rm}{\Rightarrow} E*E \quad (E \to E*E)$

$E \underset{rm}{\Rightarrow} E*(E) \quad (E \to (E))$

$E \underset{rm}{\Rightarrow} E*(E+E) \quad (E \to E+E)$

$E \underset{rm}{\Rightarrow} E*(E+I) \quad (E \to I)$

$E \underset{rm}{\Rightarrow} E*(E+b) \quad (I \to b)$

$E \underset{rm}{\Rightarrow} E*(I+b) \quad (E \to I)$

$E \underset{rm}{\Rightarrow} E*(I|+b) \quad (I \to I|)$

$E \underset{rm}{\Rightarrow} E*(a|+b) \quad (I \to a)$

$E \underset{rm}{\Rightarrow} (E)*(a|+b) \quad (E \to (E))$

$E \underset{rm}{\Rightarrow} (E+E) * (a|+b)$ $(E \to E+E)$

$E \underset{rm}{\Rightarrow} (E+I) * (a|+b)$ $(E \to I)$

$E \underset{rm}{\Rightarrow} (E+I1) * (a|+b)$ $(E \to I1)$

$E \underset{rm}{\Rightarrow} (E+b1) * (a|+b)$ $(I \to b)$

$E \underset{rm}{\Rightarrow} (I+b1) * (a|+b)$ $(E \to I)$

$E \underset{rm}{\Rightarrow} (I1+b1) * (a|+b)$ $(I \to I1)$

$E \underset{rm}{\Rightarrow} (I01+b1) * (a|+b)$ $(I \to I0)$

$E \underset{rm}{\Rightarrow} (I101+b1) * (a|+b)$ $(I \to I1)$

$E \underset{rm}{\Rightarrow} (a101+b1) * (a|+b)$ $(I \to a)$

$E \underset{rm}{\overset{*}{\Rightarrow}} (a101+b1) * (a|+b)$

Language of a Grammar:

If grammar is $(V\ T\ P\ S)$ is a CFG the language of G, denoted as $L(G)$ is the set terminal strings that have derivation the start symbol. that is

$$L(G) = \{ w \text{ in } T^* / \ S \underset{G}{\overset{*}{\Rightarrow}} w \}$$

If a language L is the language of some CFG then L is said to be Context Free Languag

Q) $P \rightarrow \epsilon/a/b/aPa/bPb$

$P \Rightarrow baabab$

$\Rightarrow bpb$

$\Rightarrow bapab$

$\Rightarrow baapaab$

$\Rightarrow baaaab \neq baabab$ (not a palindrome)

Sentential Forms

Derivation from the start symbol produce strings. If $G = \{V, T, P, S\}$ is a CFG, then any string 'α' in $(V \cup T)^*$ such that the start symbol $S \underset{lm}{\overset{*}{\Rightarrow}} \alpha$, then '$\alpha$' is a left-sentential form.

if $S \underset{rm}{\overset{+}{\Rightarrow}} \alpha$, α is a right sentential form the language $L(G)$ is those sentential form that are in T^* if they consists solely of termin

→ Parse Tree

Constructions of ParseTree: The parse tree for G are trees with the following conditions

1) Each interior node is labelled by a Variable in
2) Each leaf is labelled by either a Variable a ter
(or) epsilon (ϵ)
— If the leaf is labeled ϵ, then it must be i
only child of its parents.
— If an interior node is labelled A and its
children are labelled $x_1, x_2 - - x_k$ respectively
from the left then $A \rightarrow x_1, x_2 - - , x_k$ is
production in P. P is set of productions

The yield of the Tree

The leafs of any parse tree and concatenat
them from the left we get a string calle
the yield of the tree which is always a
string i;e derived from root of variabl.

1. The yield is the terminal string i;e all leafs
are labelled either terminal is ϵ.
2. The root is labeled by start Symbol.
3. The set of yields of those parse Tree. having
4. Start Symbol at the root & the terminal sy
derive the language of gramma.

CFG

$S \rightarrow ictS \mid ictSeS \mid a$

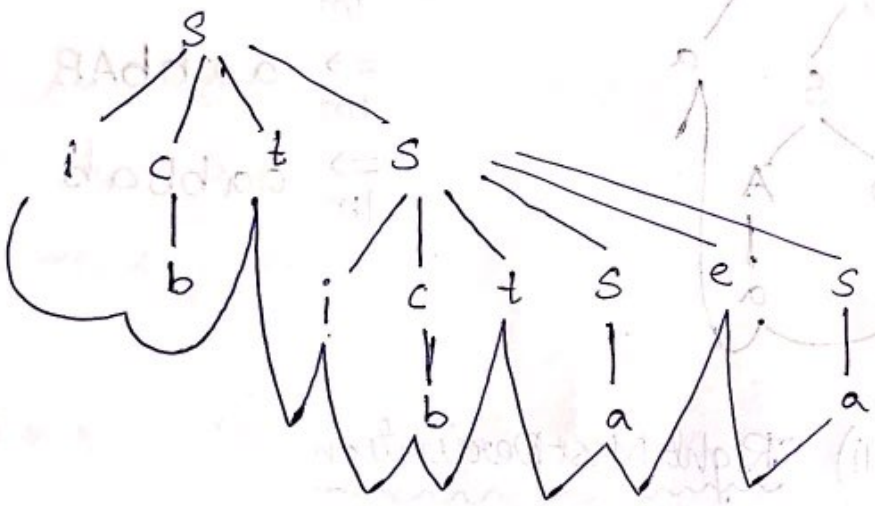
$c \rightarrow b$

Q) ibtibtaea

→ left most derivation

→ right most derivation

→ parse tree



ibtibtaea

LMD

$S \underset{lm}{\Rightarrow} icts$

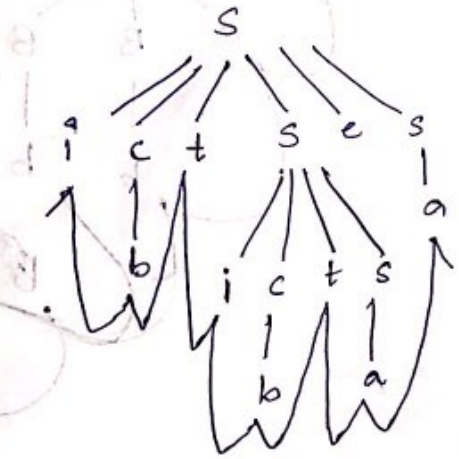
$\underset{lm}{\Rightarrow} ibts$

$\underset{lm}{\Rightarrow} ibtictSeS$

$\underset{lm}{\Rightarrow} ibtictaeS$

$\underset{lm}{\Rightarrow} ibtictaea$

left most derivation



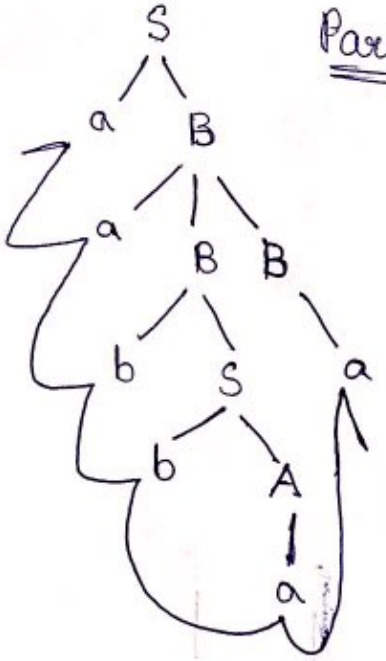
8) $S \rightarrow aB/bA$

9) $A \rightarrow as/bAA/a$

$B \rightarrow bS/bBB/b$

Verify aabbab

Parse tree



i) Left most derivation

$S \underset{lm}{\Rightarrow} aB$

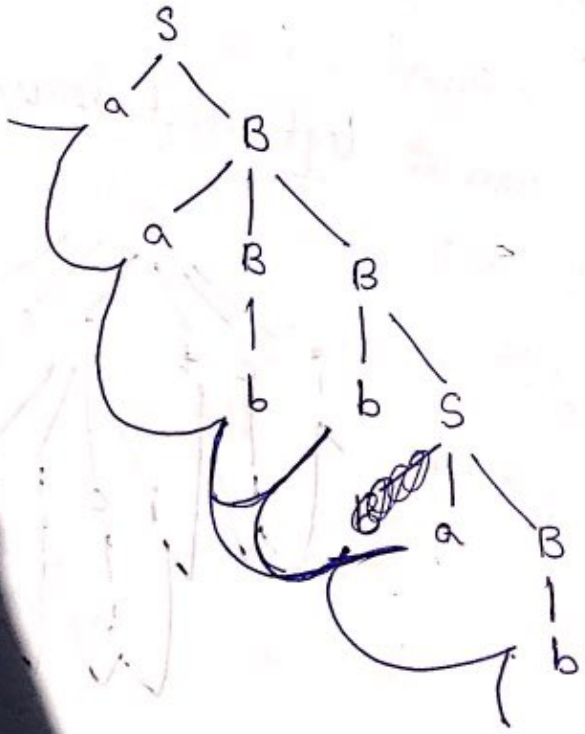
$\underset{lm}{\Rightarrow} aaBB$

$\underset{lm}{\Rightarrow} aabSB$

$\underset{lm}{\Rightarrow} aabbAB$

$\underset{lm}{\Rightarrow} aabbab$

ii) Right Most Derivation



$S \underset{rm}{\Rightarrow} aB$

$\underset{rm}{\Rightarrow} aaBB$

$\underset{rm}{\Rightarrow} aaBbS$

$\underset{rm}{\Rightarrow} aaBbaB$

$\underset{rm}{\Rightarrow} aaBbab$

$\underset{rm}{\Rightarrow} aabbab$

$\Rightarrow aabbab$

Ambiguity in Grammar

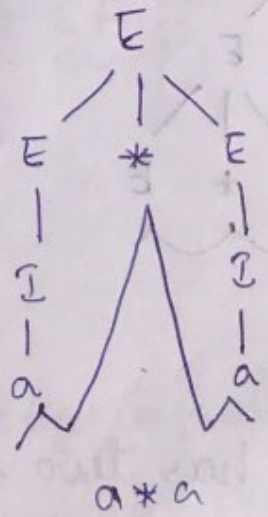
→ A Context Free Grammar $G = (V, T, P, S)$ is ambiguous if there is atleast one string 'w' in T^* for which you can find two different parse tree each with root labeled 'S' and yield 'w'.

→ If each string has atmost one parse tree in the grammar then the grammar is unambiguous.

Q) a * a using previously given production rules.

Sol° L.M.D

$$E \underset{lm}{\Longrightarrow} E * E$$
$$\underset{lm}{\Longrightarrow} I * E$$
$$\underset{lm}{\Longrightarrow} a * E$$
$$\underset{lm}{\Longrightarrow} a * I$$
$$\Longrightarrow a * a$$



RMD

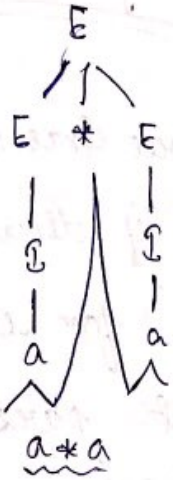
$E \underset{rm}{\Rightarrow} E*E$

$\underset{rm}{\Rightarrow} E*I$

$\underset{rm}{\Rightarrow} E*a$

$\underset{rm}{\Rightarrow} I*a$

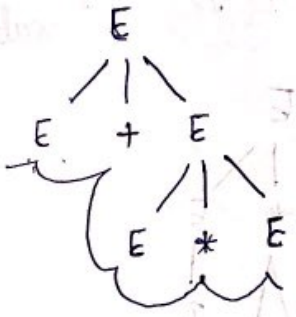
$\underset{rm}{\Rightarrow} a*a$



⇒ Since It has only one Parse Tree it is nonambi

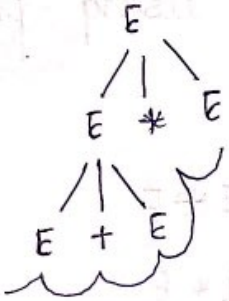
Q2. E + E * E

Sol: L.M.D



E + E * E

RMD



E + E * E

⇒ Since it has two structures for one Gramma
hence the Grammar is an ambiguous gra

Note: *) To avoid/eleminate ambiguity

i> Use precedence rule & paranthesis.

Ambiguity

LMD is one of the method to eliminate Ambiguity

For each Grammar $G=(V, T; P, S)$ and string w has two distinct parse tree if & only if w have two distinct LMD from S.

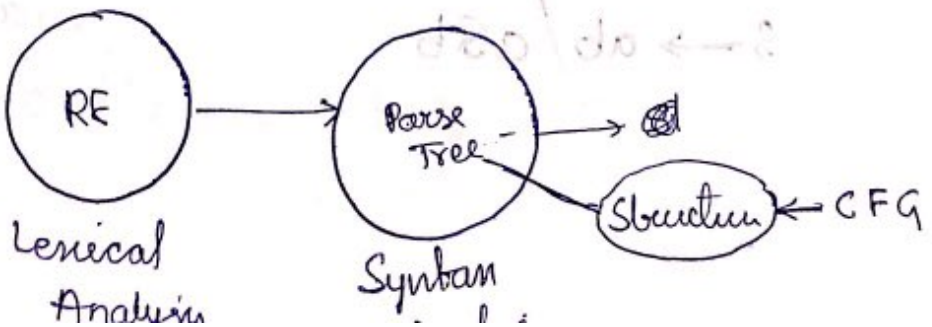
Inherent Ambiguity

A context Free Language 'L' is said to be inherently ambiguous if all its grammar are ambiguous. if even one grammar for 'L' is unambiguous then L is an unambiguous language.

Applications of Amb CFG

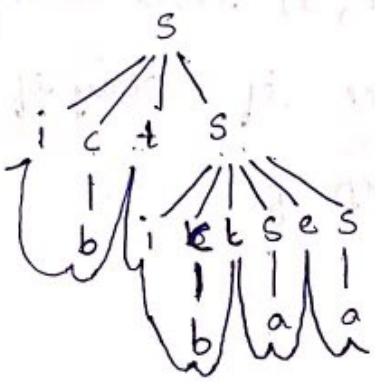
1. Parsers
2. YACC parser generator
3. Markup Languages.
4. XML and document type definitions.

Compiler



Q) ibtibtaea

LMD



ibtibtaea

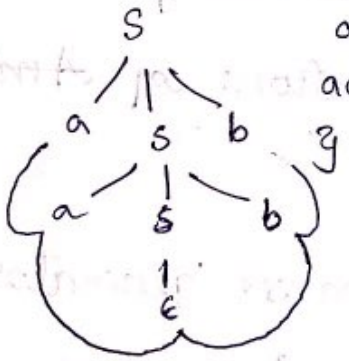
RMD

ibtibtaea

Q) CFG for $L = \{a^n b^n ; n \geq 0\}$

Sol^n $S \rightarrow \epsilon / aSb$

$\{\epsilon,$ ab, aabb, aaabbb $\}$



aabb

Q) C.FG for $L = \{a^n b^n ; n \geq 1\}$

Sol^n $S \rightarrow ab / aSb$

$\{ab, aabb, aaabbb\}$

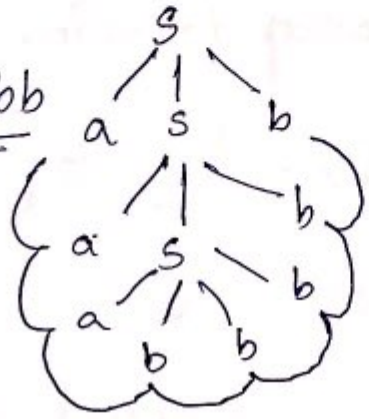
Q) $L = \{a^{n+1} b^n ; n \geq 0\}$ $W = \{a, aab, aaabb, aSb$

Sol^n $S \rightarrow a / aSb$

Q) $L = \{a^n b^{n+2}, n > 0\}$ $w = \{bb, abbb, aaabbbbb\}$

$S \rightarrow bb / aSb$

verify aaabbbbb



Q) $L = \{a^n b^{2n} ; n \geq 0\}$ $w = \{\epsilon, abb, aabbbb, aaabbbbbb\}$

Sol^n $S \rightarrow \epsilon / aSbb$

Q) CFG for Palindrome over $\Sigma = \{a, b\}$.;

Sol^n $S \rightarrow \epsilon / a / b / aSa / bSb$ $w = \{abba, abbba, abba, baab\}$

Q) CFG for non palindromes over $\Sigma = \{a, b\}$

Sol^n $S \rightarrow ab / ba / aSa / bSb / aS / bS$ abb

$S \longrightarrow aSa / bSb / A$

$A \longrightarrow aBb / bBa$

$B \longrightarrow aB / bB / \epsilon$

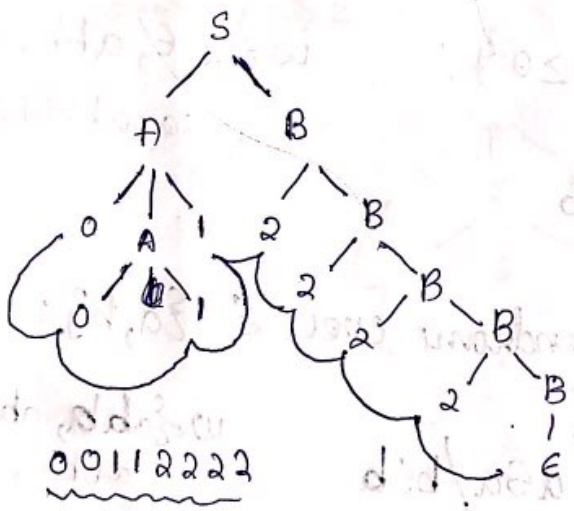
Q) $L = \{ 0^m 1^m 2^n / m \geq 1 \text{ and } n \geq 0 \}$

SOL^n $S \longrightarrow AB$

$A \longrightarrow 01 / 0A1$

$B \longrightarrow \epsilon / 2B$

Verify:- 00112222



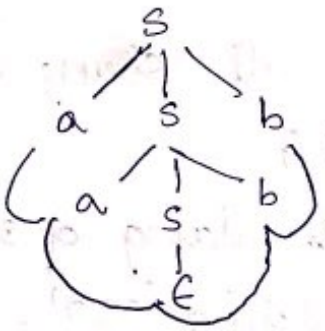
00112222

Q) $L = \{ w / n_a(w) = n_b(w) \}$ $\Sigma = \{a, b$

SOL^n $S \longrightarrow \epsilon / aSb / bSa / SS$

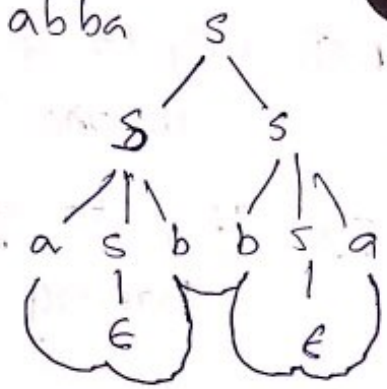
$\{ \epsilon,$
ab
abba
abab
baba
baab

verify 1) aabb



aabb

2) abba



abba

8) CFG for balanced parantheses ('(', '{', '[')

Sol: $S \rightarrow \epsilon / [S] / (S) / \{S\}$

9) CFG to generate Integers.

Sol: $S \rightarrow SN / UN$

$SN = \{+, -\}$

$SN \rightarrow +N / -N$

$UN \rightarrow N$

$D \rightarrow 0/1/2/--/9$

$D \rightarrow 0/1/2/--/9$

$N \rightarrow D/ND$

9) CFG for identifiers

$S \rightarrow$ letter / letter digit

1 Q) CFG to generate all string with exactly one a where $\Sigma = \{a, b\}$

2 Q) CFG to generate lang. of string $\Sigma = \{0, 1\}$ having substring '000'.

3 Q) CFG for lang. $L = \{w \in (0,1)^*\}$ and string is of even length.

4 Q) lang $= \{ w \in (0,1)^* \}$ and contains atleast 3 1's.

5 Q) $L = \{ w \in (0, 1)^* \text{ & } L(w) \text{ is odd} \}$

1 Ans:- $W = \{ a, abb, abbb, babb, bbbab \}$

$S \rightarrow a/bS/$

2 Ans:-

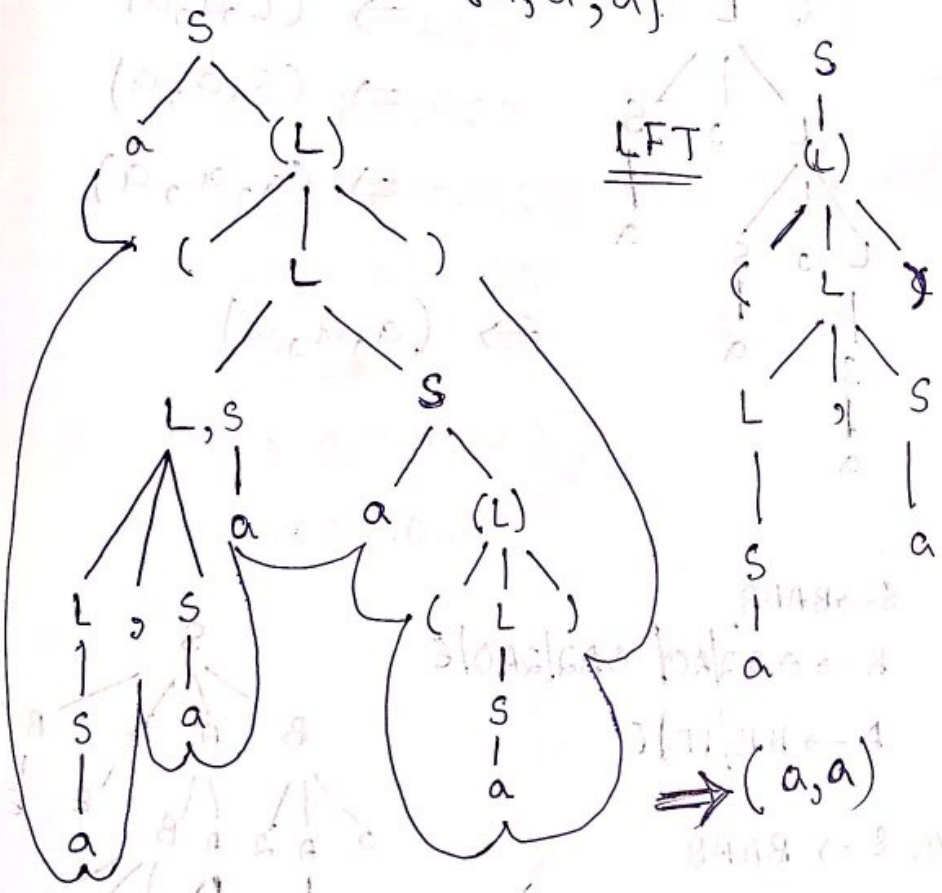
Q) Verify whether the given lang ambiguous language (or) not

i) $S \rightarrow (L) / a$

$L \rightarrow L,S/S$

~~$W = \{a,(a,a),((a,a),a)\}$~~

$W = (a,a,a)$



LFT

$\Rightarrow (a,a)$

~~$\alpha(\alpha,\alpha\alpha\alpha(\alpha))$~~

LFT :-

$S \overset{lt}{\Rightarrow} (L)$

$\Rightarrow (L,S)$

$\Rightarrow (L,S,S)$

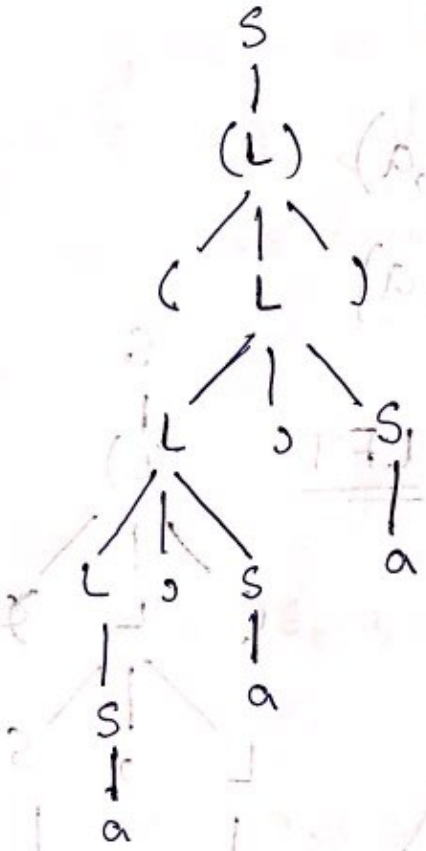
$\Rightarrow (S,S,S)$

$\Rightarrow (a,S,S)$

$\Rightarrow (a,a,S)$

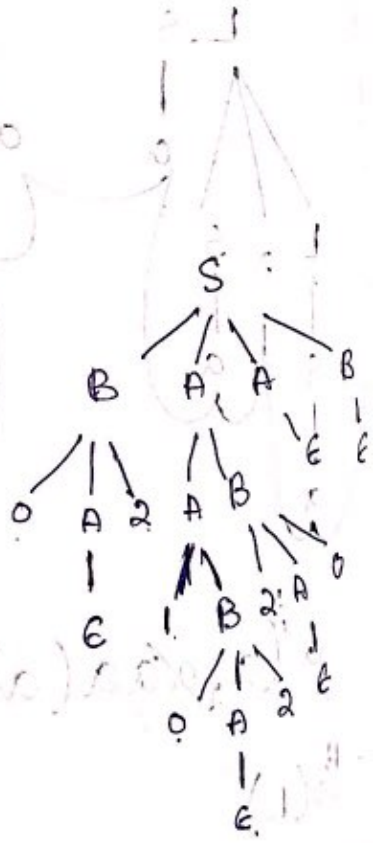
$\Rightarrow (a,a,a)$

RFT: (a, a, a)



$S \underset{rft}{\Rightarrow} (L)$
$\Rightarrow (L,S)$
$\Rightarrow (L,a)$
$\Rightarrow (L,S,a)$
$\Rightarrow (L,a,a)$
$\Rightarrow (S,a,a)$
$\Rightarrow (a,a,a)$

$\Rightarrow (a,a,a)$



29) $S \rightarrow BAAB$

$B \rightarrow$ ~~0A2/A0~~ 0A2/2A0/ϵ

$A \rightarrow AB/1B/\epsilon$

LMD $S \Rightarrow BAAB$
$\Rightarrow 0A2AAB$
$\Rightarrow$ ~~02ABBB~~
$\Rightarrow 02AAB$
$\Rightarrow 02ABAB$
$\Rightarrow 021BBAB$
$\Rightarrow 0210A2BAB$

~~0102220~~

0210220

$\Rightarrow$ 021022A0AB

$\Rightarrow$ 0210220

RMD :- S $\Rightarrow$ BAAB

$\Rightarrow$ BAA2A0

$\Rightarrow$ BAA2ε0

$\Rightarrow$ BAA20

$\Rightarrow$ BA1B20

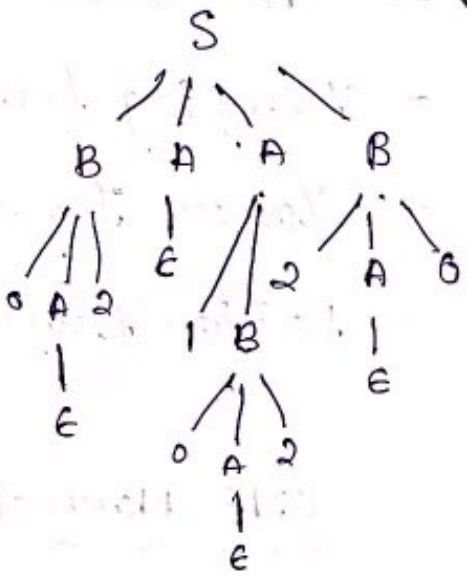
$\Rightarrow$ BA10A220

$\Rightarrow$ BA10220

$\Rightarrow$ B010220

$\Rightarrow$ 0A210220

$\Rightarrow$ 0210220



0210220

Properties of Context Free Grammars

- CNF Normal Form (Chomsky Normal Form)
- Pumping Lemma
- Closure Properties
- Decision Properties

CNF Normal Form

1. eliminate useless symbols. — Non generating Symbol / Non Reachable Symbol
2. eliminate ϵ productions
3. eliminate unit productions

~~1. Eliminate~~

Normal Forms for CFG

1. Chomsky Normal Form: In this all productions are of the form:

$$A \rightarrow BC \quad (or) \quad A \rightarrow a.$$

* The preliminary simplifications for CNF are

1. eliminate useless symbols
2. eliminate ϵ productions

1. Eliminating Useless Symbol

A symbol 'x' is useful for a Grammar G $G=(V,T,P,S)$ if there is some derivation of the form $S \overset{*}{\Rightarrow} \alpha x \beta \overset{*}{\Rightarrow} w$, where w is in T^*.

→ 'x' may be in either N (or) T and the sentential form $\alpha x \beta$ might be the first or last in the derivation.

→ If 'x' is not useful then it is called as useless symbol.

↦ Eliminating useless symbol from a gramm will not change the language gener

→ eliminating useless symbols begins by identifying two things.

i> we say x is generating if $x \overset{*}{\Rightarrow} w$ fo some terminal string 'w';

Note: Every terminal string is generating, since w can be that terminal itself which is derived by zero step.

ii> We say X is reachable if there is a derivation $S \xRightarrow{*} \alpha X \beta$ for some α & β.

Let $G = (V, T, P, S)$ be CFG and assume th $L(G) \neq \emptyset$.

Let $G_1 = (V_1, T_1, P_1, S)$ be a Grammar we obtain by following steps.

~~1. Let (V_2, T_2, P_2, S).~~

. To Compute the generating symbols of G w perform the following induction.

Basis: Every symbol of T is generating it generates itself

Induction: Seppose there is a production A and every symbol of α is already kno to be generating. Then A is generating

Note: this rule indicates the case where all variables that have ϵ as product body are surely generating.

1. $G' = (V', T', P', S)$

Basis: S is surely reachable.

Induction: Suppose we have discovered th

some variable A is reachable, then for all production with A in the head, all the symbols of the bodies of those productions are also reached.

Q) Eliminate useless symbol from the grammar

S → aA / bB

A → aA / a

B → bB

D → ab / Ea

E → aC / d

Generating rules = { ~~S, A, B, a, b~~, }

T = { a, b, d }

A → a

D → ab

E → d

(A, D, E)

(A, D, E)

S → aA

A → aA / a

D → ab / Ea

E → d

(S, A, D, E)

- Generating Symbols = { S, A, D, E, a, b, d }
- Reachable Nodes / Symbols :-

S is reachable = { S }

S → aA = { S, a, A }

A → a / aA = { S, a, A }

∴ After removing useless symbol we get :

S → aA

A → a / aA

Q) Eliminate useless symbol from the gram

Sol. $S \rightarrow aA/a/Bb/cC$

$A \rightarrow aB$

$B \rightarrow a/Aa$

$C \rightarrow cCD$

$D \rightarrow ddd$

$T = \{a, b, c, d\}$

$S \rightarrow a$

$B \rightarrow a$

$D \rightarrow ddd$

(S, B, D)

(S, B, D)

$S \rightarrow a/bB$

$A \rightarrow aB$

$B \rightarrow a/aA$

$D \rightarrow ddd$

(S, A, B, D)

- Generating Symbol = $\{S, A, B, D, a, b, c, d\}$
- Reachable symbol

S is reachable = $\{S\}$

$S \rightarrow a/bBaA = \{S, a, b, B, A\}$

$B \rightarrow a/Aa = \{S, a, b, B, A\}$

$A \rightarrow aB = \{S, a, b, B, A\}$

8) eliminate useless symbols from the gra

9) $S \rightarrow aAa / aBC$

$A \rightarrow aS / bD$

$B \rightarrow aBa / b$

$C \rightarrow abb / DD$

$D \rightarrow aDa$

$T = \{a, b, c\}$

$B \rightarrow b$

$C \rightarrow abb$

$= \{B, C\}$

$S \rightarrow aBC$

$A \rightarrow aS$

$B \rightarrow b / aBa$

$C \rightarrow abb$

Reachable Symbol

S is reachable $= \{S\}$

$S \rightarrow aBC / aAa = \{S, B, C, a, A\}$

$B \rightarrow b / aBa = \{S, B, C, a, b\}$

$C \rightarrow abb = \{S, B, C, a, b\}$

$A \rightarrow aS / bDD = \{S, B, C, A, a, b, D\}$

2. Eliminating ϵ-production:

- Let $G=(V, T, P, S)$ be a CFG we can find the nullable symbol of G by following iterative algorithm.

Basis: If $A \rightarrow \epsilon$ is a production of G then A is nullable.

Induction: If there is a production $B \rightarrow C_1 C_2 \ldots C_k$ where each C_i is nullable then B is nullable. So we only have consider production with all variab bodies.

Construction of Grammar without ϵ-pro

Let $G=(V, T, P, S)$ be CFG determine al nullable symbols of G then we constru a new grammar $G_1=(V_1, T_1, P_1, S)$ whos set of of productions P_1 is determin as follows:

→ For each production $A \rightarrow X_1 X_2 \ldots X_k$ of P where $k \geq 1$.

→ Suppose that m of the k x_i's are nullable symbols the new grammar G_1 will have 2^m versions of this production where the nullable x_i's in all possible combinations are present or absent. There is one exception.

→ if $m=k$ i.e all symbols are nullable, then we do not include the case where all x_i's are absent.

Note: If a production of the form $A \to \epsilon$ is in P, we do not place this production in P_1.

Eliminate ϵ productions from the foll. Grammar.

$S \to ABCa / bD$

$A \to BC / b$

$B \to b / \epsilon$

$C \to c / \epsilon$

$D \to d$

Sol: $B \to \epsilon$ & $C \to \epsilon$ ∴ B, C are nullable variable

$A \to BC$ ∴ A is nullable variable.

→ A, B, C are nullable variable

- S → ABCa / ABa / BCa / AC / a / Aa / Ba / Ca / a
- S → bD
- B → b
- A → BC / B / C
- A → b
- C → C
- D → d

Q) S → ABC

A → BC / a

B → bAC / ϵ

C → cAB / ϵ

Sol: B → ϵ, C → ϵ ∴ B, C are nullable

A → BC ∴ A is nullable

S → ABC ∴ S is nullable

⇒ S, A, B, C are nullable variable

S → ABC / AB / BC / AC / A / B / C

A → a

A → BC / B / C

→ A, B, C are nullable variable

- $S \to ABCa / ABa / BCa / AC / a / Aa / Ba / Ca / a$
- $S \to bD$
- $B \to b$
- $A \to BC / B / C$
- $A \to b$
- $C \to C$
- $D \to d$

Q) $S \to ABC$

$A \to BC / a$

$B \to bAC / \epsilon$

$C \to cAB / \epsilon$

Sol $B \to \epsilon$, $C \to \epsilon$ ∴ B, C are nullable.

$A \to BC$ ∴ A is nullable

$S \to ABC$ ∴ S is nullable

⇒ S, A, B, C are nullable variable

$S \to ABC / AB / BC / AC / A / B / C$

$A \to a$

$A \to BC / B / C$

$B \rightarrow bAC / bA / bC / b$

$C \rightarrow cAB / cA / cB / c$

3. Eliminate Unit Production

The steps to follow

Given a CFG $G = \{V, T, P, S\}$. construct CF $G_1 = (V, T, P_1, S)$

1> Find all the unit pairs of G

2> For each unit pair (A, B) add to P_1 all production $A \rightarrow \alpha$ where $B \rightarrow \alpha$ is non unit production in P.

Note : $A = B$ is possible in that way P_1 contains all non unit productions in P.

Q) Eliminate unit prodn from the Grammar

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow C / b$

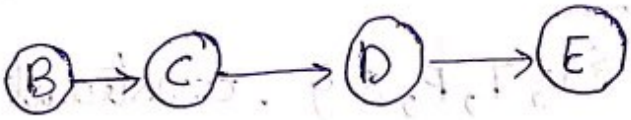
$C \rightarrow D$

$D \rightarrow E / bc$

$E \rightarrow d / Ab$

Sol

non unit production	Unit production
$S \to AB$	~~$S \to AB$~~
$A \to a$	$B \to C$
$B \to b$	$C \to D$
$D \to bC$	$D \to E$
$E \to d/Ab$	



~~$\Rightarrow$~~ $\Rightarrow D \to E$ $\Downarrow$ $D \to d/Ab/bc$	$C \to D$ $\Downarrow$ $C \to bc/d/Ab$

$S \to AB$

$A \to a$

$B \to bc/b/Ab/d$

$C \to bc/d/Ab$

$D \to d/Ab/bc$

9) $S \to Aa/B/Ca$

$B \to aB/b$

$C \to Db/D$

$D \to E/d$

$E \to ab$

non unit prod^n | unit prod^n

$S \rightarrow Aa / Ca$ | $S \rightarrow B$

$B \rightarrow aB / b$ | $E \rightarrow D$

$C \rightarrow Db$ | $D \rightarrow E$

$D \rightarrow d$

$E \rightarrow ab$

(C) → (D) → (E)

$D \rightarrow E$

$D \rightarrow ab$

$C \rightarrow D$

$C \rightarrow d/ab/Db$

(S) → (B)

$S \rightarrow aB/b$

$S \rightarrow Aa / Ca / aB / b$

$B \rightarrow aB / b$

$C \rightarrow Db / d / ab$

$D \rightarrow d / ab$

9) Eliminate Unit productions from the fol. Gramm

$S \rightarrow AO / B$

$B \rightarrow A / 11$

$A \rightarrow 0 / 12 / B$

Non-unit prod^n

$S \rightarrow AO$

$B \rightarrow 11$

$A \rightarrow 0/12$

Unit prod^n

$S \rightarrow B$

$B \rightarrow A$

$A \rightarrow B$

$S \overset{*}{\Rightarrow} A$ — $S \rightarrow AO/0/12/11$

$S \overset{*}{\Rightarrow} B$ —

$A \overset{*}{\Rightarrow} B$ — $A \rightarrow 11/0/12$

$B \overset{*}{\Rightarrow} A$ — $B \rightarrow 11/0/12$

CNF :-

- ϵ production
- unit production
- useless symbols

Productions

- In CNF $A \rightarrow BC$ (where A, B, C are variable)
- $A \rightarrow a$; A is variable & a is terminal

⇒ To convert into CNF we need to

1. eliminate ϵ productions
2. " unit "
3. " useless "

From the resulting grammar do the foll. tasks

a. Arrange that all bodies of length two (or) more consist only of variable.

b. Break bodies of length 3 (or) more into a cascade of prodⁿ each with a body consist of 2 variable

* The construction for 'a' case is as follows :-

→ For every terminal a that appears in a body of length 2 (or) more, create a new variable. say A this variable has only one prodⁿ $A \rightarrow a$. Now we use A in place of terminal a everywhere a appears in a body of length 2 or more. At this point every production has a body that is either a single terminal or atleast two variable and no terminals.

→ Construction of 'b' case :-

we must break those productions that is

→ $A \rightarrow B_1 B_2 - - B_k$ for $k \geq 3$, into a group of productions with two variables in each body.

→ We introduce $(k-2)$ new variable i;e $C_1, C_2 - -, C_{k-2}$

The original prodⁿ is replaced by the

(k-1) productions.

$A \rightarrow B_1 B_2 - - B_k$

$A \rightarrow B_1 C_1$

$C_1 \rightarrow B_2 C_2$

$C_{k-3} \rightarrow B_{k-2} C_{k-2}$

$C_{k-2} \rightarrow B_{k-1} B_k$

Q) Convert the foll. into CNF.

Soln

$E \rightarrow E + T$

$E \rightarrow T * F$

$E \rightarrow (E)$

$E \rightarrow a/b/Ia/Ib/I0/I1$

$T \rightarrow T * F$

$T \rightarrow (E)$

$T \rightarrow a/b/Ia/Ib/I0/I1$

$F \rightarrow (E)$

$F \rightarrow a/b/Ia/Ib/I0/I1$

$I \rightarrow a/b/Ia/Ib/I0/I1$

Non Terminals : {a, b, 0, 1, (,), +, *}

$A \to a$ $\quad L \to ($

$B \to b$ $\quad R \to)$

$Z \to 0$ $\quad P \to +$

$O \to 1$ $\quad M \to *$

$E \to EPT / TMF / LER / A / B / IA / IB / IZ / IO$

$T \to TMF / LER / A / B / IA / IB / IZ / IO$

$F \to LER / A / B / IA / IB / IZ / IO$

$I \to A / B / IA / IB / IZ / IO$

1) • $E \to EE_1$

$E_1 \to PT$

• $E \to TM_1$

$M_1 \to MF$

• $E \to LE_2$

$E_2 \to ER$

• $E \to A / B / IA / IB / IZ / IO / EE_1 / TM_1 / LE_2$

2) • $T \to TM_1$

~~$M_1 \to ME$~~

• $T \to LE_2$

• $T \to TM_1 / LE_2 / A / B / IA / IB / IZ / IO$

3) $F \to LE_2 / A / B / IA / IB / IZ / IO$

4) $I \to A / B / IA / IB / IZ / IO$

Q) $S \to 0A / 1B$ Convert to CNF

$A \to 0AA / 1S / 1$

$B \to 1BB / 0S / 0$

Sol. $Z \to 0$

$0 \to 1$

$S \to ZA / 0B$

$A \to ZAA / 0S / 0$

$B \to 0BB / ZS / Z$

$A \to ZA_1$ | $B \to 0B_1$

$A_1 \to AA$ | $B_1 \to BB$

$S \to ZA / 0B$

$A \to ZA_1 / 0S / 0$

$B \to 0B_1 / ZS / Z$

Q) $S \to ABC / BaB$

$A \to aA / BaC / aaa$

$B \to bBb / a / D$

$C \to CA / AC$

$D \to \epsilon$

i) elimination ϵ prodn

D → ε nullable

B → D ∴ nullable.

B, D are nullable.

S → ABC/AC/BaB/Ba/aB/a

A → aA/BaC/aC/aaa

B → bBb/bb/a/D

C → CA/AC

ii) Unit Pro

D → ε

B → D

non unit Prod

S → ABC/AC/Ba

Ba/aB/a

A → aA/BaC/aC/a

B → bBb/bb/a/

C → CA/AC

ii) unit prodn :. ~~No unit prodn~~

iii) Useless symbol

T = {a, b}

S → a

A → aaa

B → bb/a

(S, A, B)

S → BaB/Ba/aB/a

A → aA/aaa

B → bBb/bb/a

~~Correct~~

Generating Symbol = {S, A, B}

S → BaB/Ba/aB/a

A → aA/aaa

B → bBb/bb/a

A₁ → a

B₁ → b

Reachable = {S}

= {S, a, B}

= {S, a, b, B}

$S \rightarrow BA_1B / BA_1 / A_1B / A_1$

$A \rightarrow A_1A / A_1A_1A_1$

$B \rightarrow B_1BB_1 / B_1B_1 / A_1$

$S \rightarrow BC_1$

GNF

such that if z is any string L, $|z|$ is atleast n, then we can write $z = uvwxy$ subject to the following conditions.

1. $|vwx| \leq n$, middle portion is not too long
2. $vx \neq \epsilon$ since v & x are the pieces to be pumped this cond^n says that atleast one of the strings we pump must not be empty.

3. $\forall\ i \geq 0$; $uv^i wx^i y$ is in L i;e the two strings v and x may be pumped any no. of times including 0 and the resultant string be a member of L.

Push-Down-Automata

(R.L + Stack + ϵ transition)

A formal notation of P-d-automata includes seven components. i.e

$P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q = Finite set of states.

Σ = Finite set of i/p symbols.

Γ = Finite stack alphabet is the set of symbols that we are allowed to push onto the stack.

$\delta(q, a, x)$; δ = The transition function δ takes an argument a tuple q is a state in Q, a is either an i/p symbol in Σ (or) ϵ i;e empty string, x is a stack symbol that is a member of Γ

→ The o/p of δ is a finite set of pairs (p,
where P is the new state & γ is the stri
of stack symbols that replaces X at the
top of the stack.

→ if γ = ∧ then stack is unchanged.

→ if γ = γz then X is replaced by z and
is pushed onto the stack.

→ if γ = ∈ then the stack is pop.

* z_0 is start symbol initially the PDA
consist of one instance of the symbol
nothing else.

F: is set of accepting state.

Q) i) $\delta(q_0, a, z_0) = (q_0, a z_0)$

ii) $\delta(q_0, b, z_0) = (q_0, b z_0)$

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

iv) $\delta(q_0, b, a) = (q_0, ba)$

v) $\delta(q_0, a, b) = (q_0, ab)$

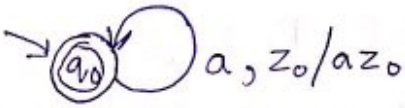
vi) $\delta(q_0, b, b) = (q_0, bb)$

vii) $\delta(q_0, c, z_0) = (q_1, z_0)$

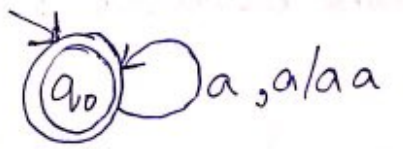
viii) $\delta(q_0, c, a) = ($

ix) $\delta(q_0, c, b) = ($

Draw the diagram for the above transitions

i) 

ii) 

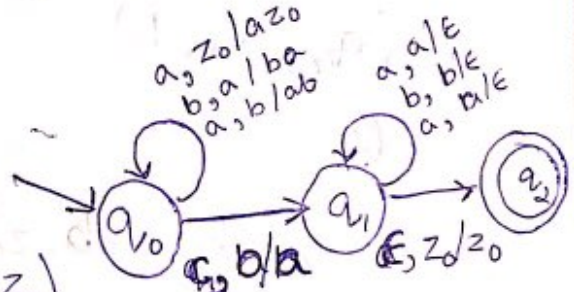
iii) 

iv) 

* triple $(q, a, z) = (p, a)$

$q \in Q$, $a \in \Sigma \cup \{\epsilon\}$, $z \in \Gamma$; $p \in Q$, $a \in \Gamma^*$

8) abacaba



Sol) i) $\delta(q_0, a, z_0) = (q_0, az_0)$

ii) $\delta(q_0, b, a) = (q_0, ba)$

iii) $\delta(q_0, a, b) = (q_0, ab)$

iv) $\delta(q_0, c, a) = (q_1, a)$

v) $\delta(q_1, a, a) = (q_1, \epsilon)$

vi) $\delta(q_1, b, b) = (q_1, \epsilon)$

vii) $\delta(q_1, a, a) = (q_1, \epsilon)$

viii) $\delta(q_1, \epsilon, z_0) = (q_2, z_0)$

$P = \{Q, \Sigma, T, q_0, \delta, z_0, F\}$

* $Q = \{q_0, q_1, q_2\}$ $\Sigma = \{a, b\}$ $T = \{a, b, z_0\}$

q_0 = start state z_0 = stack initial state $F = \{q_0$

Instantaneous Description

Ex (a) $(q_0, abacaba, z_0) \vdash (q_0, bacaba, az_0)$

$(q_0, bacaba, a) \vdash (q_0, acaba, ba)$

$(q_0, acaba, b) \vdash (q_0, caba, ab)$

$(q_0, caba, a) \vdash (q_1, aba, a)$

$(q_1, aba, a) \vdash (q_1, ba, \epsilon)$

$(q_1, ba, b) \vdash (q_1, a, \epsilon)$

Ex :- $(q_0, abacabba, z_0) \vdash (q_0, bacabba, az_0)$

$\vdash (q_0, acabba, baz_0)$

$\vdash (q_0, cabba, abaz_0)$

$\vdash (q_1, abba, abaz_0)$

$\vdash (q_1, bba, baz_0)$

$\vdash (q_1, ba, az_0)$

$\vdash$ Invalid String

→ The current config. of pda at any given instant can be described by an instantaneous description.

→ Id gives current state of the pda & remaining string to be process & entire content of the process.

→ Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA an ID is defined as 3-tuple (or) tuple $(q, w,$

→ Let the current config. of PDA be $(q, aw, z\alpha$

$\delta(q, a, z) = (p, \beta)$ then new config. with be $p, w, \beta\alpha$ i;e

$(q, aw, z\alpha) \vdash (p, w, \beta\alpha)$

→ The config $(q, aw, z\alpha)$ derives $(p, w, \beta\alpha)$ in one move

→ If an arbitary no. of moves are used to move from one config to another config. then the moves are denoted by the symbol $\vdash^+$ (or) $\vdash^*$

Ex: $(q_0, abacaba, z_0) \vdash^* (q_1, caba, abaz_0)$

Construct P.D.A for $L = \{a^n b^n \mid n > 0\}$

Soln : aaaabbbb

$\rightarrow \delta(q_0, a, z_0) = (q_0, a z_0)$

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

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

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

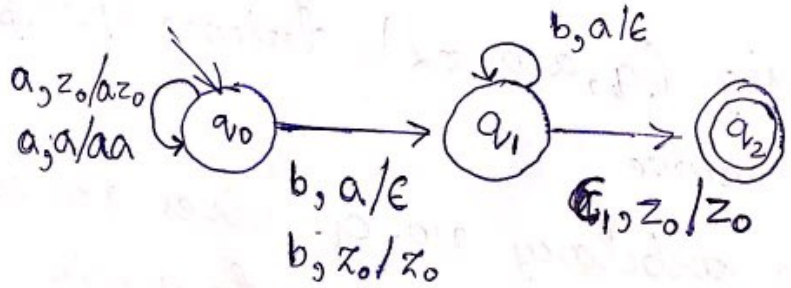
$\Rightarrow \delta(q_0, b, a) = (q_1, \epsilon)$; $\Rightarrow \delta(q_0, b, z_0) = (q_1, z$

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

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

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

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$



PDA

- Acceptance by Final State
- Acceptance by Empty Stack

8) Design PDA when $n(a) = n(b)$ when $\Sigma = \{a, b\}$

Sol 1. a a b a b b a a b b

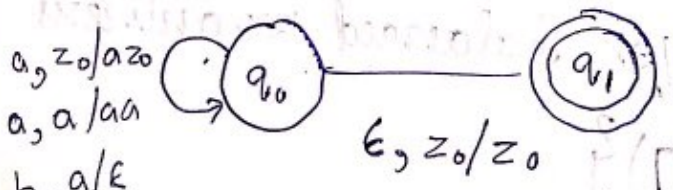
$\delta(q_0, a, z_0) = (q_0, az_0)$

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

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

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

$\boxed{\delta(q_0, \epsilon, z_0) = (q_1, z_0)}$ Acceptance by final state



* by empty stack $\boxed{\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)}$

8) Design ~~PDA for~~ balanced parenthesis

Sol 2. $\delta(q_0, abbaaabb, z_0) \vdash^* (q_0, \epsilon, z_0)$

2. $(q_0, abbaacbb, z_0) \vdash^* (q_0, bbaacbb, az_0)$

$\vdash^* (q_0, baacbb, z_0)$

$\vdash^* (q_0, aacbb, bz_0)$

$\vdash^* (q_0, acbb, z_0)$

$\vdash^* (q_0, cbb, az_0)$

$\vdash^* (q_0, bb, aaz_0)$

$\vdash^* (q_0, b, az_0)$

$\vdash^* (q_0, \epsilon, z_0)$

Q) Design P.D.A for balanced parenthesis

Sol: $\{a-(a+[b*a])\}$

$\delta(q_0, \{, z_0) = (q_0, \{z_0)$

$\delta(q_0, (, \{) = (q_0, (\{$

$\delta(q_0, [, () = (q_0, [($

$\delta(q_0,], [) = (q_0, \epsilon)$

$\delta(q_0,), () = (q_0, \epsilon)$

$\delta(q_0, \}, \{) = (q_0, \epsilon)$

$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$

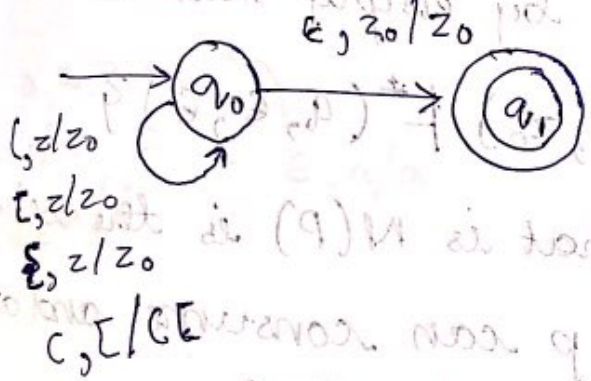
$\delta(q_0, (, z_0) = (q_0, (z_0)$

$\delta(q_0, [, z_0) = (q_0, [z_0)$

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

$\delta(q_0, [, [) = (q_0, [[)$

$\delta(q_0, \{, \{) = (q_0, \{\{)$



Languages of PDA

1. Acceptance by Final State

Let $P = (Q, \Sigma, T, \delta, q_0, z, F)$ be a PDA then $L(P)$, the language accepted by P by Final state is $\{w / (q_0, w, z_0) \vdash_P^* (q, \epsilon, \alpha)\}$ for some state q in F and any stack string α. i.e starting in the initial ID with w waiting on i/p P consumes w

from the i/p and enters an accepting sta
the contents of the stack at that time is
irrelevant

2. Acceptance by Empty Stack

for each PDA $P=(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ we also define $N(P)$ i.e language accepted by PDA by empty stack is

$N(P) = \{w \mid (q_0, w, z_0) \vdash^* (q, \epsilon, \epsilon)\}$

→ For any state q that is $N(P)$ is the set inputs w that p can consume and the same time empty its stack.

Q) Design PDA for the language $L = \{a^n b^{2n}$

Sol $\delta(q_0, a, z_0) = (q_0, aa)$

$\delta(q_0, b, z_0) \neq (q_0, bz)$

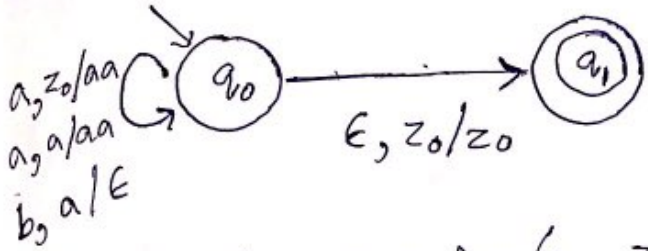
$\delta(q_0, a, a) = (q_0, aa)$

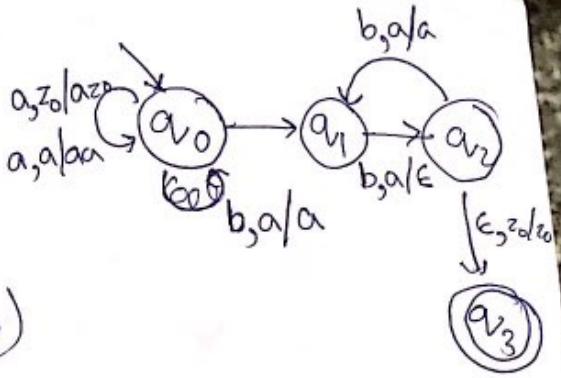
$\delta(q_0, a, b) \neq$

$\delta(q_0, b, b) \neq (q_0, bb)$

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

$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$





$$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$$

g) $n_a > n_b$